**2022 NDIA MICHIGAN CHAPTER**
**GROUND VEHICLE SYSTEMS ENGINEERING**
**AND TECHNOLOGY SYMPOSIUM**
**MODELING SIMULATION AND SOFTWARE (MS2) TECHNICAL SESSION**
**AUGUST 16-18, 2022 - NOVI, MICHIGAN**

# USING DEEP REINFORCEMENT LEARNING TO GENERATE ADVERSARIAL SCENARIOS FOR OFF-ROAD AUTONOMOUS VEHICLES

**Ted Sender[1], Mark Brudnak, PhD[2], Reid Steiger[3], Ram Vasudevan, PhD[1], Bogdan Epureanu, PhD[1]**

[1]Mechanical Engineering Department, University of Michigan, Ann Arbor, MI
[2]U.S. Army DEVCOM Ground Vehicle Systems Center, Warren, MI
[3]Ford Motor Company, Dearborn, MI

## ABSTRACT

*Modern perception systems for autonomous vehicles are often dependent on deep neural networks, however, such networks are unfortunately susceptible to subtle perturbations to their inputs. Due to the interconnected nature of perception/control systems in autonomous vehicles, it is quite difficult to evaluate the autonomy stack's robustness in different scenarios. Numerous tools have been developed to assist developers increase the robustness of these algorithms for on-road driving, but little has been accomplished for off-road driving. This work aims to bridge this gap by presenting a reinforcement learning framework to identify unsuspecting off-road scenes that confuse a custom autonomy stack with a DNN-based perception algorithm to ultimately lead the vehicle into a collision.*

**Citation:** T. Sender, M. Brudnak, R. Steiger, R. Vasudevan, B. Epureanu, "Using Deep Reinforcement Learning to Generate Adversarial Scenarios for Off-Road Autonomous Vehicles," In *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium* (GVSETS), NDIA, Novi, MI, Aug. 16-18, 2022.

## 1. INTRODUCTION

Autonomous vehicles (AVs) are complex machines with advanced vehicle dynamics and often consist of interconnected perception, navigation, and control systems. Validating the satisfactory performance of AVs is crucial to their deployment in all applications. In recent years, the algorithms that "drive" these vehicles are now relying on deep neural networks (DNNs) [1], however, DNNs have been shown to be susceptible to subtle variations in their inputs [2]; these perturbed inputs are often known as adversarial examples. The existence of these adversarial examples is primarily attributed to uncertainty (leaving aside problems inherent to

DNNs). While both on-road and off-road AVs share sources of uncertainty from weather and interactions with dynamic actors, specific to on-road AVs is the unpredictable behavior of other actors, and specific to off-road AVs is the large variation in natural, unstructured environments.

Many researchers have turned to simulation as a cost-effective alternative for testing AVs. However, due to popular interest, a majority of the published research in the field of AV testing are designed for or only demonstrate effectiveness in the on-road domain. Most of these works also specifically pertain to safety-critical cases in which vehicle failure can be catastrophic. A thorough summary of these works is given in [3]. Consequently, numerous driving simulators (CARLA [4], WeBots [5], Prescan [6], VISTA 1.0 [7], VISTA 2.0 [8] etc.) and various datasets (ApolloScape [9], Berkeley DeepDrive [10], KITTI [11], Cityscapes [12], etc.) have been created to assist developers improve the robustness of the on-road AV machine learning components. But few, if any, of these works are helpful for autonomous off-road driving.

There is a clear need for both off-road simulation platforms as well as algorithms for testing and validating AV performance in such environments. While almost every simulator provides the tools to programmatically place objects, we do not know of any library specifically designed for creating off-road driving scenarios automatically. To the best of our knowledge, we only know of one off-road simulation platform [13] which is designed to test AV driving policies. This platform is built on CARLA, however, all of the pre-made driving environments are fixed and making manual modifications would be tedious. As for off-road AV testing, we believe there should exist a common framework for defining off-road driving scenarios, similar to NHTSA's operational design domain [14] framework. Since safety-critical scenarios are not as relevant to the off-road domain, we believe there is a greater need for identifying unsuspecting scenarios that pose a challenge for the vehicle's autonomy stack and result in unexpected/poor performance. Further, given that any off-road driving scenario can easily be parameterized by tens or hundreds of parameters at a minimum, we suggest that any algorithm for testing/validating off-road AV performance be capable of working with large decision spaces.

In this paper, we aim to bridge the off-road AV virtual testing/validation gap by providing the following contributions:

- We propose a scenario decomposition strategy tailored to the off-road domain.
- We present a scalable framework based on deep reinforcement learning for generating adversarial off-road scenarios aimed at confusing an AV's autonomy stack using a Distributed Twin-Delayed Deep Deterministic Policy Gradient algorithm with Prioritized Experience Replay and a novel Action Saturation Penalty.

The rest of this paper is structured as follows: In Section 2 we will describe our proposed off-road driving scenario decomposition. In Section 3 we will describe the adversarial scene generation problem. We present our methodology in Section 4. We discuss the experimental setup, experiments, and results in Section 5. And finally we provide our conclusions in Section 6.

## 2. OFF-ROAD SCENARIO DECOMPOSITION

In this section we will introduce our proposed off-road scenario decomposition. The U.S. National Highway Traffic Safety Administration (NHTSA) developed the operational design domain (ODD) [14] as an attempt to provide a unified framework for testing (on-road) automated vehicles. The ODD is composed of attributes which effectively describe the range of scenarios/conditions that a vehicle is designed to operate in. While the ODD framework is imperfect and it can be challenging to

list/quantify all possible attributes [15], we believe that AV developers should adhere to a unified taxonomy when discussing driving scenarios. Since the focus of our work is on the off-road domain, we propose a modified set of primary attribute categories: *structural attributes*, *textural attributes*, and *operational attributes*.

Structural attributes define the geometry of all the structural objects that compose the scene. This includes the ground surface, static obstacles (e.g., trees, bushes, rocks), water bodies, etc. In a general manner, we refer to the obstacles as structural scene actors or SSAs. When defining simulated test scenarios, for most SSAs the following attributes will suffice: x-position, y-position, yaw angle, and mesh scale factor (z-position can be omitted because the simulator has access to the ground surface).

Textural attributes enhance the visual realism by describing visual effects of the scene. These attributes mostly include describing the intensities of various weather/lighting conditions.

Operational attributes describe operational constraints on any vehicle's behavior within the scene (e.g., go/no-go areas, speed ranges/limits, and areas without GPS), can define the vehicle's task (e.g., start/goal locations), as well as describe the behavior of other agents (e.g., non-player characters).

**Definition 2.1** (Off-Road Driving Scenario)**.** *An off-road driving scenario is defined by a combination of attributes from all three categories,* $x \in \mathcal{X} = \{\mathcal{A}^{str}, \mathcal{A}^{txt}, \mathcal{A}^{op}\}$*, where* $\mathcal{A}^{str}$ *is the set of all possible values for the structural attributes,* $\mathcal{A}^{txt}$ *is the set of all possible values for the textural attributes, and* $\mathcal{A}^{op}$ *is the set of all possible values for the operational attributes that describe the scenario.*

## 3. PROBLEM DEFINITION

Let $x \in \mathcal{X}$ be an instance of a driving scenario description. At a high-level, we wish to identify scenarios that are unsuspecting (i.e., not too difficult for a human driver to accomplish) and yet lead the vehicle to exhibit unexpected, and usually undesirable/sub-optimal, actions, such as taking a longer route when a shorter one exists or crashing into an obstacle. Let us define what we call the *scenario difficulty gap*, which represents the "scenario's difficulty perceived by the AV" minus the "scenario's actual difficulty". If we can define a function $d : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ to compute this difficult gap, with $\mathcal{X}$ defined above and $\mathcal{Y}$ represents the set of all possible vehicle trajectories, then our goal is to identify the scenario(s) that maximize this quantity defined by

$$x^* = \arg\max_{x \in \mathcal{X}} d(x, y(x)), \tag{1}$$

where $y(x) \in \mathcal{Y}$ represents the resulting vehicle trajectory. However, since this notion of difficulty gap is quite abstract, we instead focus our attention to a modified optimization problem

$$x^* = \arg\max_{x \in \mathcal{X}} \tilde{d}(x, y(x)), \tag{2}$$

where $\tilde{d}$ acts as a proxy for $d$ and is a concrete and computationally tractable function that should be defined to capture a desired autonomy failure mode.

## 4. GENERATING ADVERSARIAL SCENES WITH DEEP REINFORCEMENT LEARNING

Our approach for solving equation (2) is to strategically create numerous driving scenarios in simulation and evaluate them against a given proxy function. Since off-road scenarios can be defined by many attributes, we propose the use of gradient-based search methods as these tend to scale well with the number of decision variables, compared to simulated annealing [16, 17] and genetic algorithms [18] used in the literature. Specifically, we propose to frame the problem in the reinforcement learning context and use a Distributed Twin-Delayed Deep Deterministic Policy Gradient (TD3) algorithm with Prioritized Experience Replay

(PER) and a novel Action Saturation Penalty (ASP). TD3 [19] is the base actor-critic algorithm, prioritized experience replay [20] allows us to prioritize learning from past difficult scenarios, and the action saturation penalty helps prevent the actor from choosing extreme values during the early stages of training.

To help frame the problem in the RL context we first decompose the scenario space into a set of fixed attributes $\mathcal{X}_{\text{fixed}}$ and a set of variable attributes $\mathcal{X}_{\text{var}}$ (this is done for convenience as one may want to fix certain attributes). The action space is the set of variable scenario attributes $\mathcal{X}_{\text{var}}$. The state space should (ideally) contain the full scene description and the vehicle's trajectory as denoted by the set $\mathcal{X} \times \mathcal{Y}$. However, if any fixed attributes are not used to compute the difficulty proxy calculation, then we may omit those attributes from the state vector that is passed to the actor network. When running a simulation, we have access to the entire vehicle's state and control inputs, however, given the frame rate at which most simulations run at, the resulting vehicle's trajectory will contain an excessively large number of $N_{\text{simpath}}$ points. To keep the state vector at a reasonable size, we sample the vehicle's state and control information temporally along the vehicle's trajectory into $N_{\text{path}}$ data points, where $N_{\text{path}} << N_{\text{simpath}}$. Consequently, we must first run the simulated driving scenario with the previous action in order to construct the state vector. Finally, the reward function in this RL context is the difficulty proxy's evaluation.

Figure 1 provides an illustration of the process for training the RL agent and for generating adversarial scenarios. This framework consists of three layers: the ASG, an RL backbone, and a simulation platform. The actor processes the current state and outputs an action, which in this case is the variable scene description. An Unreal Engine (UE) worker is then given a scenario request consisting of both the fixed and variable scene attributes, and the UE worker begins communicating with a remote autonomy stack to carry out the navigation task. When the simulation is complete, the UE worker sends the vehicle's trajectory information to a processing block which outputs three objects: 1) the resulting next state vector, 2) the (state, action, reward, next state) RL transition to add to the replay buffer, from which we can sample from and train the critic/actor networks, and 3) a more detailed set of information regarding the entire scenario to store for further review.

## 5. EXPERIMENTAL RESULTS

To conduct experiments with our proposed approach, we will use Unreal Engine 4 (UE4) as well as our custom UE4 plugin called AutomaticSceneGeneration. This plugin is designed to quickly generate basic off-road scenes (i.e., a flat ground plane) with any composition of structural scene actors, and it will also launch basic navigation scenarios and return the vehicle's trajectory at the end of each scenario. It also requires use of the ROSIntegration plugin [21] for ROS communication. The rest of this section describes the base autonomy system under test, the experiments performed, and the results.

### *5.1. Base Autonomy System*

All of our experiments use a Polaris MRZR vehicle model (with hand-tuned model parameters) as the test vehicle with a forward-facing camera and a localization sensor. The camera is a combined RGB and depth-camera with a 90 degree field of view and a resolution of $256 \times 256$ running at 15 Hz. The localization sensor provides the vehicle's position and orientation and runs at 60 Hz. The full autonomy system is illustrated in Figure 2. The vehicle's perception system is a custom traversability semantic segmentation DNN that labels each pixel from the camera images as belonging to an object that is traversable (e.g., the ground plane), non-traversable (e.g., SSAs), or to the sky. This semantic segmentation network is based

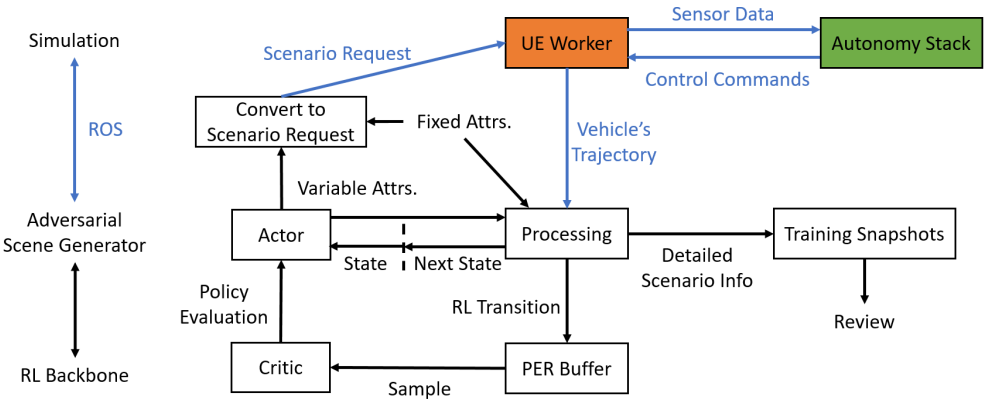



**Figure 1:** Adversarial scene generation framework. Although only one simulation instance is shown, this framework can work with any number of simulation instances running in parallel.

on the U-Net architecture [22] and is trained with categorical cross-entropy. The labeled images for the training dataset are created using an automatic image labeler from the AutomaticSceneGeneration plugin. Example raw and semantic segmentation images are shown in Figure 3. The vehicle's path planner uses an A* voting path planner. The A* planner discretizes the environment into an obstacle grid and vehicle grid. The obstacle grid indicates possible obstacle locations and keeps track of how many votes each vertex has. The vehicle grid contains nodes connected by edges that the path planner uses to generate paths. For computational performance, the obstacle grid may be finer than the vehicle grid, but in our experiments we set the node spacing to 5 meters in both grids. Example grids are depicted in Figure 4. With every image that comes in, the segmentation network produces predicted labels, and if any labels correspond to points in 3D space with a height of less than 2 meters, then the closest obstacle node is given a vote, with 50 maximum votes per obstacle node. The A* path planner waits 1 second before planning the first path, and then every 2 seconds it generates a new path from the vehicle's current location to the goal location. The cost of traversing any edge in the vehicle grid is equal to the arc-length of that edge times a scaling factor. The more obstacle votes near a vehicle grid edge, the larger the scaling factor. The path follower runs at 20 Hz and uses a basic PD controller to follow the most recently generated path with a constant tracking speed.

### 5.2. Base Experiment Parameters

As a way of mimicking a flawed autonomy system, we intentionally train our semantic segmentation network on a data set containing only barberry bushes, see Figure 5, despite the simulation also including juniper trees. The landscape is of size $300 \times 300$ meters. The vehicle's task is to start at (100,100) meters and navigate to (200,200) meters without crashing into an obstacle. We impose a safety radius around the start and goal locations such that any obstacles placed inside these regions will be removed. We also consider the vehicle to have succeeded in reaching the goal location if it enters the goal radius. Both the safety radius and goal radius are set to 11 meters. The goal for the ASG is to create scenes composed of eight barberry bushes and eight juniper trees on a flat ground plane to induce a vehicle collision. The proxy function used by the ASG is defined by

$$\tilde{d}_{\text{collision}} = 0.1\text{vis} + 0.25 d_{\text{goal}} + 10/(d_{\text{obs}} + \epsilon), \quad (3)$$

where vis $\in [-1, 1]$ indicates how visible all obstacles are in the camera's field of view at some

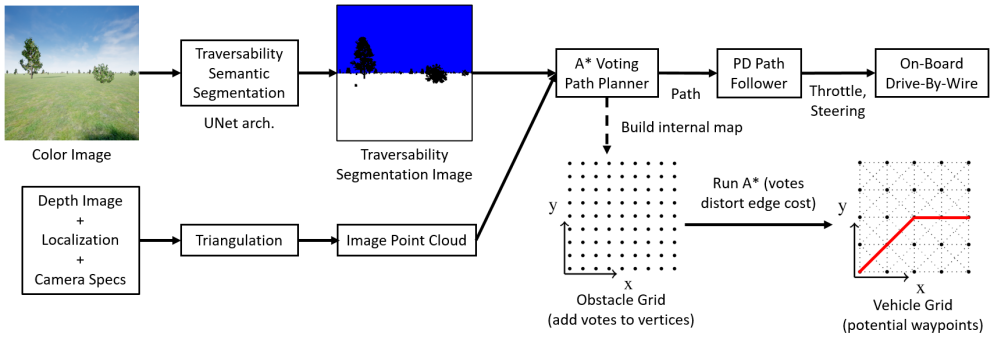
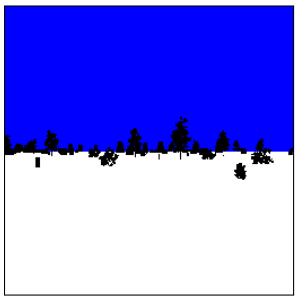


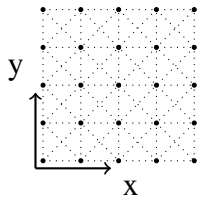

**Figure 2:** A* traversability autonomy stack.

**(a)** Raw color image.

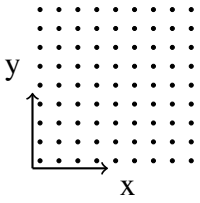


**(b)** Traversability semantic segmentation image (blue = sky, white = ground plane, and black = obstacle).

**Figure 3:** An example pair of a (a) raw color image and (b) labeled image in the traversability semantic segmentation dataset. Image in (b) is bordered for clarity.




**(a)** Vehicle grid with 10 meter spacing. Dotted lines represent graph edges.




**(b)** Obstacle grid with 5 meter spacing.

**Figure 4:** Example vehicle and obstacle grids for use by the traversability-based A* path planner.

point during the simulation, $d_{\text{goal}}$ is the distance

**Figure 5:** Vegetation meshes used in our simulations. Barberry bush on the left and juniper tree on the right.

between the vehicle's end location and the goal location, and $d_{\text{obs}}$ is the closest distance the vehicle came to any one obstacle during the simulation. We ran three separate experiments modifying only the speed of the vehicle and whether or not the obstacles can cast shadows. In each experiment, we train the RL agent for 1000 episodes with five steps per episode. Conducting a single experiment using eight parallel simulations takes approximately 12 hours.

### *5.3. Experiment and Results*

In this section, we will describe the conducted experiments and discuss some of the highlights from each experiment. Table 1 contains the experimental parameters as well as how many collisions occurred during each experiment.

In experiment 1, the vehicle was set to move at a speed of 5 m/s and the default lighting settings in UE4 were used (stationary directional light with

**Table 1:** Experiment Parameters and Number of Collisions.

| | Vehicle Speed [m/s] | Shadow Casting? | Num. Collisions |
|---|---|---|---|
| **Exp. 1** | 5 | Yes | 444 |
| **Exp. 2** | 5 | No | 175 |
| **Exp. 3** | 2.5 | Yes | 304 |

shadow casting). Of the 5000 scenarios, 444 resulted in a vehicle collision. Upon analyzing the vehicle's internal decisions on some of the most the difficult scenarios, we learned that our segmentation network, when trained only on bushes, misclassifies tree shadows as non-traversable objects, see example in Figure 6. Consequently, many scenes with a vehicle collision were due to avoiding a tree's shadow and not having enough time to adequately reroute around nearby obstacles. Figure 7 shows two scenes from experiment 1 that resulted in a vehicle collision. The dashed line in these figures represents the optimal path to take assuming perfect knowledge of the scene a priori. Figure 8 shows the path planner's state at three different time instances from the scene in Figure 7b to show how the collision occurred.

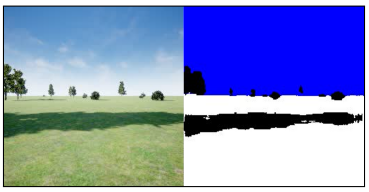
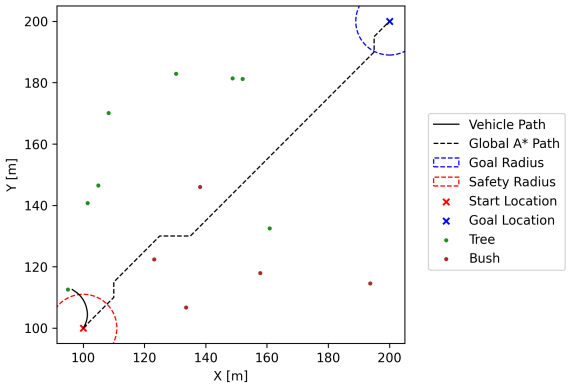
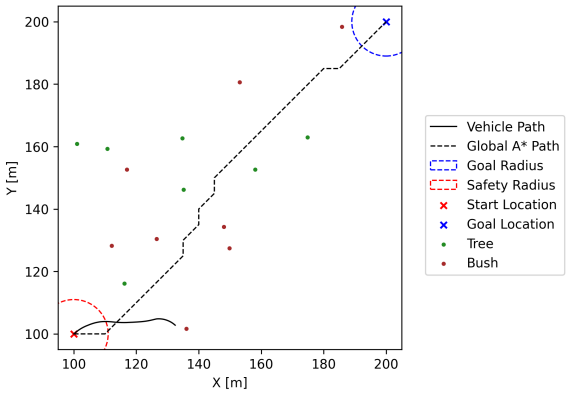


**Figure 6:** Perception network shadow misclassification. The left image is the raw camera image and right the image is the predicted semantic segmentation image, which shows the shadow being classified as an obstacle. Image is bordered for clarity.

In light of learning how shadows affect the vehicle's behavior, in experiment 2 we chose to disable shadow casting on all obstacles in the simulation and maintain the same vehicle speed of 5 m/s. Of the 5000 scenarios, 175 resulted in a vehicle collision. The fewer number of crashes is to




**(a)** Collision with tree.

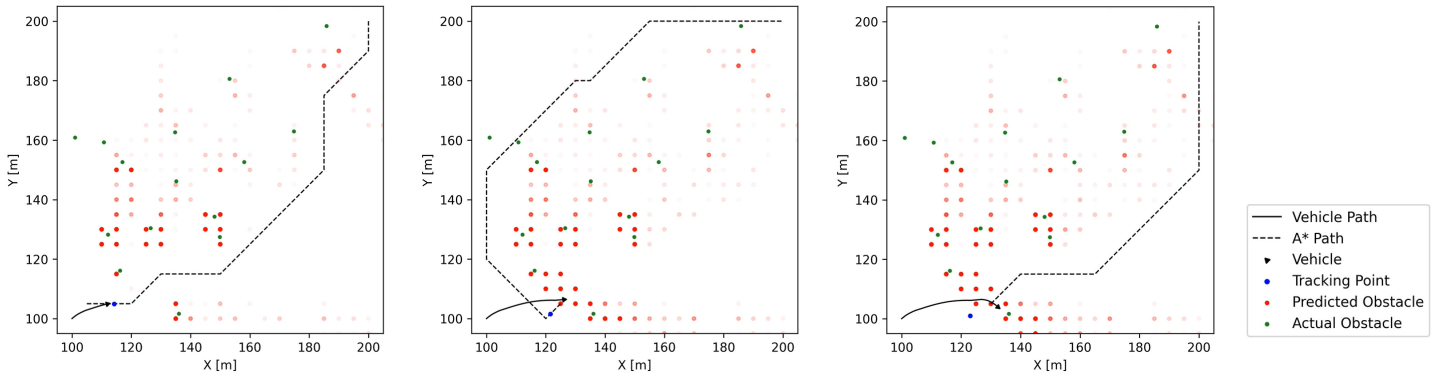



**(b)** Collision with bush.

**Figure 7:** Two collisions from experiment 1. Images show high-level scene summary.

be expected as most of the collisions in experiment 1 were due to the avoidance of shadows. The collisions that did occur were mainly due to the constraint of forward motion only and the inability to make sharp turns. As an example, sometimes the path planner would route the vehicle away from obstacles without giving the vehicle enough time to avoid the obstacle. Additionally, some collisions occurred because the segmentation network often classified tree trunks as being traversable and hence the vehicle would occasionally try to drive through trees. Figure 9 shows two scenes from experiment 2 that resulted in a vehicle collision.

Last, in experiment 3 we re-enabled shadow casting but instead lowered the vehicle speed to 2.5




**(a)** Vehicle is tracking one of the first few generated paths and everything appears okay so far.

**(b)** The perception network misclassifies the tree's shadow as an obstacle (large cluster of red dots surrounding the vehicle) and based on the newly generated path, the vehicle must change its course.

**(c)** Based on the vehicle's current pose and speed, it cannot avoid the bush and thus crashes into it, despite a new route from the path planner. The path follower has not yet received the new path, hence the tracking point shown is stale.

**Figure 8:** Snapshots of the path planner's decisions at three chronological time instances for the scene in Figure 7b. The intensity of the red indicates the number of obstacle votes assigned to that obstacle vertex. These images show how the vehicle attempts to avoid a tree's shadow but in the process crashes into a bush.

m/s to see if the additional time to process sensor data would lead to fewer collisions. Of the 5000 scenarios, 304 resulted in a vehicle collision. Since shadows were enabled, many of the collisions were still due to avoiding shadows. However, we did find one interesting scenario, shown in Figure 10, in which the path planner alternated between having the vehicle travel to the left and right around a bush, and eventually the vehicle crashes into the bush.

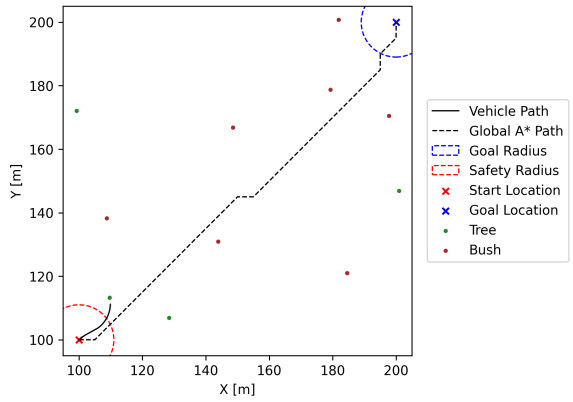
## 6. CONCLUSIONS

In this work we developed both an off-road simulation test platform and a reinforcement learning-based framework that is capable of identifying a multitude of scenarios that pose as a challenge to navigate by a custom autonomy stack with a DNN-based perception system and a basic path planner and follower. Our experiments revealed that the failure mechanism for collisions are not straightforward and often require a thorough analysis of the autonomy stack's run-time decisions. Our experiments even revealed failure mechanisms that were unexpected by the developers of the autonomy stack, which is one of the main use-cases for this work. In future work, we plan to increase the capability of our simulation platform to modify weather conditions as well as test vehicles on a non-flat ground surface, and we intend to demonstrate how our framework can identify weaknesses throughout the autonomy stack development cycle.

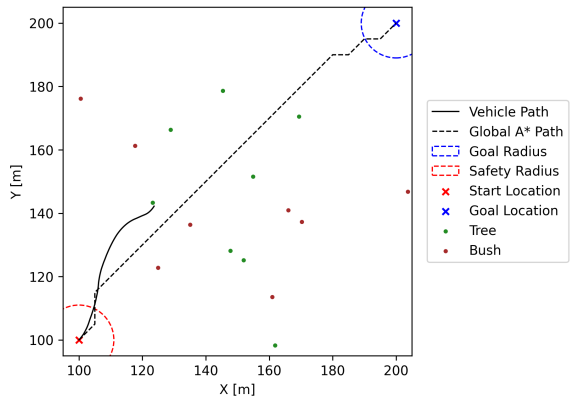
## 7. REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2014.




**(a)** Collision with tree.




**(b)** Collision with tree.

**Figure 9:** Two collisions from experiment 2. Images show high-level scene summary.

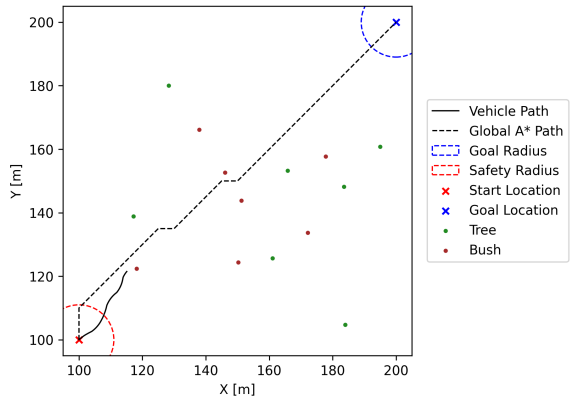



**Figure 10:** Scenario from experiment 3 where the path planner's oscillatory behavior drives the vehicle into a bush.

[3] W. Ding, C. Xu, M. Arief, H. Lin, B. Li, and D. Zhao, "A survey on safety-critical scenario generation for autonomous driving – a methodological perspective," 2022. arXiv:2022.02215 [cs.RO].

[4] 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*, pp. 1–16, 2017.

[5] O. Michel, "Webots: Professional mobile robot simulation," *Journal of Advanced Robotics Systems*, vol. 1, no. 1, pp. 39–42, 2004.

[6] F. Hendriks, M. Tideman, R. Pelders, R. Bours, and X. Liu, "Development tools for active safety systems: Prescan and vehil," in *Proceedings of 2010 IEEE International Conference on Vehicular Electronics and Safety*, pp. 54–58, 2010.

[7] A. Amini, I. Gilitschenski, J. Phillips, J. Moseyko, R. Banerjee, S. Karaman, and D. Rus, "Learning robust control policies for end-to-end autonomous driving from data-driven simulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1143–1150, 2020.

[8] A. Amini, T.-H. Wang, I. Gilitschenski, W. Schwarting, Z. Liu, S. Han, S. Karaman, and D. Rus, "Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles," 2021. arXiv:2111.12083 [cs.RO].

[9] P. Wang, X. Huang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, "The apolloscape open dataset for autonomous driving and its application," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[10] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2633–2642, 2020.

[11] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[12] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3213–3223, 2016.

[13] I. Han, D.-H. Park, and K.-J. Kim, "A new open-source off-road environment for benchmark generalization of autonomous driving," *IEEE Access*, vol. 9, pp. 136071–136082, 2021.

[14] E. Thorn, S. Kimmel, and M. Chaka, "A framework for automated driving system testable cases and scenarios," Tech. Rep. DOT HS 812 623, National Highway Traffic Safety Administration, September 2018.

[15] P. Koopman and F. Fratrik, "How many operational design domains, objects, and events?," in *SafeAI@AAAI*, 2019.

[16] C. E. Tuncali, G. Fainekos, H. Ito, and J. Kapinski, "Simulation-based adversarial test generation for autonomous vehicles with machine learning components," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1555–1562, 2018.

[17] C. E. Tuncali, G. Fainekos, D. Prokhorov, H. Ito, and J. Kapinski, "Requirements-driven test generation for autonomous vehicles with machine learning components," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 2, pp. 265–280, 2020.

[18] X. Zheng, H. Liang, B. Yu, B. Li, S. Wang, and Z. Chen, "Rapid generation of challenging simulation scenarios for autonomous vehicles based on adversarial test," in *2020 IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 1166–1172, 2020.

[19] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596, PMLR, 10–15 Jul 2018.

[20] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2016.

[21] P. Mania and M. Beetz, "A framework for self-training perceptual agents in simulated photorealistic environments," in *International Conference on Robotics and Automation (ICRA)*, (Montreal, Canada), 2019.

[22] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, eds.), (Cham), pp. 234–241, Springer International Publishing, 2015.