Safe RL for Drone Navigation Using Constrained PPO

Shishir Bhatta[*]
University of Illinois Urbana-Champaign, Champaign, IL, 61820

This report explores the use constrained proximal policy optimization (CPPO) for safe RL training and drone navigation. Reinforcement learning (RL) agents often take unsafe actions during the training phase, which can be harmful to its hardware or surroundings if deployed in real-world situations. For this reason, implementing a cost function to penalize unsafe behavior can be a way to make controllers safer to learn in application, while maintaining performance. This report takes a Lagrangian approach to constrain the optimization problem, providing a way to have the constraint be more conservative or loosen based on the performance of the controller. The algorithm was tested in gym-pybullet-drones, tasked with navigating to the destination with static obstacles in the way. If the drone enters within a specified distance of the obstacle, the drone is severely penalized by the cost function. Ultimately, CPPO performed better than a baseline PPO, but violated the constraint more due to a flawed gradient descent optimzation.

I. Nomenclature

γ	=	discount factor
λ	=	Lagrangian multiplier
π	=	policy
π^*	=	optimal policy
A	=	action space
a	=	action
G	=	goal state
H	=	obstacle state
i	=	iteration index
k	=	policy iteration index
n	=	number of states or iterations
R	=	reward function
r	=	reward
C	=	reward function
c	=	reward
ϵ	=	gradient clipping value
S	=	state space
s	=	state
s'	=	next state
d	=	distance of the agent from the goal state
o	=	observation state
t	=	time step
θ	=	weights of the neural network
$V(s)$	=	state-value function

II. Introduction

THIS report maintaining safety in drone navigation. Drone often operate in cluttered environments, such as warehouses, public parks, and possibly interact with other drones. Drones typically use traditional controllers based on concepts such as PID, LQR, and geometric controllers that are reliable and have been well-tested. However, reinforcement

[*]Undergraduate Student, Aerospace Engineering

learning (RL) controllers have become a popular alternative for better adaptability in uncertain environments. They can learn complex control strategies but fail to incorporate any safety guarantees. For this reason, RL agents often exhibit unsafe behaviors, especially during training.   The goal of this report was to implement a constrained proximal policy optimization (CPPO) algorithm to drone navigation for safe control. CPPO augments the typical PPO structure with a Lagrangian-based constraint which allows the agent to maximize reward while maintaining safety constraints.   The algorithm was implemented on a gym-pybullet-drones environment, where the drone was tasked with reaching a goal position while avoiding static obstacles [1]. The environment had to be slightly altered to support the reward and cost function for the experiment.

III. Problem Definition

The environment with which the agents will interact can be defined as a Markov Decision Problem (MDP). To show this, the state space, action space, reward and cost function, and transition states should be defined [2].

A. State Space

The continuous state space is the drone position, linear velocity, pose and angular velocity, represented as a 12x1 vector s. The objective of the drone is to reach a goal state, but the drone itself can not observe the location of the goal state, however, it can observe the relative location and size of any obstacles, H. There are max 3 obstacles in the environment, future work can expand on this. The entire observation state, o, is the only thing the drone can see.

B. Action Space

The action space consists of the 4 RPM commands for each rotor on the drone. Since this a quadrotor, the agent outputs a 4x1 vector a that has a max magnitude of 2400 RPM. If the policy output exceeds this value, the value gets clipped to the max constraint.
The transition probabilities depend on the physics, which takes in the current s_k and a_k at time k to propagate the dynamics. The physics engine in this environment is the base PyBullet physics.

C. Reward Function

The reward function has multiple components to best aid the training of the RL agent. The rewards were generally designed to guide the drone towards the goal, but prevent it from crashing. The first component of the reward is the distance reward. The distance from the goal d is defined as such.

$$d(s, G) = ||p_s - G|| \tag{1}$$

A positive reward encourages the agent to move towards the goal. For smoother gradients and sharper incentives as the drone approaches the goal, an exponential distance reward is used:

$$R_{dist}(d) = e^{-2d} \tag{2}$$

As the drone moves closer to the goal, the reward sharply increases. A small linear penalty is included, which is linear to the distance from the goal:

$$R_{penalty}(d) = -0.1d \tag{3}$$

This discourages the agent from moving away from the goal provides a counter to ensure the exponential term doesn't dominate at large distances. The crash penalty is added to help the drone learn to fly. One of the issues in previous iterations was the drone wouldn't learn to hover properly, so by penalizing crashing, the drone was incentivizes to maintain stable altitudes.

$$R_{crash}(s) = \begin{cases} -5, & \text{if } z < 0.1 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

An additional drifting penalty is included to ensure the drone doesn't deviate too far from the desired state. The penalty only occurs once the drone has deviated quite a bit, discouraging unsafe behavior.

$$R_{drift}(d) = \begin{cases} -2, & \text{if } d > 3.0 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

IV. Implementation

A. Algorithm

CPPO is based on the popular PPO algorithm, which is a policy-gradient RL algorithm designed to improve training stability while maintain sample efficiency [3]. PPO updates the policy by maximizing a surrogate objective which restricts how far the new policy π_θ can stray from the previous policy $\pi_{\theta_{old}}$, which is calculated by the probability ratio:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}(a_t|s_t)}} \tag{6}$$

This measures how much the new policy changes with respect to the old policy. PPO prevents these large policy updates by clipping this ratio. However, CPPO makes an adjustment by enforcing a safety constraint on the policy updates. The optimzation objective bcomes a constrained Markov Decision Process (CMDP):

$$\max_\theta \mathbb{E}[R] \quad \text{s.t.} \quad \mathbb{E}[C] \leq d_{\max} \tag{7}$$

where d_{max} is the allowed cost threshold. A Lagrange multiplier λ that converts this constrained problem into an unconstrained one as such:

$$L(\theta, \lambda) = \mathbb{E}[R] - \lambda(\mathbb{E}[C] - d_{max}) \tag{8}$$

where the multiplier λ is updated to penalize the agent if the costs exceeds the threshold. The multiplier is updated as such:

$$\lambda \leftarrow \max(0, \lambda + \alpha_\lambda(\mathbb{E}[C] - d_{max})) \tag{9}$$

If the agent violates the constraints, λ will increase and the cost will increase, vice versa for safe actions. This way, the agent is encouraged to take safe actions. This is the main difference between CPPO and PPO, which enforces safety constraints on the policy. Exactly like PPO, CPPO uses a clipped policy update for stable training. The only difference is the advantage function now incorporates the cost and reward function:

$$A_t^{\text{CPPO}} = A_t^R - \lambda A_t^C \tag{10}$$

The combined advantage function relies on the separate advantage functions from the reward and cost functions. Also, as λ decreases, the penalty of any cost in the advantage function decreases as well. The policy update is:

$$\max_\theta \mathbb{E}_t[\min(r_t(\theta)A_t^{\text{CPPO}}, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t^{\text{CPPO}}] \tag{11}$$

Overall, for the drone, the reward will encourage the agent to reach the target efficiently, while the cost can represent any unsafe behaviors that need to be prevented. By adjusting the λ dynamically, the importance of safety can be determined by increasing or decreasing the value. The general algorithm is below:

Algorithm 1 Constrained Proximal Policy Optimization (CPPO)

```
Require: policy $\pi_\theta$, value functions $V_\phi^R, V_\psi^C$, Lagrange multiplier $\lambda \geq 0$
Require: epochs $T$, steps per epoch $N$, PPO clip $\epsilon$, cost limit $d$
 1: for epoch = 1 to $T$ do
 2:     Collect $N$ transitions $(s_t, a_t, r_t, c_t)$ using $\pi_{\theta_{\text{old}}}$
 3:     Estimate advantages $\hat{A}_t^R, \hat{A}_t^C$ and returns $\hat{R}_t, \hat{C}_t$ (GAE)
 4:     Compute penalized advantage $\hat{A}_t = \hat{A}_t^R - \lambda \hat{A}_t^C$ and normalize
    Policy update (PPO):
 5:     Maximize
            $\mathcal{L}(\theta) = \mathbb{E}\big[\min\big(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\big)\big]$
    Value updates:
 6:     Minimize $(V_\phi^R(s_t) - \hat{R}_t)^2$ and $(V_\psi^C(s_t) - \hat{C}_t)^2$
    Dual update:
 7:     $\lambda \leftarrow \max\big(0,\ \lambda + \alpha_\lambda(\bar{C} - d)\big)$
 8:     $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$
 9: end for
10: return $\pi_\theta$
```

1. Additional Improvements

Various improvements were made to the implemented CPPO algorithm for better performance. Dense reward shaping is one example, in which an In addition, an

1) **Dense reward shaping** - exponentially-decaying distance reward provides smoother gradients and stronger rewards closer to the goal position
2) **Explicit cost function** - rather than simply terminating episodes when a collision occurs, a cost function allows for continuous safety enforcement
3) **Dual-critic architecture** - most algorithms only estimate the reward value, but this implementation estimates both the cost and reward function to stabilize the Lagrangian update and improve accuracy
4) **KL-Divergence control** - additional check to ensure the policy does not diverge too much from the old policy.

B. Environment

The environment the agent operated in was created using the gym-pybullet-drones frameworks. The environment uses the SafeHoverAviary environment, adding custom functions to fit the requirements for the algorithm. The environment simulates a single quadrotor operating in the environment with a single static obstacle in the way to get to the goal. The environment provides the reward and cost signals for the RL agent, as well as the observations and states of the drone.

1. Dynamics and Simulation

The environment simulates the dynamics of a Crazyflie 2.x quadrotor with a simulation frequency of 240 Hz and control frequency of 30 Hz. The agent controls the 4 motor RPMs, which is inputted into the environment to calculate the thrust, drag and rigid-body dynamics. Each episode has a max length of 8 seconds, which is truncated is the episode doesn't terminate for either failure or reach the goal. The goal was located at [0, 0, 1] meters and started at [0, 0, 0]. One obstacle is located at [1, 0, 1] meters, up to 6 obstacles could be added to the environment.

C. Parameters

For the CPPO agent, these are the parameters used:

- **Epochs** - 100: Likely do more epochs for further testing
- **Steps per epoch** - 4096: the epoch will terminate after this many steps, ending episodes that may run too long without reaching the goal
- **Discount factor** γ - 0.99: This factor is applied as a way to discount future rewards. The implemented value is fairly large due to better results.

- **Lagrangian multiplier** - [0, 1]: It initially starts at 0.95, but adjusts as the drone either takes safe or unsafe actions.
- **policy π, value function learning rate** - $3 * 10^{-4}$,$1 * 10^{-4}$: These are the learning rates for the policy networks and the value networks.
- **gradient clip ϵ** - 0.2: This is the clipping parameter that constrains the policy to remain within the bounds for stable training.
- **target KL threshold** - If the KL-divergence surpasses this value, the episode will terminate to prevent unstable policy updates.
- **Lagrangian learning rate** - 0.02: This is the learning rate for adjusting the Lagrangian multiplier, which is determined by a gradient descent optimization.
- **Cost limit** - 0.05: The maximum allowed expected cost per timestep.

V. Results

After implementing the RL agent in the environment, the results of drone show that the CPPO agent remains safer than PPO, but also explores the environment.

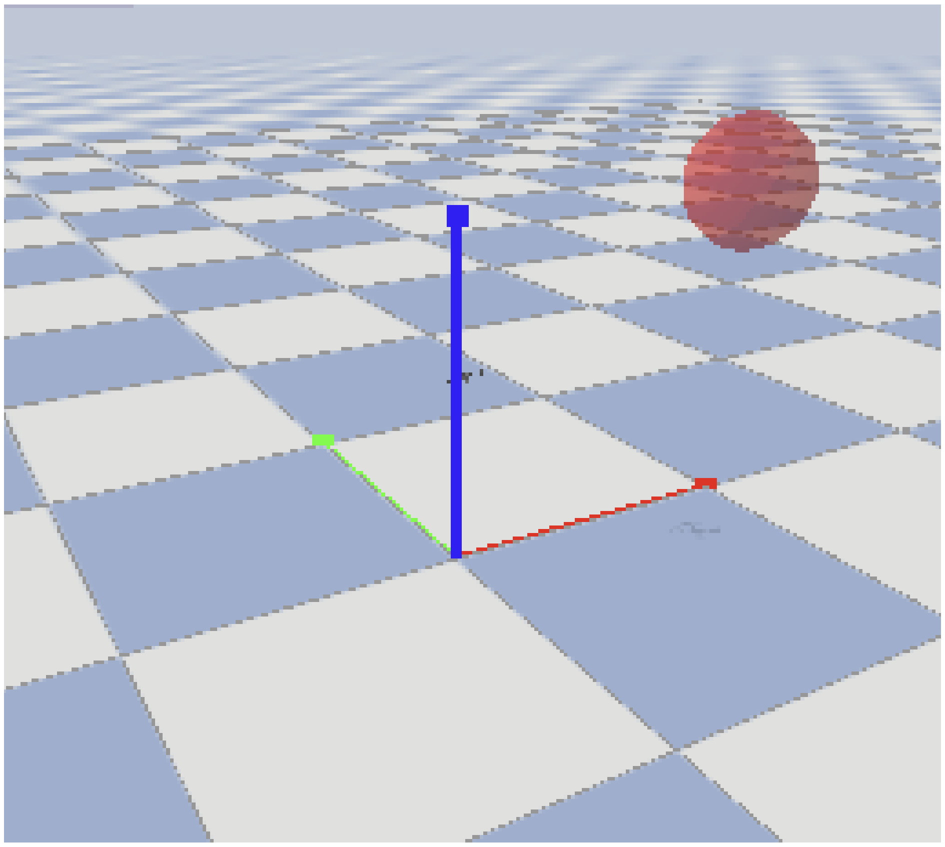

Fig. 1 Drone in Motion in the Environment

The CPP) algorithm was compared against a baseline PPO algorithm that uses the exact same parameters but without the constraints. This will provide a point of comparison for the algorithms.

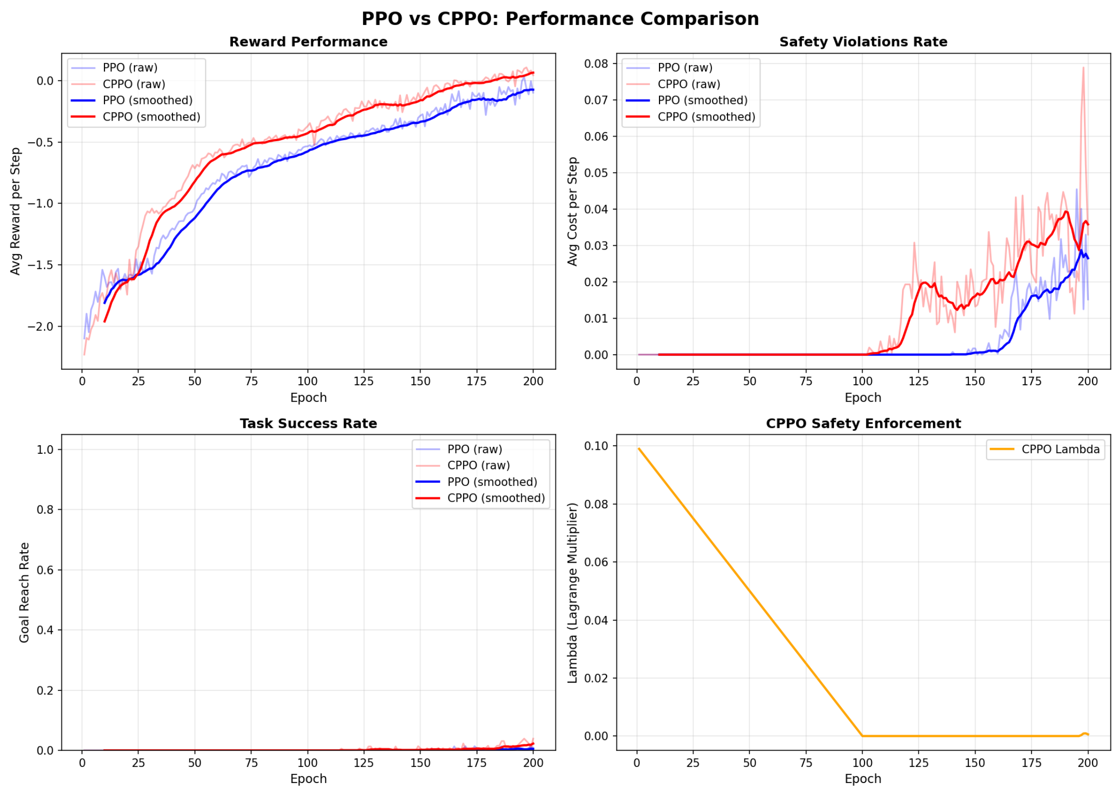

Fig. 2 Various Plots Comparing the Results of Both Algorithms

All the results are shown in Fig 2, which shows the results of CPPO in red and PPO in blue. CPPO has a higher average reward per step than the PPO, but both of them are increasing over time. Neither of the agents seem to have converged, but watching the drone interact in the environment provides proof of this, as they both fail to learn to hover at the goal position. The drone likely needs more episodes to converge. In the first 100 epochs, when the safety is enforced (since $\lambda > 0$), CPPO shows higher rewards and better rewards despite considering the cost.

Unfortunately, CPPO and PPO violate violate the safety constraints of remaining a certain distance away from the obstacle. CPPO seems to violate it further than PPO, which is unexpected, but can be explained. Based on the CPPO safety enforcement figure, the Lagrangian multiplier λ reduces to 0 at epoch 100 and fails to increase to anything meaningful afterwards. Likely the gradient descent optimization needs to be altered to further punish any safety violations, as the safety violations only start to occur once the lambda has decreased. Up until epoch 100, no safety violations are observed, so modifications to the Lagrangian optimization must be made. Since $\lambda = 0$ after epoch 100, the CPPO algorithm acts like PPO since no constraints are enforced. The higher safety costs can be contributed to CPPO being better at receiving rewards and flying high into the sky, which can often lead to safety violations.

In addition, both algorithms were poor at reaching the goal state. The goal reaching state for each epoch was very low, despite CPPO receiving higher rewards on average.

Table 2 Performance Comparison Between PPO and CPPO

Method	Final Reward	Cost	Goal Rate (%)
PPO	-0.098	0.0237	0.47
CPPO	0.046	0.0375	1.71

Cost Reduction (PPO → CPPO): −58.1%

The final rewards are summarized in Table 2, which show the cost difference between the two algorithms was significant, a 58.1% increase in cost for CPPO in comparison to PPO. The goal rate and reward are higher however, seeming to learn more efficiently than traditional PPO. Both algorithms showed they could learn effecitively, moving from rewards around -2 to close to 0, showing an increase in average reward.

VI. Conclusion

This report demonstrated the implementation of CPPO for safe drone navigation when static obstacles are present. While CPPO achieved higher success rates and rewards than PPO, it violated the safety constraints more, which is counterintuitive. For future work, the lambda learning rate or adaptive step size changes could help the algorithm adjust its safety to be more or less conservative. CPPO, despite violating the safety constrains after 100 epochs, showed promise before that, clearly receiving more rewards than PPO while remaining safe. Future work should be centered around fixing the Lagrangian multiplier learning that scales better with more epochs and allowing for more epochs to see the convergence of the policies.

References

[1] Panerati, J., Zheng, H., Zhou, S., Xu, J., Prorok, A., and Schoellig, A. P., "Learning to Fly—a Gym Environment with PyBullet Physics for Reinforcement Learning of Multi-agent Quadcopter Control," *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 7512–7519. https://doi.org/10.1109/IROS51168.2021.9635857, code available at https://github.com/utiasDSL/gym-pybullet-drones.

[2] Sutton, R. S., and Barto, A. G., *Reinforcement Learning: An Introduction*, 1st ed., MIT Press, Cambridge, MA, USA, 1998. URL https://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf.

[3] Ji, J., Zhang, B., Zhou, J., Pan, X., Huang, W., Sun, R., Geng, Y., Zhong, Y., Dai, J., and Yang, Y., "Safety-Gymnasium: A Unified Safe Reinforcement Learning Benchmark," , 2023. URL https://safe-policy-optimization.readthedocs.io/en/latest/, includes the Safe Policy Optimization (SafePO) algorithm library. Documentation available at https://safe-policy-optimization.readthedocs.io/en/latest/algorithms/lag.html.