Generating Transferable Adversarial Simulation Scenarios for Self-Driving via Neural Rendering

Anonymous Author(s)
Affiliation
Address
email

Abstract: Self-driving software pipelines include components that are learned from a significant number of training examples, yet it remains challenging to evaluate the overall system's safety and generalization performance. Together with scaling up the real-world deployment of autonomous vehicles, it is of critical importance to automatically find simulation scenarios where the driving policies will fail. We propose a method that efficiently generates adversarial simulation scenarios for autonomous driving by solving an optimal control problem that aims to maximally perturb the policy from its nominal trajectory. Given an image-based driving policy, we show that we can inject new objects in a neural rendering representation of the deployment scene, and optimize their texture in order to generate adversarial sensor inputs to the policy. We demonstrate that adversarial scenarios discovered purely in the neural renderer (surrogate scene) can often be successfully transferred to the deployment scene, without further optimization. We demonstrate this transfer occurs both in simulated and real environments, provided the learned surrogate scene is sufficiently close to the deployment scene.

1 Introduction

Safety certification of a self-driving stack would require driving hundreds of millions of miles on real roads, according to [1], to be able to estimate miles per intervention with statistical significance. This could correspond to decades of driving and data collection. Procedural generation of driving simulation scenarios has emerged as a complementary approach for designing unseen test environments for autonomous vehicles in a cost-effective way. Currently, generation of simulation scenarios requires significant human involvement, for example to specify the number of cars and pedestrians in the scene, their initial locations and approximate trajectories [2], as well as selection of assets to be added to the simulator. In addition to being challenging to scale, having a human in the loop can result in missing critical testing configurations.

In this paper, we cast adversarial scenario generation as a high-dimensional optimal control problem. Given a known image-based driving policy that we want to attack, as well as the dynamics of the autonomous vehicle, we aim to optimize a photorealistic simulation environment such that it produces sensor observations that are 3D-viewpoint-consistent, but adversarial with respect to the policy, causing it to deviate from its nominal trajectory. The objective of the optimal control problem is to maximize this deviation through plausible perturbations of objects in the photorealistic environment.

Our optimal control formulation requires differentiation through the sensor model in order to compute the derivative of the sensor output with respect to the underlying state perturbation. However, most existing photorealistic simulators for autonomous vehicles are not differentiable; they can only be treated as black boxes that allow forward evaluation, but not backpropagation. Instead of using an off-the-shelf photorealistic simulator and adding assets to match the scene, we train an editable neural rendering model that imitates the deployment scene, allowing us to insert new objects in the

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

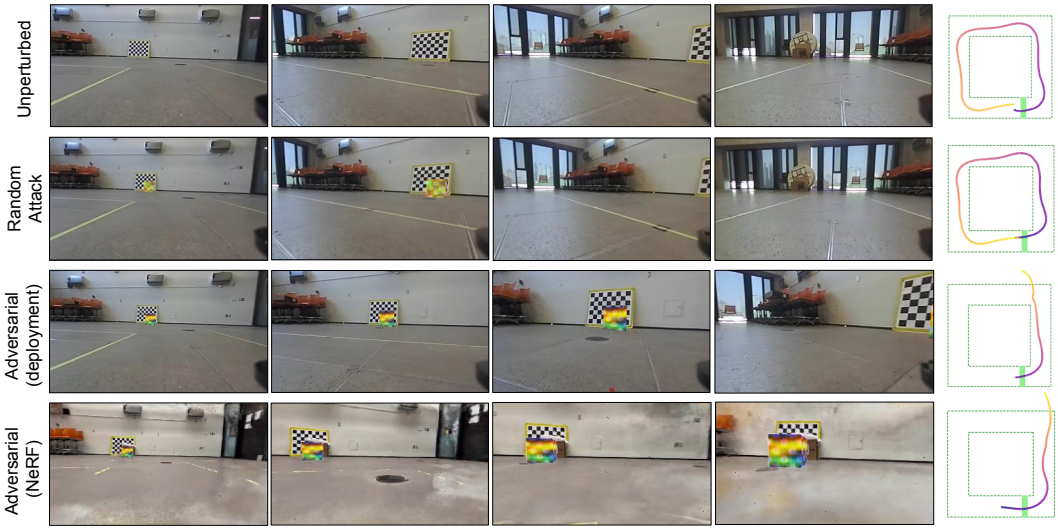

Figure 1: First-person-view (FPV) of our adversarial attack transfer to an RC car with overhead trajectory view on the right. Row 1: Unperturbed policy execution; Row 2: Random search texture attack; Row 3: Our adversarial attack directly transferred to the real deployment scene, without additional optimization; Row 4: Our adversarial attack discovered in the surrogate NeRF simulator.

simulator and to optimize their texture through gradient-based optimization. This editable neural rendering model acts as a surrogate physics and rendering simulator, enabling us to differentiate through it in an efficient way in order to attack the driving policy's input sensor observations.

Unlike many existing types of adversarial attacks in the literature [3, 4, 5], our work aims to discover environment perturbations/attacks that satisfy the following properties: (a) **They are temporally-consistent**. The influence of the attack is not instantaneous, it is amortized through time via the optimal control loss function. (b) **They are transferable**. An attack discovered in the surrogate scene should ideally be adversarial in the actual deployment scene. (c) **They are object-centric**. The attack introduces and edits objects as opposed to unstructured high-frequency perturbations across all pixels.

Specifically, we make the following contributions:

1. We formulate adversarial scenario generation across time as an optimal control problem that relies on a learned, surrogate NeRF simulator. The solution to this problem yields 3D-view-consistent, object-centric, adversarial attacks that often transfer to the deployment environment. We show how to solve this problem efficiently using implicit differentiation.
2. Differentiable rendering of our surrogate NeRF model enables gradient-based adversarial object insertion and scales to high dimensional parameterizations of multiple objects.
3. We show that our adversarial attacks discovered in the surrogate NeRF simulator can be realized in the real-world and retain their ability to disrupt the policy.

We experimentally validate our framework by reconstructing scenes using only pose-annotated images and generate adversarial object insertion attacks with multiple trajectories.

2 Related Work

Adversarial scenarios for autonomous driving. Perceptual adversarial attacks make modifications to prerecorded sensor data from real driving sessions to fool the perception system. Since this sensor data is fixed, they lack the ability to resimulate and typically only operate on the individual frame level. Previous works, [4, 6] attempt to attack a LiDAR object detection module by artificially inserting an adversarial mesh on top of car rooftops or objects in a prerecorded LiDAR sequence. They extend the scope of their attack further by incorporating textures to be able to attack image-

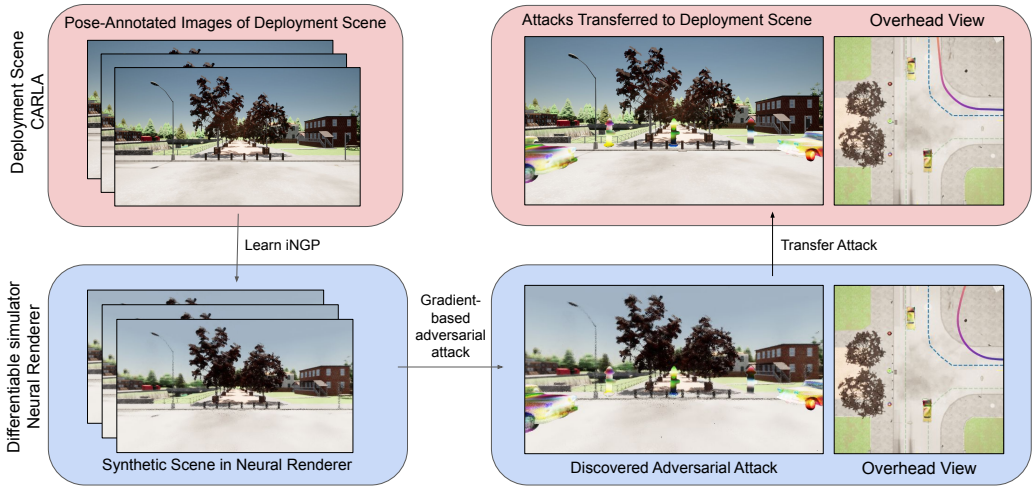

Figure 2: Our method can be summarized in the four steps shown. (a) In the top left, we obtain posed images from the deployment scene which can be a simulator or the real world. (b) In the bottom left, we reconstruct a surrogate scene by fitting a NeRF to the posed images as a differentiable simulator and observe only minor perceptual gap. (c) Having the surrogate scene, we can insert objects, which are also represented as NeRFs, and attack their color fields to generate textural attacks. (d) The discovered adversarial objects are introduced back into the deployment scene.

based object detectors as well [3]. In both these works, the inserted object has a very low resolution and nondescript geometry. Recent self-driving simulators, such as DriveGAN [7], GeoSim [8] and UniSim [9] address these issues, with the latter enabling manipulable sensor-based simulators based on prerecorded datasets. These works, however, have not dealt with discovering attacks.

Another prominent line of works produce dynamic state-level adversarial attacks. These generally target the control/planning system only by perturbing trajectories of other agents in the scene. Without considering the perception system, these methods use simplified traffic and state-based simulators that do not incorporate 3D rendering [10, 11, 12].

Closest to our work, a few methods have proposed to attack end-to-end self-driving policies that perform both perception and control. To this end, adversarial perturbations are made to existing self-driving simulators, primarily CARLA. In [13], the trajectories of other agents in a CARLA scene are modified to generate a collision event. Due to the non-differentiability of the simulator, a black-box Bayesian optimization is used to search for successful attacks. Gradient-based attacks on top of simulators have also been investigated. However, the requirement of differentiability has so far limited their scope to very simplified geometries that are composited post-hoc onto renderings from CARLA. In [5], flatly colored rectangles are composited on top of frames from the CARLA simulator and optimized to cause maximal deviation of an end-to-end image-based neural network steering controller. Similarly, work in [14] attempts to play a video sequence of adversarial images on a billboard in the scene using image composition. To our knowledge, no works in this setting have been able to demonstrate transfer of adversarial attacks to the real world, as these attacks rely on a pre-existing black box simulator that they augment. Compared to these, our attacks are entirely performed on a surrogate neural simulator that is reconstructed from only posed images captured from any deployment scene. Furthermore, these works have not demonstrated their ability to perform perceptual attacks with more complex objects, whereas our surrogate neural simulator allows for inserting arbitrary objects reconstructed from posed images.

The driving simulator VISTA [15] generates high fidelity sensor data using a large collection of real world data. In our case, we are able to train a NeRF using the data, allowing us to generalize to a wider range of novel views.

Robust adversarial examples. Adversarial attacks for classification have commonly used minimal perturbations on the input images [16] that may not always transfer to the physical world or another

domain. To enhance robustness to domain transfer [17] proposes a class of adversarial transformations that optimize for adversarial attacks under a distribution of image transforms [18].

3 Background

3.1 Neural Rendering

Neural 3D representations, such as neural radiance fields (NeRF), have seen significant activity in recent years due to their ability to reconstruct real world objects and scenes to very high detail using only posed images. A survey of recent progress in neural rendering can be found in [19].

Differentiable rendering. NeRFs represent scenes as emissive volumetric media [20]. Unlike surface rendering, volumetric rendering does not suffer from explicit hard discontinuities, which are difficult to handle for traditional surface rendering methods[19]. We exploit the differentiable volume rendering of NeRFs, to robustly compute efficient gradients for arbitrary geometries.

Complex scene reconstruction. Works such as BlockNeRF [21] and S-NeRF [22] show great potential for automatically capturing street-level scenes relevant to autonomous vehicle simulation. Unlike traditional simulators, these neural representations are directly trained on raw sensor captures, thereby obtaining high-fidelity visual reconstruction without laborious asset creation.

Composition and editing. Recent works have extended the static single scene setting of NeRF to composition of NeRFs, scene disentanglement, as well as editing and relighting. Specifically, [23] encodes scenes with latent codes from which new scenes can be generated. [24], [25] and [26] introduce compositional radiance fields to represent different objects and realize scene decomposition. [27] utilizes 2D segmentation information to perform 3D scene inpainting. [28] and [29] decompose color into different illumination components. [30] [31] learn priors from big datasets of images to disentangle existing scenes.

Control and neural rendering models. Neural rendering has seen utility in a few different optimization tasks for robotics such as pose estimation [32]. NeRFs have also seen direct application to trajectory optimization, including utilizing NeRF's density as approximate occupancy [33], collision, and friction constraints [34], effectively allowing NeRF to act as a differentiable physics simulator. On the other hand, [35] learns an additional latent dynamics of NeRF objects. In [36], physics simulations of known objects are combined with their learned NeRF to create high fidelity dynamic environments for training deep reinforcement learning agents, that can transfer to the real world. We use composition of NeRFs in our work to insert and optimize adversarial objects. This is shown in Fig. 3, and the details are in Sections 4.1 and 4.2.

Accelerated rendering. The original NeRF method has high computational cost of training and rendering. Structured grid NeRFs reduce computation cost by storing direct density and color variables [37] or latent features[38, 39] on explicit 3D grids. Instant Neural Graphics Primitives (iNGP) uses multi-scale dense grids and sparse hash grids of features that are decoded to color and density by a MLP [20]. We use iNGP like models to represent the scene and objects in our work.

3.2 Gradient computation via implicit differentiation

Gradient computation for trajectory optimization has been explored in prior works [40, 41]. A crucial tool in these methods is known as the adjoint method, which enables constant memory gradient computation with respect to trajectory length. In discrete time, the adjoint method amounts to propagating gradients through an implicit relationship G for problems of the form:

$$\min_{\theta} J(\theta) = \sum_{t=0}^{T} C(x_t) \text{ such that } G(x_{t-1}, x_t, \theta) = 0 \tag{1}$$

This can be done in two passes where we do not need to store large intermediate variables and only need to accumulate the gradient at each step. Appendix A.3 presents how we make use of implicit differentiation to efficiently solve this optimization problem.

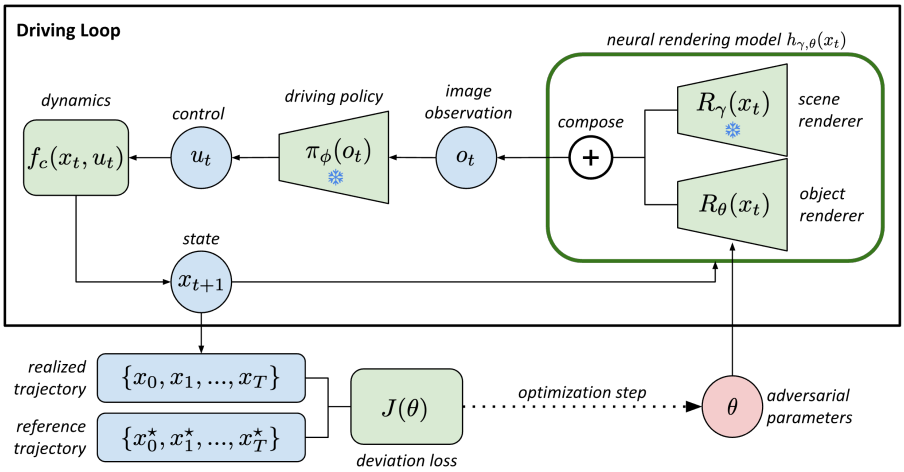

Figure 3: A computation diagram of our algorithm for generating adversarial attacks. The inner driving loop consists of three components: the neural rendering model, the driving policy, and the car dynamics. We inject the adversarial perturbation to the surrogate scene by composing the outputs of one or more neural object renderers (the single object case is shown above for simplicity) with the output of the neural scene renderer. The parameters of the object renderer(s) are optimized to maximize the deviation of the realized trajectory from the reference trajectory, while keeping the parameters of the driving policy and scene renderer frozen.

4 Method

Our framework generates successful adversarial attacks of end-to-end image-based self-driving policies with only access to posed images from the deployment scene. An overview of the high-level steps in our framework is shown in Figure 2.

We now briefly describe the setting and our adversarial attack method. More details are included in Appendix A. Let x_t denote the state of the car at time t, x^* denote a reference trajectory to track and CTE the cross-track error. Starting from Eqn. 1, we set the cost function $C(x_t)$ of our problem as the car's proximity to the reference x^*:

$$C(x_t) = -\text{CTE}(x_t, x^*) \tag{2}$$

In other words, we want to maximize deviation from the desired trajectory. We set the constraint function $G(x_t, x_{t+1}, \theta) = 0$ to be the following set of constraints:

$$u_t = \pi_\phi(o_t) \quad (3) \qquad o_t = h_{\gamma,\theta}(x_t) \quad (4) \qquad x_{t+1} = f_c(x_t, u_t) \quad (5)$$

Where π is the fixed driving policy[1], h is the neural rendering sensor model that outputs image observations o_t given the state of the car. The renderer depends on θ, the parameters of adversarial NeRF objects and γ, the fixed rendering parameters of the background scene NeRF. Finally, f_c denotes the dynamics of the ego vehicle that must be considered, since we want to find adversarial trajectories that are consistent across multiple frames.

4.1 Differentiable Renderer

Traditional simulators like CARLA do not admit computation of gradients. Thus, prior works rely on artificially compositing simplistic textured geometries on top of rendered images from CARLA and obtaining gradients with respect to the composited alteration [14]. We use NeRFs to learn surrogate models of the scene and sensor model instead. This surrogate model not only gives us an automated method to reconstruct scenes from pose-annotated images, but also provides efficient gradient computation giving us a differentiable form for the sensor h. For the purposes of optimization, we found

[1] We train our own policy and provide details in Appendix B.2.

traditional NeRF representations to be intractable in terms of compute and memory requirements (during gradient computation). Thus, we opt to use the multi-resolution hash grid representation, Instant-NGP [20].

Note that, similar to existing work, we detach the gradients of the image observation with respect to the camera coordinates (which are attached to the ego vehicle) [42]. We include more details regarding this in Appendix A.4.

4.2 Adversarial Object Insertion

We use insertion and texturing of multiple objects as our adversarial perturbations to the background scene. To do this, we first reconstruct regular objects, such as cars, as individual NeRFs from pose-annotated images. For our object NeRFs we simply store color values directly on the voxel grids of Instant-NGP, which are tri-linearly interpolated within each voxel. By choosing these color voxel grids as our adversarial parameters θ, we can perform independent adversarial texture attacks over multiple objects. The object NeRFs can be easily composed with our background scene NeRF. This is done via alpha compositing, which leverages opacity and depth values that can be easily computed.

4.3 Gradient-based Adversarial Attack

Obtaining gradients for the problem in Eqn. (1) should be possible with an autodifferentiation framework such as PyTorch [43]. We find that naively computing the gradient via backpropagation results in memory issues as we scale up trajectory lengths due to all the intermediary compute variables used to compute the integral in Eqn. A.1 being stored until the end of the trajectory. We achieve drastic memory savings by using the adjoint method [40] which only keeps track of the adjoint variables λ along the trajectory. In our case, the adjoint variables are three-dimensional, allowing us to only use as much memory as it takes to compute a single jacobian vector product of the composition of models given by (5), (3), (4) in the optimization problem in Eqn. (1).

Figure 4: Base car on the left; random texture in the middle; adversarial texture on the right.

To summarize, the computation of our gradient-based adversarial attack proceeds as follows:

1. We rollout our policy in our surrogate simulator to compute the loss and the trajectory $x_{1:T}$ in Eqn. (1).
2. We perform a backward pass to compute adjoint variables for gradient computation.
3. Using the adjoint variables, we compute the gradient $\nabla_\theta J$ and update parameters θ.

5 Experiments

To demonstrate the effectiveness of our framework, we aim to reconstruct a driving scenario from posed images, generate adversarial attacks and validate that those attacks transfer to the deployment scene. Through our experiments, we would like to answer the following key questions:

(Q1) Can gradient based optimization find better adversarial examples than random search?
(Q2) Are NeRF models suitable surrogate renderers for gradient based adversarial optimization?
(Q3) Are adversarial attacks transferable from NeRF back to the deployment domain?

5.1 Experimental Details

CARLA Deployment Scenes. We first validate our method in simulation, treating CARLA as a proxy for a real deployment scene. We perform experiments on a 3-way intersection of the CARLA [44] simulator. For the 3-way intersection, we consider 3 different trajectories to be followed by the ego vehicle. For the object models, we train surrogate NeRF models for two sample

objects, a fire hydrant and a small car using only posed images (any other object could be used). We manually insert 2 small cars and 3 fire hydrants into the driving scene in an initial placement. Our adversarial attacks jointly optimize the NeRF color parameters and object rigid transforms.

Real World Deployment Scenes. Our real-world experiments are performed on an autonomous RC car driving around a square race track in an indoor room. It is difficult to manufacture adversarial attacked objects with complex shapes in the real world. Hence, for practicality, we insert a NeRF object representing a flat square texture pattern that can be projected by a display monitor in the real world and optimize its color parameters. In order to create a first version of the attack we choose to directly compose the adversarial texture on to the robot camera feed. We then move on to a more difficult task of physically realizing this attack, for this we opted to display the texture on a monitor to simplify lighting conditions. Additional details of our real world experimental setup are given in B.4.1

5.2 Evaluation Metrics

We measure the effectiveness of an attack with our adversarial objective, the cross track error of the vehicle. We use the road center as the reference and so even an unperturbed driving policy has some non-zero deviation which we report under "Unperturbed" in Table 1. To characterize the insensitivity of our method to random seeds, we run 5 separate attacks per scenario for both the gradient-based and random attacks with different random initializations of the adversarial parameters. We report the mean and standard deviation of our metric.

Our proposed method of attack is via gradient-based optimization using the method outlined in Section 4.3. The gradient-based attack uses 50 iterations of optimization using Adam, with a learning rate of 0.1. Due to the high dimensional parameterization, detailed in A.4.1, bayesian optimization becomes computationally intractable. Therefore, as a baseline for our method, we perform a random search parameter attack on the NeRF surrogate model that samples parameters from a Gaussian distribution with mean zero and a standard deviation of 5. We chose this standard deviation to match the distribution over parameters we found in our gradient attacks. We use the same number of function evaluations, selecting the best achieved attack among the 50 random samples for the CARLA experiment. For real-world experiments, we didn't find much variation between random attacks in the surrogate simulator, showing the difficulty of random search in high dimensional parameter spaces.

5.3 Experimental Results

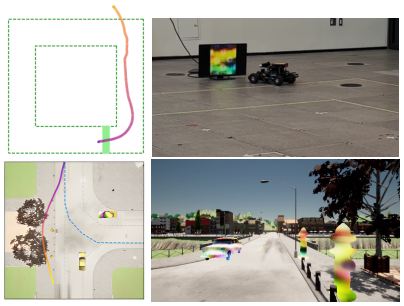

Figure 5: Selected overhead views and snapshots from adversarial deployment trajectories in the real world (top row: monitor displays adversarial texture discovered in NeRF), and in CARLA (bottom row: adversarial objects inserted in the simulator).

Example gradient attack trajectories are shown in Figure 5 We include more visualization of results for deployments of adversarial attacks, both in CARLA simulation in the real world, in Appendix C.

In Table 1 we compare the total cross track errors caused by our adversarial attack against the expert lane following controller. We observe in all 3 CARLA scenarios (averaged over 5 seeds each) that our adversarial attacks using gradient optimization consistently produce significant deviation from the lane center. When transferring these attacks back into the deployment scene, we see that although the magnitude of the deviation is reduced, we still retain a significant increase over the unperturbed or random search setting. The difference is likely due to visual imperfections in our surrogate NeRF simulator compared to the deployment scene. The random

	CARLA Deployment	Surrogate Scene		CARLA Deployment	
Scenario	Unperturbed	Random	Gradient	Random	Gradient
Straight	1166	1132 ± 7	2347 ± 49	1193 ± 19	$1702. \pm 160$
Right	1315	2084 ± 10	4105 ± 847	1476 ± 12	$2101. \pm 75$
Left	1448	1460 ± 8	4125 ± 124	1158 ± 163	$2240. \pm 574$
	Physical Deployment	Surrogate Scene		Physical Deployment	
Setup	Unperturbed	Random	Gradient	Random	Gradient
Green Screen	48	34 ± 4	157 ± 1	46 ± 3	248 ± 72
Monitor				47 ± 3	76 ± 48

Table 1: Comparison of the total cross-track error for all the scenario tested. Results are shown for the following cases: (1) no attack in the deployment scene (unperturbed), (2) an adversarial attack (random or gradient) in the surrogate NeRF scene, (3) an attack in the deployment scene. We separate results from the CARLA and physical deployments, we show that gradients in our surrogate simulator are useful for finding adversarial attacks and these attacks remain effective when transferred to the deployment environment.

search perturbations are far less effective, remaining near the baseline unperturbed trajectory for 2 out of the 3 cases.

For the real world experiment, we observed a similar result. Random attacks consistently fail to elicit deviation from the driving policy both in the surrogate and deployment scenes. Over 5 random seeds, not a single random attack was able to cause the vehicle to exit the track. Gradient attacks on the other hand are reliably able to find strong attacks with little variance in the surrogate scene. When transferring our attacks to the real world, we find the attacks to retain their strength in the green screen setup. The strength of the attack is relatively diminished when using the monitor to project the attack but is nonetheless consistently higher than the random attack and causes the vehicle to understeer and exit the track on occasion. We suspect this is due to the display properties of the monitor which can alter the appearance of the adversarial perturbation.

6 Limitations

Despite showing the ability to generate 3D-consistent adversarial scene perturbations, there are a few avenues for improvement. We currently only optimize the color of the object we place into the scene, however, our method is not limited to this and is capable of optimizing for the shape of objects as well through density parameters of the NeRF. Due to their volumetric nature, arbitrary modifications to the density may result in unusual shapes so additional geometry regularization would need to be introduced. Although gradient-based optimization scales well to high-dimensional parameters, occlusions, and long-horizon dynamics can cause our adversarial objective to be highly non-smooth. In such cases, it's worth investigating approaches for nonsmooth optimization.

7 Conclusion

We presented a method for generating 3D-consistent object-based adversarial perturbations in autonomous driving scenarios. Unlike previous approaches that rely on making edits on top of fixed pre-recorded data or black-box simulators, we develop a differentiable simulator directly with a neural radiance field representation of geometry and texture of a scene that admits gradients through the rendering of camera and depth observations. Through alpha-compositing, we can introduce new objects also represented as neural radiance fields into the scene and optimize color perturbations of the objects. We validate our framework both in simulation and on a real-world RC car race track driving scenario showing successful sim-to-real transfer of discovered attacks. While our particular implementation is only a first step towards demonstrating NeRF based adversarial attack generation, we believe that our framework shows a promising new direction for automatic evaluation of autonomous vehicles. We expect our method to benefit greatly from continued improvements being made to neural rendering and their wider adoption for AV/robotic simulation.

References

[1] K. Nidhi and S. M. Paddock. Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability? https://www.rand.org/pubs/research_reports/RR1478.html, 2016. [Online; accessed 19-July-2018].

[2] A. C. Madrigal. Inside waymo's secret world for training self-driving cars. *The Atlantic*, Aug 2017. URL https://www.theatlantic.com/technology/archive/2017/08/inside-waymos-secret-testing-and-simulation-facilities/537648/.

[3] J. Tu, H. Li, X. Yan, M. Ren, Y. Chen, M. Liang, E. Bitar, E. Yumer, and R. Urtasun. Exploring adversarial robustness of multi-sensor perception systems in self driving, Jan. 2022. URL https://arxiv.org/abs/2101.06784.

[4] J. Tu, M. Ren, S. Manivasagam, M. Liang, B. Yang, R. Du, F. Cheng, and R. Urtasun. Physically Realizable Adversarial Examples for LiDAR Object Detection, Apr. 2020. URL https://arxiv.org/abs/2004.00543.

[5] J. Yang, A. Boloor, A. Chakrabarti, X. Zhang, and Y. Vorobeychik. Finding Physical Adversarial Examples for Autonomous Driving with Fast and Differentiable Image Compositing, June 2021. URL https://arxiv.org/abs/2010.08844.

[6] Y. Cao, C. Xiao, D. Yang, J. Fang, R. Yang, M. Liu, and B. Li. Adversarial objects against lidar-based autonomous driving systems. *CoRR*, abs/1907.05418, 2019. URL http://arxiv.org/abs/1907.05418.

[7] S. W. Kim, J. Philion, A. Torralba, and S. Fidler. Drivegan: Towards a controllable high-quality neural simulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5820–5829, June 2021.

[8] Y. Chen, F. Rong, S. Duggal, S. Wang, X. Yan, S. Manivasagam, S. Xue, E. Yumer, and R. Urtasun. Geosim: Realistic video simulation via geometry-aware composition for self-driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7230–7240, June 2021.

[9] Z. Yang, Y. Chen, J. Wang, S. Manivasagam, W.-C. Ma, A. J. Yang, and R. Urtasun. Unisim: A neural closed-loop sensor simulator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1389–1399, June 2023.

[10] J. Wang, A. Pun, J. Tu, S. Manivasagam, A. Sadat, S. Casas, M. Ren, and R. Urtasun. AdvSim: Generating Safety-Critical Scenarios for Self-Driving Vehicles, Jan. 2022. URL https://arxiv.org/abs/2101.06549.

[11] D. Rempe, J. Philion, L. J. Guibas, S. Fidler, and O. Litany. Generating useful accident-prone driving scenarios via a learned traffic prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17305–17315, 2022.

[12] M. Igl, D. Kim, A. Kuefler, P. Mougin, P. Shah, K. Shiarlis, D. Anguelov, M. Palatucci, B. White, and S. Whiteson. Symphony: Learning realistic and diverse agents for autonomous driving simulation, 2022. URL https://arxiv.org/abs/2205.03195.

[13] Y. Abeysirigoonawardena, F. Shkurti, and G. Dudek. Generating adversarial driving scenarios in high-fidelity simulators. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8271–8277. IEEE, 2019.

[14] N. Patel, P. Krishnamurthy, S. Garg, and F. Khorrami. Overriding Autonomous Driving Systems Using Adaptive Adversarial Billboards. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):11386–11396, Aug. 2022.

[15] 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. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2419–2426, 2022.

[16] N. Akhtar, A. Mian, N. Kardan, and M. Shah. Advances in adversarial attacks and defenses in computer vision: A survey, Sept. 2021. URL https://arxiv.org/abs/2108.00401.

[17] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok. Synthesizing robust adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 284–293, 2018.

[18] P. Buddareddygari, T. Zhang, Y. Yang, and Y. Ren. Targeted attack on deep rl-based autonomous driving with learned visual patterns. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10571–10577, 2022.

[19] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, Z. Xu, T. Simon, M. Nießner, E. Tretschk, L. Liu, B. Mildenhall, P. Srinivasan, R. Pandey, S. Orts-Escolano, S. Fanello, M. Guo, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, D. B. Goldman, and M. Zollhöfer. Advances in neural rendering. In *ACM SIGGRAPH 2021 Courses*, SIGGRAPH '21, 2021.

[20] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):1–15, July 2022.

[21] M. Tancik, V. Casser, X. Yan, S. Pradhan, B. Mildenhall, P. P. Srinivasan, J. T. Barron, and H. Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8248–8258, 2022.

[22] Z. Xie, J. Zhang, W. Li, F. Zhang, and L. Zhang. S-neRF: Neural radiance fields for street views. In *International Conference on Learning Representations*, 2023.

[23] A. R. Kosiorek, H. Strathmann, D. Zoran, P. Moreno, R. Schneider, S. Mokrá, and D. J. Rezende. Nerf-vae: A geometry aware 3d scene generative model, 2021. URL https://arxiv.org/abs/2104.00587.

[24] M. Niemeyer and A. Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[25] B. Yang, Y. Zhang, Y. Xu, Y. Li, H. Zhou, H. Bao, G. Zhang, and Z. Cui. Learning object-compositional neural radiance field for editable scene rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13779–13788, 2021.

[26] S. Benaim, F. Warburg, P. E. Christensen, and S. Belongie. Volumetric disentanglement for 3d scene manipulation, 2022. URL https://arxiv.org/abs/2206.02776.

[27] A. Mirzaei, T. Aumentado-Armstrong, K. G. Derpanis, J. Kelly, M. A. Brubaker, I. Gilitschenski, and A. Levinshtein. Spin-nerf: Multiview segmentation and perceptual inpainting with neural radiance fields, 2022. URL https://arxiv.org/abs/2211.12254.

[28] P. P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, and J. T. Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7495–7504, 2021.

[29] W. Ye, S. Chen, C. Bao, H. Bao, M. Pollefeys, Z. Cui, and G. Zhang. Intrinsicnerf: Learning intrinsic neural radiance fields for editable novel view synthesis, 2022. URL https://arxiv.org/abs/2210.00647.

[30] Y. Xu, M. Chai, Z. Shi, S. Peng, S. Ivan, S. Aliaksandr, C. Yang, Y. Shen, H.-Y. Lee, B. Zhou, and T. Sergy. Discoscene: Spatially disentangled generative radiance field for controllable 3d-aware scene synthesis, 2022. URL https://arxiv.org/abs/2212.11984.

[31] A. Kundu, K. Genova, X. Yin, A. Fathi, C. Pantofaru, L. J. Guibas, A. Tagliasacchi, F. Dellaert, and T. Funkhouser. Panoptic neural fields: A semantic object-aware neural scene representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12871–12881, 2022.

[32] L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin. Inerf: Inverting neural radiance fields for pose estimation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 1323–1330, 2021.

[33] M. Adamkiewicz, T. Chen, A. Caccavale, R. Gardner, P. Culbertson, J. Bohg, and M. Schwager. Vision-only robot navigation in a neural radiance world. *IEEE Robotics and Automation Letters*, 7(2):4606–4613, 2022.

[34] S. L. Cleac'h, H. Yu, M. Guo, T. A. Howell, R. Gao, J. Wu, Z. Manchester, and M. Schwager. Differentiable physics simulation of dynamics-augmented neural objects, 2022. URL https://arxiv.org/abs/2210.09420.

[35] D. Driess, Z. Huang, Y. Li, R. Tedrake, and M. Toussaint. Learning multi-object dynamics with compositional neural radiance fields, 2022. URL https://arxiv.org/abs/2202.11855.

[36] A. Byravan, J. Humplik, L. Hasenclever, A. Brussee, F. Nori, T. Haarnoja, B. Moran, S. Bohez, F. Sadeghi, B. Vujatovic, and N. Heess. Nerf2real: Sim2real transfer of vision-guided bipedal motion skills using neural radiance fields, 2022. URL https://arxiv.org/abs/2210.04932.

[37] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022.

[38] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020.

[39] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su. Tensorf: Tensorial radiance fields. *European Conference on Computer Vision (ECCV)*, 2022.

[40] K. M. Jatavallabhula, M. Macklin, F. Golemo, V. Voleti, L. Petrini, M. Weiss, B. Considine, J. Parent-Levesque, K. Xie, K. Erleben, L. Paull, F. Shkurti, D. Nowrouzezahrai, and S. Fidler. gradsim: Differentiable simulation for system identification and visuomotor control. *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=c_E8kFWfhp0.

[41] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[42] N. Hanselmann, K. Renz, K. Chitta, A. Bhattacharyya, and A. Geiger. King: Generating safety-critical driving scenarios for robust imitation via kinematics gradients, 2022. URL https://arxiv.org/abs/2204.13683.

[43] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[44] 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*, pages 1–16, 2017.

[45] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi. Objaverse: A universe of annotated 3d objects, 2022. URL https://arxiv.org/abs/2212.08051.

[46] B. O. Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL http://www.blender.org.

[47] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars, 2016. URL https://arxiv.org/abs/1604.07316.

[48] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[49] F. Codevilla, M. Müller, A. Dosovitskiy, A. M. López, and V. Koltun. End-to-end driving via conditional imitation learning. *CoRR*, abs/1710.02410, 2017. URL http://arxiv.org/abs/1710.02410.

[50] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[51] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.