Sim-to-Real via Sim-to-Seg: End-to-end Off-road Autonomous Driving Without Real Data

Anonymous Author(s)
Affiliation
Address
email

Abstract: Autonomous driving is complex, requiring sophisticated 3D scene understanding, localization, mapping, and control. Rather than explicitly modelling and fusing each of these components, we instead consider an end-to-end approach via reinforcement learning (RL). However, collecting exploration driving data in the real world is impractical and dangerous. While training in simulation and deploying visual sim-to-real techniques has worked well for robot manipulation, deploying beyond controlled workspace viewpoints remains a challenge. In this paper, we address this challenge by presenting Sim2Seg, a re-imagining of RCAN [1] that crosses the visual reality gap for off-road autonomous driving, without using any real-world data. This is done by learning to translate randomized simulation images into simulated segmentation and depth maps, subsequently enabling real-world images to also be translated. This allows us to train an end-to-end RL policy in simulation, and directly deploy in the real-world. Our approach, which can be trained in 48 hours on 1 GPU, can perform equally as well as a classical perception and control stack that took thousands of engineering hours over several months to build. We hope this work motivates future end-to-end autonomous driving research.

Keywords: Sim-to-Real, Reinforcement Learning, Autonomous Driving

1 Introduction

Simplifying large autonomous driving software stacks, which are usually composed of 3D scene understanding, localisation, mapping, and control, is a promising goal. While these stacks can indeed perform well in a range of scenarios, they do suffer from error propagation through each of the modules, and tend to require a large engineering overhead. However, attempting to solve autonomous driving problem in a purely end-to-end manner, where observations are mapped directly to actions, also has its downfalls. For one, these methods are usually data intensive, and in particular for reinforcement learning (RL), collecting exploration driving data in the real world is impractical and dangerous.

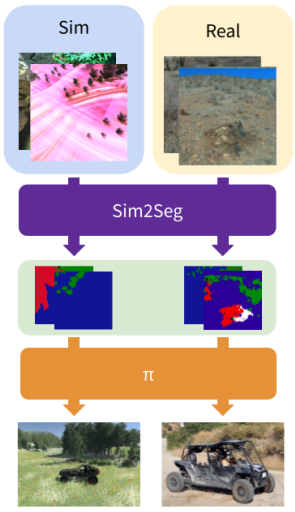

Figure 1: Sim2Seg learns a mapping between randomized images and segmentation maps for zero-shot transfer.

To overcome the data burden, large-scale simulations can be employed to collect experience from a large number of parallel agents. However, we then have to consider the visual and dynamic discrepancies between the simulation and reality. Sim-to-real transfer approaches, such as domain adaptation [2, 1] and domain randomiza-

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

tion [3, 4, 5, 6] exist for this reason. These however, have mostly been applied in tasks with a fixed camera viewpoint, such as manipulation tasks where the camera usually points down towards a bin [2, 1] or table [3], or faces a wall [4, 5]. Unlike these robot manipulation setups, autonomous navigation with full-scale vehicles in off-road settings are far less controlled, often featuring aspects such as drastically different lighting, glare, dust, long visual horizons, and complex backgrounds. In addition, unlike in standard autonomous street driving, the vehicle must traverse off-road terrains, which have less uniform dynamics and may include natural obstacles such as bushes, rocks, bumps, and more, which may not be observed during training.

In this paper, we investigate how similar sim-to-real techniques used for smaller scale robotic domains, such as robot manipulation, can be scaled to the significantly more visually challenging domain of off-road autonomous driving of a large vehicle. To this end, we propose Sim2Seg, a sim-to-real method designed with the challenges of off-terrain autonomous navigation in mind. Combined with a deep RL policy, Sim2Seg is the first work to effectively employ primarily visual sim-to-real transfer for off-road autonomous driving.

Contributions We highlight the contributions of our paper below:

- We improve the discriminator component within RCAN [1] by convolving the output segmentation maps to produce feature maps that are then fed to a discriminator to evaluate.
- We explore suitable action modes that encourage safe trajectory proposals without diminishing the model's capability.
- We show that our trained RL agent can perform as well as a sophisticated, model-based autonomous driving stack.
- We show the first sim-to-real transfer method for end-to-end RL autonomous driving that requires no real world data.

2 Related Work

Model-based Autonomous Driving Most off-road autonomous driving approaches center around scene understanding approaches. Due to the challenges of natural obstacles such as bushes, trees, and rocks; uneven terrain; and various static and dynamic uncertainties [7], the robot must constantly assess the traversability of the terrain. Geometry-based methods involve constructing a terrain map [8, 9] from depth measurements from sensors such as LiDAR, stereo cameras, etc. This terrain map is used to generate a traversability cost by performing stability analysis, using features like surface normals, maximum or minimum height of the terrain, etc., which can be used by motion planning and control algorithms to plan vehicle's actions [10, 11, 12, 13]. Papadakis [14] provides a survey of several other off-road driving algorithms. In lieu of these methods, we focus on end-to-end autonomous driving directly from pixels, avoiding these engineering layers.

End-to-End Autonomous Driving End-to-end autonomous driving at large remains an unsolved problem: most approaches have narrowed their scope to urban environments [15, 16], as it is most relevant for everyday human transportation. In addition, most methods simplify the problem of general navigation by assuming static environments [17] and real-world datasets [15], which is a limiting factor for more difficult terrains and navigation tasks like off-road autonomous driving.

Since urban environments produce unique challenges of multi-agent interactions and following traffic rules, many end-to-end approaches greatly simplify the environment to focus on navigation on roads. For instance, Chu et al. [18] and Nair et al. [19] both reduce the environment to static, toy car racing environments. Offline real-world data has also been crucial to many methods. Ram [20] trains end-to-end in CARLA [21] — an urban driving simulator, but to bridge the visual sim-to-real gap, requires real-world data in order to enhance simulator images. To produce segmentation maps that the agent utilizes in the policy, Osinski et al. [17] uses real-world images and ground-truth semantic segmentation maps. VISTA [15] focuses on street navigation and relies upon a data-driven

simulator, which synthesizes new viewpoints of a scene based on offline data. Because VISTA relies on human data collection, simulating new scenes, such as for offroad terrains, is time-intensive, making it hard to scale up and train in diverse simulations.

Sim-to-Real in the Visual Domain Perception forms the basis of many tasks, from smaller-scale robotics tasks to large-scale vehicles. To ensure consistent perception across simulators and real-world images, a popular technique used is domain randomization [22, 4, 16], in which input observations are randomized to prevent overfitting to simulator images, ensure adaptability to a variety of conditions, and encourage extraction of meaningful features such as object shapes and locations.

In the case that real-world data is easily accessible, domain adaptation is a popular method used to extract consistencies across the two domains. Pixel-level domain adaptation, for instance, improves pixel-level consistencies by restylizing simulator images [23, 24]. For robotics tasks involving objects that are critical to the scene, additional losses are penalized: RetinaGAN ensures object consistencies using a pretrained object detector [25], and RL-CycleGAN [26] penalizes differences in Q-values. Feature-level domain adaptation learns shared features across both domains [27, 28, 29]. It is notable that many of these environments are based around grasping and other robotics tasks, which have fixed camera viewpoints, controlled environments, and relatively easier data collection processes in comparison with off-road vehicles.

3 Method

In this section, we detail our method, Sim2Seg, which is summarised in Figure 1. It consists of two primary components: (1) inspired by RCAN [1], we train a Sim2Seg model that translate randomized RGB images from simulation into a shared *canonical* form, which we define as a semantic segmentation map. (2) We then train a goal-reaching policy within this canonical representation to navigate in off-terrain environments. During inference, we use a frozen pretrained Sim2Seg model to perform zero-shot transfer on real-world images.

3.1 Simulation

We create several simulation environments using the Unity engine. Unity fulfills several key desiderata, notably high-fidelity visual observations and dynamics, open-source vehicle components and scenes [30], and the ability to apply custom domain randomization techniques. Unity also integrates well with RL training with the Unity ML-Agents toolkit, which provides a Gym interface for training [31]. For our purposes, we further modify ML-Agents to support instance-level parallelism, allowing us to train multiple agents per built executable.

To maximally train our policy to a variety of off-road environments and generate a diverse dataset of simulator data, we select 3 different simulated scenes — dubbed Meadow [32], Landscapes [33], and Canyon [34] — with semantically diverse visual environments. During training, the policy trains simultaneously on all 3 environments, leveraging the most out of the simulation training phase and ensuring adaptability to a variety of scenes.

3.2 Sim2Real via Sim2Seg

To bridge the visual gaps between simulators and real-world data, we use a Sim2Seg model to convert randomized image domains into our chosen canonical form, a segmentation map consisting of six classes: trees/bushes, ground, sky, rocks, road, and logs. We believe this is useful for off-road vehicles because it simplifies the unnecessary details, textures, and colors of images and identifies different types of obstacles, which gives important information on obstacles and areas to avoid (see Figure 2).

Domain Randomization For data collection, we collect 200k pairs of paired RGB images, segmentation maps, and depth data per environment, using a random policy. We apply domain ran-

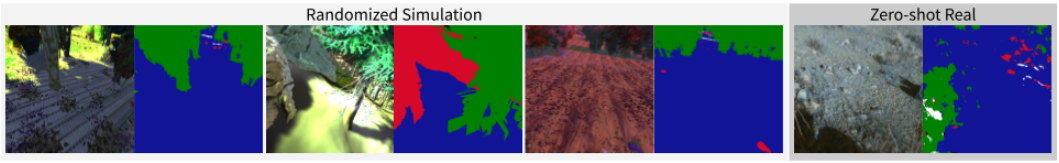

Figure 2: To bridge the visual sim-to-real gap, we apply combinations of texture randomization, camera position and intrinsic randomization, and lighting color and direction randomization to varying scenes in Unity, and learn mappings to ground truth segmentation maps.

domization to textures of all objects, lighting color and direction, camera field-of-view, and camera position. For the best map to the canonical state, we train a Sim2Seg model per each environment and a separate Sim2Seg model for the combined dataset. During training, we use each environment's Sim2Seg model, which avoids distribution shifts between the Unity environments. At real-world test time, we also consider a Sim2Seg model trained on all environments to achieve optimal performance.

3.3 Sim2Seg Training

Following RCAN [1], we use conditional GANs (cGAN) with the U-Net Architecture [35, 36] to translate pairs of randomized simulation images into their canonical segmentation map representation. During train time, we use paired simulation data $(x_s, x_c, m_d, m_o)_{j=1}^{N}$, where x_s is a randomized RGB simulation image, x_c is the canonicalized segmentation map, and m_d is the canonicalized depth map, and m_o the obstacle mask.

Unlike RCAN [1], we modify our objective to adapt to the task of autonomous driving. While RCAN predicts the canonical image, segmentation map, and depth map, we simplify our problem to predicting the segmentation map, which is simultaneously our canonical image; and the depth map, which is used as an auxiliary.

Thus, to encourage visual similarity, we optimize the following objective:

$$L_{eq}(G) = \mathbb{E}_{x_s, x_c, m_d, m_o}[\lambda_x l_{eq_x}(G_x(x_s), x_c) + \lambda_d l_{eq_d}(G_d(x_s), m_o \cdot m_d)]] \tag{1}$$

where l_{eq_x} is the cross-entropy loss, l_{eq_d} the L1 loss, $m_o \cdot m_d$ is the element-wise product, and λ_x, λ_d the weighting of the losses.

Adversarial Objective We employ a discriminator $D(x)$ which outputs the probability that the RGB image and segmentation map is a pair from the simulation dataset. Because our canonical output consists of probabilities for segmentation classes, we experimented with a few approaches to use f to featurize the combination of RGB and segmentation map.

$$L_{GAN}(G, D) = \mathbb{E}_{x_s, x_c}[\log D(f(x_s, x_c)] + \mathbb{E}_{x_s}[\log(1 - D(x_s, f(G_x(x_s))] \tag{2}$$

Because our Sim2Seg model outputs segmentation class logits while the ground truth segmentation maps are one-hot encodings, discriminating directly on image and segmentation pairs is immediately trivial. In addition, because segmentation classes are discrete, we cannot sample a segmentation class and pass the gradient back to the generator. To remedy this issue, we initially attempted two two approaches: (1) sampling with gumbel-softmax [37] and (2) approximately the arg-max with soft arg-max [38], both of which are differentiable. Despite this, GAN training remained unstable, as discriminating between real and fake pairs was easy, likely due to noisy gumbel-softmax samples and soft-argmax values in areas of uncertainty.

Thus, to circumvent these issues, we choose instead to separately convolve the simulation image and segmentation maps to an equal number of channels to use as paired features for the discriminator to evaluate [39]. This allows for gradient flow without leading to unrealistic segmentation maps. We find this to be effective for Sim2Seg, as it circumvents the problems of different modalities and instead allows for stabler discrimination in feature space.

3.4 End-to-end Driving with RL

Using Sim2Seg, we train a short-horizon navigation policy using RL; specifically, we consider a goal-conditioned policy conditioned on visual and odometry data. Sim2Seg is inherently compatible with any visual learner, but we choose to use TD3 [40] with image augmentation [41] as our backbone RL algorithm. TD3 is an off-policy algorithm, and so enables goal-conditioning and relabelling in our training pipeline. We detail specific details about our architecture and hyperparameters in the supplementary materials.

Observation Space We adjust the TD3 backbone policy with our augmented visual observations and state information for our domain. Before inference, Sim2Seg translates the input RGB image $o_t \in \mathbb{R}^{(3,256,256)}$ into a one-hot, C-class segmentation map $c_t \in \mathbb{R}^{(C,256,256)}$. Additionally, we condition on egocentric past trajectory $\tau_p \in \mathbb{R}^{10,3}$, the current 2D state $s_a \in \mathbb{R}^2$, and the goal state $s_g \in \mathbb{R}^2$. The encoded segmentation map and state information are flattened and concatenated to achieve a final representation.

Action Space We parameterize the policy's actor as an LSTM which outputs a series of 5 action tuples, each consisting of a steering angle $\theta \in [\frac{\pi}{4}, -\frac{\pi}{4}]$, and acceleration $\alpha \in [0, 1])$. The agent then performs a temporal rollout of a trajectory, which is consumed by a lower-level controller for vehicle commands. This parameterization comes with several benefits; proposing rollouts instead of vehicle torque commands mitigates the dynamics sim-to-real gap. Furthermore, proposing multiple continuous actions allows the agent to propose waypoints and consider more coherent short-term plans, while giving a notion of safety when executing in the real world. We perform an analysis of different action parameterizations in Section 5.2.

Reward Function We design a simple reward function to best achieve goals while producing safe behaviors, notably obstacle avoidance:

$$r_t(s_g, s_a, a) = \lambda_g r_g(s_g, s_a) + \lambda_u r_u(s_g, s_a, a) + \lambda_s r_s(a) + \lambda_c r_c(s, a, s') \tag{3}$$

where r_g is a goal-conditioned sparse reward, r_u is an upright reward intended to incentivize smooth terrains (i.e. less rocky and flatter terrains), r_s is a steer penalty intended to discourage bang-bang control, and r_c is a collision penalty:

$$r_g(s_g, s_a) = \begin{cases} 100 & \text{if } \|s_g - s_a\|_2 < 2 \\ 0 & \text{otherwise} \end{cases} \qquad r_c(s, a, s') = \begin{cases} -30 & \text{if collision} \\ 0 & \text{otherwise} \end{cases}$$

$$r_u(s_g, s_a, a) = -10\frac{|\theta|}{180} \qquad r_s(a_{\theta,\alpha}) = -30\|\theta\|_2$$

where θ is the max of the roll and pitch angles between the vehicle body and world frames, and collisions are detected via Unity. To supplement training, we additionally leverage Hindsight Experience Replay [42]: by relabeling sampled trajectories (τ, s_a, s_g) with a goal achieved later in the same trajectory (τ, s_a, s'_a) during training, we obtain more signal from the sparse reward r_g.

3.5 Real-world Vehicle Integration

For real-world evaluation, we use a Polaris S4 1000 Turbo RZR equipped with a variety of perception sensors, including an inertial measurement unit (IMU), stereo camera pairs, and LiDARs (see Figure 3). Note that we only use a monocular RGB camera when deploying. The Polaris also includes computing resources and a "drive by wire" system to autonomously control the vehicle (accelerate, change gears, brake, steer). NeBula [43] has been integrated with the vehicle and utilizes a ROS stack with planners for varying goal horizons. To integrate our

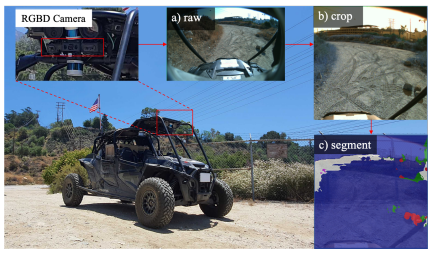

Figure 3: Polaris hardware with sensor suite. a) raw image, (b) cropped image as an input to Sim2Seg (c) segmented result using on-board computer.

model with the Nebula autonomy system, we further test in the ARL simulator, a photorealistic simulator of the Meadow environment already integrated with the ROS stack.

4 Experimental Setup

Preliminary: Classical Baseline We construct the classical baseline by leveraging the the Nebula software stack [43]. Our specific implementation uses localization estimates from a LiDAR's inertial odometry to fuse depth scans temporally which are used to estimate traversability cost using a settling-based geometric analysis [44]. This traversability map is used by a kinodynamic motion planner [45] to generate collision-free trajectories that are followed by a lower level PID-based tracking controller.

Offline Sim-to-Real Transfer Evaluation We evaluate our policy's ability to perform visual sim-to-real transfer using offline rosbag datasets of real-world, manual rollouts with the Polaris vehicle. Given rosbags of vehicle interactions, we first construct a dataset $\mathcal{D} := \{o, s_g, \tau_p^*; \tau^*\}$, where o is the observation at $t = 0$, s_g is the real achieved goal at some $t > 3$ seconds, τ_p^* is the past trajectory up to $t = 0$, amd τ is the vehicle's real-world trajectory from $t = 0$. We are then able to perform the model's full inference pipeline using o, s_g, and τ_p to produce a predicted trajectory τ. Following this method, we construct two different real-world datasets which features the vehicle navigating winding trails, rocks, and vegetation (see Figure 4).

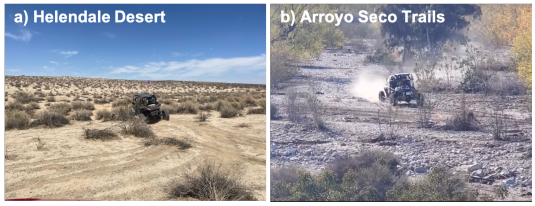

Figure 4: Real-world off-road evaluation data gathered by our platform at a) **Helendale**, Mojave Dessert and b) **Arroyo Seco Trails**, Altadena

Online Sim-to-Real Transfer Evaluation We also perform qualitative evaluation of closed-loop control with our policy fully in the real world using the system described in Section 3.5 in the Arroyo environment.

5 Experimental Results

We seek to answer two primary questions: (1) Is Sim2Seg able to efficiently reach goals and perform obstacle avoidance during zero-shot transfer? (2) What factors are most necessary for Sim2Seg's performance?

5.1 Goal Reaching and Obstacle Avoidance

We evaluate the policy's capacity for obstacle navigation through comparison against real world rollout τ^*, under the assumption that τ^* is an efficient trajectory to reach s_g while avoiding obstacles. We compare both the normalized L2 distance (denoted as **L2**) between the requested goal and the trajectory endpoint, and the cosine similarity between τ^* and τ (denoted as **GT**). A lower L2 indicates that the policy outputs trajectories that reach the goal (goal reaching), and a higher GT indicates that the policy aligns well with the expert trajectory (obstacle avoidance).

	Arroyo		**Helendale**	
Method	GT ↑	L2 ↓	GT ↑	L2 ↓
Classical	0.809	0.486	**0.859**	0.666
Sim2Seg	**0.869**	**0.408**	0.842	**0.654**

Table 1: Offline evaluation metrics of our method against a classical baseline. Sim2Seg is competitive with 'Classical' on real-world zero-shot transfer using primarily visual observations Note that the optimal L2 is 0 (passing through the goal), and the optimal GT is 1 (perfect trajectory reconstruction) .

To understand the gains provided by our method, we consider Sim2Seg's performance against a classical baseline, where we consider an existing classical autonomous stack as described in Section 3.5. The results are listed in Table 1. We note that Helendale is a difficult environment for several reasons; the operator drives fast, meaning that goals are typically more distant. In addition, the environment itself is very noisy, lacking high contrast between obstacles and consistently changing shadows.

Furthermore, we demonstrate zero-shot transfer by deploying the algorithm in a real-world off-road environment on a passenger-size vehicle shown in Figure 5. In the included example, the policy identifies the rock as an obstacle, and creates a trajectory to navigate around it. See supplementary material for videos.

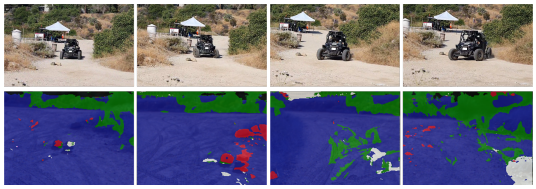

Figure 5: Timelapse of experimental demonstration of zero-shot sim2real transfer of goal following while avoiding obstacles. Please refer to video in supplementary material to see the entire run.

5.2 Ablation: Trajectory Parameterization

We additionally experiment with several different parameterizations for our policy: **'Simple'** predicts a single (θ, α) action, and manually rolls out this single action several times to artificially form a trajectory. **'MLP'** predicts all 5 (θ, α) actions simultaneously, removing the temporal dependence provided by a recurrent model. **'LSTM'** is our final model, described previously in Section 3.4. For each considered parameterization, we train a policy using identical Sim2Seg networks trained on the Meadow environment, and then perform offline-evaluation on the Arroyo dataset.

As noted in Table 2, MLP and LSTM parameterizations perform similarly, suggesting the importance of multi-step planning moreso than temporal dependence for creating trajectoies that approach the goal. We also make qualitative observations of action modes in Figure 7; the Simple trajectory rapidly shoots off-course, despite penalizing drastic steering actions through the reward function in training. We hypothesize that this behavior may be excusable in training as the agent re-plans its trajectory, effectively amounting to one-step planning. However, during inference, this results in unstable behavior, especially at high speeds.

Action	GT ↑	L2 ↓
Simple	0.854	0.807
MLP	0.879	0.709
LSTM	0.869	**0.408**

Table 2: Offline real evaluation of different action parameterizations on the Arroyo dataset using the Meadow Sim2Seg model.

5.3 Ablation: Simulation Training

Since simulation training is flexible and scalable, we consider various ablations that may improve performance on real-world offline evaluations. We explore two primary factors:

- **Diverse Simulator Scenes**: We explore the effectiveness of using a model trained on all three chosen environments versus the best performance of one trained on a single environment.

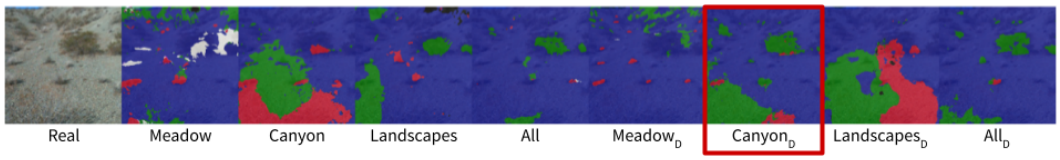

Figure 6: Qualitative zero-shot transfer results of Sim2Seg models trained on differing training datasets as described in 5.3. X_D indicates environment X trained with depth supervision. Red square depicts the model chosen to run in the real world. More examples are included in the supplementary materials.

- **Sim2Seg Depth Supervision**: Depth may provide useful learning signals to the Sim2Seg model, encouraging representations specifically to better recognize occluding obstacles such as trees and rocks, which may be useful for the agent.

Figure 6 also shows qualitative differences of Sim2Seg trained with and without depth supervision, while Table 3 shows the quantitative results of this comparison. Notably, training with segmentation maps from the depth-aware Sim2Seg seems to improve performance for the simple action mode, perhaps because accurate segmentation of obstacles is critical when the policy can only take 1-step actions. When predicting ultiple actions, training on a diverse set of simulators drastically improves performance: multi-step planning seems to reduce the importance of the depth-aware Sim2Seg, while access to diverse terrains becomes increasingly important.

Action	Sim	L2 ↓	GT ↑
Simple	Meadow	0.807	0.854
	Meadow_D	0.338	0.863
	All	0.856	0.852
MLP	Meadow	0.709	0.879
	Meadow_D	0.676	0.872
	All	0.338	0.863
LSTM	Meadow	0.408	0.869
	Meadow_D	0.671	0.874
	All	0.383	0.837

Table 3: We compare how sim training affects real-world evaluation on the Arroyo dataset. Training with depth-aware Sim2Seg Meadow_D improves performance; training with multiple simulations likewise helps.

6 Conclusion and Limitations

In this paper, we have investigated transferring end-to-end off-road autonomous driving policies from simulation to reality. To accomplish this, we have presented Sim2Seg, which converts randomized simulated RGB images into segmentation masks, and subsequently enables real-world images to also be converted. Given that our driving policy is trained in these canonical segmentation environments, it is possible to run policies trained in simulation directly in the real world. When evaluating on real-world data, we are able to perform equally as well as a classical perception and control stack that took thousands of engineering hours over several months to build.

Limitations Improving the quality of the Sim2Seg model is a key priority; qualitative analysis of segmentation maps shows difficulty with shadows (which may be perceived as new objects) and low-contrast scenes (such as the small shrubs depicted in Figure 6). We hope that such shortcomings can be addressed by introducing stronger shadow randomization techniques, and generally increasing the number of environments trained on.

Currently, Sim2Seg's policy is trained on horizons of 20 meters in length, and thus is only effective at this range. An effective horizon on the range of 50 meters or more would likely be much more practical, especially when navigating at higher speeds. Our agent currently is capable of avoiding obstacles in short range as demonstrated in Section 5.1; however, long horizon reasoning, such as needing to traverse around a forest to reach a goal, has not been tested. Training with temporally and spatially longer horizons inherently introduces more complexity to RL; this is an active area of research for us.

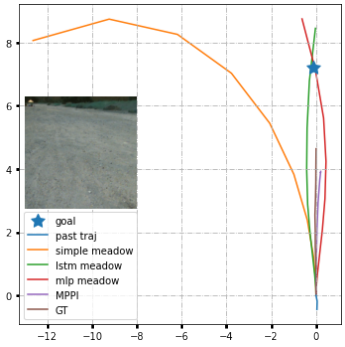

Figure 7: Sample rollouts from each of the considered action modes in 5.2, with the visual observation pictured on the left. Despite there being few obstacles in the scene, the Simple agent proposes a trajectory far from the requested goal.

One particular area of interest is trajectory proposals in terms of waypoints. Several of the action modes investigated in this work perform well during zero-shot transfer, but rely on the temporal rollout to effectively propose trajectories. Waypoint proposals would allow us to explore other methods such as Bezier curve parameterizations and discrete trajectory libraries [46], and would provide further proof that Sim2Seg can generalize to different policy architectures.

References

[1] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 12627–12637, 2019.

[2] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke. Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping. *IEEE Intl. Conference on Robotics and Automation*, 2018.

[3] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE Intl. Conference on Intelligent Robots and Systems*, 2017.

[4] S. James, A. J. Davison, and E. Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. *Conference on Robot Learning*, 2017.

[5] J. Matas, S. James, and A. J. Davison. Sim-to-real reinforcement learning for deformable object manipulation. *Conference on Robot Learning*, 2018.

[6] S. James, M. Bloesch, and A. J. Davison. Task-embedded control networks for few-shot imitation learning. In *Conference on Robot Learning*, pages 783–795. PMLR, 2018.

[7] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. Cun. Off-road obstacle avoidance through end-to-end learning. *Advances in neural information processing systems*, 18, 2005.

[8] P. Fankhauser, M. Bloesch, C. Gehring, M. Hutter, and R. Siegwart. Robot-centric elevation mapping with uncertainty estimates. In *International Conference on Climbing and Walking Robots (CLAWAR)*, 2014.

[9] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, 34(3):189–206, 2013.

[10] K. Otsu, G. Matheron, S. Ghosh, O. Toupet, and M. Ono. Fast approximate clearance evaluation for rovers with articulated suspension systems. *Journal of Field Robotics*, 37(5):768–785, 2019. ISSN 15564967. doi:10.1002/rob.21892.

[11] K. Konolige, M. Agrawal, M. R. Blas, R. C. Bolles, B. Gerkey, J. Sola, and A. Sundaresan. Mapping, navigation, and learning for off-road traversal. *Journal of Field Robotics*, 26(1): 88–113, 2009.

[12] D. D. Fan, K. Otsu, Y. Kubo, A. Dixit, J. Burdick, and A.-A. Agha-Mohammadi. Step: Stochastic traversability evaluation and planning for risk-aware off-road navigation. *arXiv preprint arXiv:2103.02828*, 2021.

[13] P. Krüsi, P. Furgale, M. Bosse, and R. Siegwart. Driving on point clouds: Motion planning, trajectory optimization, and terrain assessment in generic nonplanar environments. *Journal of Field Robotics*, 34(5):940–984, 2017.

[14] P. Papadakis. Terrain traversability analysis methods for unmanned ground vehicles: A survey. *Engineering Applications of Artificial Intelligence*, 26(4):1373–1385, 2013. ISSN 09521976. doi:10.1016/j.engappai.2013.01.006. URL `http://dx.doi.org/10.1016/j.engappai.2013.01.006`.

[15] A. Amini, I. Gilitschenski, J. Phillips, J. Moseyko, R. Banerjee, S. Karaman, and D. Rus. Learning robust control policies for end-to-end autonomous driving from data-driven simulation. *IEEE Robotics and Automation Letters*, 5(2):1143–1150, 2020. doi:10.1109/LRA.2020.2966414.

[16] Y.-J. R. Chu, T.-H. Wei, J.-B. Huang, Y.-H. Chen, and I.-C. Wu. Sim-to-real transfer for miniature autonomous car racing, 2020. URL https://arxiv.org/abs/2011.05617.

[17] B. Osinski, A. Jakubowski, P. Milos, P. Ziecina, C. Galias, S. Homoceanu, and H. Michalewski. Simulation-based reinforcement learning for real-world autonomous driving, 2019. URL https://arxiv.org/abs/1911.12905.

[18] Y. R. Chu, T. Wei, J. Huang, Y. Chen, and I. Wu. Sim-to-real transfer for miniature autonomous car racing. *CoRR*, abs/2011.05617, 2020. URL https://arxiv.org/abs/2011.05617.

[19] U. R. Nair, S. Sharma, U. S. Parihar, M. S. Menon, and S. Vidapanakal. Bridging sim2real gap using image gradients for the task of end-to-end autonomous driving, 2022. URL https://arxiv.org/abs/2205.07481.

[20] J. Ram, Jan 2022. URL https://theses.liacs.nl/pdf/2021-2022-RamJ.pdf.

[21] A. Dosovitskiy, G. Ros, F. Codevilla, A. M. López, and V. Koltun. CARLA: an open urban driving simulator. *CoRR*, abs/1711.03938, 2017. URL http://arxiv.org/abs/1711.03938.

[22] F. Sadeghi and S. Levine. CAD2RL: Real single-image flight without a single real image. In *Robotics: Science and Systems*, 2017.

[23] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks, 2017.

[24] Z. Yi, H. Zhang, P. Tan, and M. Gong. Dualgan: Unsupervised dual learning for image-to-image translation, 2018.

[25] D. Ho, K. Rao, Z. Xu, E. Jang, M. Khansari, and Y. Bai. Retinagan: An object-aware approach to sim-to-real transfer, 2020. URL https://arxiv.org/abs/2011.03148.

[26] K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, and M. Khansari. Rl-cyclegan: Reinforcement learning aware simulation-to-real. *CoRR*, abs/2006.09001, 2020. URL https://arxiv.org/abs/2006.09001.

[27] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. 2015. doi:10.48550/ARXIV.1505.07818. URL https://arxiv.org/abs/1505.07818.

[28] B. Sun, J. Feng, and K. Saenko. Return of frustratingly easy domain adaptation, 2015.

[29] N. Sünderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford, and P. Corke. The limits and potentials of deep learning for robotics, 2018.

[30] Realistic buggy kit. https://assetstore.unity.com/packages/tools/physics/realistic-buggy-kit-62978. Accessed: 2022-02-13.

[31] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, and D. Lange. Unity: A general platform for intelligent agents, 2020.

[32] Meadow environment - dynamic nature. https://assetstore.unity.com/packages/3d/vegetation/meadow-environment-dynamic-nature-132195. Accessed: 2022-03-02.

[33] Real landscapes - valley forest. https://assetstore.unity.com/packages/3d/environments/real-landscapes-valley-forest-194338. Accessed: 2022-04-18.

[34] Castle valley collection 1. https://assetstore.unity.com/packages/3d/environments/landscapes/castle-valley-collection-1-165032. Accessed: 2022-04-18.

[35] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks, 2016. URL https://arxiv.org/abs/1611.07004.

[36] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL http://arxiv.org/abs/1505.04597.

[37] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax, 2017.

[38] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel. Learning visual feature spaces for robotic manipulation with deep spatial autoencoders. *CoRR*, abs/1509.06113, 2015. URL http://arxiv.org/abs/1509.06113.

[39] A. Abdollahi, B. Pradhan, S. Gite, and A. Alamri. Building footprint extraction from high resolution aerial images using generative adversarial network (gan) architecture. *IEEE Access*, 8, 11 2020. doi:10.1109/ACCESS.2020.3038225.

[40] S. Fujimoto, H. Van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. *Intl. Conference on Machine Learning*, 2018.

[41] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning, 2021. URL https://arxiv.org/abs/2107.09645.

[42] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay, 2017. URL https://arxiv.org/abs/1707.01495.

[43] A. Agha, K. Otsu, B. Morrell, D. D. Fan, R. Thakker, A. Santamaria-Navarro, S.-K. Kim, A. Bouman, X. Lei, J. Edlund, et al. Nebula: Quest for robotic autonomy in challenging environments; team costar at the darpa subterranean challenge. *arXiv preprint arXiv:2103.11470*, 2021.

[44] R. Thakker, N. Alatur, D. D. Fan, J. Tordesillas, M. Paton, K. Otsu, O. Toupet, and A.-a. Agha-mohammadi. Autonomous off-road navigation over extreme terrains with perceptually-challenging conditions. In *International Symposium on Experimental Robotics*, pages 161–173. Springer, 2020.

[45] G. Williams, A. Aldrich, and E. A. Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017.

[46] N. Dashora, D. Shin, D. Shah, H. Leopold, D. Fan, A. Agha-Mohammadi, N. Rhinehart, and S. Levine. Hybrid imitative planning with geometric and predictive costs in off-road environments, 2021. URL https://arxiv.org/abs/2111.10948.