DriveEnv-NeRF: Exploration of A NeRF-Based Autonomous Driving Environment for Real-World Performance Validation

Mu-Yi Shen[1*], Chia-Chi Hsu[1*], Hao-Yu Hou[1*], Yu-Chen Huang[1*], Wei-Fang Sun[2], Chia-Che Chang[3], Yu-Lun Liu[4], and Chun-Yi Lee[1]

***Abstract*— In this study, we introduce the DriveEnv-NeRF framework, which leverages Neural Radiance Fields (NeRF) to enable the validation and faithful forecasting of the efficacy of autonomous driving agents in a targeted real-world scene. Standard simulator-based rendering often fails to accurately reflect real-world performance due to the sim-to-real gap, which represents the disparity between virtual simulations and real-world conditions. To mitigate this gap, we propose a workflow for building a high-fidelity simulation environment of the targeted real-world scene using NeRF. This approach is capable of rendering realistic images from novel viewpoints and constructing 3D meshes for emulating collisions. The validation of these capabilities through the comparison of success rates in both simulated and real environments demonstrates the benefits of using DriveEnv-NeRF as a real-world performance indicator. Furthermore, the DriveEnv-NeRF framework can serve as a training environment for autonomous driving agents under various lighting conditions. This approach enhances the robustness of the agents and reduces performance degradation when deployed to the target real scene, compared to agents fully trained using the standard simulator rendering pipeline. Please note that more results can be found on our project page: `https://github.com/muyishen2040/DriveEnvNeRF`.**

I. Introduction

Autonomous driving is a complex task involving real-time decision-making in intricate environments. Although various methods for learning driving policies in virtual environments have been proposed, they often exhibit performance degradation when applied to real-world scenarios. This degradation frequently stems from the discrepancy between simulated and real-world domains in both visual perception and control dynamics, commonly referred to as the sim-to-real gap [1–7]. The sim-to-real gap encompasses a wide range of aspects, including perceived observations, environmental conditions, as well as vehicle's physical dynamics. Each of these aspects can affect the practical deployment behavior of the agent and its interaction with real-world objects. As a result, forecasting such discrepancies poses a crucial challenge, particularly when attempting to predict the deployment behavior or performance during the transition from simulated environments to the real world. Therefore, correctly evaluating the real-world performance of a policy during the training phase in a simulated environment is a formidable task. Despite previous endeavors that have attempted to leverage validation environments to anticipate possible outcomes, they often fail to accurately predict actual performance [8, 9]. Moreover, validating the efficacy of trained driving policies directly in real-world scenarios is time-consuming, labor-intensive, and potentially expensive. An appropriate method to address the need for forecasting and validating real-world performance in simulated environments is thus necessary, especially for paradigms concentrating on virtual-to-real transfer learning.

Fig. 1: Visualization of the scenes rendered by DriveEnv-NeRF for emulating diverse lighting conditions using different setups of the appearance embedding for training policies.

Photorealistically mirroring real-world environments in a simulator that accurately captures the appearance of the to-be-deployed environment might offer a potential solution to mitigate such a gap. However, reconstructing photorealistic digital twin objects and scenes in virtual environments can be expensive. Factors such as challenging lighting and shadowing conditions, as well as dynamics, introduce complexity to the simulation process. A number of prior research endeavors from different perspectives have been put forth to tackle the virtual-to-real gap, including domain randomization [10–13], domain adaptation [14–16], knowledge distillation [17–21], meta-learning [22–24], learning with disturbances [12], and others [13, 25, 26]. In addition, some researchers have attempted to address the issue from a different angle by enhancing the simulator's fidelity through traditional or data-driven rendering [27–32]. While these previous efforts have been demonstrated valuable, the challenges of accurately forecasting a model's real-world performance still remain.

In light of the above, this study introduces a framework named DriveEnv-NeRF, capable of forecasting the performance of a driving agent in a targeted real-world environment through building a simulation environment. Specifically, the simulation environment replicates a target scene

*Equal contribution
[1]Elsa Lab, National Tsing Hua University
[2]NVIDIA AI Technology Center, NVIDIA Corporation
[3]MediaTek Inc.
[4]National Yang Ming Chiao Tung University

from the real-world environment (referred to as the '*target real scene*' or the '*target real environment*' hereafter) in which our agent intends to operate by utilizing a Neural Radiance Field (NeRF) [33, 34] model and a physics simulator. The NeRF model is trained to fit images gathered from the target real environment using a cost-effective camera setup. After being trained, it can reconstruct renderings within the target real scene, and its extracted 3D meshes are integrated into a simulator for physics simulations. This offers insights into the forecasted real-world performance when the agent is deployed, which facilitates adjustment of the training procedure to mitigate the impacts of the domain gap between the simulated and the targeted real-world environments.

To more realistically reflect the target real scene, we further utilize the concept of appearance embedding proposed in NeRF-W [35] as a means of data augmentation. This enables diversifying the lighting conditions in the simulated scene, which allows our agents to be deployed in various conditions such as daytime, dawn, or even in the evening, as illustrated in Fig. 1. The contributions can be summarized as follows:

- We introduce the DriveEnv-NeRF framework for building a simulation environment, which enables forecasting the performance of an agent's policy in real scenarios.
- We employ NeRF to model and reconstruct the target real environment and extract 3D meshes from it to provide a realistic emulation of the target real environment.
- We validate the effectiveness of the simulation environment constructed by DriveEnv-NeRF as a performance indicator for the policy trained by deep reinforcement learning (DRL) by conducting real-world experiments.
- We propose the use of appearance embedding to augment the training effectiveness by altering the lighting conditions to emulate more diverse training conditions.

II. Related Works

NeRF has gained considerable attention for its capability of synthesizing novel photo-realistic views with implicit scene representation. The success of NeRF has led to an explosion of subsequent methods addressing quality, speed, accurate scene geometry, and other aspects [36–47]. While NeRF performs well on static scenes with controlled lighting conditions, it encounters challenges when dealing with image collections from unconstrained real-world scenarios, such as variable weather, lighting, or transient occluders. NeRF-W [35] attempts to address these issues with appearance embeddings and transient networks. The existence of appearance embedding further enables implicit control of illumination. In addition to photo-realistic rendering, NeRF also allows for the extraction of accurate geometry. Methods such as Marching Cubes [48] or Poisson Surface Reconstruction [49] can extract 3D meshes from NeRF, which can be further utilized for physical simulations. The combination of 3D meshes and rendered images presents the possibility of constructing a real-to-sim simulation environment with NeRF [50–52]. NeRF2Real [50] employs NeRF to reconstruct real-world environments in a simulator, which enables novel view synthesis and collision detection. Based on such a simulator, a bipedal robot control policy trained with DRL can be deployed to the real world in a zero-shot manner.

III. Methodology: DriveEnv-NeRF Framework

Fig. 2 illustrates an overview of the proposed DriveEnv-NeRF framework, which aims to construct a simulation environment that better reflects a real-world deployment scene. To achieve this, DriveEnv-NeRF utilizes images of the target real scene to train a NeRF model that serves two purposes. First, the trained NeRF model is utilized to generate novel views for the agent during validation and DRL training. Second, 3D meshes of the scene are extracted from the trained NeRF model to emulate physical presence. As a result, DriveEnv-NeRF is able to reflect the environment appearance for visual observations and utilize the derived 3D meshes in the simulated environments to model collisions. The components of DriveEnv-NeRF are detailed as follows.

A. Construction of the Simulation Environment

The DriveEnv-NeRF framework encompasses four main stages: (1) Data Collection and Preprocessing, (2) Scene Reconstruction, (3) Physics Simulation, and (4) Hardware Alignment. Each of these stages is elaborated on as follows.

Data Collection and Preprocessing. To train the NeRF model, videos of the target real scene are captured under various lighting conditions, such as different times of the day and weather scenarios. Image frames are then extracted at regular intervals to ensure high-quality and unobstructed views. These frames are processed using the structure-from-motion (SfM) process with COLMAP [53, 54] to establish correspondences, which assist in determining the camera frames' extrinsic and intrinsic parameters for NeRF training.

Scene Reconstruction. To reconstruct the simulation environment for DriveEnv-NeRF, the NeRF model is first trained to generate novel view renderings, which are then utilized to derive the 3D meshes of the target real scene. The rendered images serve as input observations for the agent, while the 3D meshes are employed for collision detection. The NeRF model further incorporates an appearance embedding, which enables the representation of a scene under various lighting conditions. During inference, the appearance embedding can be specified to render images with different illuminations.

Physics Simulation. The simulation environment is built upon the Unity engine [55], a game engine with physics simulation capabilities for collision detection and 3D object management. The reference frames of the NeRF model and the 3D meshes in the simulator are aligned, which enables translation of coordinates between the simulated frames and the NeRF-rendered frames. Another key feature of this environment is its support for the incorporation of various 3D objects, such as pedestrians and obstacles, into the rendered scenes to enrich the environmental details and complexity.

Hardware Alignment. To faithfully capture real-world performance using the simulation environment built by DriveEnv-NeRF, it is essential to align the agent's configuration with real-world conditions. This involves adjusting parameters such as camera intrinsics, the camera's height

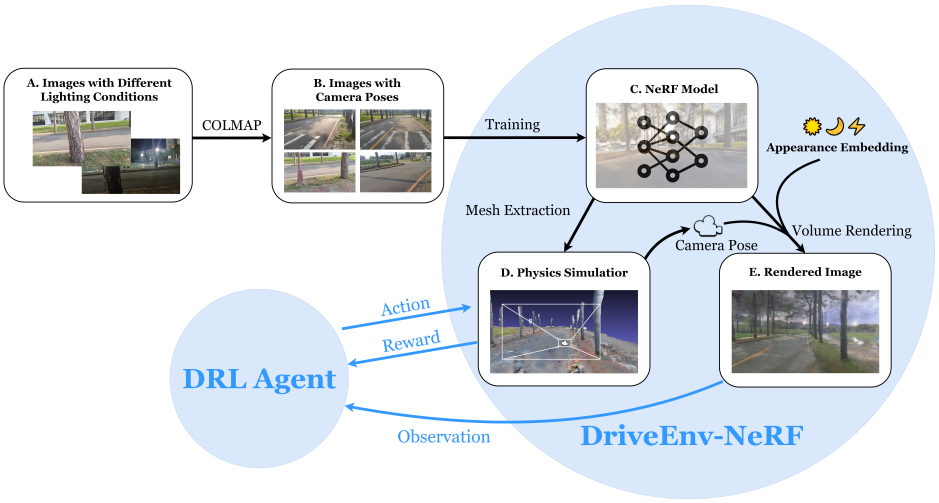


Fig. 2: An overview of the DriveEnv-NeRF framework.

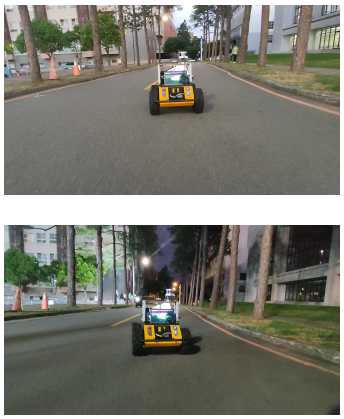


Fig. 3: The application of DriveEnv-NeRF shows agent's adaptability to diverse lighting.

(a) (b) (c)

Fig. 4: Examples of (a) an image rendered by the simulator, (b) a NeRF rendered image, and (c) a real image.

and angle relative to the ground and the vehicle, as well as aligning the vehicle's speed and turning angle. Other pertinent factors are taken into account for distinct use cases.

B. Validation and Training of DRL Agents

Once the simulation environment is built, the training and evaluation of driving policies can be fully conducted within it. At each timestep, the camera coordinates of the driving agent in the simulator are transformed to the corresponding coordinates in the NeRF model for rendering an image with NeRF. This image is then used as the agent's observation. Subsequently, the action selected by the agent is sent to the simulator, which executes it to simulate the agent's movements. This simulation environment can be used to both train and validate the agent's driving policy with different settings. More specifically, the road layout and lighting conditions can be fixed to serve as a standardized and reproducible validation condition. If more complex environments are desired for validation, DriveEnv-NeRF allows alternation of the appearance embedding to evaluate the policy's generalizability over different lighting or weather conditions, as depicted in Fig. 1. This feature is crucial in training a driving policy as it reduces the negative impacts of domain discrepancy caused by factors such as weather and lighting. Moreover, static or dynamic obstacles can be generated using the simulator. As a result, the simulation environment built with the DriveEnv-NeRF framework can serve as a benchmark for validating driving policies trained using different procedures through the emulation of these policies within the environment. By leveraging the realistic images rendered by NeRF, as illustrated in Fig. 4, the trained DRL policies can be validated within the reconstructed simulation environment using success rate or other types of metrics to forecast their performance in the target real scene.

IV. Experimental Results

A. Experimental Setup

Tasks. Two different navigation tasks, namely *Straight Road* and *Single Right Turn*, are utilized to demonstrate the effectiveness of our DriveEnv-NeRF framework for forecasting the performance of autonomous driving policies in the real world. In these tasks, the DRL agent is expected to safely navigate the road without deviating from the boundaries.

Validation Environments. Two validation environments, (1) the baseline simulator and (2) the simulation environment built by DriveEnv-NeRF, are utilized to assess their effectiveness in predicting the model's performance in the real world. In the baseline simulator, the agent can only observe the manually-built mesh of the scene. DriveEnv-NeRF, on the other hand, allows the agent to observe the renderings produced by NeRF. The DRL agent can be evaluated under an appearance embedding not encountered during training.

DRL Training Setup. To validate the effectiveness of our proposed environment, we employ the Soft Actor-Critic (SAC) algorithm [56] as the DRL policy. In addition, the agent is trained from scratch with raw RGB observations in

an end-to-end manner, without relying on prior datasets for pretraining. This specific setup aims to highlight the benefits offered by DriveEnv-NeRF. All training settings and model parameters are fixed across experiments. In our comparisons, we consider three baseline approaches described as follows:

- *Simulator:* This baseline takes input image observations generated by the standard simulator rendering pipeline.
- *DriveEnv-NeRF w/o appearance embedding:* The agent receives NeRF renderings as inputs. Nevertheless, the appearance embedding is kept constant during training.
- *DriveEnv-NeRF w/ appearance embedding:* This is the default DriveEnv-NeRF with varying appearance embeddings between episodes for diverse appearances.

Evaluation Metric. In the experiments, the success rate is employed to evaluate the performance of the trained DRL agents. An episode is considered successful if the DRL agent reaches its destination without deviating from the road boundaries, causing collisions, or exceeding the time limit.

B. Quantitative Results

TABLE I: The success rate of the agents trained under different baseline setups. 'A.E.' stands for appearance embedding. Each experiment is performed with ten independent runs.

Task 1 - Straight Road			
Baselines	Training	Validation	Real-World Testing
Simulator	0.94	0.57	0.00
DriveEnv-NeRF w/o A.E.	1.00	0.19	0.00
DriveEnv-NeRF w/ A.E.	0.99	0.98	0.80
Task 2 - Single Right Turn			
Baselines	Training	Validation	Real-World Testing
Simulator	0.73	0.54	0.00
DriveEnv-NeRF w/o A.E.	1.00	0.95	0.67
DriveEnv-NeRF w/ A.E.	1.00	1.00	0.83

Table I demonstrates the performances of the agents in different environments. The *Simulator* setup trains and validates the agent entirely using the standard simulator rendered image. In contrast, the *DriveEnv-NeRF* setup trains and validates the agent in the simulation environment built by DriveEnv-NeRF. One of the *DriveEnv-NeRF* setup utilizes appearance embeddings, while the other does not. In the *Real-World Testing* environment, we conduct experiments on a ClearPath Husky Unmanned Ground Vehicle (UGV) equipped with a ZED-2 camera, as depicted in Fig 3. Please note that in this study, the agent use raw image only, and does not rely on the depth estimation capability of ZED-2.

Results of Straight Road Task. The results indicate that despite achieving a high success rate during assessments within the training simulated environment for all three cases, only the policies trained under the NeRF environment with varying appearance embedding demonstrate the capability to reach the end-point without deviating from the straight road. These policies exhibit superior adaptability to different lighting conditions in the real-world scenarios and can achieve approximately an 80% success rate when driving through a straight road. In contrast, the other baselines exhibit a significantly higher probability of deviating from the road.

Results of Single Right Turn Task. The task of executing a right turn poses greater challenges when applied to the target real world. It can be observed that only the DRL models trained under NeRF environments can successfully complete the task. Similar to the previous task, providing the scene's various light conditions with appearance embedding during training can result in a more robust driving policy.

C. Sim-to-Real Gap Analysis

In both straight driving and right turn tasks, success rates below 0.6 in the *Validation* environments consistently lead to failure in *Real-World Testing* environments. A model that outperforms others in the simulation environment built by DriveEnv-NeRF still maintains its advantage in the real world. However, the results indicate that high success rates in the simulator do not guarantee similar performance in real-world scenarios due to the sim-to-real domain gap. These findings validate that DriveEnv-NeRF can serve as an effective predictor for real-world performance and can be used to forecast the relative performance of different policies.

Although DriveEnv-NeRF demonstrates performance alignment with the real world, a subtle disparity persists due to the inherent challenges in bridging the sim-to-real gap. Achieving precise physical calibration between simulation and real-world scenarios presents a significant obstacle. Fluctuations in camera extrinsics caused by UGV vibration and inaccuracies in UGV wheel encoders during slipping may further impact the performance of the DRL policy. Such discrepancies can complicate the application of experiences gained in the simulation environment to real-world scenarios.

V. Discussion and Conclusion

In this work, we presented DriveEnv-NeRF, a simulation framework using NeRFs to validate and forecast the real-world performance of autonomous driving agents. Our method adjusts appearance embeddings to reflect varying lighting conditions, and the results demonstrate its effectiveness as the success rates in simulated and real-world settings align. DriveEnv-NeRF also serves as a beneficial training platform for enhancing the adaptability and robustness of policies through the alteration of environmental appearances.

While DriveEnv-NeRF demonstrates promising results, opportunities for further refinement exist in novel view synthesis methods and simulator selection. Recent advancements introduced 3D Gaussian Splatting (3DGS) [41] for novel view synthesis, which offered realistic rendering at significantly faster speeds. 3DGS facilitates direct collision detection on point clouds and can potentially be enhanced with tailored mesh extraction methods such as SuGaR [57]. DriveEnv-NeRF, built on top of the Unity engine, faces challenges such as integration with practical robotics due to the gap between hardware simulation and real-world implementation. To address these, a potential avenue is shifting to Isaac Sim [58] together with ROS 2 [59] and Isaac ROS [60], which offer better tool integration and simulation fidelity.

ACKNOWLEDGEMENT

The authors gratefully acknowledge the support from the National Science and Technology Council (NSTC) in Taiwan under grant numbers MOST 111-2223-E-007-004- MY3, as well as the financial support from MediaTek Inc., Taiwan. The authors would also like to express their appreciation for the donation of the GPUs from NVIDIA Corporation and NVIDIA AI Technology Center (NVAITC) used in this work. Furthermore, the authors extend their gratitude to the National Center for High-Performance Computing (NCHC) for providing the necessary computational and storage resources.

REFERENCES

[1] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[2] Yunpeng Pan, Ching-An Cheng, Kamil Saigol, Keuntaek Lee, Xinyan Yan, Evangelos Theodorou, and Byron Boots. Agile autonomous driving using end-to-end deep imitation learning. *arXiv preprint arXiv:1709.07174*, 2017.

[3] Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end autonomous driving. *arXiv preprint arXiv:1605.06450*, 2016.

[4] Alexander Amini, Igor Gilitschenski, Jacob Phillips, Julia Moseyko, Rohan Banerjee, Sertac Karaman, and Daniela Rus. Learning robust control policies for end-to-end autonomous driving from data-driven simulation. *IEEE Robotics and Automation Letters*, 5(2):1143–1150, 2020.

[5] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. In *Proc. International Conf. on Robotics and Automation (ICRA)*, pages 8248–8254, 2019.

[6] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926, 2021.

[7] Xinlei Pan, Yurong You, Ziyan Wang, and Cewu Lu. Virtual to real reinforcement learning for autonomous driving. *arXiv preprint arXiv:1704.03952*, 2017.

[8] Abhishek Kadian, Joanne Truong, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, Manolis Savva, Sonia Chernova, and Dhruv Batra. Sim2real predictivity: Does evaluation in simulation predict real-world performance? *IEEE Robotics and Automation Letters*, 5(4):6670–6677, 2020.

[9] Błażej Osiński, Adam Jakubowski, Paweł Zięcina, Piotr Miłoś, Christopher Galias, Silviu Homoceanu, and Henryk Michalewski. Simulation-based reinforcement learning for real-world autonomous driving. In *Proc. IEEE international Conf. on robotics and automation (ICRA)*, pages 6411–6418, 2020.

[10] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Proc. IEEE/RSJ international Conf. on intelligent robots and systems (IROS)*, pages 23–30, 2017.

[11] Wenshuai Zhao, Jorge Pena Queralta, Li Qingqing, and Tomi Westerlund. Towards closing the sim-to-real gap in collaborative multi-robot deep reinforcement learning. In *Proc. International Conf. on robotics and automation engineering (ICRAE)*, pages 7–12, 2020.

[12] Jingkang Wang, Yang Liu, and Bo Li. Reinforcement learning with perturbed rewards. In *Proc. AAAI Conf. on artificial intelligence*, pages 6202–6209, 2020.

[13] Zhuangdi Zhu, Kaixiang Lin, Anil K Jain, and Jiayu Zhou. Transfer learning in deep reinforcement learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[14] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proc. IEEE Conf. on computer vision and pattern recognition (CVPR)*, pages 3722–3731, 2017.

[15] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *Proc. International Conf. on machine learning (ICML)*, pages 1989–1998, 2018.

[16] Alex Bewley, Jessica Rigley, Yuxuan Liu, Jeffrey Hawke, Richard Shen, Vinh-Dieu Lam, and Alex Kendall. Learning to drive from simulation without real world labels. In *Proc. International Conf. on robotics and automation (ICRA)*, pages 4818–4824, 2019.

[17] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

[18] René Traoré, Hugo Caselles-Dupré, Timothée Lesort, Te Sun, Natalia Díaz-Rodríguez, and David Filliat. Continual reinforcement learning deployed in real-life using policy distillation and sim2real transfer. *arXiv preprint arXiv:1906.04452*, 2019.

[19] Corentin Sautier, Gilles Puy, Spyros Gidaris, Alexandre Boulch, Andrei Bursuc, and Renaud Marlet. Image-to-lidar self-supervised distillation for autonomous driving data. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 9891–9901, 2022.

[20] Divya Kothandaraman, Athira Nambiar, and Anurag Mittal. Domain adaptive knowledge distillation for driving scene semantic segmentation. In *Proc. IEEE/CVF Winter Conf. on Applications of Computer Vision (ICCV)*, pages 134–143, 2021.

[21] Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Proc. International Conf. on machine learning (ICML)*, volume 1, page 2, 2000.

[22] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

[23] Karol Arndt, Murtaza Hazara, Ali Ghadirzadeh, and Ville Kyrki. Meta reinforcement learning for sim-to-real domain adaptation. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, pages 2725–2731, 2020.

[24] Richard Liaw, Sanjay Krishnan, Animesh Garg, Daniel Crankshaw, Joseph E Gonzalez, and Ken Goldberg. Composing meta-policies for autonomous driving using hierarchical deep reinforcement learning. *arXiv preprint arXiv:1711.01503*, 2017.

[25] Chao Lu, Fengqing Hu, Dongpu Cao, Jianwei Gong, Yang Xing, and Zirui Li. Transfer learning for driver model adaptation in lane-changing scenarios using manifold alignment. *IEEE transactions on intelligent transportation systems*, 21(8):3281–3293, 2019.

[26] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *Proc. International Conf. on Machine Learning (ICML)*, pages 2817–2826, 2017.

[27] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Proc. Field and Service*, pages 621–635, 2018.

[28] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Proc. Conf. on robot learning*, pages 1–16, 2017.

[29] Fadri Furrer, Michael Burri, Markus Achtelik, and Roland Siegwart. Rotors—a modular gazebo mav simulator framework. *Robot Operating System (ROS) The Complete Reference*, 01:595–625, 2016.

[30] Alexander Amini, Tsun-Hsuan Wang, Igor Gilitschenski, Wilko Schwarting, Zhijian Liu, Song Han, Sertac Karaman, and Daniela Rus. Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles. In *Proc. International Conf. on Robotics and Automation (ICRA)*, pages 2419–2426, 2022.

[31] Seung Wook Kim, Jonah Philion, Antonio Torralba, and Sanja Fidler. Drivegan: Towards a controllable high-quality neural simulation. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5820–5829, 2021.

[32] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *Proc. IEEE Conf. on computer vision and pattern recognition (CVPR)*, pages 9068–9079, 2018.

[33] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Proc. European Conf. on Computer Vision (ECCV)*, 2020.

[34] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, et al. Nerfstudio: A modular framework for neural radiance field development. In *Proc. ACM SIGGRAPH Conf.*, pages 1–12, 2023.

[35] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 7210–7219, 2021.

[36] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 8248–8258, 2022.

[37] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 12922–12931, 2022.

[38] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022.

[39] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proc. IEEE/CVF International Conf. on Computer Vision (ICCV)*, pages 5752–5761, 2021.

[40] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5459–5469, 2022.

[41] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4):1–14, 2023.

[42] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf−−: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021.

[43] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *Proc. IEEE/CVF International Conf. on Computer Vision (ICCV)*, pages 5741–5751, 2021.

[44] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proc. IEEE/CVF*

International Conf. on Computer Vision (ICCV), pages 14335–14345, 2021.

[45] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proc. IEEE/CVF International Conf. on Computer Vision (ICCV)*, pages 5855–5864, 2021.

[46] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5470–5479, 2022.

[47] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. *arXiv preprint arXiv:2304.06706*, 2023.

[48] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987.

[49] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proc. Eurographics symposium on Geometry processing*, volume 7, page 0, 2006.

[50] Arunkumar Byravan, Jan Humplik, Leonard Hasenclever, Arthur Brussee, Francesco Nori, Tuomas Haarnoja, Ben Moran, Steven Bohez, Fereshteh Sadeghi, Bojan Vujatovic, et al. Nerf2real: Sim2real transfer of vision-guided bipedal motion skills using neural radiance fields. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, pages 9362–9369. IEEE, 2023.

[51] Allan Zhou, Moo Jin Kim, Lirui Wang, Pete Florence, and Chelsea Finn. Nerf in the palm of your hand: Corrective augmentation for robotics via novel-view synthesis. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 17907–17917, 2023.

[52] Michal Adamkiewicz, Timothy Chen, Adam Caccavale, Rachel Gardner, Preston Culbertson, Jeannette Bohg, and Mac Schwager. Vision-only robot navigation in a neural radiance world. *IEEE Robotics and Automation Letters*, 7(2):4606–4613, 2022.

[53] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proc. IEEE Conf. on computer vision and pattern recognition (CVPR)*, pages 4104–4113, 2016.

[54] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *Proc. European Conf. on Computer Vision (ECCV)*, pages 501–518, 2016.

[55] Unity technology. "unity engine". URL `https://unity.com`.

[56] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proc. International Conf. on machine learning*, pages 1861–1870, 2018.

[57] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *arXiv preprint arXiv:2311.12775*, 2023.

[58] NVIDIA. Isaac Sim - Robotics Simulation and Synthetic Data Generation. `https://developer.nvidia.com/isaac-sim`, 2024. (accessed on March 16, 2024).

[59] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science robotics*, 7(66):eabm6074, 2022.

[60] NVIDIA. Isaac Robot Operating System (ROS). `https://developer.nvidia.com/isaac-ros`, 2024. (accessed on March 16, 2024).

[61] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5481–5490, 2022.

[62] Yunzhu Li, Shuang Li, Vincent Sitzmann, Pulkit Agrawal, and Antonio Torralba. 3d neural scene representations for visuomotor control. In *Proc. Conf. on Robot Learning*, pages 112–123. PMLR, 2022.

[63] Danny Driess, Ingmar Schubert, Pete Florence, Yunzhu Li, and Marc Toussaint. Reinforcement learning with neural radiance fields. *Advances in Neural Information Processing Systems*, 35:16931–16945, 2022.

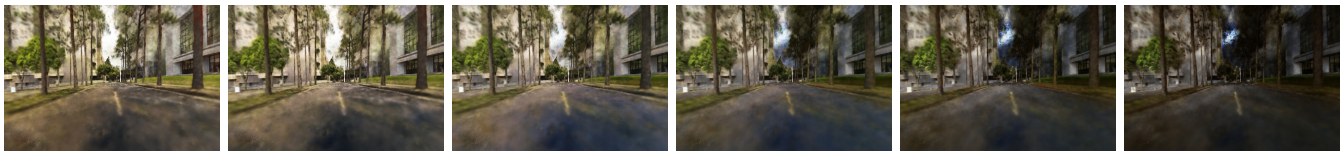
[64] Dongseok Shim, Seungjae Lee, and H Jin Kim. Snerl: Semantic-aware neural radiance fields for reinforcement learning. *arXiv preprint arXiv:2301.11520*, 2023.

[65] Zirui Wu, Tianyu Liu, Liyi Luo, Zhide Zhong, Jianteng Chen, Hongmin Xiao, Chao Hou, Haozhe Lou, Yuantao Chen, Runyi Yang, Yuxin Huang, Xiaoyu Ye, Zike Yan, Yongliang Shi, Yiyi Liao, and Hao Zhao. Mars: An instance-aware, modular and realistic simulator for autonomous driving. *CICAI*, 2023.

[66] Guohang Yan, Jiahao Pi, Jianfei Guo, Zhaotong Luo, Min Dou, Nianchen Deng, Qiusheng Huang, Daocheng Fu, Licheng Wen, Pinlong Cai, et al. Oasim: an open and adaptive simulator based on neural rendering for autonomous driving. *arXiv preprint arXiv:2402.03830*, 2024.

[67] Ze Yang, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel 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.

[68] Adam Tonderski, Carl Lindström, Georg Hess, William Ljungbergh, Lennart Svensson, and Christoffer Petersson. Neurad: Neural rendering for autonomous driving. *arXiv preprint arXiv:2311.15260*, 2023.



Day → Night

Fig. 5: DriveEnv-NeRF enables the interpolation of lighting conditions between two appearance embeddings, which allows generation of a spectrum from a daylight view to a nighttime view. This rendering capability, achieved through the use of appearance embedding, enriches the training and validation process for DRL policies by offering diverse and realistic scenes.

APPENDIX

A. An Extended Review of the Related Work

NeRF. Nerfstudio [34] is a modular framework for NeRF development that allows rapid exploration and implementation of new concepts. Within this framework, Nerfacto is the default model, which combines a number of state-of-the-art (SOTA) methods, namely Mip-NeRF 360 [46], Instant-NGP [38], NeRF-W [35], NeRF−− [42], Ref-NeRF[61], etc.NeRF [61], etc.

Training DRL with NeRF for Robotics. SPARTN [51] augments expert data in imitation learning scenarios by leveraging NeRFs to generate new viewpoints based on perturbed expert camera poses. Subsequently, the associated expert actions are adjusted based on the known camera-to-end-effector transform. The study in Adamkiewicz et al. [52] establishes a robot navigation pipeline based on a pre-trained NeRF for representing a 3D environment. Its trajectory optimization algorithm is designed to circumvent high-density regions of the pre-trained NeRF model, which ensures the generation of collision-free trajectories. In addition, a pose filtering mechanism leveraging NeRF's renderings is incorporated to estimate the robot's pose and velocities, which further improves the overall accuracy of the estimated robot state.

Apart from directly using the generated renderings, another line of research [62–64] explores latent-conditioned NeRF to train an encoder-decoder model that learns the 3D representations of a scene. The intermediate 3D representations generated by the encoder are utilized in downstream DRL tasks, which enhances the model's learning capabilities by incorporating pre-trained information. This 3D-aware neural implicit representation derived from NeRF empowers the DRL agent to generalize across diverse scene geometries. Nevertheless, these methods are orthogonal to this study.

NeRF-based Simulators. Concurrent to our study, several previous works [65–68] have attempted to build autonomous self-driving simulators using NeRF as a foundation. These approaches focus more on the development of realistic simulators for general datasets, including the insertion of dynamic objects and the generation of corner cases. However, our methodology diverges from these studies as our primary focus is on replicating specific target scenes from the real-world environment where the robot is intended to be deployed, rather than creating a general-purpose simulator.

B. Details of the Experimental Setups

Target Real Scenes. The experiments focus on two specific real-world environments: a straight road and a right turn, measuring 30 meters and 60 meters in length, respectively. Approximately 500 photographs were captured to train the NeRFs employed in our proposed pipeline. These images were taken under various weather conditions to ensure a diverse representation of environmental appearances in the NeRF models.

DRL Environment Setup. The DRL agent receives a survival reward of 0.1 at each timestep. On the other hand, a penalty of -4.0 is imposed for collision with obstacles, and -5.0 points are deducted for moving out of boundaries or exceeding the time limit. The observation of the agent consists of four stacked RGB frames, where each frame is individually rendered by NeRF with a resolution of 90 x 160.

NeRF Training Setup. In our experiments, we utilize Nerfacto [34] for the reconstruction of our targeted real-world scenes. The training process involves 200,000 iterations with a batch size of 4,096, and the training can be completed in approximately 10 minutes on an NVIDIA RTX Titan GPU.

Appearance Embedding. To generate appearance embeddings for evaluating and training our driving policy within the proposed framework, we sample a diverse range of lighting conditions by applying randomness to each dimension of the embedding. These embeddings are subsequently selected manually based on their ability to realistically represent various environmental lighting scenarios. The dimension of the appearance embedding used in the experiment is 32.

Real-World Experimental Setup. We deploy a trained driving policy on the Clearpath Husky UGV, which receives commands from a Jetson Xavier NX. The DRL model is executed on a laptop equipped with a GeForce RTX 4080 GPU, and the derived actions are forwarded to the Jetson Xavier NX, which transforms the action commands into control signals for the Husky UGV for real-time execution.

C. Extended Discussion and Conclusion

In addition to the potential future improvements on novel view synthesis methods using 3DGS [41], there is also room for enhancement in the deployment process of the driving agent by leveraging dedicated simulators such as Isaac Sim [58]. A software transition from ROS 1 to ROS 2 [59], further augmented by the integration of Isaac ROS [60], can facilitate improved real-time performance. Moreover, ROS

controller nodes can be encapsulated within containers to simplify deployment across diverse hardware platforms. To facilitate further exploration of our project, we've shared a portion of our implementation code in Isaac Sim and Isaac ROS on the project page. This page will be regularly updated as we develop more open-source implementations. In conclusion, DriveEnv-NeRF paves the way for cost-effective validation by facilitating the replication of real-world scenes and enhancing continuous testing of driving policies. Employing DriveEnv-NeRF in common real-world failure scenarios can strengthen the testing pipeline by faithfully mirroring these challenging real-world situations. Future research directed towards refining and expanding the capabilities of DriveEnv-NeRF to achieve a more precise alignment with real-world performance presents a promising avenue for further investigation and advancement in the field.