A Safety Fallback Controller for Improved Collision Avoidance

Daniel Genin
JHU Applied Physics Laboratory
Laurel, MD USA
Daniel.Genin@jhuapl.edu

Elizabeth Dietrich
JHU Applied Physics Laboratory
Laurel, Maryland USA
Elizabeth.Dietrich@jhuapl.edu

Yanni Kouskoulas
JHU Applied Physics Laboratory
Laurel, Maryland USA

Aurora Schmidt
JHU Applied Physics Laboratory
Laurel, Maryland USA
Aurora.Schmidt@jhuapl.edu

Marin Kobilarov
JHU Whiting School of Engineering
Baltimore, Maryland USA
marin@jhu.edu

Kapil Katyal
JHU Applied Physics Laboratory
Laurel, Maryland USA

Shahriar Sefati
JHU Whiting School of Engineering
Baltimore, Maryland USA

Subhransu Mishra
JHU Whiting School of Engineering
Baltimore, Maryland USA

Ivan Papusha
JHU Applied Physics Laboratory
Laurel, Maryland USA

2023 IEEE International Conference on Assured Autonomy (ICAA) | 979-8-3503-2601-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICAA58325.2023.00026

***Abstract*—We present an implementation of a formally verified safety fallback controller for improved collision avoidance in an autonomous vehicle research platform. Our approach uses a primary trajectory planning system that aims for collision-free navigation in the presence of pedestrians and other vehicles, and a fallback controller that guards its behavior. The safety fallback controller excludes the possibility of collisions by accounting for nondeterministic uncertainty in the dynamics of the vehicle and moving obstacles, and takes over the primary controller as necessary. We demonstrate the system in an experimental set-up that includes simulations and real-world tests with a 1/5-scale vehicle. In stressing simulation scenarios, the safety fallback controller significantly reduces the number of collisions.**

***Index Terms*—Formal Methods, Autonomous Vehicles**

I. Introduction

While artificially intelligent technology can enable increasing levels of ground vehicle autonomy, the question of how to safely adopt such autonomy remains critical to the successful translation of this performance to operational settings. We investigate the use of an approach to collision avoidance that has been formally verified to construct a safe fallback controller that guards the behavior of primary navigation planner. Through the use of an independent fallback system, our approach enables the safe adoption of complex autonomy systems that optimize a number of performance factors. Furthermore, this approach could enable better flexibility of the system to incorporate new autonomous planning algorithms, while keeping the risk of such adoption low. This approach has been demonstrated in a system for the safe testing of autonomous aircraft, [1]. In this work, we explore the use of formally verified methods for safety controllers in the fundamental design of the autonomous vehicle system.

This work was funded by the Johns Hopkins University Institute for Assured Autonomy, https://iaa.jhu.edu

The design of the Verified Assured Learning for aUtonomous Embedded Systems (VALUES) is illustrated in Figure 1. A primary controller employs physics- and learning-based algorithms to arrive at a fused policy for vehicle path planning. The secondary controller (box T2) monitors the system and only intervenes on the planned navigation actions if the system is in a critical state; i.e., a state in which only a restricted set of actions can guarantee the future ability to avoid collisions. In this investigation, the primary controller employs greedy optimization of a cost function that uses a finite horizon roll out. The cost function of the primary controller combines a number of components pertaining to the vehicle dynamics, such as jerk and progress towards goal, as well as safety metrics, such as distance to obstacles, into a single scalar value. By optimizing a scalar-valued control function, the primary controller policy attempts to simultaneously balance performance sometimes resulting in significantly reduced safety margins, which in the presence of noise, may result in collisions. The watchdog fallback controller design is based on our prior work in designing a formally verified fallback controller for aircraft collision avoidance and is reviewed in sections II and III. We demonstrate with simulations and vehicle tests that in such scenarios the safety fallback controller significantly reduces the number of collisions. As shown in Fig. 1, this architecture also allows for autonomous systems that continually learn, as the fallback controller maintains the safety of the overall system even when the behavior of the primary controller is altered.

A. Prior Work

Early work proposing the use of a fallback control architecture includes the *simplex architecture* of [2]. Indeed, the safe adoption of autonomy in vehicles necessitates a means of

979-8-3503-2601-7/23/$31.00 ©2023 IEEE
DOI 10.1109/ICAA58325.2023.00026

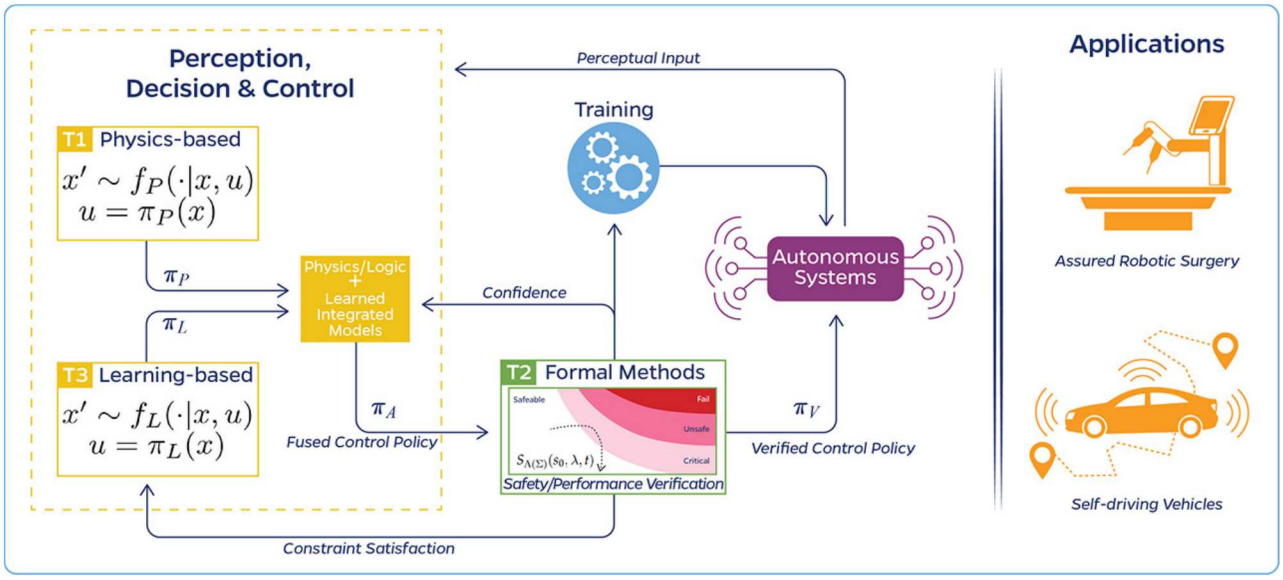

Fig. 1. The Verified Assured Learning for aUtonomous Embedded Systems (VALUES) System Architecture

monitoring and guarding during safe operations, [3]. Outside of the fields of robotics and control, the idea of having a trusted code base is relevant to this approach, [4]. There has been much additional work using reachability models as safeguards. We do not summarize all such work, but some examples that are relevant to our particular approach include [5], [6] and [7]. In our past work, we applied formally verified collision avoidance safety predicates to aircraft systems, [8]–[10], and created an approach for correct-by-construction fallback control using those safety predicates, [11]. We now apply this approach, and the verified collision avoidance theorems to collision-free ground vehicle navigation. Our methods are applicable to the avoidance of pedestrians and other moving obstacles, such as other vehicles, and employ non-deterministic uncertainty models to formulate envelopes that contain the uncertain trajectories of both the ego car and the mobile intruders.

II. Background

Our work on collision avoidance began with the stress testing and verification of Airborne Collision Avoidance System X, the Federal Aviation Administration's (FAA) next-generation collision avoidance advisory system for aircraft [12], [13]. We developed an approach for verifying safety of ACAS X collision resolution advisories based on the geometric analysis of reachable envelopes [8]. By separately treating the horizontal and vertical dynamics of the aircraft, we reduce the problem of collision avoidance to the avoidance of vertical proximity during the time period in which the two aircraft are within horizontal proximity. Control is modeled in the vertical dimension only, and the horizontal components of the trajectory simply determines the timing of horizontal proximity. The aircraft collision avoidance maneuvers are specified in terms of target vertical rates and compliance acceleration ranges. Our approach to formulating and verifying theorems for safety used the fact that the corresponding reachable envelopes are piece-wise quadratic curves [10]. As a result, we obtained closed form safety predicates that determine whether two such envelopes overlap, indicating in a violation of separation constraints. In the case of ACAS X, this allowed us to efficiently test the over 4 million two-aircraft encounter geometries and the corresponding collision resolution advisories stored in the policy score table, which served as the basis for optimal online collision resolution advisory computation [14].

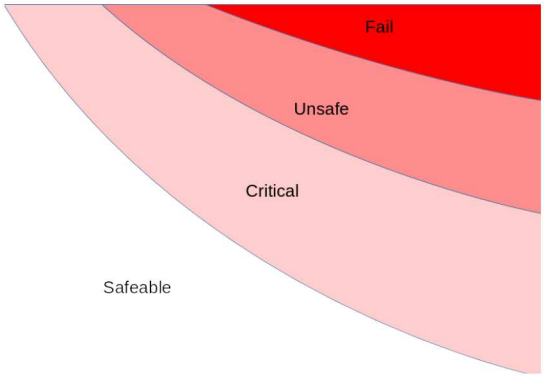

Fig. 2. Illustration of the state space separated into safeable, critical, unsafe, and failing subsets.

We developed a key property, termed *safeability*, that evaluates whether, at the next decision time, the system could be made safe through the use of an available maneuver. This property allowed us to determine precisely when the current policy must be intervened upon to maintain safety. As illustrated in figure 2, the system may move from a safeable state, in which no restriction is needed on its policy, to a critical state, in which only a subset of available maneuvers can maintain future safety. When the system is in a critical state, we restrict the control to the set of safe maneuvers, so that we can avoid entering unsafe states. We demonstrated the

use of safeability in the construction of fallback control in [11]. The result was a formalization of a simplex architecture that could be used to derive provable safety guarantees (under the assumption of correctness for reachable envelope analysis).

We realized then that the geometric analysis developed for ACAS X verification could also be used as the core of a collision avoidance fallback controller for ground vehicles. Many navigation path planning algorithms attempt to optimize their actions by balancing multiple competing demands. For example, this could involve avoiding collisions while minimizing deviation from the planned shortest path. As a consequence, a heuristic system may achieve overall performance that comes at the expense of undesirable behavior in some limited set of circumstances. Moreover, with the introduction of machine learning and other autonomous learning methods it becomes difficult to precisely characterize the set of states in which the primary collision avoidance controller may underperform, making it impossible to address problematic sets of states algorithmically. It is therefore desirable to have a trusted fallback controller that provides the final layer of safety. The fallback controller can be constructed in an explainable and verifiable way because its only goal is maintaining safety. The simplex architecture with a primary and fallback controller [2] allows the system to maintain its multi-objective performance optimizations in the vast majority of circumstances while at the same time guaranteeing safety in *all* circumstances.

Here we demonstrate an application of the simplex control approach by providing an implementation of the safety fallback controller for ground vehicle collision avoidance. This work assumes that our fallback controller will not use turning motions as avoidance maneuvers, and rather controls the velocity down the road, braking to avoid any intruders. Due to the fact that the primary controller engages in turning maneuvers, this limitation could impact the accuracy of the fallback controller predictions. In [15], we show how to perform a similar reachability analysis of turning maneuvers using a provably sound approximation but this computation is considerably more complex to implement. In this work, we show that even with restricted fallback maneuvers, our implementation provides a significant safety improvement in some stressing situations.

III. Approach

In this section we describe the unmanned ground vehicle (UGV) system design and the implementation of the safe fallback controller.

A. UGV architecture

The autonomous ground vehicle used for experimentation and testing was the Johns Hopkins University (JHU) all-terrain agile ground vehicle, a $1/5$-scale ground vehicle model. This vehicle used Redcat Racing Rampage XB-E as base, which was heavily modified with additional hardware (Figure 3). The drive and steering servo motor controllers were upgraded to enable control and sensing of rotational position and velocity. An ATmega2560 board was added as an interface between radio receiver and motor controllers. The vehicle was also outfitted with a computer subsystem for onboard perception processing and motion planning.

Fig. 3. JHU all-terrain agile ground vehicle

This subsystem comprised an Intel NUC CPU, Nvidia Jetson AGX Xavier, an ethernet switch and a wifi router. For sensing and perception (in addition to wheel odometry feedback), the vehicle was equipped with an Intel Realsense D455 RGB-D camera, a LORD Microstrain 3DM-GX4-25 IMU and two u-blox ZED-F9P RTK-GPS units. The latter provided readings of absolute heading as well as global position.

The vehicle state estimation and motion planning was implemented in the Robot Operating System (ROS), [16]. The ROS architecture is illustrated in Figure 4. Vehicle state is computed by fusing wheel odometry, inertial measurement unit (IMU), and global positioning system (GPS) position and heading using the ROS extended Kalman filter implementation *robot_localization* [17], [18]. This test vehicle was designed to operate on pedestrians sidewalks of the JHU campus. For this reason, the perception module is focused on identifying pedestrians. Pedestrian detection was performed by passing the RGB-D camera feed to Yolov3 neural network classifier, which itself uses Darknet-53 neural network for feature extraction. Yolov3 was selected among other algorithms for its ability to produce accurate bounding boxes while maintaining real time performance. A video showing an example of the operation of the UGV is available at https://drive.google.com/file/d/1AXMqpmbTT-4L2bGDTxHjbXO09lUqwqIx/view.

The overall software architecture (Figure 5) was organized to maximally facilitate software/hardware in the loop testing. This allowed us to test the full ROS software stack in the CARLA virtual simulation environment, [19], as well as replay ROS bags recorded during vehicle testing for debugging and analysis. By testing in the virtual environment we were able to eliminate bugs and fine tune performance so that physical vehicle tests could be focused on addressing issues specific

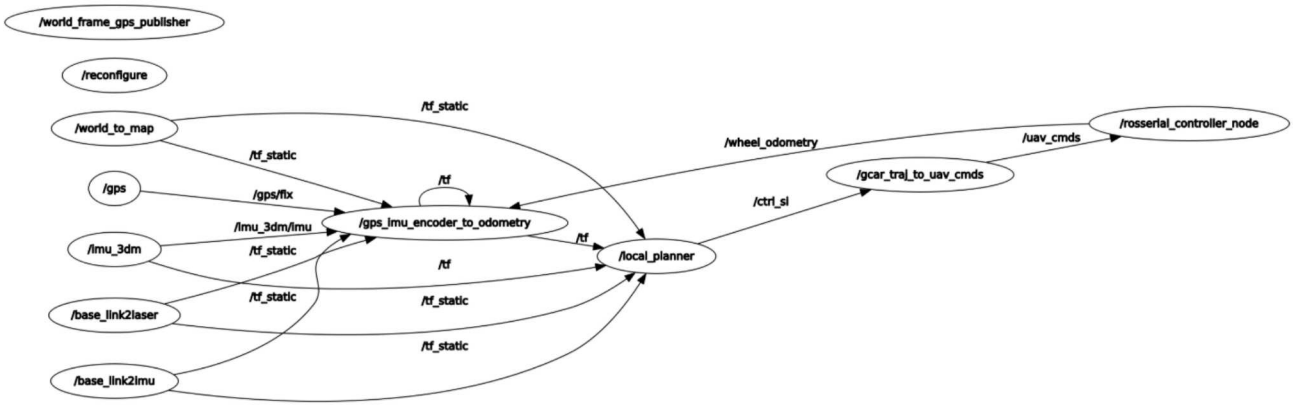

Fig. 4. ROS architecture

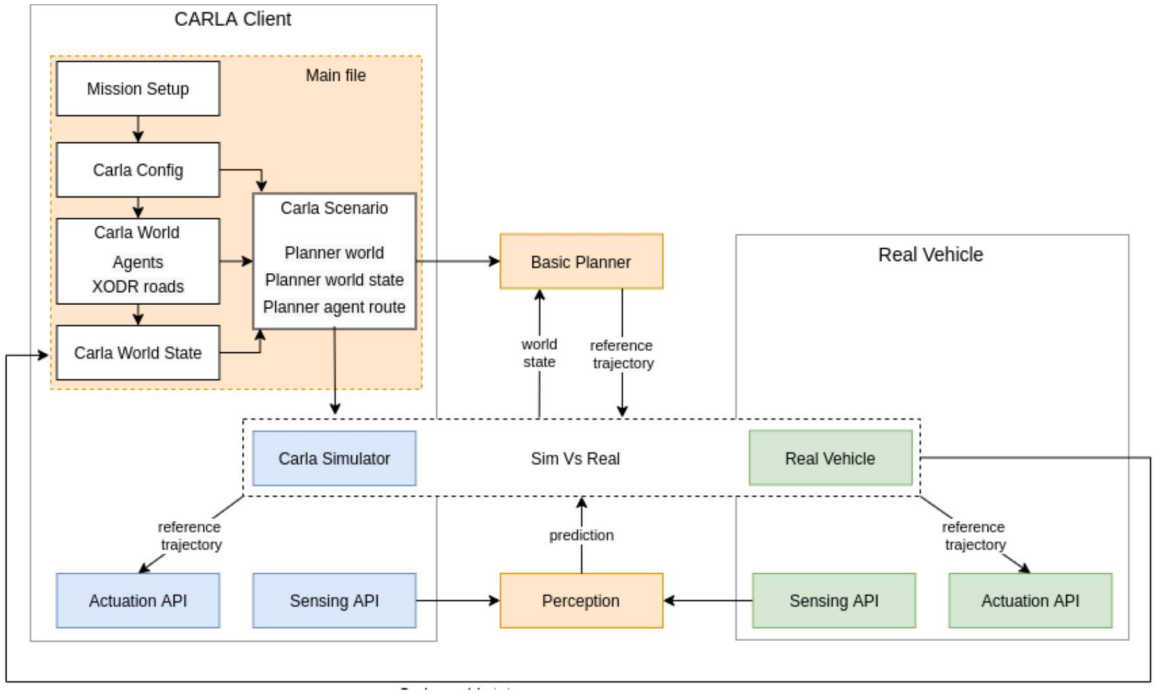

Fig. 5. Simulation Versus Real Test Framework Software Architecture

to interactions with the physical environment such as noise, camera calibration, etc.

The design of our UGV primary control policy used the methods of "Robust Policy Search for an Agile Ground Vehicle Under Perception Uncertainty", also called PROPS, [20]. However, in the testing of a safety fallback controller, we substituted the PROPS algorithm with a greedy policy roll-out method. The greedy method is less conservative in accounting for uncertainty and allowed us to better demonstrate the value of the safety guard.

B. Safe Defensive Driving Fallback Controller

The objective of the Safe Defensive Driving (SDD) fallback controller is to improve the overall safety of the UGV. While the primary controller does take into account pedestrian proximity, it weighs this metric against several other performance metrics, which sometimes results in undesirable tradeoffs between planned trajectory smoothness and safety. By having a separate fallback controller focused entirely on pedestrian collision avoidance, we can substantially reduce the likelihood of such undesirable trajectories.

Furthermore, the primary controller considers possible future pedestrian states resulting only from propagation of the Gaussian uncertainty of the pedestrian detection. So, if the pedestrian speeds up or slows down significantly the primary controller is likely to miscalculate the future cost of a collision. In contrast, the SDD takes into account a fairly broad range of possible pedestrian dynamics, making the system much more robust to unexpected pedestrian actions.

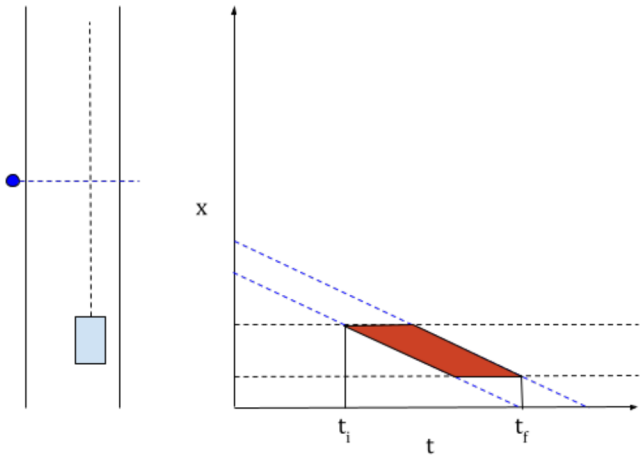

Fig. 6. Lateral conflict interval computation

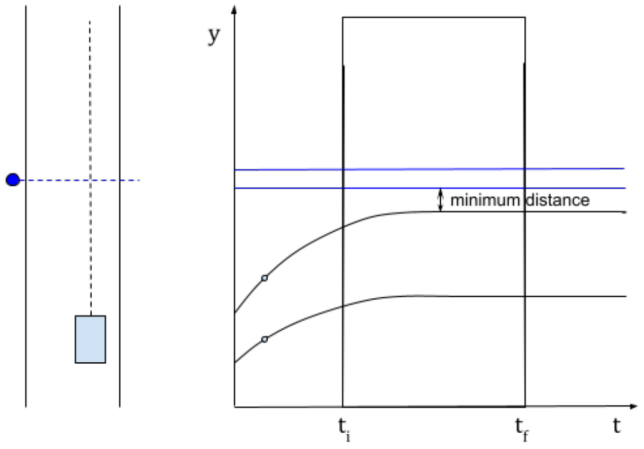

Fig. 7. Down-the-road conflict computation

For the down-the-road direction, we compute the reachable envelopes for each pedestrian and the vehicle, taking into account dynamic behavior by including all trajectories satisfying selected acceleration bounds (Figure 7). Note, that the vehicle's reachable envelope includes a segment – between the y-axis and the blue dots – representing the reachable positions of the vehicle as it continues to move under the current primary controller command for one controller iteration (the duration on the t-axis is exaggerated). This segment is what allows safety to be guaranteed for all future time provided the vehicle starts in a safe state and the acceleration ranges, that determine the shapes of reachable envelopes, encompass all possible vehicle and pedestrian accelerations.

The results of the lateral and down-the-road computations are combined to obtain the estimated distance of closest down-the-road approach within the lateral conflict time interval $[t_i, t_f]$ (Figure 7). If this distance is positive, it is safe for the vehicle to continue to move at its current speed until the next fallback controller invocation. If, on the other hand, the computed distance of closest approach is zero or negative, then the vehicle should still try to avoid collision by braking immediately [11].

The SDD fallback controller was integrated into the motion planner by attaching it as a filter on the primary controller output. Specifically, the fallback controller was implemented as a class that had an interface identical to the primary controller, so that the two were indistinguishable from the point of view of the motion planner. When active, the fallback controller called the primary controller to determine what velocity goal the primary controller planned to set. It then performed the previously described safety computation to determine whether it was safe to proceed. If the safety computation returned a positive result the original primary controller actions was returned to the planner, otherwise, a braking command was issued.

This approach had the advantage that the primary controller could be developed and run without any special modifications. In particular this allowed us to quickly and easily compare vehicle safety performance with and without the SDD fallback controller. It also allowed for very easy parallel and independent development on both the primary and the fall back controller.

A special emergency brake case was added to handle the situations when the pedestrian popped up in the vehicle's field of view. In our tests this occurred almost exclusively when the pedestrian overtook the vehicle from behind but could also occur if a pedestrian steps out from behind an occlusion or due to perception failure.

IV. Results

In order to stress test the primary and SDD fallback controllers, we used a custom simulation environment based on simple Newtonian dynamics that provides planning for vehicles and pedestrians. This provides lower fidelity than other environments, as it does not possess the ability to model metrics such as wheel friction or rigid body dynamics that sophisticated simulation engines provide. However, it remains representative of the system under ideal road and perception conditions, allowing us to interrogate the policy of the controllers while removing perception artifacts. This simple environment also afforded more efficient computations for simulation, allowing us to test thousands of scenarios.

Additionally, to validate our stress testing results in a more realistic setting, we performed targeted testing in a custom CARLA environment. CARLA is a simulator for autonomous driving research that is implemented over the Unreal Engine 4; this provides state-of-the-art rendering quality, realistic physics, basic NPC logic, and an ecosystem of interoperable plugins [19].

We stressed the system by creating *pop-up* pedestrian encounters. While the set-up is non-physical, it successfully tests cases where a pedestrian's position is mis-estimated or an undetected pedestrian enters the field of view at very close range. Such sensing failures are possible in the actual system, and so our test investigates the navigation system's ability to respond to these stressing cases. These simulated encounters involved a vehicle traveling along a two lane road with a

pedestrian initially located at the center line 40 meters ahead of the vehicle as seen in Figure 8. The vehicle will move down the road attempting to reach a goal velocity, 10 m/s, and avoid collisions. If there are no obstacles, the vehicle will just move in a straight line down the center of the lane. Additionally, the pedestrian moves towards a goal position and velocity. For the purpose of our simulations, these values are tested over large ranges to encompass a variety of scenarios.

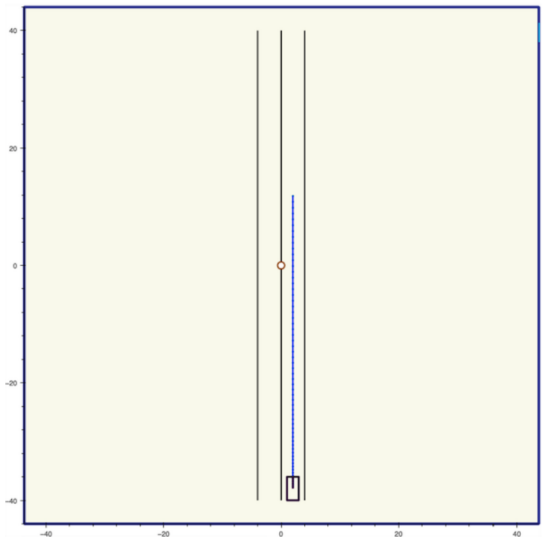

Fig. 8. Initial state of simulation with the pedestrian represented as a circle and the vehicle represented as a rectangle.

During the simulation, we changed the pedestrian's current position to be in front of the vehicle at some time, effectively teleporting the pedestrian. In addition, we added random positional offsets from a uniform distribution over [0, 1) to the pedestrian's trajectory to mimic sensor noise. The teleportation interrupts the pedestrian's trajectory abruptly and places the pedestrian's position in front of the vehicle with a sampled vertical and horizontal offset. These offsets were calculated to ensure the pedestrian was not teleported to a position that would result in an unavoidable collision. We teleported the pedestrian in this manner for a range of offsets, goal positions and velocities. The pedestrian was only teleported at one time stamp for each scenario, but our scenarios tested all possible time stamps for every range of values. The controllers were given ground truth position and velocity values for the pedestrian, so there were no filtering delays in estimating the state of the new intruder. For each encounter, we tested the primary controller and the SDD fallback controller.

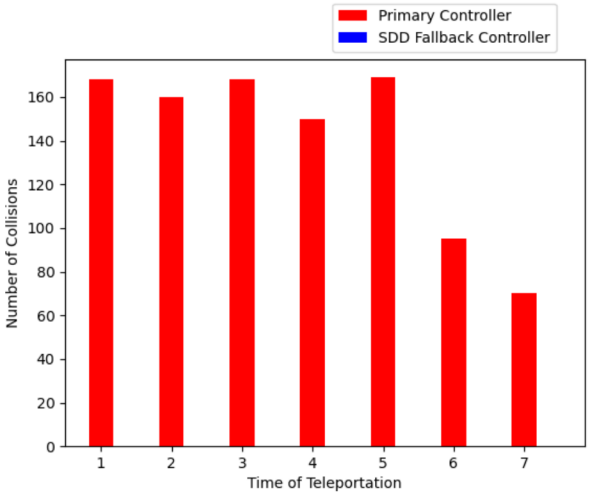

Fig. 9. Safety performance comparing the primary controller and safety defensive driving controller under stress from a set of pop-up encounters over times of encounter. A total of 980 collisions were observed by the primary controller. No collisions were observed using the SDD fallback controller.

We found that the SDD fallback controller was 100% effective at improving the safety of these pop-up encounters. Figure 9 shows the number of collisions the primary controller encountered during our 24,500 test scenarios over each time of teleportation. We see a decrease in the number of collisions (without SDD) as time of teleportation increases. This is because as the pedestrian is teleported later in the simulation, it is possible that the vehicle will never reach that pedestrian. The collisions encountered at these higher time stamps are attributed to the positional noise in the pedestrian's trajectory rather than the teleportation. There are fewer opportunities to stress the controller at these higher time stamps. The fallback controller anticipates a broader range of pedestrian trajectories, meaning it will anticipate positional noise and can start slowing down even before the pedestrian is teleported in front of the vehicle, making it easier for the vehicle to avoid the pedestrian.

Accordingly, this phenomenon can be seen through scenarios in which the primary controller caused a collision. During these scenarios, the vehicle had an average velocity of 8.842 m/s when the primary controller was activated and an average velocity of 6.742 m/s when the fallback controller was activated. We note that the primary controller makes almost no attempt to slow down for these encounters, as the average velocity of the vehicle under the primary controller during scenarios in which there were *no* collision was 8.847 m/s.

Taking a closer look at an example collision from the primary controller, Figure 10, on the left, displays a scenario where the primary controller does not sufficiently react to a pedestrian that is in front of the vehicle. We see the vehicle hitting the pedestrian without slowing down, as seen with the constant red in the trajectory. The vehicle tries to swerve at the last minute, but does not manage to miss the pedestrian. The vehicle remains on its original trajectory even though the pedestrian is still in the vicinity. In comparison, the right side of Figure 10 shows the reaction of the SDD fallback controller.

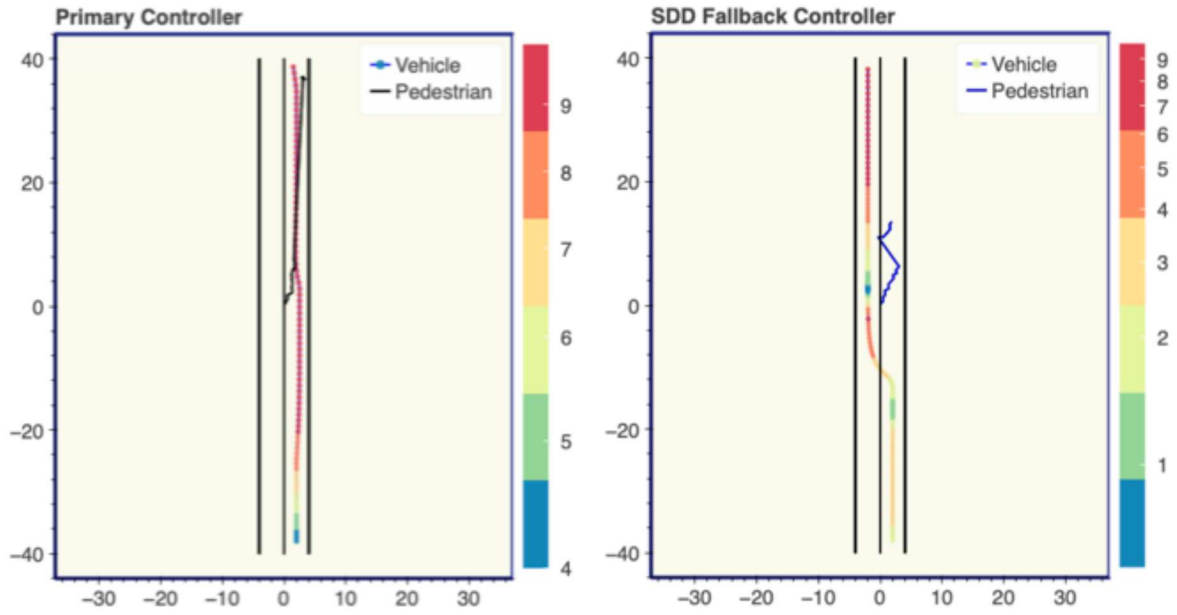

Fig. 10. Example primary controller (left) and SDD fallback controller (right) vehicle and pedestrian trajectories that do and do not, respectively, result in a collision. The vehicle's trajectory is colored according to its velocity with red denoting the highest velocity and blue denoting the lowest as shown by the colorbar to the right of each figure.

The vehicle slows to a complete stop multiple times denoted by the blue in the trajectory, as it anticipates how the pedestrian might move. The vehicle remains at a reduced speed, as seen by the green and yellow, until the pedestrian is at a safe distance away from the vehicle. After the vehicle passes the pedestrian, it begins gaining speed, as seen by the eventual return to red. At the beginning of this simulation the vehicle begins to slow well before the pop-up occurs, allowing ample time for the vehicle to stop and avoid the pedestrian.

The behavior of our fallback controller is rather conservative, as described in the scenario above. During the 24,500 stress tests, the fallback controller triggered the *STOP* action 3,151 times. Of these times, the fallback controller triggered the *STOP* action 2,638 times when the greedy controller did *not* have a collision nor triggered a *STOP* action. This corresponds to the false positive rate of 83.7% for the fallback controller.

As mentioned above, we validated our stress testing results by performing targeted testing in CARLA. We reproduced the random positional offsets and teleportation of the pedestrian's trajectory, and tested over a similar range of time stamps to ensure our stress testing setup was represented appropriately in the CARLA simulation. We found that in this higher fidelity testing environment, the SDD fallback controller remained 100% effective at improving the safety of pop-up encounters. For the 60 CARLA test scenarios, we saw 5 collisions while the primary controller was activated and 0 collisions under the control of the SDD fallback controller. While this is about double the collision rate we saw in our custom simulation environment above, this is expected due to the targeted nature of this test setup.

V. Conclusions

We developed a safe defensive driving fallback controller to act as a fallback controller, providing run-time monitoring of a primary autonomous vehicle system. To develop the safety logic, we took advantage of collision avoidance logic that had been formally verified for aircraft applications. We integrated the safety fallback on a 1/5-sized test vehicle and demonstrated the system in experimental tests. We further stress tested the safety fallback by pairing it with a simple primary controller, showing that in simulations with pedestrians that pop up in front of the vehicle the safety system greatly reduced the collision rates. We confirmed these test results in higher fidelity CARLA simulation tests. The results are encouraging and demonstrate the value of implementing independent safety monitoring as a means of enabling the safe adoption of complex autonomy.

Acknowledgments

The authors would like to thank the Johns Hopkins University Institute for Assured Autonomy for funding this project and for their guidance and support. We are, in particular, grateful for the advice and support of Dr.'s Cara LaPointe, David Silberberg, Anton Dahbura, Jim Bellingham, and Ken Schmidt. The authors also thank Prof. Gregory Hager, Dr. Molly O'Brien, Dr. Joshua Brulé, and Dr. Reed Young for their useful discussions and insights during this project.

References

[1] D. Scheidt, R. Lutz, W. D'Amico, D. Kleissas, R. Chalmers, and R. Bamberger, "Safe testing of autonomous system performance," in *Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)*, 2015.

[2] D. Seto, B. Krogh, L. Sha, and A. Chutinan, "The simplex architecture for safe online control system upgrades," in *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No. 98CH36207)*, vol. 6. IEEE, 1998, pp. 3504–3508.

[3] P. Koopman and M. Wagner, "Toward a framework for highly automated vehicle safety validation," *SAE Technical Paper, Tech. Rep*, 2018.

[4] E. Kang and D. Jackson, "Patterns for building dependable systems with trusted bases," in *Proceedings of the 17th Conference on Pattern Languages of Programs*, ser. PLOP '10. New York, NY, USA: Association for Computing Machinery, 2010. [Online]. Available: https://doi.org/10.1145/2493288.2493307

[5] N. Aréchiga and B. Krogh, "Using verified control envelopes for safe controller design," in *American Control Conference*, June 2014, pp. 2918–2923.

[6] Y. S. Shao, C. Chen, S. Kousik, and R. Vasudevan, "Reachability-based trajectory safeguard (rts): A safe and fast reinforcement learning safety layer for continuous control," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3663–3670, 2021.

[7] S. Kaynama, J. Maidens, M. Oishi, I. M. Mitchell, and G. A. Dumont, "Computing the viability kernel using maximal reachable sets," in *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 55–64. [Online]. Available: https://doi.org/10.1145/2185632.2185644

[8] J.-B. Jeannin, K. Ghorbal, Y. Kouskoulas, R. Gardner, A. Schmidt, E. Zawadzki, and A. Platzer, "A formally verified hybrid system for the next-generation airborne collision avoidance system," in *TACAS*, ser. LNCS, C. Baier and C. Tinelli, Eds., vol. 9035. Springer, 2015, pp. 21–36.

[9] J.-B. Jeannin, K. Ghorbal, Y. Kouskoulas, A. Schmidt, R. Gardner, S. Mitsch, and A. Platzer, "A formally verified hybrid system for safe advisories in the next-generation airborne collision avoidance system," *International Journal on Software Tools for Technology Transfer*, pp. 1–25, 2016.

[10] Y. Kouskoulas, D. Genin, A. Schmidt, and J.-B. Jeannin, "Formally verified safe vertical maneuvers for non-deterministic, accelerating aircraft dynamics," in *ITP*, ser. LNCS, M. Ayala-Rincón and C. A. Muñoz, Eds.,

[20] S. Sefati, S. Mishra, M. Sheckells, K. D. Katyal, J. Bai, G. D. Hager, and M. Kobilarov, "Robust policy search for an agile ground vehicle under vol. 10499. Springer, 2017, pp. 336–353.

[11] Y. Kouskoulas, A. Schmidt, J.-B. Jeannin, D. Genin, and J. Lopez, "Provably safe controller synthesis using safety proofs as building blocks," in *7th International Conference in Software Engineering Research and Innovation (CONISOFT)*, 2019, pp. 26–35.

[12] M. J. Kochenderfer and J. P. Chryssanthacopoulos, "Robust airborne collision avoidance through dynamic programming," MIT Lincoln Laboratory, Tech. Rep. ATC-371, January 2010.

[13] R. W. Gardner, D. Genin, R. McDowell, C. Rouff, A. Saksena, and A. Schmidt, "Probabilistic model checking of the next-generation airborne collision avoidance system," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, 2016, pp. 1–10.

[14] J. Jeannin, K. Ghorbal, Y. Kouskoulas, A. Schmidt, R. Gardner, S. Mitsch, and A. Platzer, "A formally verified hybrid system for safe advisories in the next-generation airborne collision avoidance system," *International Journal on Software Tools for Technology Transfer (STTT)*, vol. 9035, pp. 21–36, 2017.

[15] Y. Kouskoulas, T. J. Machado, and D. Genin, "Formally verified timing computation for non-deterministic horizontal turns during aircraft collision avoidance maneuvers," in *Formal Methods for Industrial Critical Systems: 25th International Conference, FMICS 2020, Vienna, Austria, September 2–3, 2020*. Berlin, Heidelberg: Springer-Verlag, 2020, pp. 113—129. [Online]. Available: https://doi.org/10.1007/978-3-030-58298-2_4

[16] Stanford Artificial Intelligence Laboratory et al., "Robot operating system." [Online]. Available: https://www.ros.org

[17] ——, "ROS state estimation nodes." [Online]. Available: http://docs.ros.org/en/kinetic/api/robot_localization/html/state_estimation_nodes.html

[18] T. Moore and D. Stouch, "A generalized Extended Kalman Filter implementation for the Robot Operating System," in *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, July 2014.

[19] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.
perception uncertainty," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 154–161.