

Learning to Take Cover on Geo-Specific Terrains via Reinforcement

Learning

Tim Aris, Volkan Ustun, Rajay Kumar

University of Southern California Institute for Creative Technologies, Playa Vista, CA, USA
timjaris@gmail.com, ustun@ict.usc.edu, kumar@ict.usc.edu

Abstract

This paper presents a reinforcement learning model designed to learn how to take cover on geo-specific terrains, an essential behavior component for military training simulations. Training of the models is performed on the Rapid Integration and Development Environment (RIDE) leveraging the Unity ML-Agents framework. We show that increasing the number of novel situations the agent is exposed to increases the performance on the test set. In addition, the trained models possess some ability to generalize across terrains, and it can also take less time to retrain an agent to a new terrain, if that terrain has a level of complexity less than or equal to the terrain it was previously trained on.

Introduction¹

Reinforcement learning (RL) aims to produce optimal policies in given environments. While there has been significant progress, learning to navigate a 3D terrain remains a challenge for RL systems. To date, most work in such environments uses pixels as the primary input to the models. This paper poses the question of what if rays are used instead. The motivation would be to provide the agent with observations that are more abstract and hence, less inferential distance to understand the correlation between the observation and good actions. One example of ray-based observations would be the LIDAR system for autonomous cars.

The specific task chosen was for the agent to take cover in realistic environments from a stationary opponent by finding a location where the agent is not visible from the opponent's perspective. The motivation behind this

selection is to explore learning robust neural behavior models that can be used as a component of more complex behaviors in the Rapid Integration and Development Environment (RIDE), a Unity-based military training simulation environment (Hartholt et al. 2021). Thus, agents were trained in the RIDE platform using geo-specific terrains and leveraging the ML-Agents framework within Unity.

This paper shows that increasing the number of novel situations the agents are trained on increases the generalizability of the agents. Furthermore, it shows that the trained models possess some ability to generalize across terrains, and it can take much less time to retrain an agent on a new terrain than it does to train a new agent from scratch on that terrain.

Background

Many RL systems operating in 3D environments use pixels for observations. For example, DeepMind's open-ended learning agents (O.E.L. Team et al. 2021) take information about their goal and an object they may be holding, but the main observation is an RGB pixel image. Likewise, the primary observation in the MineRL environment (Guss et al. 2019) is a grayscale image.

One example of something similar to this paper is OpenAI's hide-and-peek environment (Baker et al. 2019), which uses LIDAR-inspired rays for agents to find ways to be occluded from other agents. However, their focus is on multi-agent scenarios rather than navigating real world terrain. Other work on analyzing raycasts in Unity include

¹Copyright © 2022, by the authors. All rights reserved.

dodgeball (Gorgan et al. 2019), navigation (Alonso et al 2020), and self-driving cars (Hossain et al. 2019).

ML-Agents (Juliani et al. 2018) is a toolkit that facilitates the training of agents from within the Unity game engine. RIDE is a middleware built on the Unity game engine that facilitates rapid development and prototyping of simulated environments. Furthermore, RIDE supports Machine Learning experiments through the ML-Agents framework, allowing to directly train RL agents within the high-fidelity military training simulation environments based on geo-specific terrains.

Taking Cover

“Taking cover” is defined as having more than 88% of the sampled points on the agent occluded from a stationary opponent. In other words, the goal is to have an object in between the agent and the opponent such that the agent is no longer visible. An agent is also considered in cover if they are sufficiently far away from the opponent because it would be impractical for an opponent to shoot the agent from that distance. However, in the terrains that have been used, it’s almost always faster to take cover by hiding behind an object. Finally, an agent is considered to have failed if they have not taken cover within the maximum number of timesteps for the episode, which is 5,000 for our experiments.

The Agent

Agents were trained using the PPO algorithm (Schulman et al. 2017). The primary observation of the agent consists of Unity’s rays. The shape of these rays is described in Figure 1. Additionally, the location of the opponent, relative to the agent’s location, is added to the observation space. The observation space is stacked from the previous two timesteps.

The default action space with seven actions - do nothing, go forward, go left, go right, go backward, turn to the left, and turn to the right - is leveraged in our experiments.

Upon successfully completing an episode, agents are rewarded with a quantity equal to $5 + 5X$, where X is the percentage of the timesteps left in the episode. Such reward design gives a solid signal to distinguish just barely completing the episode before the time runs out and provides an additional incentive for speed. The agents also have a step penalty of $1/\text{max_steps}$, so in practice, rewards range from -1 to just under 10. Exploratory experiments have shown that our models are not very sensitive to changes in reward structure, so we have kept this as the reward design.

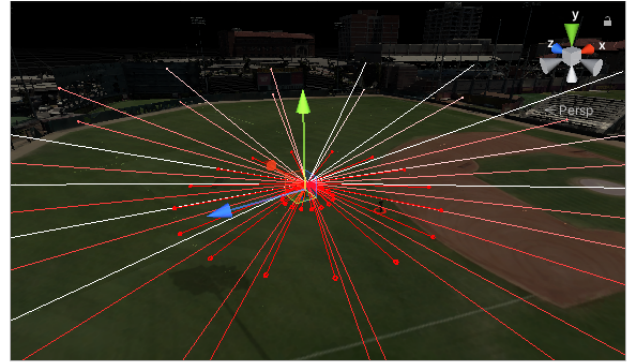


Figure 1: The rays that make up the majority of the agent’s observations. These rays are represented to the agent as, for each ray, the length of the ray, and whether the ray hit the opponent, terrain, or nothing. The shape above was chosen for the virtue of being the most useful for a human player when the human tried the task with only rays visible.

The Environment

Agents were trained on the geo-specific, drone-captured terrains of the University of Southern California (USC) campus and Catalina island. Agents and opponents spawn in 16×16 unit square areas on the terrain. Areas are chosen by sampling a random point on the Nav Mesh, checking whether the point is contiguous with the main mesh (to avoid roofs or places where occlusion is impossible), and checking if the square around the point is on the mesh to avoid agents spawning into walls. Finally, agents are trained on a timescale of 5, as values higher than 5 seemed to destabilize the agent’s movements.

Experiments

This paper showcases four training runs, and two distinct tests on them. In addition, it includes 3 more runs designed to answer the question of whether or not curriculum learning is helpful for the task of taking cover, as well as how area selection during training impacts performance in testing.

First, agents are trained on the complete USC and Catalina sets, gradually being introduced to areas via curriculum. Then, those two models are retrained on the opposite terrain. The base USC model is tested at multiple points from its training, as shown in Figure 3, then all four models are tested on the USC and Catalina terrains, on both training and testing sets. Figure 2 shows the cumulative reward across the various training runs.

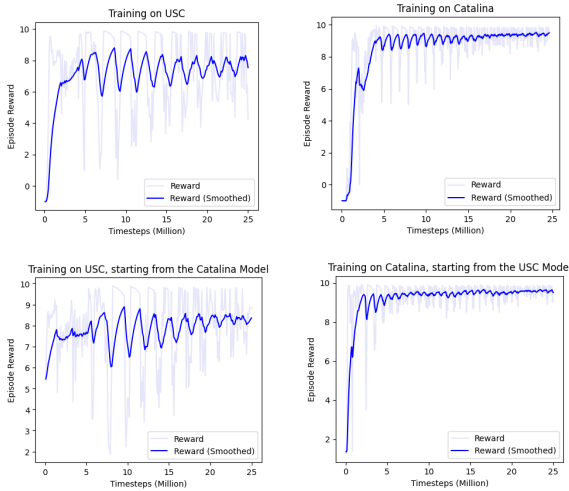


Figure 2: Training plots for USC and Catalina with curriculum learning. The cyclical pattern can be explained by the way areas are scheduled. When an area is finished, the area is turned off, meaning the high-reward areas finish first. As time goes on, agents get less reward, until eventually all the areas that were never solved complete at once. The purpose of the schedule is to prevent high-earning areas from being over-represented in the data, as they would if they were instantly refreshed.

Increasing the number of areas trained on increases generalizability

Agents were trained via curriculum learning. Agents started training in 1 area, and when they received an average reward of 8 or higher, the number of areas trained in would double, until they got to the maximum number of areas for that terrain (192 for USC and 152 for Catalina). This was later found not to be significantly better than training on all areas at once, but we use this to show that the generalizability of the agent increases with the number of areas it is trained on.

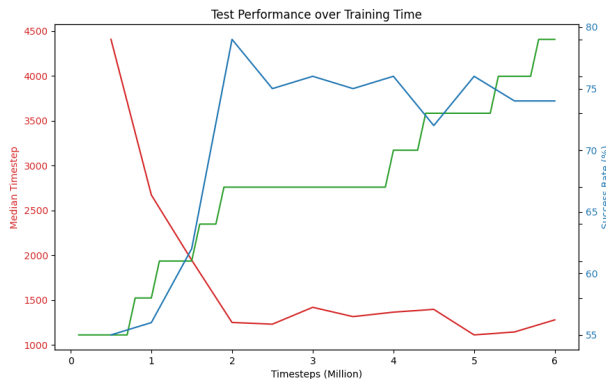


Figure 3: Median finish time (in timesteps) and success rate of the base USC model on the USC test set as a function of how many timesteps the model was trained on. The green line represents the lesson number, with the lowest being 1 area, then doubling with

every step, until the highest point which represents 192 areas being trained on.

Figure 3 shows that as the number of areas the agent trains on increases, the success rate on the testing set increases, and the median time to finish an episode decreases. However, Figure 3 also depicts diminishing returns over time, with no appreciable performance gain past 16 areas. Such diminishing returns indicate that the number of areas required to train an agent on a terrain is roughly 16, though finding that number for different terrains or different parameters will require further research.

Transfer Learning

In Catalina, the USC model achieved peak performance at around 2.2 million timesteps, while model training from scratch did so at around 4.3 million. This mirrors when each model got to the final lesson to train on the full training set showing that using a pre-trained model can substantially reduce the time required to train a model.

In USC, the Catalina model achieved peak performance at around 6.8 million timesteps, while the model training from scratch took only 5.8. One possible explanation for this would be that Catalina is an easier terrain, and pre-training only helps when transferring to a terrain of equal or lower complexity. However, it remains the case that having a pre-trained model is not a guarantee to cut the training time in half.

Test Results

All models were tested in 4 environments: USC training, USC testing, Catalina training, and Catalina testing. Additionally, a human was also tested on the environments to provide a baseline. The human was tested with both pixels, as well as a version of the environment with only rays visible, such that the human could only use the information the agents had access to. Based on this baseline, the RL agents achieve human-level performance in both the training and test sets of the terrains they were trained on.

Table 1: Test results, the numbers are the median finish time of the agents (lower is better), followed by the success rate for. The rows are the environment the model was trained on (or the observations for the human player), and the columns are where the model was tested.

Model \ Tested on	USC 192	USC Test 34	Catalina 152	Catalina Test 15
USC 192	1092 (84%)	932 (83%)	1546 (73%)	2294 (80%)
Catalina	2498	1481	723	819

152	(63%)	(72%)	(97%)	(91%)
USC to Catalina	2261 (61%)	1093 (71%)	565 (98%)	648 (99%)
Catalina to USC	1012 (89%)	895 (82%)	1401 (72%)	3396 (56%)
Human (Pixels)	507 (100%)	389 (100%)	314 (100%)	510 (100%)
Human (Rays)	1455 (92%)	715 (97%)	596 (100%)	536 (100%)

The primary takeaway is that performance is comparable to a human level on the terrains that models were trained if the human is restricted to the same information the agent has. Such comparison is accurate for both the training and testing set, which the agents had not seen before they were tested on it. Furthermore, performance on training and test sets also indicates that the models are not over-fitted and are indeed learning the specifics of how to navigate their particular terrain. Other interpretations of these numbers include that those pre-trained models have a better final performance, but the effect size does not seem to be large enough to say it is significant. It is not surprising, but agents lose some of their applicability to their original terrains when retrained to a new one. Such an effect could be mitigated by training on both. Another interpretation is that these agents do contain some applicability to alternate terrains. Their reliability goes down, and they go from roughly human-level to taking 2-3x as long as a human to complete an episode, but they still succeed more often than they fail. One final point is there does seem to be potential for improvement by incorporating pixel data. Some exploratory experiments showed that rays outperformed pixels, but a more in depth comparison between pixels and rays is left for future work.

16 Areas

To get a more reliable indicator of whether 16 areas was the optimal number, and if curriculum learning was helpful, 3 scenarios with 4 models each were trained. The median finish time and standard deviation were recorded per scenario. For curriculum learning, the average median finish time was 2540, with a standard deviation of 1370. Without curriculum learning, those numbers were 1470 and 191. Trying again with no curriculum, but selecting a different 16 area set from the original resulted in the numbers 3140, and 1280. Interpreting these numbers, the amount of areas as well as final performance are functions of some qualities of the areas being trained on. Additionally, curriculum performance gives the same best-case performance, but increases volatility.

Conclusions

This work shows that RL ray-agents can learn to take cover in novel terrains, and it shows that the more areas an agent is exposed to, the more it will generalize, to a point. Furthermore, it shows that training time can be substantially reduced by starting from a model that was trained in a different terrain of greater or equal complexity. In addition to comparing pixel observations, future work will be experimenting on more terrains to further investigate transferability, as well as what qualities of a training area lead to the best test set performance.

Acknowledgments

Part of the effort depicted is sponsored by the U.S.Army Research Laboratory (ARL) under contract number W911NF-14-D-0005, and that the content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

- Alonso, E., Peter, M., Goumar, D. and Romoff, J., 2020. Deep reinforcement learning for navigation in aaa video games. *arXiv preprint arXiv:2011.04764*.
- Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B. and Mordatch, I., 2019. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*.
- Gorgan, D., 2019. Performance Analysis in Implementation of a Dodgeball Agent for Video Games. *International Journal of User-System Interaction*, 12(4), pp.225-240.
- Guss, W.H., Houghton, B., Topin, N., Wang, P., Codel, C., Veloso, M. and Salakhutdinov, R., 2019. MineRL: A large-scale dataset of Minecraft demonstrations. *arXiv preprint arXiv:1907.13440*.
- Hartholt, A., K. McCullough, E. Fast, A. Reilly, A. Leeds, S. Mozgai, V. Ustun, and A. S. Gordon. 2021. "Introducing RIDE: Lowering the Barrier of Entry to Simulation and Training through the Rapid Integration & Development Environment". 2021 Virtual Simulation Innovation Workshop.
- Hossain, S. and Lee, D.J., 2019. Autonomous-driving vehicle learning environments using unity real-time engine and end-to-end CNN approach. *The Journal of Korea Robotics Society*, 14(2), pp.122-130.
- Juliani, A., Berges, V. P., Teng, E., Cohen, A., Harper, J., Elion, C., ... & Lange, D. (2018). Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*.
- Schulman J., Wolski F., Dhariwal F., Radford A., and Klimov O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- Team, O.E.L., Stooke, A., Mahajan, A., Barros, C., Deck, C., Bauer, J., Sygnowski, J., Trebacz, M., Jaderberg, M., Mathieu, M. and McAleese, N., 2021. Open-ended learning leads to generally capable agents. *arXiv preprint arXiv:2107.12808*.