2019 IEEE Intelligent Transportation Systems Conference (ITSC)
Auckland, NZ, October 27-30, 2019

Rapidly-exploring Random Trees for Testing Automated Vehicles

Cumhur Erkan Tuncali and Georgios Fainekos

Abstract— **One of the expectations from fully or partially automated vehicles is to never cause an accident and actively avoid dangerous situations. However, an automated vehicle may not be able to avoid all collisions, e.g., collisions caused by other vehicles. Hence, it is important for the system developers to understand the boundary case scenarios where an autonomous vehicle can no longer avoid a collision. In this paper, an automated test generation approach that utilizes Rapidly-exploring Random Trees is presented to explore these boundary scenarios. An important advantage of the approach is the openness of the test scenarios: one can set the road geometry and the number of adversarial objects and let the system search for interesting trajectories and environment parameters. A cost function is proposed which guides the test generation toward almost-avoidable collisions or near-misses.**

I. Introduction

Autonomous vehicles are safety-critical systems, and they should be tested thoroughly before they are deployed on public roads. Although testing in real traffic environments is always going to be a necessary step in any development process, simulation-based testing provides many advantages such as fully controllable environments, ground-truth information, ability to try a massive number of scenarios, and creating risky scenarios without risking human life or the vehicle under test. In addition, even though real-life testing evaluates system performance in average or usual driving conditions, simulations are necessary to discover and evaluate system performance in risky low probability scenarios. In fact, industry leaders, such as Waymo [1], recognize simulation-based testing as an important component in establishing automated vehicle safety.

Optimization-guided falsification techniques [2]–[5] utilize optimization engines to generate challenging scenarios for an Automated Driving System (ADS) under test with the ultimate goal of finding a scenario in which the Vehicle Under Test (VUT), also referred to as *Ego vehicle*, fails to satisfy its safety requirements (for an extensive comparison of the different methods see [6]). One fundamental challenges of optimization-based methods is that if the test scenario is not properly constrained, or the safety requirements are not in assume-guarantee form, then it is very easy to discover low-cost bad behaviors which are not interesting. For instance, in an open (not overly constrained) test environment, a collision

This work has been partially supported by awards NSF CNS 1350420, NSF IIP-1361926 and the NSF I/UCRC Center for Embedded Systems.

C.E. Tuncali, and G. Fainekos are with School of Computing, Informatics and Decision Systems, Arizona State University, Tempe, AZ 85281, USA (e-mail:{etuncali, fainekos}@asu.edu)

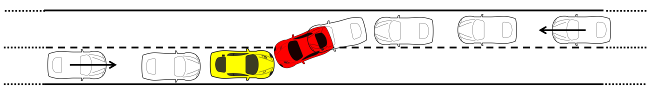

Fig. 1. An unavoidable collision example. Red car suddenly moves into the Ego vehicle's lane. In an open test environment, it is trivial for an optimization based tester to produce such uninteresting scenarios.

with a vehicle that drives into the VUT could be such a low-cost worst-case scenario (see Fig. 1). However, such a scenario is potentially uninteresting since typically such collisions are unavoidable. In order to improve the safety-related capabilities of the system, it is more interesting to identify the boundaries between barely avoided collisions and collisions that could have been avoided with minor changes in the control or perception. A distinguishing characteristic of our previous falsification work [2] from the related literature is that we have used an optimization-guided falsification tool, S-TaLiRo [7], to search for such boundary cases.

In this paper, we propose an alternative framework to the falsification-based approach in [2]. We consider the test generation problem as a motion planning problem for a number of powerful adversarial agents and we utilize Rapidly-exploring Random Trees (RRTs) [8] in order to discover interesting boundary-case scenarios. To guide the search toward that boundary, we propose a new cost function that captures time-to-collision as well as a measure of the collision severity. In order to efficiently explore the search space under the new cost function, we combine ideas from Transition-based RRT (T-RRT) [9] with ideas from RRT* [10]. The RRT code and case studies are distributed with the S-TaLiRo toolbox [11].

The major advantage of the RRT based testing framework over the optimization-based approach in [2] is that now the test scenarios can be as open as possible. In other words, we do not need anymore a finite number of variables to parameterize the search space for the adversarial vehicle trajectories. Rather, the system tester can place a number of adversarial agents in an existing test scenario and discover critical behaviors not originally conceived by the engineers.

A. Background and Related Work

RRTs were first developed for robot path/motion planning problems [8]–[10]. However, thanks to their ability to efficiently search over high-dimensional spaces, RRT-based approaches also deliver promising results in the test generation domain [12]–[16].

Since their first introduction, many variants of RRTs have been proposed. In [9], a method called Transition-based RRT (T-RRT) was introduced. Transition-based RRT method extends the classical RRT by incorporating additional cost criteria to the explored paths rather than only aiming to reach a target configuration. T-RRT borrows the notion of transition tests from stochastic optimization approaches. Hence, it can be considered as a method that is merging RRTs with stochastic optimization. Furthermore, T-RRT controls exploration versus refinement using a method called *minimal expansion control* which helps to promote the expansion of a tree to the unexplored areas of the search space. In another line of work, a provably asymptotically optimal RRT approach, RRT*, was proposed and studied in [10].

II. RRT FOR ADVERSARIAL TESTING

A. Problem Formulation and Solution Overview

In this paper, informally, we address the following problem: Given a set of n adversarial actors with dynamics $\dot{y}_i = f_i^A(y_i, u_i)$ with $y_i(0) \in Y_0$, a set of m automated vehicles under test (Ego vehicles) with dynamics $\dot{x}_i = f_i^E(x_i, w_i)$ with $x_i(0) \in X_0$ and sensing capabilities $w_i = g_i^E(x, y)$, a set of constraints C^E on $x = [x_1^T \dots x_m^T]^T$, a set of constraints C_y^A on $y = [y_1^T \dots y_n^T]^T$, and a set of input constraints C_u^A on $u = [u_1^T \dots u_n^T]^T$, compute initial states $y(0), x(0)$ and adversarial inputs $u(t)$ that lead to "almost-avoidable" collisions or "near" misses.

Here, the vectors y_a for each adversary a and x_e for each Ego vehicle e represent the state variables of the road participants with dimensionality capturing the desired model fidelity, e.g., a 2D kinematic model vs a 4D dynamic model, etc. We highlight that $f_a^A(y_a, u_a)$ implies that the adversarial road participant a does not sense the environment and that we directly control its actions through the input u_a. On the other hand, each Ego vehicle perceives the environment and, hence, it needs to receive information y and x about all the other road participants through a sensing system, i.e., g_i^E.

The constraints C^E and C_y^A capture the free configuration space of each vehicle. Namely, the free workspace defined by the environment modified by the vehicles' geometries and states (see [17] Ch. 4.1). We remark that the free workspace for the adversaries does not necessarily coincide with the road boundaries – it could be a subset, a superset, or any other desired geometry. Even though in most cases, the free configuration space cannot be explicitly represented but only sampled, in this paper, for brevity in the presentation, we will not make this distinction. In addition, C_y^A could be split into soft constraints (e.g., used only for path sampling) and hard constraints (e.g., used for collision detection). The constraints C_u^A simply capture the limitations on actuation, e.g., maximum possible acceleration or braking.

To solve this problem, we utilize notions from RRT* [10] and T-RRT [9] with a custom cost function that we propose. We implement our version of *minimal expansion control* using the notion of *sparseness* from evolutionary algorithms that perform novelty search [18], [19]. In the following, due to space limitations, we assume that the reader has a general understanding of planning using RRT methods [10], [20].

Since we have to solve kinodynamic planning problems [20] for the adversarial agents, we assume that a simulation function sim is available for the overall system:

$$\dot{z} = \begin{bmatrix} \dot{y} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} F^A(y,u) \\ F^E(x, G^E(x,y)) \end{bmatrix}, \; x \in C^E, y \in C_y^A, u \in C_u^A$$

where $z = [y^T \, x^T]^T$ for brevity, and F^A, F^E and G^E are vector functions with components f_i^A, f_i^E and g_i^E, respectively. In other words, given a time interval $[t_1, t_2]$, a vector input $u(t)$ for $t \in [t_1, t_2]$, and an initial state (configuration) $\tilde{z}$ at time t_1, then the state (configuration) at time t_2 is $\hat{z} = sim(\tilde{z}, [t_1, t_2], u)$ subject to C^E, C_y^A and C_u^A. Finally, we represent a simulation (output) trajectory for a sequence of requested time points $t_0 \; t_1 \dots t_f$ by $\mathbf{z}$ (where t_f is the time at a leaf of a search tree).

The flowchart of our RRT-based approach is shown in Fig. 2. In the rest of this section, we will describe the key components of our approach and we will formally define the notion of "almost-avoidable" collisions and "near" misses.

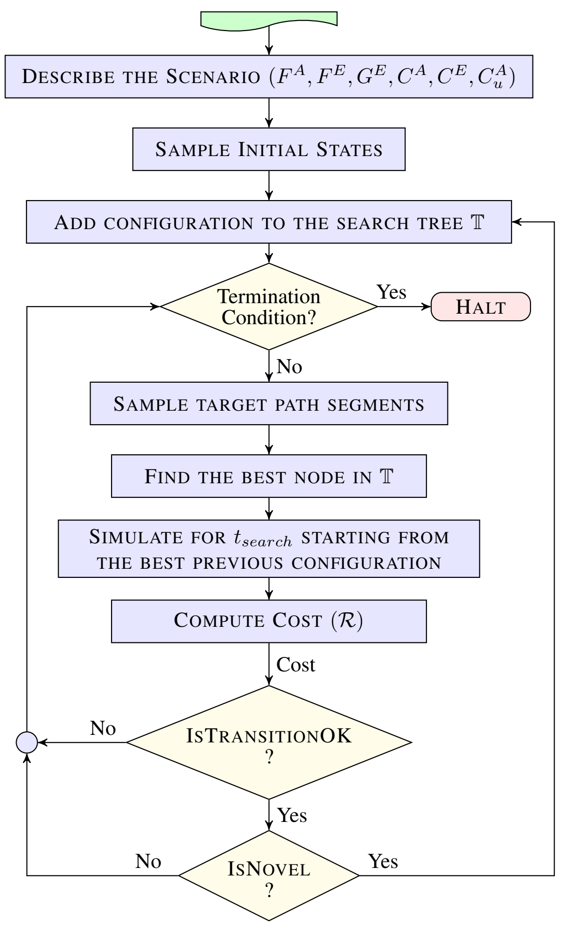
DESCRIBE THE SCENARIO ($F^A, F^E, G^E, C^A, C^E, C_u^A$)
SAMPLE INITIAL STATES
ADD CONFIGURATION TO THE SEARCH TREE $\mathbb{T}$
Termination Condition?
Yes
HALT
No
SAMPLE TARGET PATH SEGMENTS
FIND THE BEST NODE IN $\mathbb{T}$
SIMULATE FOR t_{search} STARTING FROM THE BEST PREVIOUS CONFIGURATION
COMPUTE COST ($\mathcal{R}$)
Cost
No
ISTRANSITIONOK ?
Yes
No
ISNOVEL ?
Yes

Fig. 2. Flowchart illustrating the RRT-based approach.

B. Initializing the Search

After the general road structure for the driving scenario is described in C^E and C_y^A, the sampling spaces for the initial states X_0 and Y_0 are used to sample initial configurations of the simulation entities (ego vehicles and adversaries). The initial sampled states form the root node for the tree.

C. Information Stored on RRT Tree Nodes

A tree grows while seeking to discover interesting behaviors. While growing the tree, instead of executing simulation traces starting from the initial configuration, only a partial simulation is executed starting from an existing node in the tree. For that purpose, the state of the system, i.e., z (which includes the state of the controllers – if stateful), and the simulation time are stored on the tree nodes. If a simulation ends with a state such that $x \notin C^E$ or $y \notin C_y^A$, then the newly added tree node is marked as a "terminal node" which means that the tree can no longer grow from that node.

D. Sampling a Target Path Segment

A sample target path segment is simply a set of waypoints which is used as an immediate target for an adversarial vehicle. For an adversarial agent a, a waypoint is denoted as $\mathtt{w} = (\mathtt{w}_x, \mathtt{w}_y, \mathtt{w}_\theta, \mathtt{w}_v) \in W_a$, where $\mathtt{w}_x$, $\mathtt{w}_y$, $\mathtt{w}_\theta$, and $\mathtt{w}_v$ are the x-coordinate, y-coordinate, target driving direction and the target speed at the waypoint. The sampling space for the waypoints is defined by a corresponding parameter space W_a. The exact structure of W_a indirectly depends on f_a^A. For example, if the adversary f_a^A is a dynamic model, then $W_a = Proj_a(C_y^A)$, where $Proj_a(\cdot)$ projects the constraints for agent a. On the other hand, if f_a^A is a kinematic model, then $W_a = Proj_a(C_y^A) \times Proj_a(C_u^A)$.

An example waypoint sampled on a straight road (ignoring the vehicle geometry) is shown in Fig. 3. A coordinate transformation can be applied for sampling from curved roads. Although the example waypoint in Fig. 3 is sampled from a road, the sample space of the waypoint does not have to be the same as the area of a road in the simulation. It may be defined to go beyond the road limits, it may be limited to only a part of a road, or it may be completely irrelevant to a road in the simulation environment.

Once a waypoint is sampled, the next step in sampling a target path segment is to add an endpoint at a predefined distance d_{leg} from the waypoint, along the direction of the waypoint. Figure 3 shows a target path segment formed using this approach. If the endpoint of a target path segment is outside the sampling space of the waypoint, we simply break the segment at the boundary of the sampling space and add a second leg along the boundary in the direction closest to the waypoint direction. The sampled target path segment for this example can be denoted by $\mathtt{p} = ((\mathtt{w}_x, \mathtt{w}_y, \mathtt{w}_\theta, \mathtt{w}_v), (\mathtt{w}_{x2}, \mathtt{w}_{y2}, \mathtt{w}_{\theta 2}, \mathtt{w}_v), (\mathtt{w}_{x3}, \mathtt{w}_{y3}, \mathtt{w}_{\theta 2}, \mathtt{w}_v))$.

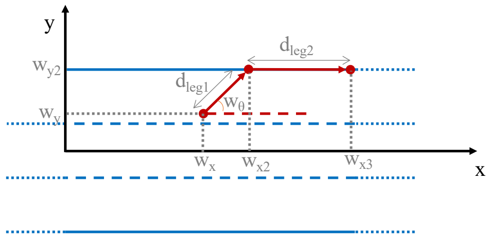

Fig. 3. Sampling a waypoint and a target path segment with space constraints on a three lane road.

This approach is applied to each agent vehicle for sampling their target path segments. Note that our framework allows using a different algorithm for selecting a target path and/or any other type of input for the agent vehicles.

E. Selecting the Best Node from the Search Tree

Once target path segments for the agent vehicles are sampled, we pick one node from the existing tree, as the initial configuration for the simulation that will be executed with the sampled target path segments. There is no single correct approach to decide which node of the tree would be the best choice.

In our study, the notion of selecting the optimal node is adopted from the RRT* algorithm [10]. In [10], when adding a new node to the tree, existing nodes in a neighborhood of the new node are all checked and the one which minimizes the cost is selected as the parent node. Our approach has similarities to the RRT* one. We compute the sum of distances from each vehicle to the starting point of their target path segment with the constraint that the configurations of all vehicles are behind the start position of the initial target waypoint with respect to the driving direction of that waypoint. Then, we execute partial simulations from the best n candidate previous nodes ($n = 5$ for our case studies) and pick the one which gives the minimum cost. We believe that this approach is promising to create relatively natural-looking vehicle trajectories while still allowing enough randomness in the maneuvers.

We would like to emphasize that the function used for selecting the best previous node is user-configurable in our framework and, depending on how much randomness is plausible in the generated driving paths, a different algorithm can be utilized, *e.g.*, simply selecting the closest node.

In [10], after adding the new node to the tree, there is a rewiring step which modifies existing connections to other nodes in the neighborhood of the newly added node. Application of the rewiring step is straightforward for path planning problems in Euclidean spaces where the tree nodes represent planned waypoints on a path instead of the vehicle configurations. The rewiring step can be computationally costly in our approach, and so, we do not apply the rewiring step and leave it as a future work for which the applicability should be analyzed.

F. Simulating the System

After obtaining a set of target path segments for agent vehicles and deciding the initial configuration for the simulation, we create the simulation scene in the simulation environment using the data stored in the selected node of the search tree. That is, we set the initial states of the simulation entities and initialize the Ego vehicle controllers with the previous inputs and the saved controller states. We also pass the sampled target path segments to the agent vehicles as inputs. Finally, we simulate the system for t_{search} time and collect state and input histories at each time step of the simulation. For our setup, we use MATLAB simulations, however, this is not mandatory and other simulators can be used as well. Note that if the simulator and the Ego vehicle controllers allow saving the state and continuing simulation from a saved state, *which is the case in our setup*, the time spent in the simulations can be radically reduced because it would be enough to simulate only the new part of the simulation. Otherwise, the simulation must start from the root node of the tree and run until the current target time.

G. Cost Function

After a simulation is executed, a cost function is used to compute how close the simulation trace is to an interesting behavior. The approach we describe here can be utilized to discover other types of interesting/failing behaviors; however, our target in this work is to explore the behaviors that are on the boundary between safe and unsafe operation. Hence, an interesting behavior for our purposes would be (i) a collision between an Ego vehicle and an agent that could have been avoided with a minor change in the control applied or agent trajectories, (ii) an almost-collision (near-collision) which could have resulted in a collision with a minor change in the control applied or agent trajectories.

The properties of a good cost function that would guide the search toward an interesting behavior for our purposes can be listed as follows:

- Among two similar collisions between an Ego vehicle and an agent vehicle, the one which has the smaller magnitude in the relative speed between the vehicles should have a smaller cost, as a smaller change in the speed of the Ego vehicle may have avoided the collision.
- Among two similar collisions, the one which has the smaller impact area, *i.e.*, the area of the collision surface, should have a smaller cost, as a smaller change in the steering maneuver of the Ego vehicle would be enough to avoid the collision.
- For vehicle trajectories without a collision, a smaller time-to-collision at any point of the trajectories, a smaller relative speed between vehicles and a smaller collision surface area for a possible future collision (assuming the vehicles continue driving without changing directions) should have a smaller cost.

Given a simulation trajectory $\mathbf{z}$ of the overall system, we propose the following cost function:

$$\mathcal{R}(\mathbf{z}) = (1 + s_{coll,\mathbf{z}})(v_{coll,\mathbf{z}}^2 + ttc_{min,\mathbf{z}}^2) \qquad (1)$$

where $s_{coll,\mathbf{z}} \in [0, 1]$ is the ratio of the collision surface to the overall surface on the collision side of the vehicle, $v_{coll,\mathbf{z}}$ is the relative speed of the vehicles at the moment of collision, and $ttc_{min,\mathbf{z}}$ is the mimimum time-to-collision encountered during the simulation output trace $\mathbf{z}$. For simulations with a collision, $ttc_{min,\mathbf{z}}$ is 0. For simulations without a collision, $s_{coll,\mathbf{z}}$ and $v_{coll,\mathbf{z}}$ are computed at the instance of smallest time-to-collision with the assumption that the vehicles continue their motion without changing their speeds and orientations. When the simulation output trace $\mathbf{z}$ contains collision(s) with Ego vehicle, we only consider the first collision of an Ego vehicle with any object for computing the cost using Eq. (1). Figure 4 shows the function with respect to the minimum time-to-collision and collision speed variables for a fixed collision surface. The effect of the collision surface to the cost is linear.

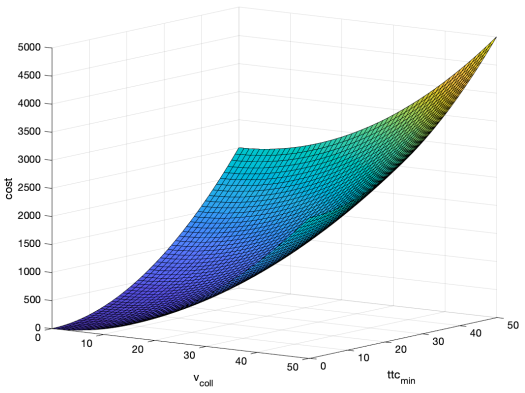

Fig. 4. Cost function to guide the search toward the boundary between collisions and near collisions.

H. Transition Check Function

The function we use for accepting a new configuration based on the cost is similar to the one proposed in [9], which we don't repeat here due to space considerations. Further details can be found in [21].

I. Novelty Function

To get better coverage of the state space and to avoid local minima, we reward novelty in our search. For an Ego vehicle e and an adversarial agent a, we define $\mathbf{x}_{\mathsf{e},\mathsf{a}}[k]$ as the vector of relative states and change in relative states at discrete simulation time k, *i.e.*, $\mathbf{x}_{\mathsf{e},\mathsf{a}}[k] =$

$$(\mathbf{x}_{\mathsf{e}}[k] - \mathbf{x}_{\mathsf{a}}[k], (\mathbf{x}_{\mathsf{e}}[k] - \mathbf{x}_{\mathsf{a}}[k]) - (\mathbf{x}_{\mathsf{e}}[k-1] - \mathbf{x}_{\mathsf{a}}[k-1])).$$

We compute the novelty of $\mathbf{x}_{\mathsf{e},\mathsf{a}}[k]$ as follows:

$$\mathcal{N} = \sum_{i=0}^{q} dist(\mathbf{x}_{\mathsf{e},\mathsf{a}}[k], \mu_i) \qquad (2)$$

where $\mu_i \in X_{rel,k-1}$ is the i^{th} nearest neighbor of $\mathbf{x}_{\mathsf{e},\mathsf{a}}[k]$ in the set $X_{rel,k-1}$ which contains $\mathbf{x}_{\mathsf{e},\mathsf{a}}$ vectors for all ego-agent pairs for all times before k. The function $dist$ computes the Mahalanobis distance between $\mathbf{x}_{\mathsf{e},\mathsf{a}}[k]$ and the elements of its q-nearest neighbors set. We choose to use the Mahalanobis distance because of its ability to provide a dissimilarity measure between two observations by utilizing the sample covariance matrix [22].

J. Termination Condition

Our algorithm checks a set of termination conditions to stop the search and returns the configuration which has the minimum cost associated with it. One of the termination conditions we use is a threshold for the minimum interesting cost. Another termination condition is a preset maximum overall time spent. Alternative termination conditions can be used, *e.g.*, a maximum number of nodes in the search tree.

III. Case Study

Here, we present a case study and compare our RRT-based approach with our falsification-based approach [2]. Further details and case studies can be found in [21].

In this case study, we have 4 agent vehicles and 1 Ego vehicle on a multiple-lane straight road. Figure 5 gives a high-level overview of our simulation setup. The initial lateral position of agent $\mathtt{a}_1$ is randomly sampled between $1.25\,\mathrm{m}$ and $6\,\mathrm{m}$, the initial lateral positions of agent vehicles $\mathtt{a}_2, \mathtt{a}_3, \mathtt{a}_4$ are randomly sampled between $-2.25\,\mathrm{m}$ and $-1.75\,\mathrm{m}$. The initial speed of $\mathtt{a}_1$ is randomly sampled between $5\,\mathrm{m/s}$ and $15\,\mathrm{m/s}$, and its target speed at each waypoint is sampled between $0\,\mathrm{m/s}$ and $30\,\mathrm{m/s}$. The initial and target speeds of all other vehicles are set to $15\,\mathrm{m/s}$. All other initial states of the vehicles are fixed.

The Ego vehicle has 5 sensors. Figure 6 visualizes the sensor placement and ranges of the sensors. A long-range sensor with a $22.5°$ field of view and $50\,\mathrm{m}$ range is placed at the front of the vehicle. Two $5\,\mathrm{m}$-range sensors with $90°$ field-of-view are placed on the sides, facing left and right. Two $7\,\mathrm{m}$-range sensors with $90°$ field-of-view are placed at the rear-left and rear-right corners with an angle to scan the area behind the rear corners of the vehicle.

Agent vehicle $\mathtt{a}_1$ is controlled by the *move-to-pose* controller described in [17]. Agent vehicles $\mathtt{a}_2, \mathtt{a}_3$, and $\mathtt{a}_4$ are driven with a constant speed on a straight line. The Ego vehicle controller is a rudimentary reactive controller which actively attempts to avoid collisions by moving to a direction opposite from the future projected collision direction (see [21]). Our goal for this work is not to design a robust collision avoidance controller, but rather to show how different testing methods explore the search space.

For this case study, we only search for an optimal trajectory for $\mathtt{a}_1$ with the target of minimizing the cost function described in Eq. (1). The existence of agent vehicles $\mathtt{a}_2, \mathtt{a}_3$, and $\mathtt{a}_4$ between Ego vehicle $\mathtt{e}$ and agent vehicle $\mathtt{a}_1$ creates many local minima for the selection of trajectories for $\mathtt{a}_1$.

In this case study, we have executed 100 experiments with both the RRT-based approach and the falsification-based approach [2] in MATLAB. Each run of both approaches had a time-out duration of $30\,\mathrm{min}$. Out of 100 runs, only 5 of the minimum-cost trajectories returned by the falsification-based approach were able to make $\mathtt{a}_1$ move into the lane of the Ego vehicle, and only 2 trajectories were able to cause a collision (1 high-speed collision and 1 boundary-case collision) and, other than the collision cases, only 1 trajectory was able to challenge Ego vehicle by activating its collision avoidance system. All other trajectories returned by the falsification approach were stuck in local-minima where $\mathtt{a}_1$ tries to get closer to Ego vehicle and ends up colliding with one of the other agent vehicles. On the other hand in 28 of the minimum-cost trajectories returned by the RRT-based approach, agent $\mathtt{a}_1$ was able to get into the lane of Ego vehicle and it was able to cause the Ego vehicle to collide in 11 of those cases.

Figure 7 shows one of the interesting collision cases discovered by the RRT-based approach. Agent $\mathtt{a}_1$ first forces Ego to move to the right to avoid a collision and then to the left where it ends up colliding with Agent $\mathtt{a}_3$. Histories of $\mathtt{a}_1$ (red) and Ego (yellow) vehicles are numbered to show their evolution over time. Figure 8 shows the only low-speed collision case discovered by the falsification-based approach. Agent $\mathtt{a}_1$ moves into the Ego vehicle's lane, accelerates and rear-ends with Ego vehicle even though Ego vehicle tries to avoid the collision by accelerating and steering away. Figure 9 shows a typical trajectory that is stuck in a local minimum. Agent $\mathtt{a}_1$ tries to move closer to Ego vehicle and reduces the time-to-collision but collides with one of the other agents, which is $\mathtt{a}_2$ in this figure. Although both approaches can get stuck in a local minimum, this case is significantly more common for the falsification-based approach as discussed above.

As a numerical comparison of the minimum costs discovered by the two approaches, the minimum, mean and maximum costs achieved by the falsification approach were $0.0043, 13.7134$, and 50.6017, respectively. The minimum, mean and maximum costs achieved by the RRT-based approach were $4.8955, 10.2571, 15.0856$. Figure 10 provides box and whisker diagrams of the minimum costs achieved by the two approaches among the 100 experiments we have executed. The black diamonds plotted on top of the box plots show the mean values for the returned minimum costs. While

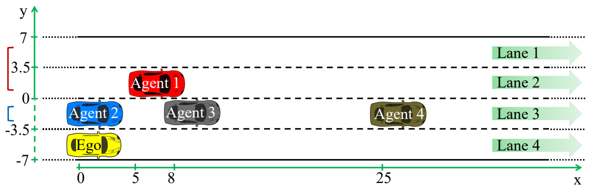

Fig. 5. Initial states of the vehicles in the setup for Case Study 2.

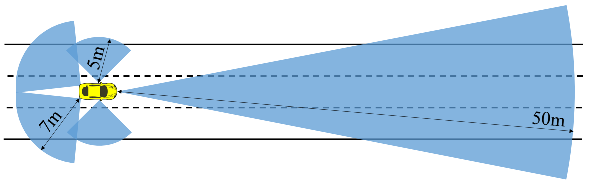

Fig. 6. Ego vehicle sensor setup for Case Study 2.

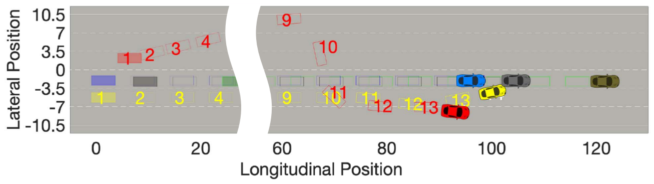

Fig. 7. One of the collision cases discovered by the RRT-based approach.

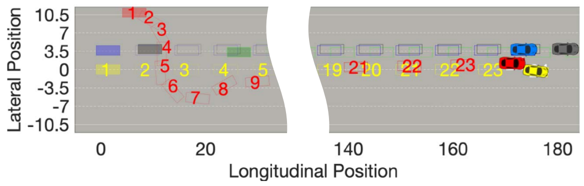

Fig. 8. The low-speed collision found by the falsification-based approach.

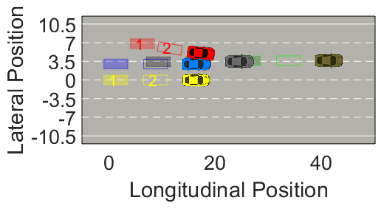

Fig. 9. An example of the falsification-based approach getting stuck in a local minimum.

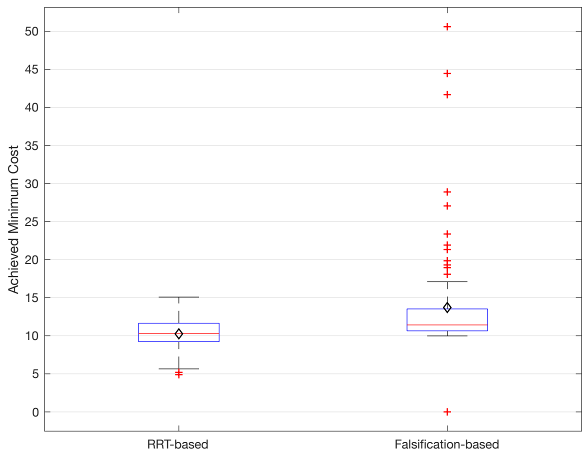

Fig. 10. Comparison of the minimum cost achieved by the falsification approach and the RRT-based approach in Case Study 2.

44 out of 100 RRT-based approach experiments were able to avoid local minima near the cost 10, only 2 of the falsification approach experiments were able to achieve this.

An observation we would like to share is that in the single case where the falsification approach was able to cause a low-speed collision, the achieved minimum cost was significantly smaller than any of the 11 collision cases discovered by the RRT-based approach. Typically, when the falsification-based approach enters into the neighborhood of a local minimum, it is more successful in getting closer to the minimum point than the RRT-based approach. The advantage of the RRT-based approach in avoiding local minima and the advantage of the falsification-based approach in getting closer to local minima suggests that an approach combining these two methods has the potential to improve the overall test generation performance. In such an approach, firstly, RRT-based approach would discover neighborhoods of minima and then the falsification-based approach would guide the test toward the minima starting from the results of the RRT-based approach.

IV. Conclusions and Future Work

We proposed an approach that explores maneuvers for adversary agents using rapidly-exploring random trees to discover interesting scenarios for automated driving systems under test. We have adopted notions from transition-based RRT [9] and RRT* [10] methods. The proposed approach delivered more promising results compared to the stochastic optimization-based falsification approach [2] for the test scenarios that create many local minima in the cost landscape of the system safety. Another substantial benefit of the RRT-based approach over the falsification-based approach is the avoidance of the finite parameterization of the trajectories of the adversarial agents. This enables the possibility of discovering adversarial driving scenarios without resorting to the intuition of the test engineers and their potential biases. Our future work will concentrate on developing a graphical user interface for easily specifying arbitrary test scenarios.

References

[1] "Waymo safety report: On the road to fully self-driving," Waymo, Tech. Rep., 2017. [Online]. Available: https://waymo.com/safety/
[2] C. E. Tuncali, T. P. Pavlic, and G. Fainekos, "Utilizing S-TaLiRo as an automatic test generation framework for autonomous vehicles," in *IEEE Intelligent Transportation Systems Conference*, 2016.
[3] C. E. Tuncali, G. Fainekos, H. Ito, and J. Kapinski, "Simulation-based adversarial test generation for autonomous vehicles with machine learning components," in *IEEE IV*, 2018.
[4] T. Dreossi, S. Ghosh, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Systematic testing of convolutional neural networks for autonomous driving," in *Reliable Machine Learning in the Wild (RMLW)*, 2017.
[5] H. Abbas, M. O'Kelly, A. Rodionova, and R. Mangharam, "Safe at any speed: A simulation-based test harness for autonomous vehicles," in 7^{th} *Workshop on Cyber-Physical Systems (CyPhy)*, 2017.
[6] H. Abbas, I. Saha, Y. Shoukry, R. Ehlers, G. Fainekos, R. Gupta, R. Majumdar, and D. Ulus, "Embedded software for robotics: Challenges and future directions," in *EMSOFT*. ACM, 2018.
[7] Y. S. R. Annapureddy, C. Liu, G. E. Fainekos, and S. Sankaranarayanan, "S-TaLiRo: A tool for temporal logic falsification for hybrid systems," in *TACAS*, ser. LNCS, vol. 6605. Springer, 2011.
[8] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.
[9] L. Jaillet, J. Cortés, and T. Siméon, "Transition-based RRT for path planning in continuous cost spaces," in *IEEE/RSJ IROS*, 2008.
[10] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.
[11] S-TaLiRo Tools, "https://sites.google.com/a/asu.edu/s-taliro/."
[12] J. Kim, J. M. Esposito, and V. Kumar, "Sampling-based algorithm for testing and validating robot controllers," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1257–72, 2006.
[13] M. S. Branicky, M. M. Curtiss, J. Levine, and S. Morgan, "Sampling-based planning, control and verification of hybrid systems," *IEE Proceedings-Control Theory and Applications*, vol. 153, no. 5, pp. 575–590, 2006.
[14] T. Dang, A. Donzé, O. Maler, and N. Shalev, "Sensitive state-space exploration," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*. IEEE, 2008, pp. 4049–4054.
[15] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Hybrid systems: from verification to falsification by combining motion planning and discrete search," *Formal Methods in System Design*, vol. 34, no. 2, pp. 157–182, 2009.
[16] T. Dreossi, T. Dang, A. Donzé, J. Kapinski, X. Jin, and J. V. Deshmukh, "Efficient guiding strategies for testing of temporal properties of hybrid systems," in *NASA Formal Methods (NFM)*, ser. LNCS, vol. 9058. Springer, 2015, pp. 127–142.
[17] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms In MATLAB®, 2nd, Completely Revised*. Springer, 2017, vol. 118.
[18] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.
[19] J. Lehman and K. O. Stanley, "Exploiting open-endedness to solve problems through the search for novelty." in *ALIFE*, 2008.
[20] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.
[21] C. E. Tuncali and G. Fainekos, "Rapidly-exploring random trees-based test generation for autonomous vehicles," arXiv:1903.10629, Tech. Rep., 2019.
[22] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart, "The mahalanobis distance," *Chemometrics and intelligent laboratory systems*, vol. 50, no. 1, pp. 1–18, 2000.