ROADS: Randomization for Obstacle Avoidance and Driving in Simulation

Samira Pouyanfar[1]*, Muneeb Saleem[2], Nikhil George[2], Shu-Ching Chen[1]
[1]Florida International University, Miami, FL, USA
[2]Electronics Research Laboratory, Volkswagen Group of America, Belmont, CA, USA
spouy001@cs.fiu.edu, mmuneebsaleem@gmail.com, nikhil.george@vw.com, chens@cs.fiu.edu,

Abstract

End-to-end deep learning has emerged as a simple and promising approach for autonomous driving recently. However, collecting large-scale real-world data representing the full spectrum of road scenarios and rare events remains the main hurdle in this area. For this purpose, this paper addresses the problem of end-to-end collision-free deep driving using only simulation data. It extends the idea of domain randomization to bridge the reality gap between simulation and the real world. Using a range of domain randomization flavors in a primitive simulation, it is shown that a model can learn to drive in realistic environments without seeing any real or photo-realistic images. The proposed work dramatically reduces the need for collecting large real-world or high-fidelity simulated datasets, along with allowing for the creation of rare events in the simulation. Finally, this is the first time domain randomization is used for the application of "deep driving" which can avoid obstacles. The effectiveness of the proposed method is demonstrated with extensive experiments on both simulation and real-world datasets[1].

1. Introduction

Deep neural networks have shown promising results in visual data analysis [13, 21, 1]. In recent years, it has also been leveraged for different tasks in self-driving cars such as car and pedestrian detection [30, 15], car-following behavior analysis [27], and controlling the steering of an automotive using raw image sensor data [3]. The latter also known as "Deep driving" applies deep neural networks directly on raw pixels of images taken from a car's front camera (raw visual sensor data) to generate a sequence of steering commands for autonomous cars. This technique is an end-to-end approach that eliminates the need for defining modules and the interfaces between them as this is implicitly handled when the model is trained.

Figure 1: Transferring the knowledge to the real world (center) from a photo-realistic simulation[2](left) vs a primitive simulation (right)

Yet, one of the main challenges in autonomous driving is collecting large-scale data with sufficient variety representing different real-world scenarios and conditions. For instance, the dataset should represent different locations, obstacles, movements, lighting, etc. Collecting such a large and heterogeneous dataset is costly and time-consuming. Also, it is sometimes impossible to collect data for rare scenarios (e.g., accidents, bad weather, and unusual driver behavior). To alleviate this problem, simulators can be used to quickly generate a huge amount of synthetic data. Until now, simulators have been widely used in various computer vision and autonomous driving applications [6, 8, 19, 9, 5, 7]. However, the question is how to effectively make the network trained on synthetic data operate on real-world data, in other words, how to bridge the reality gap.

The existing solutions to overcome the reality gap include generating photo-realistic worlds [7], Generative Adversarial Networks (GANs) for image-to-image translation [31], and Domain Randomization (DR) [25]. The latter is the most inexpensive, yet effective technique recently proposed by researchers to manage this challenge, mostly in the field of robotics [22, 24]. DR aims to expose the network to simulation's data with a wide range of variability (e.g., lighting, texture, objects, etc.) during the training to address the reality gap. This simple technique is able

*This project was supported by Volkswagen Group of America, Electronics Research Lab during Samira Pouyanfar's internship.

[1]Please see the color online copy of the paper. For supplementary video see: https://youtu.be/PsTneDbqClY

[2]This photo is taken from http://sonify.psych.gatech.edu/research/driving/index.html

to reduce (or eliminate) the need for large-scale real-world data since it forces the model to generate the representation invariant to the appearance of the object and environment [22]. In other words, the models trained on a wide variety of object meshes and scenes can generalize to the realistic scenes that may be completely different from the renderings generated for training [24]. DR has been recently investigated in very few specialized object localization and detection tasks [26, 4, 25].

This paper explores the DR potential for the application of autonomous driving. More specifically, we investigate whether networks trained using non-realistic simulation data can be used for collision-free driving in photo-realistic simulators and to generate collision-free driving paths in the real world. This is the first application of DR for collision-free autonomous driving using an end-to-end deep neural network. Different from some existing work on DR, the collected synthetic images are not photo-realistic and do not need to reflect real complex objects such as cars, pedestrians, and traffic signs. In other words, the goal is to transfer the knowledge from a primitive simulator with simple randomization techniques to a complex world instead of using expensive photo-realistic simulation as shown in Figure 1. This technique significantly reduces the time and cost of collecting realistic synthetic data.

Moreover, we extended DR to generate dynamic randomized scenarios during the training. More specifically, a wide range of random scenarios (events) is generated each representing a completely new world. Each world contains a new terrain, texture, road, light, shadow, and multiple objects with different sizes, randomly moving in various directions. We call this approach "Scenario Randomization" (SR) which not only includes randomization for static objects and their texture but also it randomizes the dynamic of objects (e.g., moving direction of obstacles). After training the network on those primitive simulation worlds, it is tested on the existing realistic simulation worlds for autonomous driving as well as two real-world image datasets including a self-collected parking-lot dataset and Kitti. To be able to evaluate the network in the real world without really driving a car, the future driving path of the car is also predicted. The network receives a single image and predicts the few next steering angles that are later translated to an estimated path. The extensive experiments on simulators' data demonstrate the effectiveness of DR in training deep neural networks for collision-free autonomous driving in simulation and also show interesting performance on real-world images.

The remaining of the paper is organized as follows. Section 2 provides recent work in autonomous driving and domain randomization. In section 3, the proposed framework is presented in details. Section 4 discusses the experimental results. Finally, conclusion and future directions are given in Section 5.

2. Related Work

This section provides an overview of existing studies in domain adaptation and randomization, as well as the state-of-the-art in autonomous driving.

2.1. Domain adaptation and randomization

Deep learning achievements heavily depend on the existence of large-scale datasets [13, 16]. Collecting such datasets in real-world is expensive and laborious. In particular, for the perception modules of autonomous driving, it is challenging to collect large-scale data for a diverse set of scenarios (e.g., day and night, various lighting conditions, large space of road, users movements, etc.) that would allow the perception systems to work robustly.

An alternative technique to collect a large amount of data is data augmentation with label-preservation that is commonly used in various computer vision applications [23, 29]. However, data augmentation cannot generate a high variability in the environment. In recent years, GANs have also shown promising results in domain adaptation and deep learning generalization [10, 2, 11]. Nevertheless, GAN models still need real-world data for training which is usually expensive.

Another practical approach is utilizing simulators. Collecting data from game engines and generating synthetic data for deep learning training has attracted significant attention in recent years. It is used for car and pedestrian detection [26, 12, 17] as well as robotic grasping and motion control [24, 5, 22]. To handle the reality gap problem, conventional approaches usually use synthetic data to train the network and then fine tune it on the real-world data [9]. Moreover, some studies used photo-realistic synthetic images to bridge the reality gap in object and scene detection [18, 12]. However, generating photo-realistic images is expensive and often requires laborious manual designing.

Compared to the aforementioned approaches, DR is a relatively new topic among deep learning training methods. DR is originally used in various robotics applications to transfer deep learning from simulation to the real world [22, 25]. CAD^{2}RL [22] is one of the first applications of DR that flies a quadcopter through indoor environments using reinforcement learning. Although CAD^{2}RL is based on only simulation data, it is still leveraging realistic scenes (e.g., chairs, doors, etc.) in the simulation which represent the real world quite well. Tobin *et al.* [25] proposed DR for object localization and robot manipulation. Similarly, Bousmalis *et al.* [5] leveraged both domain adaptation and randomization to transfer simulation to the real world for robotic grasping systems. In that work, a GAN is employed to make synthetic images more realistic. There are still few studies exploring DR in autonomous driving applications. In a recent work by Tremblay *et al.* [26], it is shown how DR can be used for object (cars) detection and it also illus-

trates the benefit of fine-tuning deep neural networks on real data after training on simulated data.

This work aims to use primitive and non-realistic synthetic data to train deep neural networks performing collision-free autonomous driving in simulation. Using novel domain and scenario randomization, the proposed framework is able to transfer the knowledge from a simulator without any realistic component to more realistic simulated worlds and further to real-world images. It also handles both static and dynamic objects in these worlds.

2.2. End-to-end learning for autonomous driving

The model architecture used in this paper is a single end-to-end differentiable neural network. This reduces the complexity otherwise associated with designing an autonomous driving architecture where a human needs to select and design a variety of modules (e.g. perception, tracking, prediction, planning etc.). It also removes the need for expensive data annotations, as the network is trained using imitation learning, where labels are provided directly by human driving trajectories. We call this approach "Deep Driving".

There are multiple experiments conducted by different groups over the years on testing Deep Driving in the real world. Some examples using raw camera pixels to predict steering angles or path include Alvinn [20], one of the first work in this area, and Dave, another DARPA-funded project [14] by Lecun *et al.* More recently, Nvidia evaluated this approach on a car in multiple road scenarios [3]. Also, Xu *et al.* added time dimension to this approach using recurrent neural networks and an auxiliary loss [28].

3. Proposed Framework

The proposed framework is shown in Figure 2. As can be seen from the figure, we generated a very simple simulation world consisting of primitive object shapes and a road using the Unity 3D game engine[3]. Thereafter, we apply various SR techniques on this world and collect both images and the corresponding steering angles by driving the car in simulation. These data are later used to train the neural network and finally predict the future steering angles path on a realistic image.

3.1. Domain randomization

In this work, we proposed three different domains (worlds) as shown in Figure 3. The first domain is designed for the training purpose while the other two are used only for testing the framework. We need these three domains to see how the model trained on Domain 1 can drive on more realistic simulated worlds (Domain 2) and how it predicts the path in the real-world images (Domain 3).

[3] `https://unity.com`

Domain 1: The original domain is designed using a simulator (i.e., Unity game engine). It contains a simple road and some basic primitives that the ego-car tries to avoid as shown in Figure 3a. This domain does not include real texture (e.g., roads with lane marking) or any real objects such as vehicles, pedestrians, tree, bridge, and lake[4]. In Domain 1, we apply various DR and SR techniques and use the collected data from this domain for model training. To collect the data (images and steering angles), human users control the ego-car in this domain to keep the car on the road while avoiding the obstacles.

Domain 2: The next domain is also designed in the simulator but it includes more photo-realistic elements similar to the real world. In Figure 3b, two samples of Domain 2 are shown that include realistic objects such as trees, a lake, cars (static and dynamic), real road texture, etc. This domain is only utilized for the validation phase.

Domain 3: Finally, the last domain includes real-world images/video of outdoor environments (e.g. highways, urban, and parking lots). Two image samples of this domain are shown in Figure 3c. It must be noted that Domain 3 is only used for final testing of the deep driving network and never used during training.

It is worth mentioning that Domains 1 and 2 are both designed based on the Lake Track scene of the Udacity's self-driving car simulator[5]. For Domain 1, we only used the basic road track and removed all the realistic components. We also modified the road size, shapes, and curves.

This work aims to add a variety of randomization to the original simulated domain (Domain 1) which helps the model generalize to more realistic domains (Domain 2 and Domain 3). In fact, the model is trained to see the new domain merely as one more randomized flavor of the original domain. The randomization factors include:

Terrain: in order to cover various surroundings in the real world, we utilize several unrealistic terrains (grounds) with various textures selected from a small set of terrain textures (20 textures) which can help the model to drive in different real-world environments (e.g., mountains, jungle, parking, highway, etc.)

Road texture: for each scenario, a new road texture is randomly selected from a set of road textures. This texture can be unrealistic and does not include any lane marking or other realistic road texture. However, its variation can help the model to generalize to real-world unseen textures. In particular, we use 20 unrealistic road textures for training and 10 realistic textures for testing.

Novel objects (types, size, color, texture): the 3D objects in Domain 1 include three simple primitives (cube, cylinder,

[4] The only realistic component in this domain is rocky mountains (is simply generated by Unity) which can be used for shadow generation or representing tall buildings in the real world.

[5] `https://github.com/udacity/self-driving-car-sim`

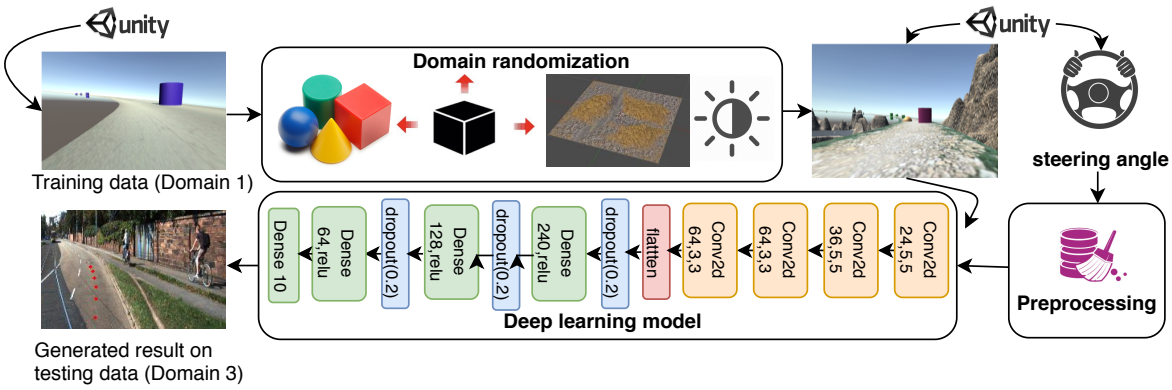
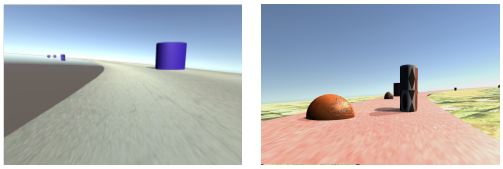


Figure 2: The proposed framework for collision-free autonomous driving based on domain randomization



(a) Domain 1 without DR (left) and with DR (right)

(b) Domain 2 designed by us (left) and Udacity lake track (right)

(c) Domain 3 parking data collected by us (left) and Kitti (right)

Figure 3: Samples of three domains including Domain 1 or primitive simulation (a), Domain 2 or photo-realistic simulation (b) and Domain 3 or the real world (c)

sphere). In each scenario, the program selects a random number (between 10 to 40) of objects from a list of random object types with random scale, color, and texture and place them in random positions (x, y, z) on the road using 3D waypoints in Unity. Different from existing work [26], we use simple pattern-based textures instead of using realistic textures from large-scale datasets.

Light: in each scenario, a random intensity of light (between 0 to 2) is selected which represents different time (day, night) and conditions of the real world.

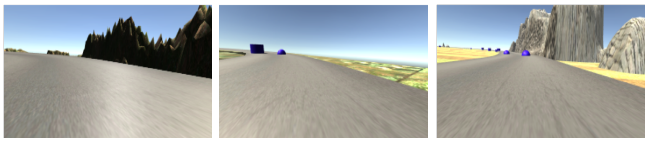
Shadow: for shadow randomization, a special terrain with various heights is designed. This terrain generates shadows in different parts of the road and helps the model to learn object's and the environment's shadows.

Dynamic objects: one of the major novelties of this work is handling both static and dynamic objects with various movement trajectories. For this purpose, a random path (right or left lane) with a random velocity (0 for static objects, negative numbers for reverse driving, and positive numbers for normal driving) is assigned to each object. Thus, the object can automatically follow the waypoints in the assigned lane.

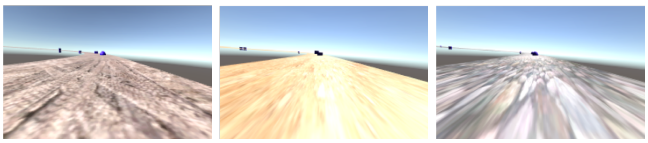
Through extensive research and experiments, it is shown that the aforementioned factors play important roles in generalizing the model to the real worlds. Examples of different randomizations are shown in Figure 4. The last row shows three samples of the combined randomization.

3.2. Data preparation and model training

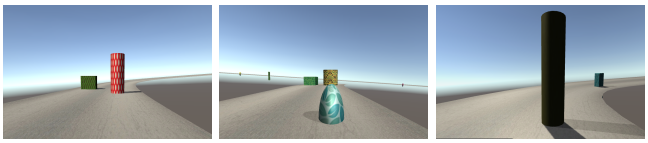
In this work, the goal is to steer the car on a road while avoiding the static and dynamic obstacles. Therefore, the speed of the car is fixed using the auto-cruise component in the simulator and only the steering angle for each frame is collected. For training, the video frames from three cameras, placed in left, center, and right positions of the ego-car, together with the corresponding steering angle applied by the user are collected from Domain 1 with different domain and scenario randomization. Specifically, ten frames and steering angles are selected per second from the simulation video at 40 frames per second (fps) rate. As mentioned in [3], images from left and right cameras are required to train the agent on how to recover from non-optimal positions, which is essential because of the cascading errors that occur due to behavior cloning based imitation learning. Specifically, this method helps the model to avoid drifting off the road by augmenting with images that are shifted laterally relative to the longitudinal axis of the car.



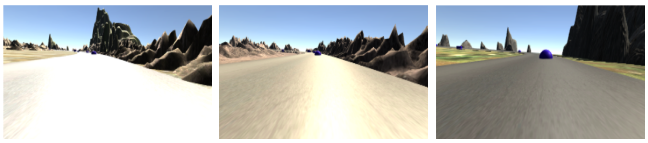
(a) Samples of terrain randomization



(b) Samples of road texture randomization

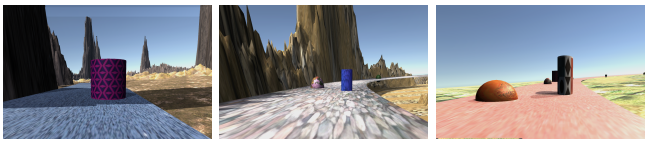


(c) Samples of object color/texture/scale randomization



(d) Samples of light intensity randomization

(e) Samples of shadow randomization



(f) Samples of all combined randomizations

Figure 4: Examples of various randomizations applied to Domain 1

After collecting the data from the simulator, it is necessary to handle data outliers and smooth the steering angles since human are not always able to drive smooth trajectories. Outliers are replaced by mean and the steering angle curve is smoothed with a moving average with a window size of 20 based on our empirical study.

In this work, rather than generating steering angles as traditionally done [3], we chose to generate a path to be able to test our model in the real world. To do so, we collect the current and $N-1$ future steering angles. Thus, the network can predict the path while driving on the road. The generated path can help us to evaluate the model on real-world images and videos in an open-loop manner without driving a real car. After the preprocessing step, the steering angles of the left and right cameras are set by shifting the center cameras' steering angle by a factor of γ (to avoid driving off the road). In this work, the correction factor for the next N consecutive steering angles is calculated as:

$$\alpha_{L(R)} = \alpha_C \pm (\gamma * (N - i^2)/N) \quad (1)$$

where α refers to the steering angle, L, C, and R refer to the left, center, and right cameras, respectively, and $i \in \{0...N-1\}$ ($i = 0$ refers to the first steering angle).

The neural network architecture of the proposed end-to-end deep driving model is shown in Figure 2. It takes a single image which goes through several convolutional layers followed by four dense (fully-connected) layers. The last dense layer generates ten outputs which correspond to the N steering angles. In other words, given a single image, this regression model is able to predict the next N steering angles in an end-to-end manner. The model outputs are later converted to the path in order to visualize the performance of the model in the real world data.

4. Experiments

4.1. Datasets

Multiple datasets are used for training and evaluation, with a focus on covering diverse simulated and real-world scenarios. The details of each dataset is described below:

Simulation dataset: As described before, our training data is collected from Domain 1 (refer to Figure 3a). For each scenario, a new combination of all randomization factors are automatically generated. To collect the images and the corresponding steering angles, we had test subjects drive the car in the simulated world akin to playing a computer game (each played between 10-30 minutes). In total, the combined DR dataset contains around 200k images. We also collected 30K-100K images for each flavor of randomization (e.g. road texture, object, terrain, light, and shadow). For the baseline model ("No Rand"), we used Domain 1 data without applying any randomization technique for training, while keeping the total number of images fixed. For evaluation, we utilized two different photo-realistic simulation worlds (please refer to Domain 2 in Figure 3b). The first version is the lake track from Udacity in which several obstacles (cars, objects, etc.) are added on the road and the second simulation world designed by us includes moving cars, and also photo-realistic road/terrain textures.

Parking dataset: A dataset is collected by driving a car around our corporate campus in California. This dataset is useful in evaluating the behavior of our model in a complex real-world environment. Presence of a large number

of stationary parked vehicles is an additional benefit of this dataset for evaluation of our obstacle avoidance model.

Kitti: Finally, we downloaded several sets of Kitti raw data[6] including city, residential, and road categories to further evaluate our framework on real-world images.

4.2. Experimental settings

In this work, the steering angle correction factor is empirically set to $+0.25$ and -0.25 for the left and right shifts, respectively. N is set to ten consecutive steering angles. The image size is set to 160*320px. The car speed is fixed to 30mph during both data collocation and testing on simulation. For preprocessing, the moving average window is set to 20. The sequence of steering angles and the camera calibration are used to place a 2D path on the image for visualization purposes. This projection assumes a flat ground plane for the road surface.

The deep learning model includes preprocessing layers to normalize the image (mean centered) and crop the top parts of the image (remove the sky and only focus on the road and obstacles). Similarly, the real-world images from parking and Kitti datasets are preprocessed to follow the same format of the simulation data.

For deep learning training, the following settings are used: Number of epochs=5, Optimizer=Adam, learning rate=0.0005, batch size=32, Loss function= Mean Squared Error (MSE), activation functions=RELU, dropout=20%. The images collected from 2/3 of the road from Domain 1 is used for the training and the remaining is used for the validation. In all the experiments, the same deep learning model is used for training and all the training images are obtained from Domain 1 without using any real images.

We evaluate the performance of the deep learning model using Nvidia's autonomy metric [3] for simulation. This metric counts the number of human interventions to retake the control of the car. In our case, there are two types of errors while the model is driving the car: (1) collisions with an object ($\#Collisions$), (2) events where the car ends up outside of the road boundary ($\#Off-roads$). We assign the same penalty as [3] (6 seconds) when an error happens. Thus, autonomy is calculated as:

$$autonomy = (1 - \frac{(\#Collisions + \#Off-roads) * 6 \quad [sec]}{total \quad time \quad [sec]}) * 100 \tag{2}$$

When a collision or off-road event happens, we programmatically count the number of errors and reset the car's location to the next waypoint on the road. This reduces human intervention while testing the model.

[6]KITTI Vision Benchmark Suite: `http://www.cvlibs.net/datasets/kitti/raw_data.php`

4.3. Experimental results

4.3.1 Ablation study for simulation environment

The goal of this experiment is to see how the model trained on Domain 1 can drive in a photo-realistic simulation environment. The first set of experiments is executed in a new simulation environment (Domain 2) which is never seen during the training. To test the impact of each randomization factor, we trained a model for every single randomization and compared them with no randomization (No Rand) and our model (DR), which is trained on all the randomizations together. Specifically, a fixed amount of image data (30K) are collected for each randomization model (e.g., terrain, road texture, light, object scale/color/texture, and shadow). After training, each model is tested on four different scenarios as follows: (1) Our designed Domain 2 including 7 static cars, (2) Our designed Domain 2 including 8 dynamic cars moving in various directions, (3) Our designed Domain 2 including 6 static cars in the middle of the road and 16 cars on the side (simulating a narrow parking space) (4) Domain 2 from Udacity (lake track) including 4 static cars and 2 cubes. In total, each model is tested for 20 minutes on these four fixed scenarios. The result of this experiment is shown in Table 1. This table shows total number of collisions, off-roads, and autonomy. As can be seen from the table, with adding terrain randomization (R1), the number of collisions and driving off the road decreases. With road texture randomization (R2) the model confuses objects with the road textures causing more collisions. However, this greatly helps the ego-car stay on the road. Similarly, light (R3) and shadow (R4) are important factors for avoiding off the road driving. Object randomization factors (R5 & R6) are obviously the best parameters for reducing collisions. This is powerful as the object randomization appears to teach the model the concept of avoiding obstacles. This is indicated by the fact that despite the training set only containing simple geometric objects, the model avoids more complex obstacles like cars and pedestrians. Finally, the combined domain randomization model can reduce the collision with a great margin while staying on the road all the times. The autonomy of our DR model reaches 0.98 in this set of experiments which is 11% higher than the one from the "No Rand" model.

To further investigate the impact of DR on simulation, randomization components (R1 to R6) are added one by one to the dataset (while keeping the size of the dataset fixed) and a model is trained for that specific combination. Although the previous experiment shows our model is able to avoid dynamic objects without seeing them during the training, we also use "object movement randomization" (R7) to further enhance the model reaction to the moving objects. Each model is tested in our designed Domain 2 for 30 minutes while changing the environment components (e.g.,

Method	# collision	# off–roads	autonomy
No Rand	16	11	87
Terrain (R1)	6	8	94
Road (R2)	20	1	90
Light (R3)	13	1	93
Shadow (R4)	23	1	88
Obj Scale (R5)	13	4	92
Obj color & texture (R6)	6	4	95
DR (Ours)	5	0	98

Table 1: Comparison results on simulation (Domain 2)

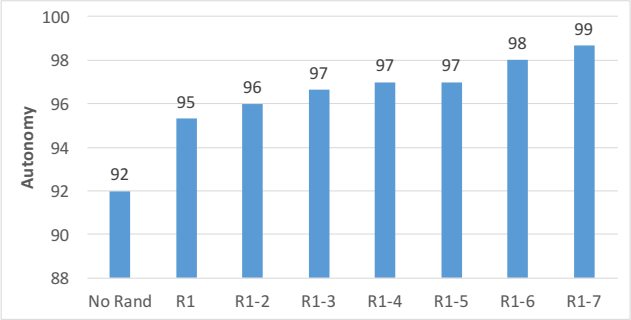


Figure 5: Impact of adding individual randomization to the model on autonomy

light, shadow, terrain, static and dynamic objects, etc.) after each cycle. More specifically, each model continuously tested over multiple cycles where each cycle used a different environment (to have a fair comparison, we keep these changes fixed for all the models). Figure 5 depicts the results of this experiment regarding the autonomy metric. As can be inferred from this plot, R1-2 (terrain+road) can extensively enhance the performance (especially decreasing the off-road) and adding other randomization factors can gradually increase the model's generality to realistic environments. The full randomization model (also includes object movement randomization) achieves 99% autonomy.

4.3.2 Ablation study for realistic environment

To evaluate the performance of our model in the real world, we utilized our collected parking dataset and the public Kitty dataset as explained before. To do so, the model receives a single image and generates a path demonstrating the future direction of the car. Figure 6 shows several samples from both datasets with DR and without DR. It can be clearly seen that when there are objects in its path the DR model changes its path to avoid the objects, while the "No Rand" model can barely stay on the road (it can be seen from the sharp trajectory to the either left or right) or goes directly towards the object. For the Kitti dataset, the "No Rand" model is showing a sharp turn to the left in almost all the images, while our DR model smoothly changes its direction when observing an object close to it (e.g., first and second rows in Figure 6 (b)). These results also show that our model can deal with extra shadows (e.g., fourth rows in Figure 6 (a) and (b)) and detect obstacles in noisy images (e.g., last row in Figure 6 (a) which is an image taken through the windshield of the car using a cellphone camera with reflections from the dash).

Method	Accuracy (%)	
	Parking	Kitti
No Rand	18.60	12.44
DR (Ours)	58.14	73.42

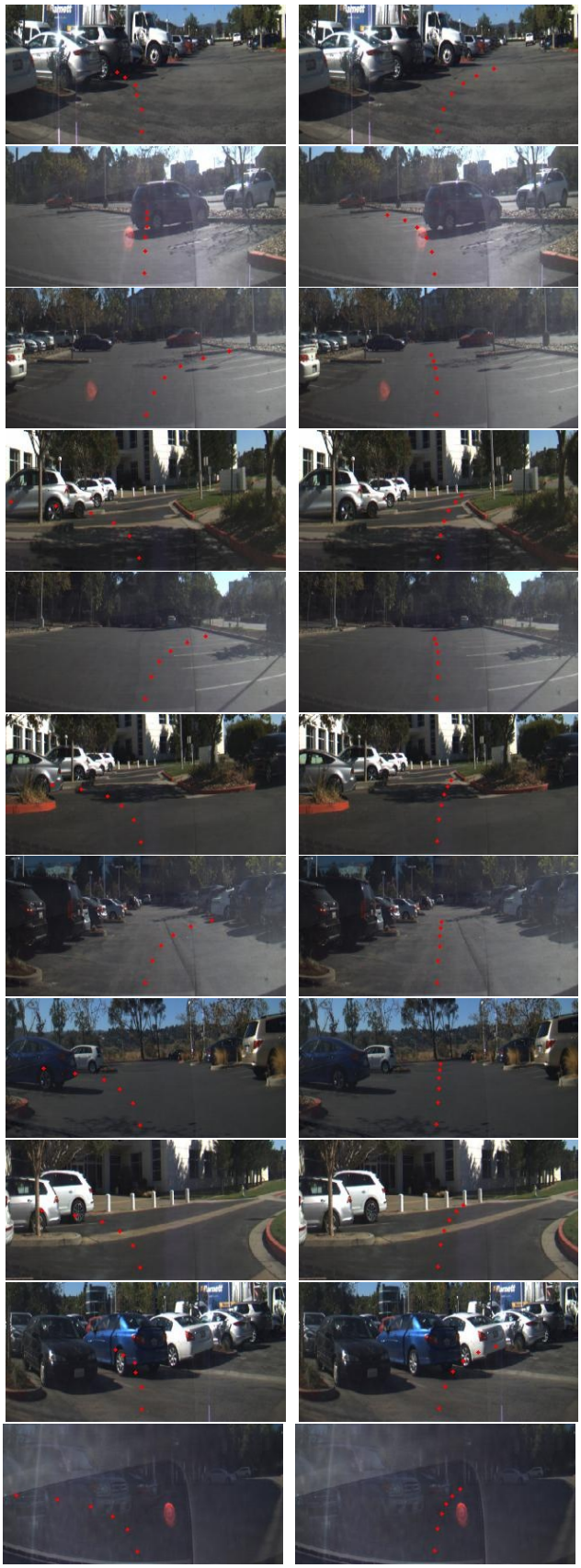
Table 2: Comparison results on real world (Domain 3)

Finally, the accuracy of these two models (No Rand and ours) is compared in Table 2. Accuracy is calculated by $\frac{(number\ \ of\ \ images\ \ with\ \ correct\ \ trajectories)}{total\ \ number\ \ of\ \ images} * 100$. It can be seen from the table that DR can greatly enhance the performance of our obstacle avoidance framework on the real-world images. These results show the effectiveness of DR in bridging the reality gap for this application.

5. Conclusion and future directions

This work aims to provide an alternative avenue of research for solving one of the challenging topics in machine learning and robotics, i.e. autonomous driving, with a simple but effective approach compared to the current state-of-the-art. The proposed work utilizes primitive simulated data and applies a wide range of domain and scenario randomization to reduce the gap between simulation and the real world. The main contributions of this work include: (1) Applying DR to a new and complex application (collision-free deep driving); (2) Extending the idea of DR to SR with leveraging dynamic objects with random movements in addition to the static domain randomization; (3) Demonstrating that obstacle avoidance can be learned with simple geometric shapes rather than expensive photo-realistic objects; (4) Conducting comprehensive experiments to show the importance of various randomization factors in making deep driving work in the simulation that also reveals interesting results in the real-world.

Future research can focus on extending these ideas to more structured and rules-driven environments while avoiding making the framework intricate. This will also help us test our models directly on the real world by coupling the network to the controller on a real car.



(a) Parking samples without DR (left) and with DR (right)

(b) Kitti samples without DR (left) and with DR (right)

Figure 6: Examples of results on parking and Kitti datasets

References

[1] Arian Azarang, Hafez E Manoochehri, and Nasser Kehtarnavaz. Convolutional autoencoder-based multispectral image fusion. *IEEE Access*, 7:35673–35683, 2019.

[2] 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. *arXiv preprint arXiv:1812.03823*, 2018.

[3] 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.

[4] João Borrego, Atabak Dehban, Rui Figueiredo, Plinio Moreno, Alexandre Bernardino, and José Santos-Victor. Applying domain randomization to synthetic data for object category detection. *arXiv preprint arXiv:1807.09834*, 2018.

[5] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *IEEE International Conference on Robotics and Automation*, pages 4243–4250, 2018.

[6] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.

[7] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2016.

[8] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. Understanding real world indoor scenes with synthetic data. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4077–4085, 2016.

[9] Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. On pre-trained image features and synthetic images for deep learning. In *European Conference on Computer Vision*, pages 682–697, 2018.

[10] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213*, 2017.

[11] Weixiang Hong, Zhenzhen Wang, Ming Yang, and Junsong Yuan. Conditional generative adversarial network for structured domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1335–1344, 2018.

[12] Matthew Johnson-Roberson, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, Karl Rosaen, and Ram Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? *arXiv preprint arXiv:1610.01983*, 2016.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[14] Y Lecun, E Cosatto, J Ben, U Muller, and B Flepp. Dave: Autonomous off-road vehicle control using end-to-end learning. *DARPA-IPTO Final Report*, 2004.

[15] Jianan Li, Xiaodan Liang, ShengMei Shen, Tingfa Xu, Jiashi Feng, and Shuicheng Yan. Scale-aware fast r-cnn for pedestrian detection. *IEEE Transactions on Multimedia*, 20(4):985–996, 2018.

[16] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pages 740–755, 2014.

[17] Javier Marin, David Vázquez, David Gerónimo, and Antonio M López. Learning appearance in virtual scenarios for pedestrian detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 137–144, 2010.

[18] John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J Davison. Scenenet rgb-d: 5m photorealistic images of synthetic indoor trajectories with ground truth. *arXiv preprint arXiv:1612.05079*, 2016.

[19] Matthias Müller, Vincent Casser, Jean Lahoud, Neil Smith, and Bernard Ghanem. Sim4cv: A photo-realistic simulator for computer vision applications. *International Journal of Computer Vision*, pages 1–18, 2018.

[20] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems*, pages 305–313, 1989.

[21] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[22] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[23] Naoya Takahashi, Michael Gygli, Beat Pfister, and Luc Van Gool. Deep convolutional neural networks and data augmentation for acoustic event detection. *arXiv preprint arXiv:1604.07160*, 2016.

[24] Josh Tobin, Lukas Biewald, Rocky Duan, Marcin Andrychowicz, Ankur Handa, Vikash Kumar, Bob McGrew, Alex Ray, Jonas Schneider, Peter Welinder, et al. Domain randomization and generative models for robotic grasping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3482–3489, 2018.

[25] 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 23–30, 2017.

[26] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *IEEE Conference on Computer*

Vision and Pattern Recognition Workshops, pages 969–977, 2018.

[27] Xiao Wang, Rui Jiang, Li Li, Yilun Lin, Xinhu Zheng, and Fei-Yue Wang. Capturing car-following behaviors by deep learning. *IEEE Transactions on Intelligent Transportation Systems*, 19(3):910–920, 2018.

[28] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. End-to-end learning of driving models from large-scale video datasets. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2174–2182, 2017.

[29] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. *arXiv preprint arXiv:1708.04896*, 2017.

[30] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.

[31] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision*, pages 2223–2232, 2017.