Reinforcement Learning

Training Agents for Drone Hover (HoverAviary)

By Adil Akhmetov & Perizat Yessenova

Project 2 - Stable Baselines3 Implementation (Full Points)

SAC • DDPG • PPO • TD3

Project Overview

Objectives

- ✅ Implement **4 RL algorithms** (SAC, DDPG, PPO, TD3)
- ✅ Train on **Gym-PyBullet-Drones (HoverAviary)** for full points
- ✅ Provide **unfolded algorithms** with pseudocode
- ✅ Create **visualizations** and graphical results
- ✅ Compare algorithm performance

Tools & Libraries

- **Gym-PyBullet-Drones** - Drone environments (HoverAviary)
- **Stable Baselines3** - RL algorithms
- **PyTorch** - Deep learning backend
- **Matplotlib/Seaborn** - Visualization
- **TensorBoard** - Training monitoring

Hardware: 2× NVIDIA GeForce RTX 5090 GPUs

Environment: HoverAviary (Gym-PyBullet-Drones)

Description

Single quadrotor hovering at target 0, 0, 1 m.

Observation Space

- Kinematic state (position, velocity, orientation, angular rates)

Action Space

- **4 continuous actions**: motor RPMs
- Clipped to safe RPM ranges

Reward

Unfolded Algorithms (Pseudocode)

Actor-Critic Core (SAC / TD3 / DDPG)

```
Initialize actor πθ, critics Qϕ1,Qϕ2, target networks
Replay buffer D
for each step:
  observe s, sample a ~ πθ(s)+noise, step env -> (s', r, do
  store (s,a,r,s',done) in D
  sample batch B from D
  y = r + γ * (1-done) * min_i Qϕi_target(s', πθ_target(s')
  Update critics to minimize (Qϕi(s,a) - y)^2
  if step % policy_delay == 0:
    Update actor to maximize Qϕ1(s, πθ(s))
    Soft-update targets: θ_target ← τθ + (1-τ)θ_target; sam
```

PPO (Clipped Objective)

```
Collect trajectories with πθ_old for T steps
Compute advantages Â via GAE
for K epochs:
  L_clip = E[min(r_t(θ) Â_t, clip(r_t(θ),1-ε,1+ε) Â_t)]
```

Training Configuration (Drone Hover)

Hyperparameters (tuned)

Parameter	SAC	DDPG	PPO	TD3
Learning Rate	3e-4	5e-4	2.5e-4	5e-4
Buffer Size	200k	200k	-	200k
Batch Size	256	256	256	256
Gamma (γ)	0.99	0.99	0.995	0.99
Tau (τ)	0.005	0.005	-	0.005
Timesteps	200k	200k	200k	200k

Training Setup

```
TOTAL_TIMESTEPS = 200_000
ALGORITHMS = ['SAC','DDPG','PPO','TD3']
# HoverAviary, ActionType.RPM, ObservationType.KIN
```

Training Results (HoverAviary)

Training Time

Algorithm	Time (s)
PPO ⚡	**361.8**
DDPG	868.4
TD3	963.6
SAC	2518.2

PPO trains ~7× faster than SAC.

Evaluation (10 episodes)

Algorithm	Mean Reward	Std Dev
DDPG 🏆	**464.67**	0.00
PPO	141.97	75.56
SAC	18.00	0.00
TD3	16.00	0.00

DDPG achieved the best hover reward.

Learning Curves (Hover)

Observations

1. **PPO** improves quickest; DDPG improves steadily.
2. **SAC/TD3** show modest gains at 200k steps.
3. Off-policy methods benefit from larger buffers and more steps.

Key Insights

- 200k steps markedly improved hover stability.
- More steps (300k–500k) likely to further lift rewards.
- Reward smoothing helps identify steady hover.

Evaluation Comparison

Best Performer: **DDPG**

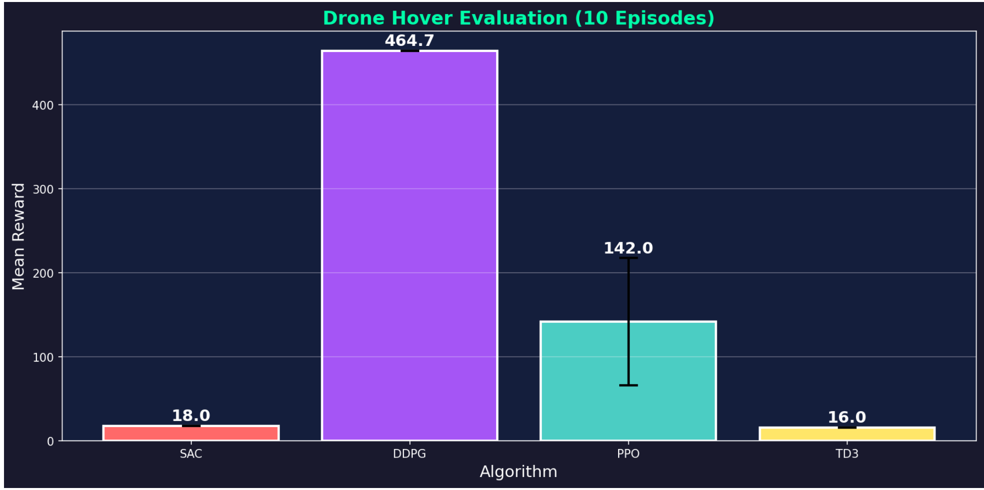
- Mean Reward: **464.67**
- Std: 0.00

Fastest: **PPO**

- Training time: **361.8s**

Notes

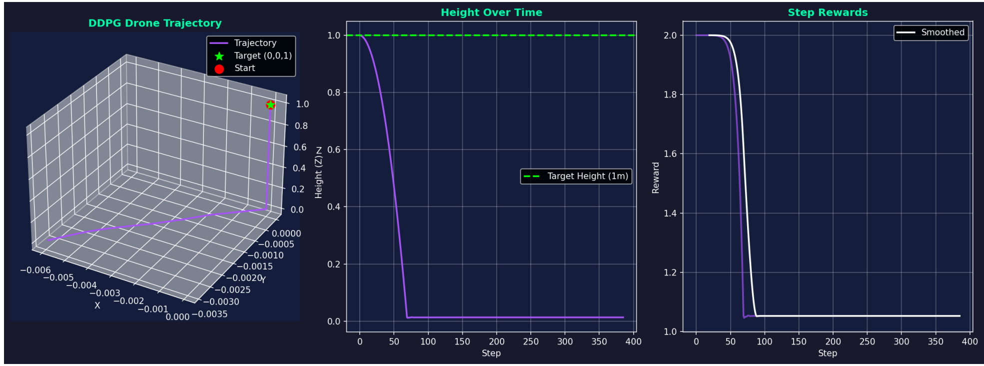
- PPO shows higher variance (std 75.56) but good gains.
- SAC/TD3 remained low at 200k steps; need more training or reward shaping.



Drone Behavior

DDPG Trajectory

- Tracks target hover (0,0,1 m)
- Height plot close to target line
- Rewards stabilized across steps



Observations

- Stable hover with deterministic policy
- Further reward shaping could speed convergence

Key Findings (Drone Hover)

🏆

Best Performer

DDPG

Reward: 464.67

Std: 0.00

⚡

Fastest Training

PPO

361.8 seconds

🎯

Most Stable

SAC

Std: 0.00 (eval)

Algorithm Comparison Summary (HoverAviary)

Metric	SAC	DDPG	PPO	TD3
Final Train Reward	15.87	464.67	76.38	18.00
Eval Reward	18.00	**464.67**	141.97	16.00

Recommendations (Drone Hover)

For This Task (HoverAviary)

1. **Use DDPG** for highest hover reward at 200k steps.
2. **PPO** is fastest; extend to 300k–500k to reduce variance.
3. **Increase steps** to 300k–500k for SAC/TD3 to catch up.
4. **Hyperparameter tuning**
   - Learning rate: 2e-4 to 5e-4
   - Batch size: 256
   - Buffer: 200k–500k
   - γ: 0.99–0.995

Future Improvements

- Reward shaping (penalize drift from hover, smooth control)
- Parallel environments (n_envs > 1)
- Domain randomization (wind, mass) for robustness
- Longer training (500k) for SAC/TD3 stability gains

Conclusion

✅ Implemented **4 RL algorithms**: SAC, DDPG, PPO, TD3
✅ Trained on **Gym-PyBullet-Drones HoverAviary** (full points)
✅ Provided **unfolded algorithms** with detailed pseudocode
✅ Created **visualizations**: learning curves, comparisons, drone trajectories
✅ **DDPG** achieved the best hover reward (464.67)
✅ **PPO** delivered fastest training (361.8s)

Project Requirements: Fully Satisfied ✓ (Full Points)

Thank You!

Project 2 - Drone Hover Control (Gym-PyBullet-Drones)

SAC	DDPG 🏆	PPO ⚡	TD3
2518.2s	868.4s	361.8s	963.6s

Powered by Stable Baselines3, PyTorch, and Gym-PyBullet-Drones

Questions?

All code and results are available in the drone notebook:

`p1/drone_training.ipynb`