Hindawi
Journal of Robotics
Volume 2022, Article ID 9916292, 10 pages
https://doi.org/10.1155/2022/9916292

Research Article

Sim-to-Real Reinforcement Learning for Autonomous Driving Using Pseudosegmentation Labeling and Dynamic Calibration

Jiseong Heo and Hyoung woo Lim

Agency for Defense Development, Daejeon, Republic of Korea

Correspondence should be addressed to Hyoung woo Lim; hwlim@add.re.kr

Received 17 January 2022; Accepted 31 May 2022; Published 26 June 2022

Academic Editor: Keigo Watanabe

Applying reinforcement learning algorithms to autonomous driving is difficult because of mismatches between the simulation in which the algorithm was trained and the real world. To address this problem, data from global navigation satellite systems and inertial navigation systems (GNSS/INS) were used to gather pseudolabels for semantic segmentation. A very simple dynamics model was used as a simulator, and dynamic parameters were obtained from the linear regression of manual driving records. Segmentation and a dynamic calibration method were found to be effective in easing the transition from a simulation to the real world. Pseudosegmentation labels are found to be more suitable for reinforcement learning models. We conducted tests on the efficacy of our proposed method, and a vehicle using the proposed system successfully drove on an unpaved track for approximately 1.8 km at an average speed of 26.57 km/h without incident.

1. Introduction

Due to the improvements in deep convolutional neural network architectures and graphical processing units, recent research has aimed at applying deep learning algorithms to autonomous driving tasks. Previously, traditional computer vision algorithms, including edge detection and template matching, were used to infer how the vehicle should drive. Researchers utilized deep learning methods, including convolutional neural networks (CNNs) to leverage their complicated features and enable the self-driving algorithm to behave more intelligently.

Imitation learning has been a common approach for autonomous driving [1]. During imitation learning, a CNN is trained to learn human-like control from a given image and features. However, there are some drawbacks associated with imitation learning. First, imitation learning cannot encompass the very diverse number of possible cases that can occur in association with driving. Additionally, imitation learning requires large amounts of labeled data for training, which must be collected from actual driving environments and is therefore cost-ineffective and labor-intensive.

Off-road driving environments are quite different from road driving environments, wherein the path is kept much more standard and uniform. In off-road environments, there exists much more variability, such as grass growing in the middle of the road, road curvature changes due to seasonal factors like rain and snow, and even color changes in the road itself before and after rain. Autonomous driving based on images in off-road environments is inevitably affected by these various disturbances. To address this problem, we require the ability to generalize off-road driving environments, and we also need a robust algorithm that can make the best choice in a wide variety of driving situations.

In contrast, reinforcement learning (RL) has several advantages over imitation learning. Agents can learn how to drive over many trials in a simulation, and they can be trained from a near-infinite number of possible cases without the need for labeled data. Moreover, reinforcement learning has the potential to outperform human drivers because the driving performance of systems trained through reinforcement learning is not limited by the training dataset.

Nevertheless, deploying reinforcement learning in a real vehicle remains challenging, in part because of distribution

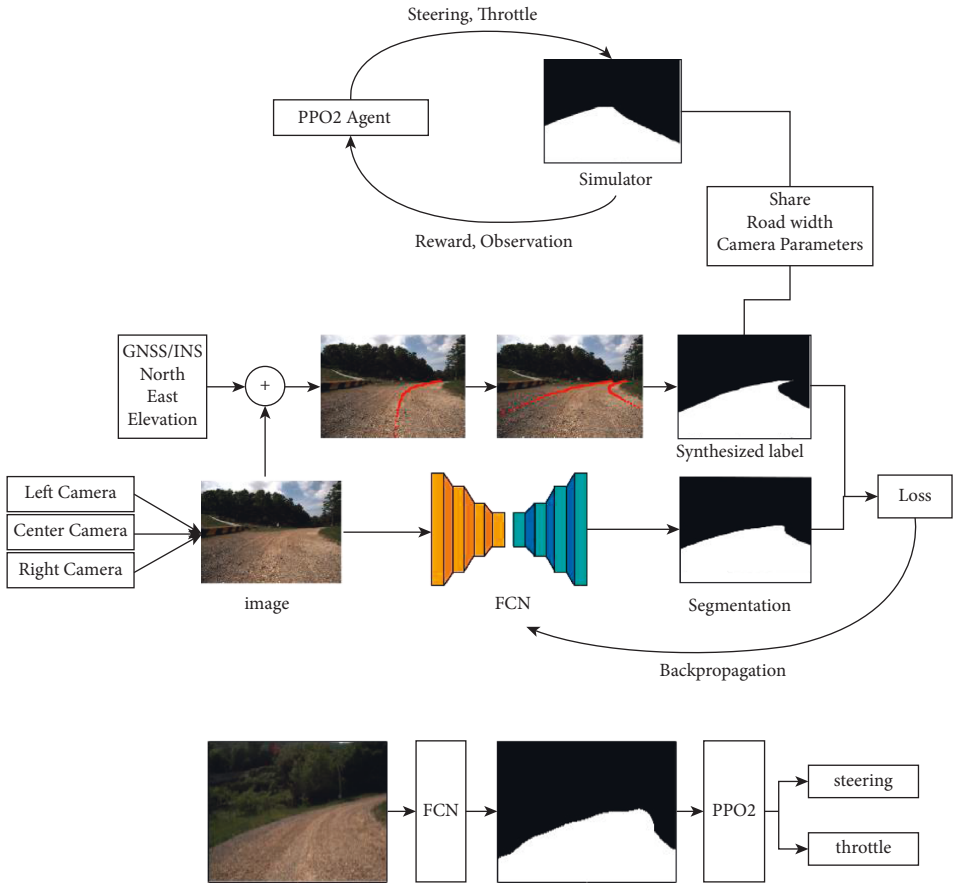

Figure 1: Overall architecture of our method.

shifts between the simulations in which they are trained and the real world [2]. Distribution shift is one of the main reasons why a trained model might perform poorly in a real-world test environment. Covariate shift, specifically, refers to the difference between trained input data and testing input data [3]. Because a simulation cannot perfectly reconstruct the real world, there are often mismatches between the real and simulation scenes, which results in negative effects on a model's driving performance [4]. Alternatively, concept shift refers to the difference in the relationships between labels and their given inputs [2]. For example, the correct control command for a given image might be different in a simulation and in the real world because of the differences in dynamics of the two environments.

The covariate shift between a simulator and the real world can be relieved using intermediate representations of the input [5]. For example, two-class semantic segmentation narrows the gap between the simulator and the real world. For this to work, simulators can easily produce binary images. In the real world, images can be processed into binary images by semantic segmentation networks. Using these binary images instead of the raw images can help reduce covariance shift.

In this article, dynamic calibration was used to enable a trained model to behave similarly in both simulations and the real world. A simple vehicle dynamics model was used, and only four parameters were fine-tuned using linear regression to mediate problems that may occur from distribution shift, covariate shift, and dynamics. The experiments conducted in this article demonstrated that this simple method was effective in reducing the concept shift between the simulation and real environment. However, modeling the complicated car dynamics in a way that considers the many relevant parameters requires significant computational effort and can often not be generalized to other types of vehicles, our simple and data-driven approach can be used on different devices and vehicles with little modification.

The overall architecture of the proposed method, and in particular its training phase, is illustrated in Figure 1. There are two parts of the training phase. The first is for training the semantic segmentation network with synthesized labels from the global navigation satellite systems (GNSS) and inertial navigation systems (INS) data. The second is to train the RL model using our calibrated simulator. Our simulator and synthesized labels share the same road width and camera parameters, including focal length, camera height, and tilt angle.

Figure 2 shows the testing phase of our method, wherein semantic segmentation and RL model inference are conducted sequentially. The steering and throttle values produced by the RL model are then passed to the control system of the test vehicle.

In this article, our main contributions are as follows:

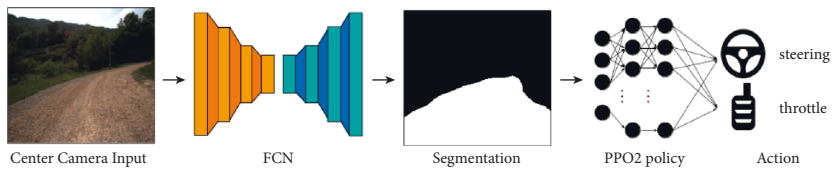

Figure 2: Pipeline of our method used in inference mode.

(i) Reconstruction of pseudoroad segmentation labels from GNSS/INS records

(ii) Simulator dynamic calibration through the linear regression of actual driving records and execution of the RL model

(iii) Driving test of our algorithm in a real vehicle and road, demonstrating its efficacy

2. Related Works

Autonomous driving has become a key research area in the field of artificial intelligence. Pomerleau [6] introduced ALVINN, a military vehicle driven by an algorithm, and demonstrated that it could successfully drive on paved roads. In 2016, Bojarski et al. [1] applied and achieved end-to-end learning of a self-driving task using a convolutional network. Their model had five convolutional layers and two fully connected layers to output throttle and steering control. The authors demonstrated that each of the convolutional filters was able to successfully recognize the edges of roads without explicitly being provided that information.

Semantic segmentation is a traditional computer vision task that classifies every pixel in a given image. Long et al. [7] used fully convolutional networks (FCNs) and skip connections to leverage both coarse and fine-grained features from the image for semantic segmentation. They achieved state-of-the-art performance on several segmentation challenges, including PASCAL-VOC [8] and SIFT flows. Chen et al. [9] proposed DeepLab, which uses Atrous convolution and conditional random fields to increase the performance of semantic segmentation, and achieved state-of-the-art performance at PASCAL-VOC-2012 [8].

Reinforcement learning is one of the most important branches of artificial intelligence. Mnih et al. [10] applied deep convolutional networks to reinforcement learning to allow an agent to play Atari games. Their trained model outperformed humans for most of the games it played, and the authors showed that only a few consecutive images were necessary to train the reinforcement learning algorithm. Deep reinforcement learning has also been applied to self-driving to handle various scenarios, which cannot be solved with traditional rule-based algorithms [11–13].

Sim-to-real transfer, lastly, regards transferring a model that was trained in a virtual environment to the real world. Domain randomization is one of the representative techniques of sim-to-real. It randomizes various property of the inputs, including brightness, contrast, and dynamics, to allow a trained model to consider the real-world input as just one of the randomized simulation data. Researchers have applied domain randomization techniques to make vehicles trained in simulations perform well in the real world [14–17].

3. Method

3.1. Semantic Segmentation Using GNSS/INS. For the semantic segmentation of the road area, a fully convolutional network (FCN) [7] was trained with images and labels. As semantic segmentation is trained using supervised learning, segmentation labels are necessary for each corresponding raw image. To gather the tremendous number of necessary segmentation labels, GNSS/INS data were utilized to synthesize pseudosegmentation labels, rather than rely on human labor.

GNSS/INS data can accurately measure the current location of a vehicle, localizing its position within an error of 0.40 meters at 20 Hz. As a vehicle drives around a track, its location data, including longitude and latitude, are recorded for every frame. Each location component is estimated in meters.

Figure 3 shows how the segmentation labels are produced from GNSS/INS data. The left image shows the location points projected on a camera input, which is taken at one of the recorded locations. The image in the middle shows the lateral points of each location points. They are predefined distance away from its corresponding location point. Lastly, the road segmentation label is produced by polygons composed of lateral points of each location point.

An FCN [7] with ResNet50 [18] backbone was used as the semantic segmentation network. The network consists of 57 layers, which takes 224×224 sized RGB image (3 channels) as an input and produces an output tensor with the shape of $21 \times 224 \times 224$. We used the default number of classes, which is 21, to load the pretrained weights of FCN. The training dataset consisted of two types of labeled datasets. The first was the dataset with synthesized segmentation labels from the GNSS/INS records. The other contained labels produced by humans. The number of labels in the two datasets was 6,492 and 969, respectively. The model was trained for 100 epochs, with a learning rate of 0.001. Adam was used as the optimizer during training.

3.2. Simulator. The simulator was designed to simulate real-world situations. The global map in the simulator was reconstructed as a $10{,}000 \times 10{,}000$ array filled with zeros. The global map was derived through accumulating the trajectory positions driven by an expert driver. Figure 4 shows the global map of the simulator and one of the rendered images.

The car dynamics in the simulator were designed to be as simple as possible. In the simulator, the car was considered a

Figure 3: Segmentation labeling process using GNSS/INS data. The left image shows the location points projected on the corresponding image. The center image shows the lateral points of each location. The right image shows a segmentation label constructed from those lateral points.

Figure 4: Simulator preview. The left image shows the global map of our simulator. The right image is a rendered image taken from the camera's point of view.

point with a heading vector. The movements depend on the steering and throttle inputs. Steering $s \in [0, 1]$ determines the change in the heading angle ($\Delta\theta$), and the throttle $t \in [0, 1]$ determines the distance advanced along the heading vector (Δn). We assume these variables have linear correlations between them. Thus, those relationships can be modeled as the following equations. The weights (w_s, w_t) and biases (b_s, b_t) in the equations are parameters optimized by linear regression:

$$\begin{aligned} \Delta\theta &= w_s * s + b_s, \\ \Delta n &= w_t * t + b_t. \end{aligned} \tag{1}$$

The outputs from RL models often show unrealistic actions. For example, the steering values from an RL model may rapidly change between the maximum and minimum values, or the trembling of steering can help an agent to maximize rewards within the learning paradigm. However, these unrealistic movements can often cause a catastrophic breakdown of wheel motors in the real world. For this reason, the maximum steering change was limited to M_s, which is obtained from actual driving data.

3.3. Reinforcement Learning. Proximal Policy Optimization (PPO2) [19] with Multilayer Perceptron Policy (MlpPolicy) was used as the reinforcement learning model. MlpPolicy consists of two layers, with 64 features each. The depth of the policy network is shallow, and the number of features is significantly lower than many modern CNN networks. However, MlpPolicy is still sufficient to train tasks where inputs are simple and state transitions are very consistent. Our simulator provides binary images of roads and uses simple dynamics. Moreover, by using complicated and deep networks can cause an overfitting problem, which is critical for sim-to-real transfer.

Observation refers to the data that are input to an RL model. The simulator provides binary images of roads from the camera's point of view. To compress the input image to a feature vector, the lengths of ten evenly spaced vertical lines were taken as observations. Figure 5 visualizes the observation from the perspective of the simulation. In addition, to provide the agent with temporal information, the previous (i.e., historical) steering and throttle values were added to the observation. Therefore, the total number of observation features was 12.

The objective of reinforcement learning is to maximize the rewards. The reward function is composed of four types of reward. We describe each reward in detail:

$$\begin{aligned} R &= R_t + P_i + R_e + P_c, \\ R_t &= \lambda_t T. \end{aligned} \tag{2}$$

Here, R_t is the throttle reward. The throttle reward induces the vehicle to move forward. T refers to the throttle value. λ_t is the weight of the throttle reward, which can be empirically determined. The imbalance penalty is given as

$$P_i = -\frac{|l - r|}{l + r}. \tag{3}$$

Here, P_i is the imbalance penalty, which measures how close the vehicle is to the center of a road. l is the distance of the left side to the road boundary from the car, and r is the distance from the opposite direction. If the vehicle gets closer to one of the road boundaries, the penalty increases. The imbalance penalty was found to be useful in preventing vehicles from driving in zigzags and encouraged a straighter path:

$$R_e = \begin{cases} \dfrac{1000}{N}, & \text{if } c \notin V, \\ 0, & \text{else.} \end{cases} \tag{4}$$

Here, R_e is the exploration reward that induces an agent to visit an unseen area. It outputs $1000/N$ when the car reaches a new track tile [20], where N is the total number of location points used to build the global map. Our simulator determines that the agent arrives at an unvisited point only when the current closest location point c is not included in a

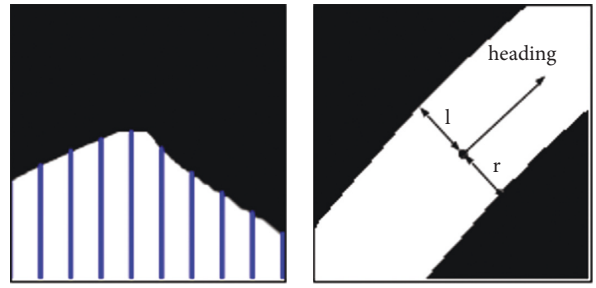

Figure 5: (a) Visualization of line segments for constructing the observation, with the lengths of 10 lines taken as an observation. (b) Visualization of l and r for calculating the imbalance penalty.

Table 1: Hardware specification used in experiments.

GPU	GTX 2080 Ti
Camera field of view (v)	66.9
Camera field of view (h)	82.4
Total road length	1.8 km
Camera height	1.4 m
Average road width	8 m
Vehicle width	2.5 m
Image resolution	224×224
Action frequency	10 Hz
Camera tilt degree	10 degrees
GNSS/INS error	<0.40 m
GNSS/INS frequency	>20 Hz
Weight of the vehicle	6,480 kg

visited point set V. This reward prevents the vehicle from continuously driving along a small circle. Lastly, the crash penalty is given as

$$P_c = \begin{cases} -\lambda_c T, & \text{if } l = 0 \text{ or } r = 0, \\ 0, & \text{else.} \end{cases} \quad (5)$$

Here, P_c is the crash penalty that an agent receives whenever the vehicle touches the edges of the road. P_c is proportional to the throttle value.

3.4. Experimental Settings. The total distance of the test road was 1.8 km. The road was unpaved and covered with gravel and dirt. The average road width was approximately 8 m. The boundaries of the road were not clear, and there were grasses and trees outside the road. The height of the camera attached to the vehicle was approximately 1.4 m from the ground. The test vehicle had six wheels and was a skid-type vehicle, which can reach speeds of up to 50 km/h. Table 1 depicts the details of the experimental setup.

4. Results and Analysis

To gather driving data, an expert driver drove along the whole course of the test road. At each frame, the information of the vehicle including steering, throttle, heading angle, and position coordinates is recorded. We obtain $\Delta\theta$ by calculating the difference between heading angle of the recorded two consecutive frames. Also, Δn is computed by projecting

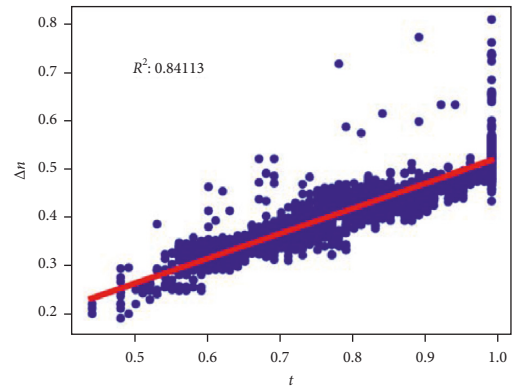

Figure 6: Results of linear regression between throttle and Δn, which is the distance advanced toward the heading direction.

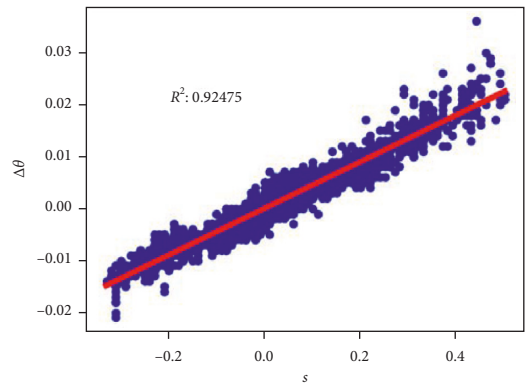

Figure 7: Results of linear regression between steering and $\Delta\theta$, which is the change of heading angle.

the difference between the two consecutive GNSS/INS positions to heading angle vector of the vehicle.

Figure 6 shows the scattered blue points, which represent the throttle and Δn pair recorded at each moment. Likewise, the blue points in Figure 7 denote the pairs of steering and $\Delta\theta$ recorded at each frame. The red lines in Figures 6 and 7 visualize the results of linear regression conducted by the least squared error method. Table 2 shows the calibrated value of the model parameters w_s, b_s, w_t, and b_t, which were

Table 2: Results of dynamic calibration.

Parameter	Regression result
w_s	0.04495
b_s	$1.25525e-05$
w_t	0.51856
b_t	0.0022277

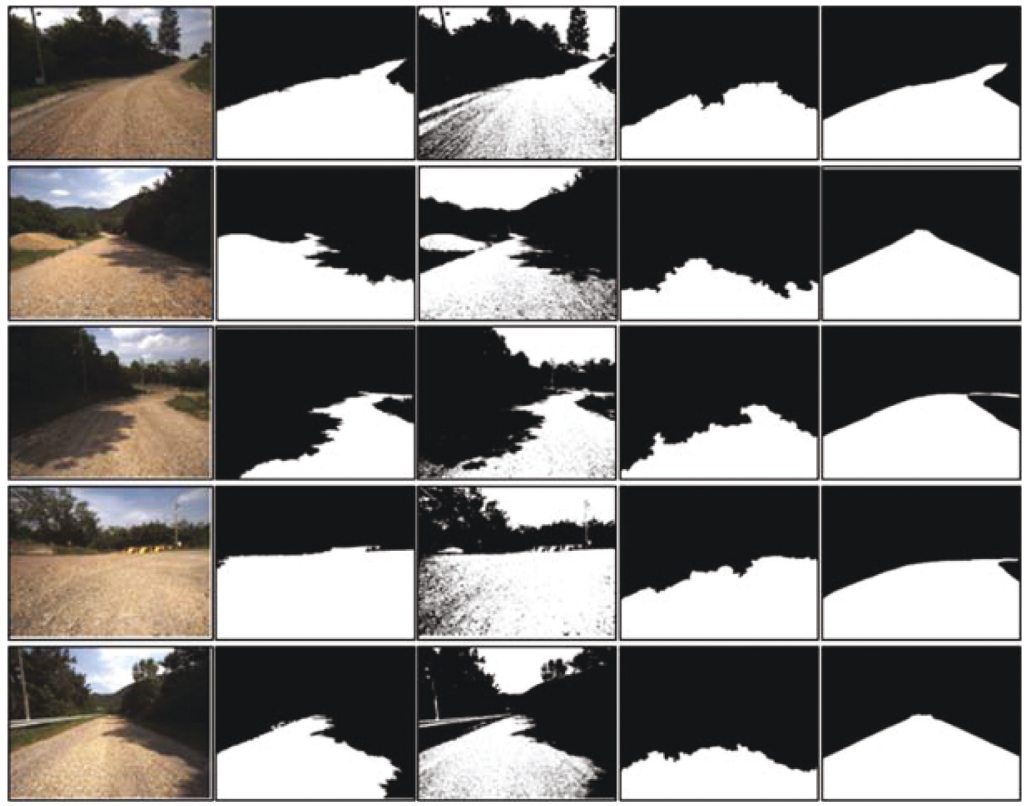

Figure 8: Qualitative results of various image processing methods for road segmentation.

obtained from linear regression. Our simulator used this model and these parameters to mimic the real-world dynamics.

Several pseudosegmentation labeling methods were implemented and compared in the present study. SLIC [21] and Watershed [22] are methods based on superpixel algorithms. The threshold method filters pixels whose values are below a threshold. The specific threshold was determined using the method from Otsu [23].

According to Figure 8, the SLIC algorithm appears to be the most promising method, relative to our own. However, the SLIC method is vulnerable to discrepancies resulting from shade. Similarly, the threshold method produces noisy labels and misclassifies the sky as part of the road. Lastly, Watershed barely provided any useful segmentation labels.

Two methods have recently been published for pseudo-semantic segmentation labeling [24, 25]. Those methods use class activation maps from GradCAM [26]. Unfortunately, the activation maps of our road images were not suitable for obtaining the road areas, because these pretrained classification models classify roads as a part of the background.

The first row of Figure 9 shows the input images. The last two rows are the segmented images that were inferred from the two different models. The first model was trained using the ground-truth labels of the test road, whereas the second model was trained using the synthesized pseudolabels.

Both Figures 9(b) and 9(c) show reasonable segmentation performance. The intersection over union (IoU) was higher in (b), except for the three rightmost columns. (c) often predicted a road area that was narrower than the ground truth because the road width of the synthesized labels was fixed at 6 m. The road widths of the synthesized labels and simulator were the same. Thus, our model can produce segmentation images that is more similar to the simulation scenes. The input to our RL model was in the form of the lengths of 10 lines in a segmentation image. The Kullback–Leibler divergence was calculated to compare the similarity between the distributions in simulator observations and the observations from the segmentation models. To calculate the KL divergence, histograms of each line length were generated. The formula is as follows:

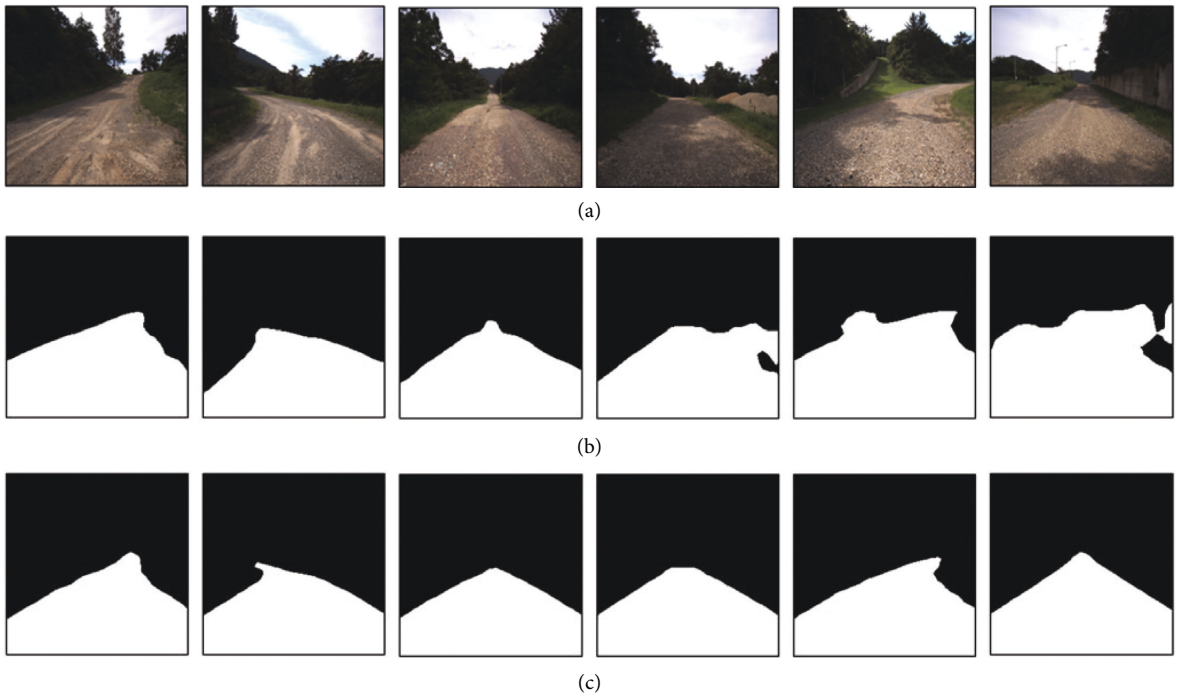

Figure 9: (a) Camera input image. (b) Inferenced segmentation results from the model trained with the ground-truth labels. (c) Segmentation results from the model trained with pseudolabels.

$$\mathrm{KL} = \Sigma p_i(x) \log \frac{p_i(x)}{q_i(x)}. \tag{6}$$

where $p_i(x)$ is the value of the i-th bin in the histogram of a segmentation model outputs and $q_i(x)$ represents that of the simulator outputs. The KL divergence of each line is shown in Figure 10. According to Figure 10, the KL divergences were lower in our segmentation model for each line, which implies that our model produces much more similar output to the simulator scenes than the comparator models. The average KL divergences for the model trained with ground truths and our model were 1.5450 and 0.42644, respectively. Therefore, our pseudosegmentation labeling algorithm significantly reduced the covariate shift between the simulator and the real world.

To compare the suitability of the segmentation outputs from these models, a dataset collected through actual human driving was used. The dataset contains images and corresponding values of the throttle and steering. The images were processed by both segmentation models, and the RL model produced values of steering and throttle from the segmentation images. The steering values were compared with the values from the dataset, which are representative of actual human decisions.

In Figure 11, the blue lines represent the visualization of the steering values from the dataset collected from manual driving. The above orange line shows the steering outputs from segmentation model trained by the ground-truth labels. The below orange line represents the steering values obtained from our model, which is trained with pseudolabels. From the figure, it is clear that our segmentation model is more suitable for use in

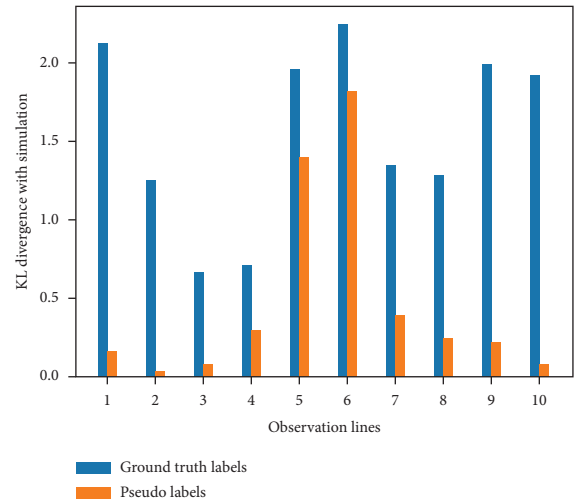

Figure 10: Comparison by KL divergence. Our segmentation model output results that are more similar to the simulation than does the model trained by the ground-truth dataset.

the RL model than the model trained from the ground truth. It is remarkable that the RL model behaved similarly to human driving without requiring any steering or throttle data.

In the real environment experiment, our model was deployed in the test vehicle. Figure 12 shows the trajectory of our model and the trajectory from human driving. Our model drove around the entire track without crashing, driving at an average speed of 26.57 km/h. The minimum, maximum, and the speed during the 270-degree hairpin curve were 23.2 km/h, 28.7 km/h, and 23.4 km/h, respectively.

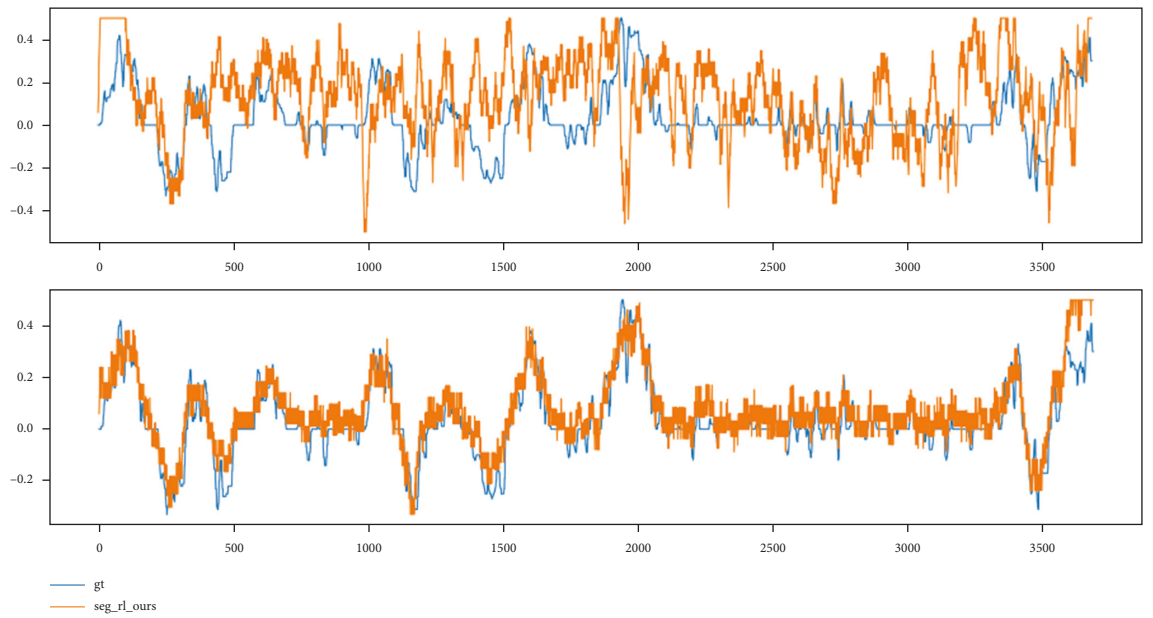

Figure 11: Steering recorded by human decisions (blue) and our algorithm (orange). Our method is scaled down to half for better comparison.

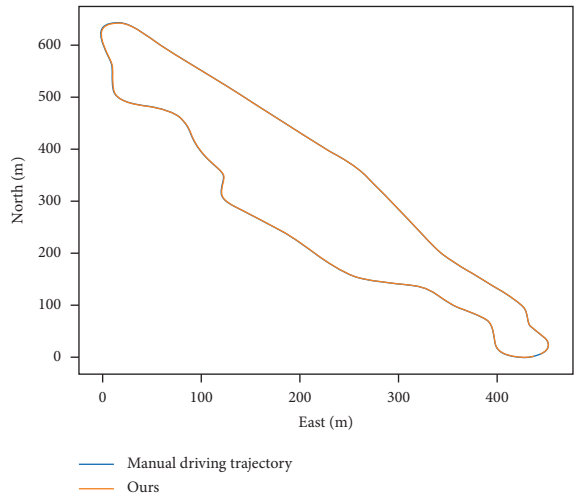

Figure 12: Recorded locations of the vehicle during the real environment test.

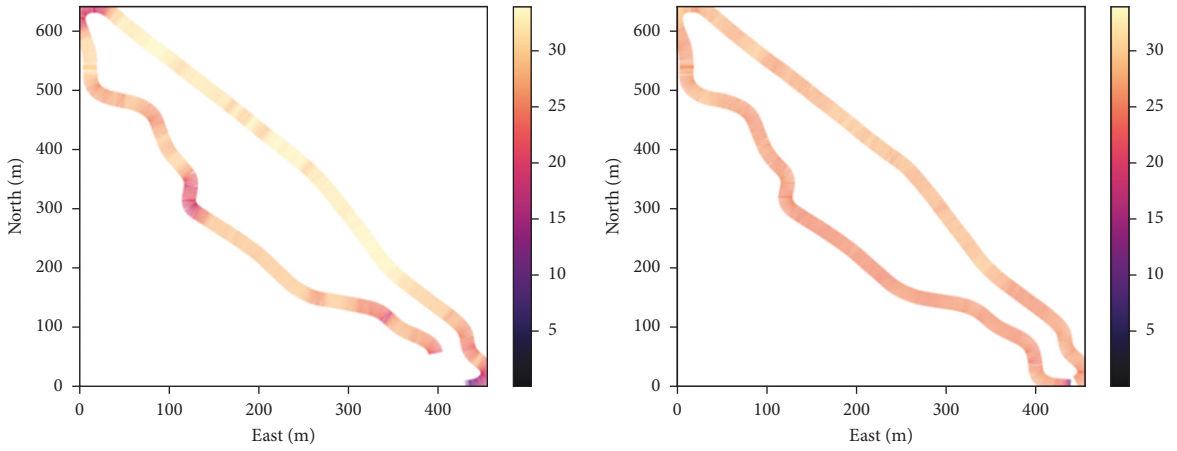

Figure 13: Visualized velocities at each point on the track.

Table 3: Results obtained from various deep reinforcement learning models.

Deep RL method	Reward
PPO2	3855.0 ± 5.00
SAC	3756 ± 42.47
A2C	3455.0 ± 105.19
TD3	3054.0 ± 793.85

Table 4: Student's t-test results for comparing PPO2 with other RL algorithms.

Method	T	P value	Conclusion
SAC	6.94	$1.7276e-06$	Reject H_0
A2C	11.40	$1.1552e-09$	Reject H_0
TD3	3.03	0.007248	Reject H_0

Table 5: Mean squared error of testing models.

Pseudolabeling	Dynamic calibration	MSE
O	O	**0.00643**
O	X	0.01697
X	O	0.05227
X	X	0.07953

Figure 13 shows the velocities recorded at each point on the track. The left image shows the velocities from when the human was driving, and the right image is from our RL model. The human driver was instructed to drive run the track clockwise along the center of the road at about 30 km/h. According to the figure, the driver slowed down the vehicle at each turning points and accelerated at the straight parts of the road. In contrast, our model drove at an almost constant speed. Human drivers consider the safety of the human and vehicle when driving. However, the RL model was trained in a way that it drives as fast as possible without considering safety.

Table 3 shows the results of applying various deep RL models to our simulator. For performance comparison, we used the representative deep reinforcement learning algorithms including Proximal Policy Optimization (PPO2), Soft Actor Critic (SAC), Advantage Actor Critic (A2C), and Twin Delayed Deep Deterministic Policy Gradients (TD3). Those algorithms are chosen because they show state-of-the-art performances with appropriate hyperparameters and are recommended for continuous action environments. According to the reward comparison results, PPO2 turned out to provide the highest reward among the methods. To validate statistical superiority of PPO2, we conducted t-test with other RL methods, and the results are shown in Table 4. Therefore, PPO2 was chosen to be the main RL algorithm to test on our vehicle.

To validate the effectiveness of pseudolabeling and dynamic calibration, we evaluated the mean squared error (MSE) between the steering values from the manual driving and the values from the four types of testing models. The testing models were trained with and without pseudolabeling and dynamic calibration. The steering values from the testing models were adjusted to half because the testing models typically output full throttle values. Table 5 shows that using both pseudolabeling and dynamic calibration resulted in steering values that were most similar to manual driving.

The equally distributed velocity heatmap in Figure 13 represents the optimal steering and throttle values that can be provided to the vehicle to drive the course. According to the results, the speed was maintained during the curved course unless it was a 270° hairpin curve course. This may be considered an un-human-like driving method, but this style of driving can be useful for strategic defense purposes. Strategic purposed vehicles, such as self-propelled artillery and armored vehicles, are required to move swiftly in curves without decelerating, because doing so will leave them vulnerable to enemy fire.

5. Conclusion

Applying reinforcement learning to autonomous driving has been a significant challenge for researchers because of the severe mismatches between simulations and the real world. Our simulator used dynamic calibration to predict the vehicle's next location from the given control commands. Moreover, two-class semantic segmentation, which distinguishes the road from the background, was found to be effective in reducing the gap between simulation scenes and real images. These methods demonstrated a positive effect on the sim-to-real performance of self-driving RL models. As a result, our model successfully drove on an unpaved road track without derailment.

6. Discussion

When the driving algorithm passes the simulation stage on the computer and tested in the real driving environment, there are many restrictions besides the core algorithm that must be considered. This is because it is no longer a simulation, but a real driving in the off-road condition. When a large vehicle weighing nearly 6.5 tons drives off-road with a high altitude difference at an average of 28 km/h, the restrictions are more severe. This is because driving in a situation where errors and problems exist in the overall system integration. Problems between tests continuously occur, which causes delay, and can continue only when these problems are resolved. It was practically difficult to prove the superiority and performance of one method to another through autonomous driving results in a real environment by implementing various reinforcement learning-based autonomous driving due to the project schedule and realistic conditions. Instead, the most probable and realistic algorithm through selection and concentration process was chosen through simulation, and then, the goal was to implement it in actual driving.

Data Availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Disclosure

The authors are with the Advanced Defense Technology Research Institute, Agency for Defense Development, Daejeon, 34186, South Korea.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. Bojarski, D. D. Testa, D. Dworakowski et al., "End to end learning for self-driving cars," 2016, https://arxiv.org/abs/1604.07316.

[2] M. Kull and P. Flach, "Patterns of dataset shift," in *Proceedings of the First International Workshop on Learning over Multiple Contexts*, LMCE) at ECML-PKDD, Bristol, UK, 2014.

[3] K. Kisamori, M. Kanagawa, and K. Yamazaki, "Simulator calibration under covariate shift with kernels," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 1244–1253, PMLR, San Diego, CA, USA, 2020.

[4] J. D. Chang, M. Uehara, D. Sreenivas, R. Kidambi, and W. Sun, "Mitigating covariate shift in imitation learning via offline data without great coverage," 2021, https://arxiv.org/abs/2106.03207.

[5] H. Zhang-Wei, Y.-M. Chen, H.-K. Yang et al., "Virtual-to-real: learning to control in visual semantic segmentation," in *Proceedings of the ACM International Joint Conferences on Artificial Intelligence (IJCAI)*, Vienna, Austria, 2018.

[6] D. A. Pomerleau, "Alvinn: an autonomous land vehicle in a neural network," in *Proceedings of the (NeurIPS) Neural Information Processing Systems*, La Jolla, CA, USA, 1989.

[7] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, Honolulu, HI, USA, 2015.

[8] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: a retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, 2015.

[9] L.-C. Chen, P. George, L. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[10] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Playing atari with deep reinforcement learning," 2013, https://arxiv.org/abs/1312.5602.

[11] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 29, no. 19, pp. 70–76, 2017.

[12] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," 2016, https://arxiv.org/abs/1610.03295.

[13] S. Wang, D. Jia, and X. Weng, "Deep reinforcement learning for autonomous driving," 2018, https://arxiv.org/abs/2002.00444.

[14] Y. Chebotar, A. Handa, V. Makoviychuk et al., "Closing the sim-to-real loop: adapting simulation randomization with real world experience," in *Proceedings of the 2019 International Conference on Robotics and Automation (ICRA)*, 2019.

[15] S. James, W. Paul, M. Kalakrishnan et al., "Sim-to-real via sim-to-sim: data-efficient robotic grasping via randomized-to-canonical adaptation networks," 2019, https://arxiv.org/abs/1812.07252.

[16] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3803–3810, IEEE, Philadelphia, PA, USA, 2018.

[17] Z. Xie, X. Da, M. v. d. Panne, B. Buck, and A. Garg, "Dynamics randomization revisited: a case study for quadrupedal locomotion," 2020, https://arxiv.org/abs/2011.02404.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Silver Spring, MD, USA, 2016.

[19] J. Schulman, W. Filip, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, https://arxiv.org/abs/1707.06347.

[20] R. Tan, J. Zhou, H. Du, S. Shang, and L. Dai, "A modeling processing method for video games based on deep reinforcement learning," in *Proceedings of the 2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, pp. 939–942, IEEE, Chongqing, China, 2019.

[21] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[22] Z. Hu, Z. Qin, and Q. Li, "Watershed superpixel," in *Proceedings of the 2015 IEEE International Conference on Image Processing (ICIP)*, Quebec City, Canada, 2015.

[23] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[24] X. Shi, S. Khademi, Y. Li, and J. V. Gemert, "Zoom-cam: generating fine-grained pixel annotations from image labels," in *Proceedings of the 2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 10289–10296, IEEE, Milano, Italy, 2021.

[25] Y. Zou, Z. Zhang, H. Zhang et al., "Pseudoseg: designing pseudo labels for semantic segmentation," 2020, https://arxiv.org/abs/2010.09713.

[26] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 618–626, Venice, Italy, 2017.