Shielding Induced Safe Reinforcement Learning For Drone Navigation

Daniel Bramblett, Vivek Sahukar
Group #1
School of Computing and Augmented Intelligence
Arizona State University, AZ, USA
{drbrambl,vsahukar}@asu.edu

Introduction

Autonomous drones have been an active area of research, with work including safely learning how to traverse in real-world environments. One use case for these types of drones is agriculture, where they are increasingly used to improve farm operations' efficiency and yield, such as surveillance, crop health monitoring, and pesticide treatment. In general, drone users require the ability to either safely train a drone on their specific task or have some guarantees of safety.

Drone navigation needs to fly agile and autonomously to complete its tasks in all environments. Unsafe behavior can cost the drone while also putting other people and agents who share that environment at risk. This makes reinforcement learning challenging since unsafe behavior has to be performed to learn that it is unsafe.

This paper examines using shielding on reinforcement learning agents for faster and safer drone navigation training. We evaluate constructing shields for learning obstacle avoidance in continuous state and action spaces. We explore three shield construction approaches for shielding agents in a continuous state and action space.

Related Work

A framework, "FoRShield", was proposed by (Potok 2018) for ensuring safety in reinforcement learning (RL) systems with a particular application to drones for fire-fighting. FoRShield employs Actor-Critic RL algorithms and reachability analysis to create a protective "shield" that filters out unsafe actions during training and operation. The shield dynamically evaluates actions based on safety and efficiency, adapting to different environmental conditions. The paper (Ugurlu, Pham, and Kayacan 2022) introduces a novel deep reinforcement learning (DRL) framework tailored for autonomous aerial robots navigating in dense environments. The authors employed a forward-facing depth camera to enable the DRL agent to identify and navigate around obstacles within a rough global path. The framework focused on ensuring safety during end-to-end path planning by embedding safety boundaries during training, designed to prevent the agent from encountering hazardous situations. (Hodge, Hawkins, and Alexander 2021) discusses safety assurance for the drone navigation system that uses deep RL to guide drones based on sensor data. Firstly, the authors conduct functional failure analysis (FFA) to identify all possible system function failures that could lead to hazardous situations, such as collisions. This helps define specific safety requirements the system must meet. Then, the paper discusses the three assurances, viz. training, learned model, and overall drone performance, to generate confidence that the navigation system will not lead to harmful outcomes.

Our research is based on work by Alshiekh et al. (2018) on converting LTL safety specifications into a shield for training an RL agent. We also evaluate safety performance during training. The main difference is that we are working with a continuous state and action space, meaning that it is impossible to derive the agent's safe region for the active shield.

Technical Approach

Overall Technical Approach

The problem we are solving is reducing the time and safety risks associated with training an autonomous drone to navigate an environment safely. To do this, we found an existing drone environment and modified it to create a source of risk the drone must learn to avoid. We then implemented three shield variations and evaluated the safety and efficiency of training agents with each.

Environment We used the Ingenuity environment based in Isaac Gym (Makoviychuk et al. 2021) for our project. Isaac Gym was created to perform parallel end-to-end GPU training, resulting in faster training and rendering. These benefits come at the cost of increased difficulty in creating and/or modifying environments because each piece of the code must work with all parallel instances simultaneously. Also, Gym manages the objects making it not trivial to figure out how forces are being applied.

The Ingenuity environment is based on NASA's Ingenuity helicopter, which has 6 degrees of freedom actions for the force vectors of the lower and upper rotors. A target location is randomly generated between [-5,5] on the x and y axis and [1,2] on the z. Using its orientation and relative distance to the target, the helicopter must traverse to the target.

A dense reward function is provided that gives a positive reward based on how close the helicopter is to the target, how upright it is, and how little it is spinning. These rewards help the helicopter learn to remain upright, not spin out of control, and try to get closer to the target. The rewards received each timestep is bounded between (0, 7].

We introduced an obstacle into the environment at a fixed location $(0.5, 0.5, 0.5)$. If the agent hits the obstacle, it is penalized with a -10,000 reward. The agent's cumulative reward is negative if the policy is unsafe.

The environment's observations were also modified to include the drone's relative location to the obstacle. Otherwise, the obstacle would be invisible to the agent, making it impossible to train to avoid it.

Shield Implementations To construct the shield, we decided to make the shield satisfaction condition *"the agent must always be 0.6 or greater away from the obstacle"*. While this seems arbitrary, such a format allows the user's preferences to be used as a shield to allow the reinforcement learning agent to learn how to maximize its rewards while remaining compliant.

In the case the safety condition is not satisfied, there are two ways the shield can interact: give feedback to the agent through its reward function and replace the action with a safer action. For the feedback, we remove the positive reward received by the agent. Due to the agent learning to maximize its rewards, it must learn how to satisfy the shield. Also, by not punishing the agent with a negative reward, we can still differentiate whether the agent's behavior is safe or not using just the expected value.

To find a safe action, we used the locations of the obstacle and agent to calculate the quickest direction to move the agent away from the obstacle. Using the agent's location $\{x_a, y_a, z_a\}$ and the obstacle's location $\{x_o, y_o, z_o\}$, the direction vector that moves the agent the quickest away is $\overrightarrow{d} = \{x_a - x_o, y_a - y_o, z_a - z_o\}$. The unsafe action is replaced with an action equivalent to pushing the agent with a certain force using $\overrightarrow{d}$.

Using the feedback and safe action replacement approaches, we propose three different shields:

1. **Hard Shield:** Replaces the action with the safe action.
2. **Soft Shield:** Gives feedback to the agent.
3. **Hybrid Shield:** Both replace the action with the safe action and give feedback to the agent.

Tasks Conducted by Each Team Member

We used pair programming when modifying the Ingenuity environment and writing the shield variations. Beyond that, here are the individual tasks we each performed:

Daniel Bramblett: Explored other drone environment options, including implementing the same task of avoiding the obstacle in the OpenAI environment gym-pybullet-drones (Panerati et al. 2021). He also constructed the algorithm for selecting the safe action.

Vivek Sahukar: Set up and modified Isaac Gym and ran the experiments. Additionally, researched existing work on safe AI with regards to drones.

Results and Observations

We evaluated the safety and convergence when Advantage Actor-Critic (A2C) trained an agent on this task for each shield for the same ten seeds. We evaluated the average and standard deviation of the rewards for the first 10,000 steps. We trained without a shield to use as a baseline. The results are shown in Figure 1.

The main results are that the shield performed worse than the no-shield baseline. All experiments performed similarly badly in the first steps with an extremely high negative reward. However, it took the baseline only 16 steps to start receiving a positive reward, while it took 175 to 262 steps for the shield approaches. Then, the baseline learned faster by reaching 6 thousand rewards seven to ten times quicker than the hard or soft shield. Furthermore, the hybrid shield caused the agent to fail to learn 20% of the time. However, the soft shield was more stable than the baseline.

Observing the agents running, we noticed that the no-shield baseline causes the agent to go higher up than the shielding approaches to avoid the obstacles. Implying that it learned quickly to avoid the area around the obstacle.

A key observation explaining these results is that the initial step of all these approaches results in the same negative reward for each seed. We examined whether the drone is starting in the obstacle's crashing range and found that, while this does reduce the negative reward slightly, it does not prevent the hard shield from becoming negative, implying this is only part of the problem.

One hypothesis we could not test is that Isaac Gym applies forces based on the object's orientation. This means we must adjust our safe actions to apply force in the desired direction based on the orientation. This seems to be the case when examining the hard shield run video where the drone maintains an orientation of facing away from the obstacle. Learning to maintain this orientation explains why the hard and hybrid shields trained slower.

The soft shield also trained slower because we removed the rewards, reducing the information the agent needed to learn. For example, if the target was close to the obstacle, the agent might not receive a higher positive reward if it went closer to the target, resulting in it learning slower.

We reran these results multiple times to verify they are repeatable and are not a coding bug.

Safety Dimensions Addressed

Using shielding for safe reinforcement learning prevents side effects and helps the agent learn how to complete its objective using safe behavior. Furthermore, shielding protects the agent and environment against unsafe computed behavior by replacing unsafe actions. As such, our work addresses two main safety dimensions: **AI Objective vs AI Behavior** and **Computed Behavior vs Real Outcome**.

In the case the shield is derived from the user's preferences, **Intent vs Specification** and **Intent vs Outcome** are also addressed since the shield is being used to help the user convey their intent on what the agent should not do while completing its objective.

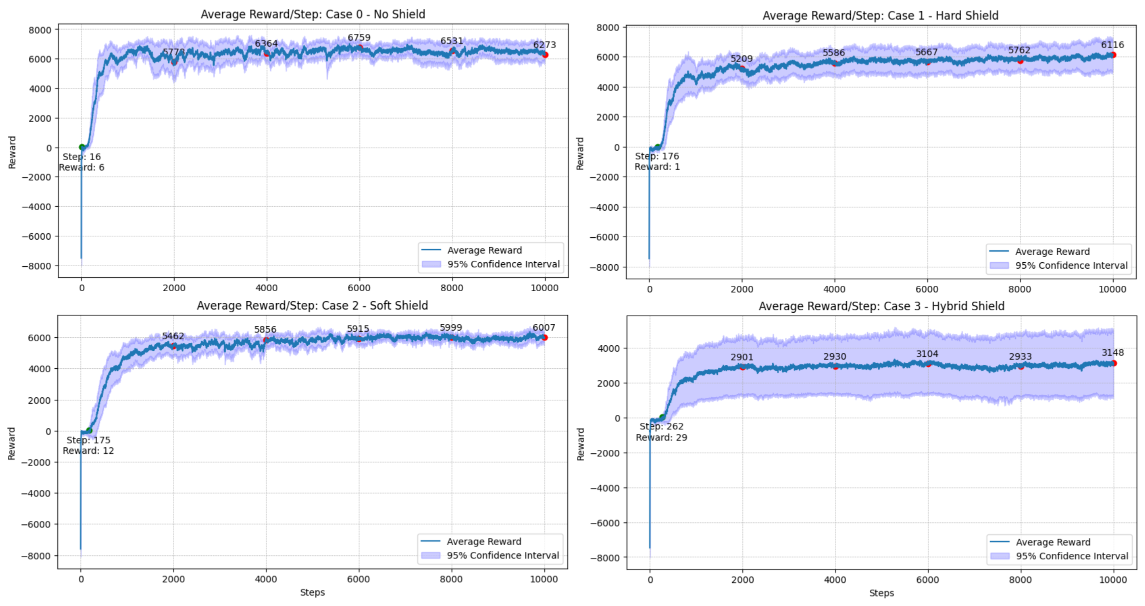

Figure 1: Empirical results of the rewards of training each agent using reinforcement learning with different shield variations. The top left graph is the baseline when the agent learned without a shield.

Conclusions

For this project, we examined using shielding for safe reinforcement learning training in drone navigation. We implemented three shields for training an A2C agent in a modified Ingenuity environment. We found that adding a shield isn't trivial, resulting in the shield approaches performing worse. The hard shield made it more challenging for the agent, and the soft shield prevented the agent from learning.

Based on the poor performance of the current safe action and feedback, we want to explore different safe action and feedback approaches and see how they affect the training performance.

Due to not having direct access to the physics engine, another area of research would be exploring how to apply shielding to a black box agent. We have considered two approaches. First, use an environment-level shield during training, reset the environment, or give the agent direct feedback on the outcome. However, this approach cannot be active outside of training. The second approach would be to learn the underlying model through training.

Another major direction for future work is exploring creating shields that can automatically handle more dynamic environments. For example, if we add horizontal wind or create different obstacles and targets, the shield should automatically guarantee some degree of safety.

An interesting question we also want to examine is how to extend shielding into partial-observable environments. This would require querying the current belief state and determining the confidence needed to decide whether an action satisfies a safety requirement. Then, the question is how to figure out safe actions.

The objective of the current environment implementation is to avoid obstacles while reaching the target location, which allows for constructing a clear region that the drone should avoid. This is why a hard shield is even possible for this problem, or the baseline can quickly find a strategy around the obstacle. For example, if the target location is in close proximity to the obstacle, the agent cannot avoid the obstacle. This would require a soft shield or a hard shield that is precise enough to allow for these proximity cases. We are planning to explore solving such problems.

We discussed how to handle learning the shield similar to how the paper by Alshiekh et al. (2018) learned the safe region. Automatically learning this for continuous, partially observable would give a user-friendly shield. This could be expressed as solving a three-agent environment containing an agent, environment, and the shield. The agent's and environment's objective is to reach a state that does not satisfy the safety requirements. The shield's objective is to prevent this while minimizing the interference. Using a heuristic to approximate the closeness to the threshold of passing/failing a safety requirement, we think it might be possible to solve this problem creating something similar to a safe region.

Another question worth exploring is creating a shield to handle multiple risks. For example, extending the current shield for preventing the drone from crashing into the ground. We started examining this question by putting a high negative reward for that case, resulting in the drone failing to learn due to always receiving a negative reward. There is an active area of research on merging multiple shields in such cases, which, in our case, would require finding a safe action that satisfies each shield. The main challenge is figuring out how to perform this search in a continuous space.

The GitHub repository containing the modified code files and steps for running it can be found at *https://github.com/viveksahukar/ai-safety-drone*.

References

Alshiekh, M.; Bloem, R.; Ehlers, R.; Könighofer, B.; Niekum, S.; and Topcu, U. 2018. Safe reinforcement learning via shielding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Hodge, V. J.; Hawkins, R.; and Alexander, R. 2021. Deep reinforcement learning for drone navigation using sensor data. *Neural Computing and Applications*, 33(6): 2015–2033.

Makoviychuk, V.; Wawrzyniak, L.; Guo, Y.; Lu, M.; Storey, K.; Macklin, M.; Hoeller, D.; Rudin, N.; Allshire, A.; Handa, A.; et al. 2021. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*.

Panerati, J.; Zheng, H.; Zhou, S.; Xu, J.; Prorok, A.; and Schoellig, A. P. 2021. Learning to Fly—a Gym Environment with PyBullet Physics for Reinforcement Learning of Multi-agent Quadcopter Control. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 7512–7519.

Potok, M. 2018. Safe reinforcement learning: An overview, a hybrid systems perspective, and a case study.

Ugurlu, H. I.; Pham, X. H.; and Kayacan, E. 2022. Sim-to-real deep reinforcement learning for safe end-to-end planning of aerial robots. *Robotics*, 11(5): 109.