Study of ROS-Based Autonomous Vehicles in Snow-Covered Roads by Means of Behavioral Cloning Using 3DCoAutoSim

Yuzhou Liu[1], Walter Morales-Alvarez[1] Georg Novotny[1,2], and Cristina Olaverri-Monreal[1]

[1] Johannes Kepler University Linz; Chair Sustainable Transport Logistics 4.0, Altenberger Straße 69, 4040 Linz, Austria,
{yuzhou.liu, walter.morales_alvarez, georg.novotny, cristina.olaverri-monreal}@jku.at,
[2] UAS Technikum Wien, Hoechstaedtplatz 6, 1200 Vienna, Austria

Abstract. Autonomous driving in winter weather conditions has always been a unique challenge, and as such it is an interesting research topic. Due to reasons related to safety and local laws, simulators have become one of the first choice for the required research. This paper extends the capabilities of the 3DCoAutoSim simulation platform with a realistic simulation environment for the study of autonomous driving with ROS-controlled vehicles in adverse weather conditions such as snow-covered roads. The weather-related details of the environment such as snow fall and car tracks on the snow were implemented by using Unity3D's physics and graphics engine. A series of autonomous driving experiments based on behavioral cloning were performed to test the performance of the environment and its scalability for ROS-based machine learning applications. Results from the experiments conducted to validate the approach demonstrated a good driving performance. Moreover, results from the model trained with the data set generated in the snowy environment, showed that car tracks features in the snow promoted the learning and generalization steps in the machine learning process.

Keywords: Unity 3D, Driving Simulator, ROS, Autonomous Driving

1 Introduction

In autonomous driving scenarios, driving in winter weather conditions, with snow-covered roads, is particularly challenging because snow covers the visual features of the ground, in addition to the fact that any kind of driving in snow can also be more dangerous as it affects breaking and it can lead to accidents. At the same time, vehicles passing by will add a new feature to the snow-covered road, fresh car tracks, which can affect the visual features of the road and create issues with vehicle handling. In order to more conveniently study such problems, a simulator that can replicate the real world and simulate the laws of physics

is always one of the first choices for researchers. In the past few years, we have developed the flexible, modular tailored 3DCoAutoSim simulation tool to investigating the effect of automation and V2X communication on drivers [1]. The platform is linked to Unity3D, the traffic microscopic Simulation of Urban MObility (SUMO) and the Robot Operating System (ROS) [2], and it has been used to address several research questions such as interaction with pedestrians [3] or other vehicles on the road [4]. The excellent graphic performance and physics of Unity3D as the core part of the 3DCoAutoSim simulation platform makes it possible to address the tasks necessary for examining the issues mentioned above. Robot Operating System (ROS) is currently one of the most popular intelligent vehicle control systems. Its use is not only limited to robots in the narrowest sense of the word, such as delivery robots, but can also be used for self-driving cars and unmanned aerial vehicles. Many previous works have studied the simulation of robots combining ROS and Unity3D [5–7].

This work aims to extend the capabilities of the 3DCoAutoSim to create a realistic simulation environment for the study of autonomous driving with ROS-controlled vehicles in snow-covered roads. To this end we use Unity's particle system to create the snow falling effect, and use technologies such as tessellation, Unity RenderTexture, Unity Shader programming to achieve the function of leaving car tracks on the snow. In addition, we also design an architecture combined with ROS-based machine learning and vehicle control based on our previous works.

We then perform a series of autonomous driving experiments based on behavioral cloning and assess the performance of the environment and its scalability for ROS-based machine learning applications.

The remainder of this paper is organized as follows: Section 2 introduces the previous works in the field. The basic architecture and implementation method of the entire system as well as the experimental environment is presented in Section 3. Then, we describe the experiments performed to validate the whole design in Section 4. In Section 5, the results are presented and interpreted. Finally, Section 6 concludes the paper and presents possible future work.

2 Related Work

In previous work, we developed the 3DCoAutoSim simulation platform that links a driver-centric Unity3D-based simulator with the SUMO traffic simulator incorporating communication capabilities [8, 9]. It can also simulate ROS-based intelligent vehicles and robots [2, 10] by adapting and extending the ROSBridgeLib library [11]. Driving simulation platforms that replicate winter weather conditions often rely on replacing texture maps, such as the Develter [12] and CarSim simulator [13]. This process affects not only the level of realism, but also makes difficult to simulate complex snow trace changes, especially when there are multiple vehicles in the simulated environment, or a single vehicle repeatedly passes through a certain location. In the case of being the simulation platform intended for machine learning research aimed at autonomous driving, the inability of real-

time interaction and the dynamic simulation environment conditions will greatly reduce the generalization ability of the final machine learning model. Other techniques such as adding a map to make the snow look natural and replicate snow bump add a certain realism to the environment [14], however this approach is not able to reproduce the conditions to provide a real-time interaction that bases on snow traces. Unity3D can provide powerful physics and graphics effects, which are essential for the simulation of realistic driving conditions in snow-covered roads. In this context, they can be achieved by manipulating the plane meshes in real time. We rely on this approach and also use particle effects commonly used in game engines to replicate the effect of snowfall.

It is worth mentioning that Unity3D has already been used in machine learning research, and its development team has also added packages for implementing machine learning functions [15–17]. However, studies that combine its machine learning capabilities with intelligent vehicles are scarce and existing studies such as [18, 19] are rather theoretical and difficult to apply to practical situations due to the fact that existing vehicle control systems have not been considered. Our approach however, contributes to the state of the art by replicating drivers skills in the simulator by means of behavioral cloning, resulting thus the model trained on the data set generated in the simulation environment in a direct application for the control of real vehicles.

This behavioral cloning technique has been applied in the field of autonomous driving in several works [20][21]. In addition, the Gazebo advanced robotics simulator [22] is often used in the simulation research of ROS-based intelligent vehicles, and many teams have developed machine learning projects based on it [23–25].

We rely on [20] to implement our approach and refer to some of the Gazebo architecture designs to achieve the replication of snow-covered roads and be able to interact with ROS and the ROS-based machine learning module.

3 Methodology

This section describes the implementation of snow fall and the car tracks on snow. We additionally present the architecture of the 3DCoAutoSim extension, as well as some important components and parameters in the simulation, to finalize with the description of the simulator architecture combining ROS and the machine learning algorithms.

3.1 Simulation Implementation

To replicate roads covered with snow, we designed a mountain road that consisted of a two-way four-lane ring road, with fences and trees on the side of the road. The background was mainly mountains and a small number of buildings. Mountains, trees, terrain and part of the road surface were added with white textures to simulate the appearance of snow, which enhanced the sense of immersion and realism. The track was designed to include left and right turns to avoid biased data.

The effect of snowfall was achieved through the Unity's Particle System. The approach allowed 2D or 3D models to be released as particles within a range with certain rules and density. The trajectory and speed of the particles could also be controlled after they were released (Fig 1 shows the implemented scenario with snowfall).

To investigate whether and how the machine learning model would be affected after several laps of driving tests in the snow we developed a scenario with multiple car tracks as follows: We first applied a plane to the corresponding white texture and normal map to resemble its appearance to a snow ground. We then performed the tessellation process on the mesh of the plane to increase the triangles on the mesh, resulting this in an detailed representation of the snowy environment with bumps, hills and valleys. This technique made it possible to better replicate the snow ground plane to simulate the effect of the depression after the snow was compressed, as well as the folds of the snow. Next, we incorporated a camera and the corresponding RenderTexture to record the path of the wheel passing on the plane. To this end, we added the wheel with an additional Unity material (Shader), so that the camera could record its trajectory through the car body from above. Finally, we developed and executed a script to make the "meshes where the wheels pass" on the plane sink down to form a car track. Fig 2 shows the result. After this final step we covered the road in the entire simulation environment by the such planes.

Fig. 1. Illustration of the snowfall effect in the simulated environment

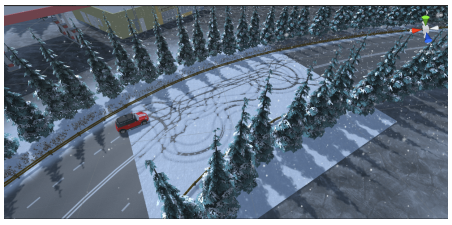

Fig. 2. Visualization of the vehicle tracks on the developed snow plane

3.2 Architecture and Settings

The basic architecture of the implemented 3DCoAutoSim extension is shown in Fig 3, and mainly consists of two parts. Unity is responsible for simulating the experimental scene and the vehicle itself and various physical effects. ROS is responsible for providing a stable middleware, while the neural networks (NNs) are responsible for learning and control. The neural network is built through the ROS Python API, so it actually runs under the ROS framework, therefore we can also consider it as a part of ROS.

Within the present work we aimed at investigating vision-based autonomous driving in snowy conditions. To this end we just required a color camera which

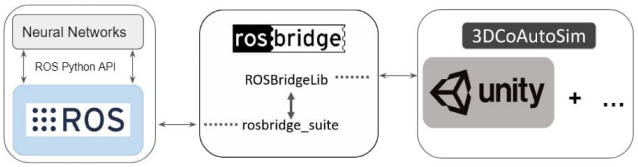

Fig. 3. Basic architecture of the implemented 3DCoAutoSim extension

we installed in the front of the simulated vehicle. Each frame of the camera image are cached by the Unity's RenderTexture and then sent to ROS with a frame rate of 15 fps, and a resolution of 128×96. Data such as the speed and the angle of the steering wheel, etc. can be directly read from the Unity system which connected with the hardware components. Further details regarding transmitting these data can be found in [2] and [10].

3.3 ROS and Behavioral Cloning

We used behavioral cloning (BC) to test the reliability of the 3DCoAutoSim simulation extension. Fig 4 depicts a high-level overview of the overall architecture of the BC model. Initially the car is operated by a human driver through the 3DCoAutoSim cockpit (steering wheel and the foot pedals for braking/accelerating).

The collected data points are divided into a training set and a validation set (ratio 8:2). An approximate time synchronization was afterwards implemented to correctly map the images with the prediction values. The data set was recorded by leveraging the rosbag, and cv_bridge ROS python API.

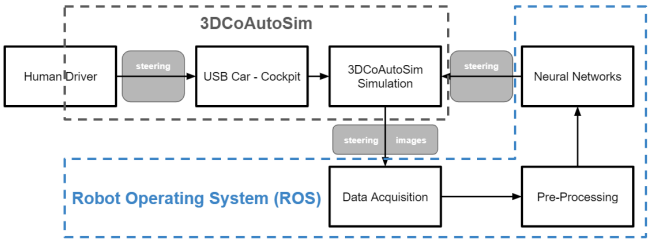

Fig. 4. High-level overview of the overall architecture of the behavioral cloning model

Pre-processing was applied by normalizing the image and re-scaling it to 66×200 pixels. The pre-processed images and the ground truth value for the steering angle were then utilized to build the base for the BC model. The trained model was next used as a control system for the vehicle inside the simulation environment by predicting the required steering angle. The priority of ROS control commands is lower than that of human control, so the driver can take over

autonomous driving at any time. To enable the model to imitate the human driving behavior in winter weather conditions, to smoothly perform autonomous driving, while also being able to stay in the corresponding lane, we followed the approach for data collection, training and testing described in the next section.

4 Experimental Setup and Data Analysis

4.1 Data Collection and Training

We invited several drivers to participate in the creation of the data set. They were required to drive as safely as possible, and to stay on lane. The road was covered by snow. We created a total of 35 minutes ROS bag, where a normal driving lap took about 2 minutes. The relevant parameters to their driving performance such as the angle of the steering wheel, the strength of the throttle, etc. were then sent to ROS together with the correspondent video records to build the data set. The data was initially recorded as ROS bag and saved for later transmission through ROSBridge Next, in order to transform the data into a more readable format, the images were extracted and a csv file containing the path to each image with the corresponding ground truth values was saved to the local hard drive.

We used video recordings and steering wheel angles for the training process. To this end, the throttle and the brake were set to a constant value or were controlled by the 3DCoAutoSim to be able to manually control the vehicle speed and verify the performance of the model for dealing with the lane keeping problem in the snowy environment at different speeds.

4.2 Experiments Design and Development

After the training phase was completed, we used the models to perform the pertinent tests and compare the experimental results of different situations. The first autonomous driving experiment was conducted on a road completely covered with snow. The acceleration and brake values were set as manually adjustable constants. The upper limit of the speed of the vehicle model was set to 50 km/h.

When the vehicle speed was low, although the lane could not be seen, the vehicle could still be stabilized in the correct lane range. This shows that the model could correctly imitate the driver's driving mode, keeping the vehicle within the corresponding lane. Fig 5 shows part of the car tracks left on the snow after the vehicle had driven several laps autonomously with low speed. The car trajectory overlapped almost completely with the previous tracks. It should be noted that when the experiment started, there were no car tracks on the snow.

Then, a second experiment was carried out in which a lot of car tracks were added in advance to simulate more complex snowy road conditions. Fig 6 shows that the vehicle could still maintain a centered position on the lane. To verify that the model did not completely ignore the road surface, we removed the snow.

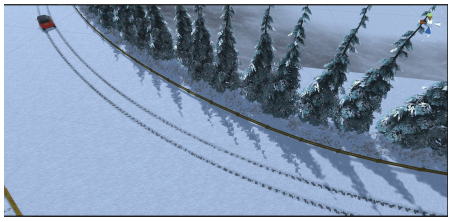

Fig. 5. Initial low-speed experiment showing that the car trajectory overlapped almost completely with the previous tracks on the snow

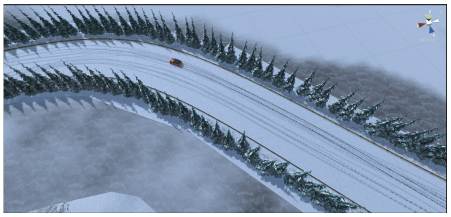

Fig. 6. Second low-speed test with previously added multiple car tracks. Results show the centered vehicle position on the lane.

In this case the autonomous driving model could no longer generate a reasonable output.

Next, the experiment was performed at a higher speed. The results showed that a vehicle could stay in the correct lane range most of the time. However, when turning at a larger angle, the trajectory of the vehicle shifted significantly.

This behavior seemed to be due to the model being unable to control the speed of the vehicle. When the turning angle was too large, the excessively fast speed caused the vehicle to deviate from the lane. We noticed that since the neural network we used only had a single image input and no storage capacity, it could not perceive or record the current state of the vehicle itself. Therefore, gas and brake inputs could not participate in training. In order to verify that the model could output an accurate steering wheel angle under a real-time controllable speed, we involved the real drivers again to drive cooperatively with the trained model. This time the drivers were requested to only control the accelerator and brake without steering. The limit of the vehicle's speed was canceled so that the speed could be freely controlled.

4.3 Data Analysis

To evaluate the results of the above experiments, we recorded and compared the positions of the vehicle in the four sets of experiments with regards to deviation from the center of the lane. The distances from the center were calculated by using linear interpolation to synchronize the number of sample points and calculate the absolute value between each pair of samples.

5 Results

The average and maximum deviation of the vehicles in each group of experiments relative to the center of the lane is quantified and listed in Table 1. Since the width of the lane is about 3.5 m, and the virtual vehicle model used in the experiment is about 1.73 m wide, the experimental results showed that the vehicle could maintain a centered position on the lane under low speed and semi auto conditions.

Table 1. Comparison of the lateral deviation from the center of the lane in the 4 performed experimental conditions

	Maximum distance [m]	Average distance [m]
Initial Experiment	2.28	0.97
Second Experiment	2.28	0.97
With Higher Speed	3.99	1.51
Semi Auto	2.47	1.16

Results from the initial and semi auto experimental showed a good model performance as the trajectories overlapped (Fig 7 and Fig 8). As previously mentioned, at higher speeds however, the vehicle deviated from the center of the lane (depicted by the green dashed line in Fig 7). The purple trajectory depicted in Fig 8, represents the results of the low-speed experiment with several vehicle tracks, which is visualized in Fig 6. In this case the purple and red lines (initial experiment) overlap almost everywhere, showing that the model performs well in different snow conditions.

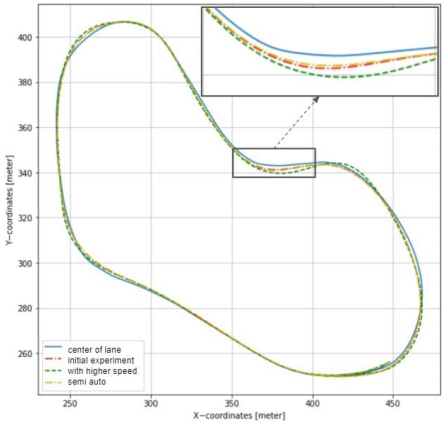

Fig. 7. Trajectories' deviation from the center of the lane at different speeds

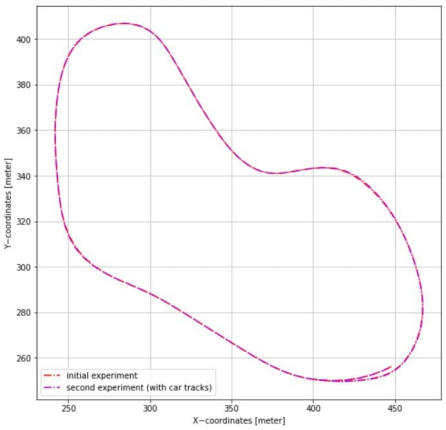

Fig. 8. Visualization of the impact of different car tracks on snow

6 Conclusion and Future Work

In this work we extended the 3DCoAutoSim simulation platform to create a realistic simulation environment for the study of autonomous driving with ROS-controlled vehicles under snowy weather conditions. Behavioral cloning was used to assess the approach. We carried out multiple experiments in different snowy environments, and analyzed the impact on the performance of the newly implemented modules. Experimental results showed that the single visual input model

makes it possible for the vehicle to maintain its position on the lane in snow-covered roads if vehicle speed is kept within a certain range. We showed that car tracks features in the snow promoted the learning and generalization steps in the machine learning process and therefore conclude that the proposed approach is appropriate to develop simulations under snowy conditions. In future work, light snow will be added to the system, as it may lead to false conclusions such as the existence of traffic congestion [26]. More complex scenarios such as more diverse surroundings, other vehicles, etc. will also be investigated additionally to reinforcement learning capabilities.

ACKNOWLEDGMENT

This work was supported by the Austrian Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK) Endowed Professorship for Sustainable Transport Logistics 4.0., IAV France S.A.S.U., IAV GmbH, Austrian Post AG and the UAS Technikum Wien.

References

1. Hussein, A., Díaz-Álvarez, A., Armingol, J.M., Olaverri-Monreal, C.: 3dcoautosim: Simulator for cooperative adas and automated vehicles. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC). pp. 3014–3019. IEEE (2018)
2. Hussein, A., García, F., Olaverri-Monreal, C.: Ros and unity based framework for intelligent vehicles control and simulation. In: 2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES). pp. 1–6. IEEE (2018)
3. Artal-Villa, L., Olaverri-Monreal, C.: Vehicle-pedestrian interaction in sumo and unity3d. In: World Conference on Information Systems and Technologies. pp. 198–207. Springer (2019)
4. Smirnov, N., Liu, Y., Validi, A., Morales-Alvarez, W., Olaverri-Monreal, C.: A game theory-based approach for modeling autonomous vehicle behavior in congested, urban lane-changing scenarios. Sensors 21(4), 1523 (2021)
5. Meng, W., Hu, Y., Lin, J., Lin, F., Teo, R.: Ros+ unity: An efficient high-fidelity 3d multi-uav navigation and control simulator in gps-denied environments. In: IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society. pp. 002562–002567. IEEE (2015)
6. Hu, Y., Meng, W.: Rosunitysim: Development and experimentation of a real-time simulator for multi-unmanned aerial vehicle local planning. Simulation 92(10), 931–944 (2016)
7. Katara, P., Khanna, M., Nagar, H., Panaiyappan, A.: Open source simulator for unmanned underwater vehicles using ros and unity3d. In: 2019 IEEE Underwater Technology (UT). pp. 1–7. IEEE (2019)
8. Michaeler, F., Olaverri-Monreal, C.: 3D Driving Simulator with VANET capabilities to assess cooperative systems: 3DSimVanet. In: 2017 IEEE Intelligent Vehicles Symposium (IV). pp. 999–1004. IEEE (2017)
9. Validi, A., Olaverri-Monreal, C.: Simulation-based impact of connected vehicles in platooning mode on travel time, emissions and fuel consumption. arXiv preprint arXiv:2105.10894 (2021)

10. Liu, Y., Novotny, G., Smirnov, N., Morales-Alvarez, W., Olaverri-Monreal, C.: Mobile delivery robots: Mixed reality-based simulation relying on ros and unity 3d. In: 2020 IEEE Intelligent Vehicles Symposium (IV). pp. 15–20. IEEE (2020)
11. Thorstensen, M.C.: Ros bridge lib ([Last Accessed: 2019-03-31]), https://github.com/MathiasCiarlo/ROSBridgeLib
12. Develter: Simulator and driving on snow, [Last Accessed: 2021-11-3]. [Online] Available: http://www.develter.com/simulator-and-driving-on-snow-_l_EN_r_32_a_36.html
13. CarSim: Mechanical simulation, [Last Accessed: 2021-11-3]. [Online] Available: https://www.carsim.com
14. Carla: Open-source simulator for autonomous driving research, [Last Accessed: 2021-10-3]. [Online] Available: https://carla.org
15. Sallab, A.E., Abdou, M., Perot, E., Yogamani, S.: Deep reinforcement learning framework for autonomous driving. Electronic Imaging 2017(19), 70–76 (2017)
16. Youssef, A.E., El Missiry, S., El-gaafary, I.N., ElMosalami, J.S., Awad, K.M., Yasser, K.: Building your kingdom imitation learning for a custom gameplay using unity ml-agents. In: 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON). pp. 0509–0514. IEEE (2019)
17. Cao, Z., Wong, K., Bai, Q., Lin, C.T.: Hierarchical and non-hierarchical multi-agent interactions based on unity reinforcement learning. In: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems. pp. 2095–2097 (2020)
18. Hossain, S., Lee, D.J.: Autonomous-driving vehicle learning environments using unity real-time engine and end-to-end cnn approach. The Journal of Korea Robotics Society 14(2), 122–130 (2019)
19. Li, X., Cao, Z., Bai, Q.: A novel mountain driving unity simulated environment for autonomous vehicles. In: 35th AAAI Conference on Artificial Intelligence (AAAI) (2021)
20. Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., et al.: End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316 (2016)
21. Farag, W., Saleh, Z.: Behavior cloning for autonomous driving using convolutional neural networks. In: 2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT). pp. 1–7 (2018)
22. Koenig, N., Howard, A.: Design and use paradigms for gazebo, an open-source multi-robot simulator. In: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566). vol. 3, pp. 2149–2154 vol.3 (2004)
23. Zamora, I., Lopez, N.G., Vilches, V.M., Cordero, A.H.: Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo. arXiv preprint arXiv:1608.05742 (2016)
24. Lopez, N.G., Nuin, Y.L.E., Moral, E.B., Juan, L.U.S., Rueda, A.S., Vilches, V.M., Kojcev, R.: gym-gazebo2, a toolkit for reinforcement learning using ros 2 and gazebo. arXiv preprint arXiv:1903.06278 (2019)
25. Nuin, Y.L.E., Lopez, N.G., Moral, E.B., Juan, L.U.S., Rueda, A.S., Vilches, V.M., Kojcev, R.: Ros2learn: a reinforcement learning framework for ros 2. arXiv preprint arXiv:1903.06282 (2019)
26. Wiseman, Y.: Real-time monitoring of traffic congestions. In: 2017 IEEE International Conference on Electro Information Technology (EIT). pp. 501–505. IEEE (2017)