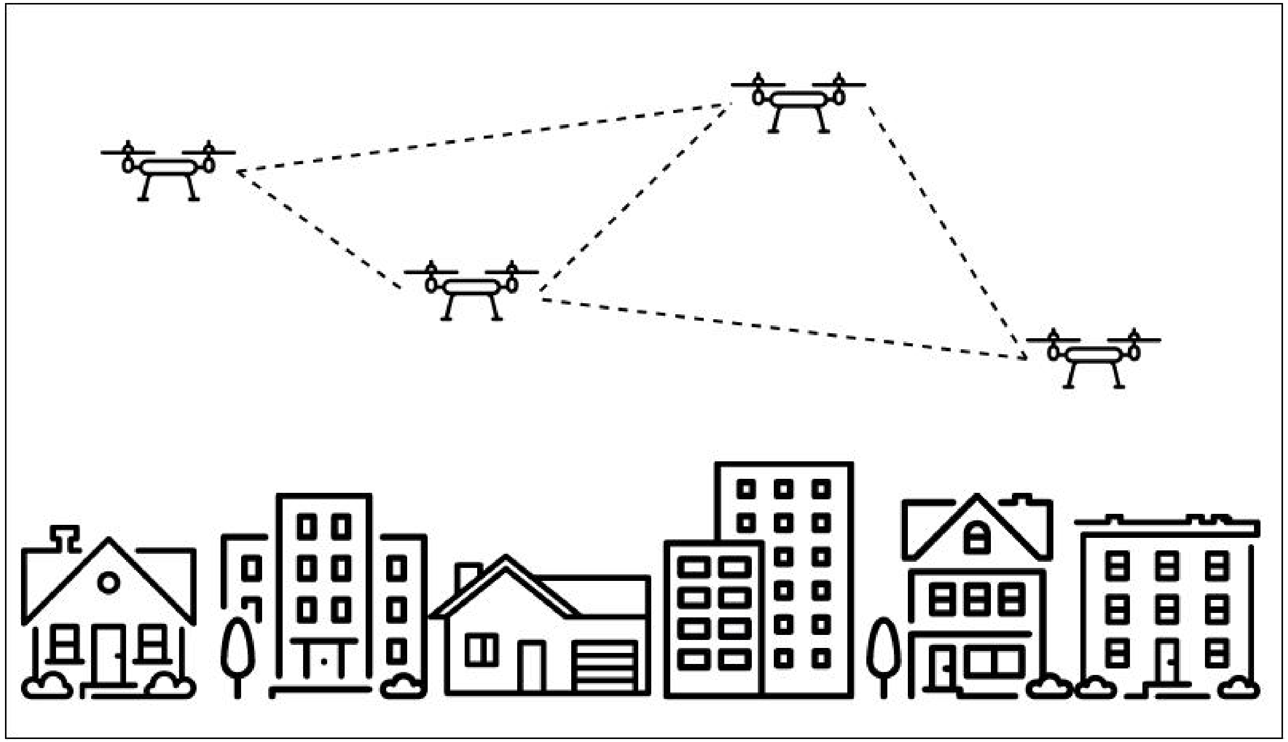
Learning Safe Cooperative Policies in Autonomous Multi-UAV Navigation

Arshdeep Singh
Indian Institute of Technology Ropar
2019csm1001@iitrpr.ac.in

Shashi Shekhar Jha
Indian Institute of Technology Ropar
shashi@iitrpr.ac.in

Multi-UAV Applications

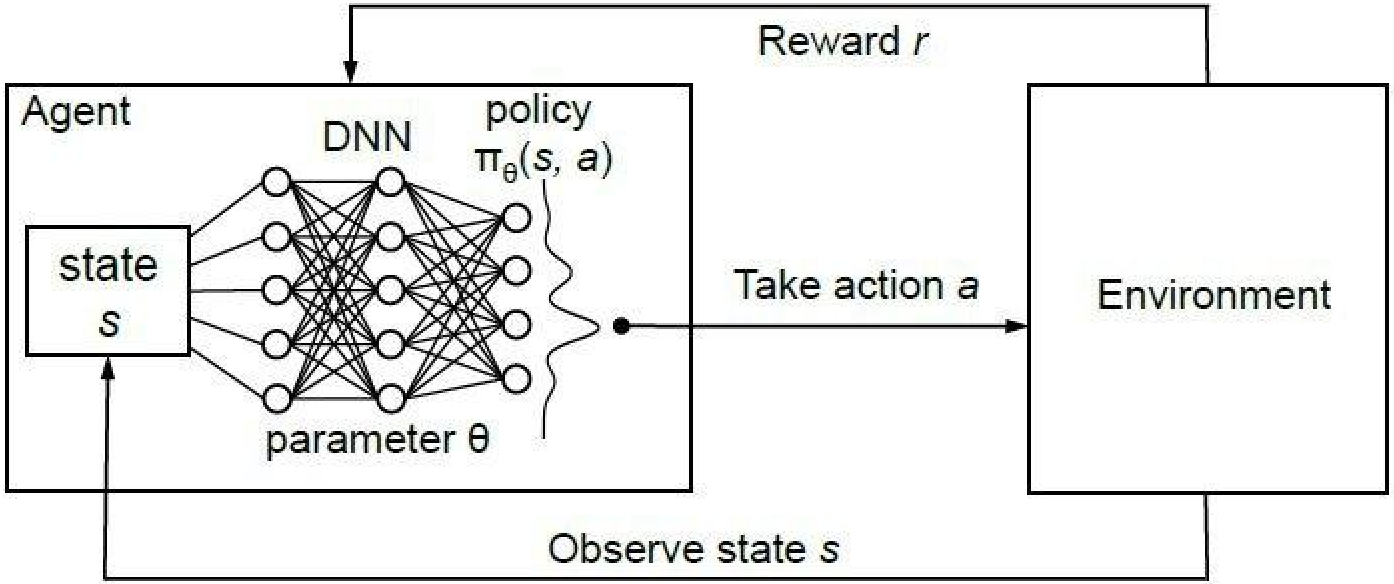
- Application Domains
  - Surveying, Remote Sensing, Search & Rescue, E-Commerce Delivery



Problem Formulation

- Multi-UAVs learning to autonomously reach their target destination through safe navigation
  - Each UAV should avoid collision with other UAVs
- Formulated as a Multi Agent Reinforcement Learning(MARL) problem
  - MARL is when we are considering various intelligent agents interacting with an environment
- Multi Agent scenario
  - Cooperative
  - Competitive
  - Mixture of both

Reinforcement Learning at a glance



RL feedback loop[1]

System Dynamics

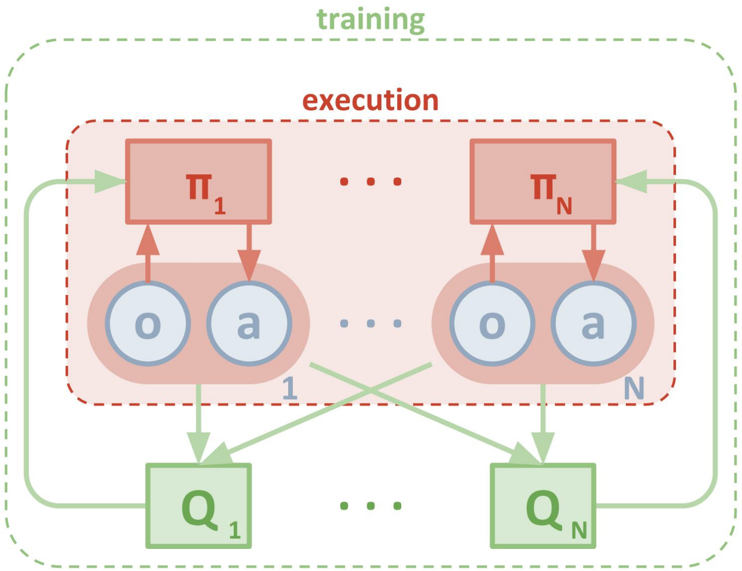
- N number of agents
- Observation space : $O_t = \{o_1, o_2, o_3, \ldots o_N\}$
  - $o_i = \{s_1, s_2, .. s_i, ..., s_N\}$
- Action space : $A_t = \{a_1, a_2, a_3, \ldots a_N\}$
- Parameterized policy : $\pi_{\theta_i} : o_i \rightarrow a_i$
- Transition function : $T : O_t \times A_t \rightarrow O'_{t+1}$
- Reward : $r_i^t : o_i^t \times a_i^t \rightarrow \mathbb{R}$
- Goal : $G_i = \sum_t \gamma^t r_i^t$

System Dynamics

- Actor Critic algorithms
  - Optimizes a parameterized *policy* π_θ through policy gradients
  - Learning both policy model and the value function
- Each agent is modelled as an Actor
  - Updates its policy as suggested by the Critic
- The Critic estimates the values of the state
  - It guides the actor to prefer good actions over bad ones

MADDPG (Lowe et al)[2]

- Multi Agent Deep Deterministic Policy Gradient
  - Centralized Critic network, Decentralized execution (multiple independent Actors), Deterministic Policy function
- Architecture
  - An actor network for using observations for deterministic actions
  - An identical target actor network for training stability
  - A critic network that uses joint states action pairs to estimate Q-values



MADDPG[2]

Continued

- Update of the Critic network through loss
  - $L(\theta_c) = \frac{1}{M}\ \sum_j \left(y_i^j\ -\ Q_i^{\mu}(o_i^j,\ a_1^j, \ldots, a_N^j)\right)^2$ (1)
    - where $y_i^j\ =\ r_i^j\ + \gamma\, Q_i^{\mu'}\left(o_i'^j,\ a_1', \ldots, a_N'\right)|_{a_i' = \mu_i'(o_i'^j)}$
- Update of the Actor network through the sampled policy gradient
  - $\nabla_{\theta_i} J \approx \frac{1}{M}\ \sum_j \nabla_{\theta_i}\mu_i(o_i^j)\nabla_{a_i}Q_i^{\mu}(o_i^j,\ a_1^j, .., a_N^j)|_{a_i=\mu_i(o_i^j)}$ (2)

Issues

- No obstacle avoidance : static, dynamic
- No safety guarantees
- Large state space configuration
- Techniques
  - Safety Controllers[3]
  - Control Barrier functions[4]

Safe Reinforcement Learning[4]

- Learning optimal policies while satisfying Safety Constraints
- Safe RL will help in transitioning the control to physical systems
- A major challenge in using reinforcement learning is safety - control policies learned using reinforcement learning typically do not provide any safety guarantees

Safety Layer(Dalal et al)[3]

- Constrained policy optimization
  - $\max_{\theta} E[\sum_t \gamma^t R(o_i, \mu_{i|\theta}(o_i))]$ where $c_k(o_i) \leq C_k, \forall k \in K$ (3)
- Safety Layer
  - $g(s, w)$
- Learning Safety Layer
  - $a_i^* = \arg min_{a|i} \frac{1}{2} \| a_i - \mu_{i|\theta}(o) \|^2$ where $c_k(o_i) + g(o, w)^T a_i \leq C_k, \forall k \in K$ (4)

Safe-MADDPG

- Step 1

Algorithm 1: Safe-MADDPG for N agents

while epochs $\leq$ TOTAL EPOCHS **do**
    while step $\leq$ TOTAL SAMPLE STEPS **do**
        Reset Environment(state = $o_i, \forall i \in N$);
        Calculate constraint values c_i;
        Execute actions $(a_1, ..., a_N)$ and observe reward r_i and new obs o_i' for each agent $\forall i \in N$;
        Calculate constraint values ${c_i}^{next}$;
        Store $(a_i, o_i, c_i, {c_i}^{next})$ in replay buffer D;
    end while

- Safety Network update

    while update via mini-batch **do**
        Sample a mini-batch for each agent $i \in N$: $(a_i, o_i, c_i, {c_i}^{next})$;
        Observe safety layer : $g_i(o_i, w_i)$;
        Predict next constraint values;
        ${c_i}^{next.predicted} = c_i + g_i(o_i, w_i)^T a_i$;
        Calculate loss and update safety layer weights;
        $L^k = \| {c_i}^{next} - {c_i}^{next.predicted} \|^2, \forall k \in K$
    end while
end while

Safe-MADDPG

- Step 2

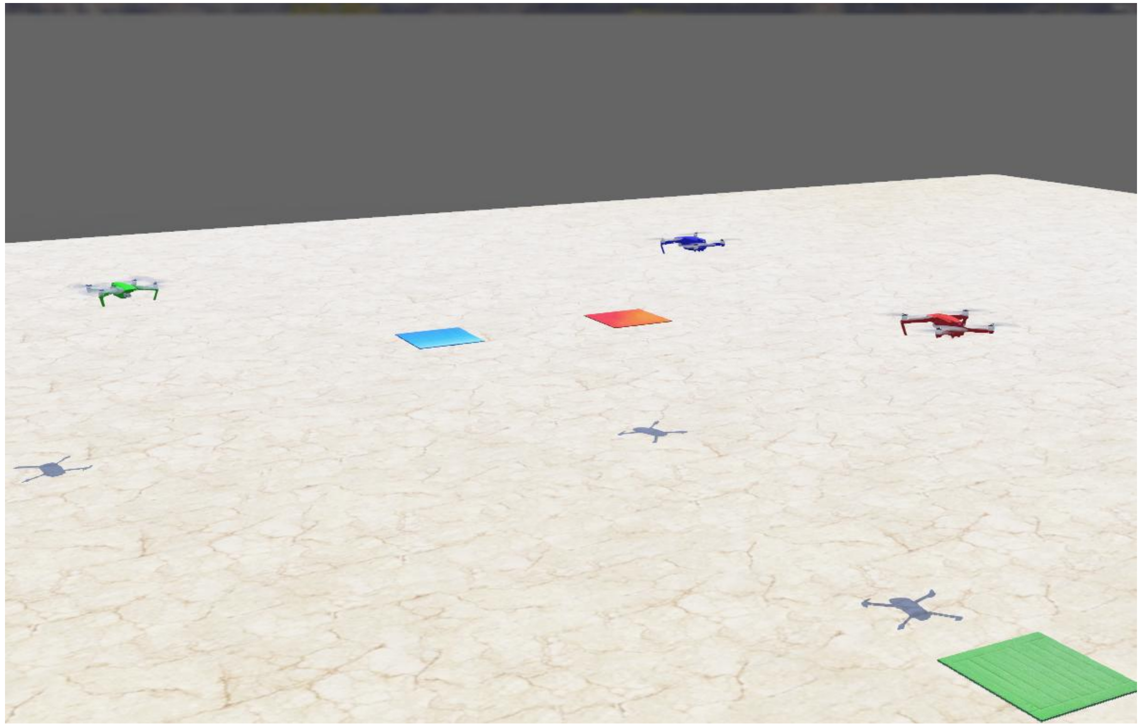
while episode number $\leq$ TOTAL EPISODES **do**
Initialize exploration noise;
Reset environment (state = o_i, $\forall i \in N$);
while timestep(t) $\leq$ EPISODE LENGTH **do**
For an agent i, select action $a_i = \mu_{i|\theta}(o_i) + \mathcal{N}_t$
w.r.t. the current policy and exploration;
calculate lagrange multipliers :
$\lambda_i{}^* = \frac{g(o_i,w_i)^T \mu_{i|\theta}(o_i) + c_i(o_i) - C_i}{g(o_i,w_i)^T g(o_i,w_i)}$;
correct each agent's action as :
$a_i{}^* = \mu_{i|\theta}(o_i) - \lambda_i{}^* g(o_i, w_i)$;
Execute corrected actions $(a_1{}^*, ..., a_N{}^*)$ and
observe reward r_i and new obs o'_i for agent i;
Store $(o_i, a_i{}^*, r_i, o'_i)$ in replay buffer D;
$o_i \leftarrow o'_i$;

- Next

while agent i = 1 to N **do**
Sampling of a randomized minibatch of M
samples $(o_i^j, a_i^j, r_i^j, {o'}_i^j)$ from D;
set $y_i{}^j =$
$r_i{}^j + \gamma Q_i{}^{\mu'}({o'}_i^j, a'_1, ..., a'_N)|_{a'_i = \mu'_{i|\theta}({o'}_i^j)}$;
Update of the critic through loss $L(\theta_c) =$
$\frac{1}{M} \sum_j (y_i{}^j - Q_i{}^\mu(o_i{}^j, a_1{}^j, ..., a_N{}^j))^2$;
Update of the actor by the use of sampled
policy gradient;
$\nabla_{\theta_i} J \approx \frac{1}{M} \sum_j \nabla_{\theta_i} \mu_i(o_i{}^j) \nabla_{a_i} Q_i{}^\mu(o_i{}^j, a_1{}^j, .., a_N{}^j)$
end while
Update of the target network parameters
for each agent i;
$\theta'_i \leftarrow \tau \theta_i + (1 - \tau)\theta'_i$;
end while
end while

Experimentation

- Multiple UAVs and their Target locations
- UAV : DJI Mavic
- Webots Simulation by Cyberbotics[5]
  - Real world physics engine
  - Deepbots[6]: Deepbots is a simple framework which is used as middleware between the Webots Simulator and Reinforcement Learning algorithm
- Emitter-Receiver scheme
- Combined robot-supervisor controller scheme



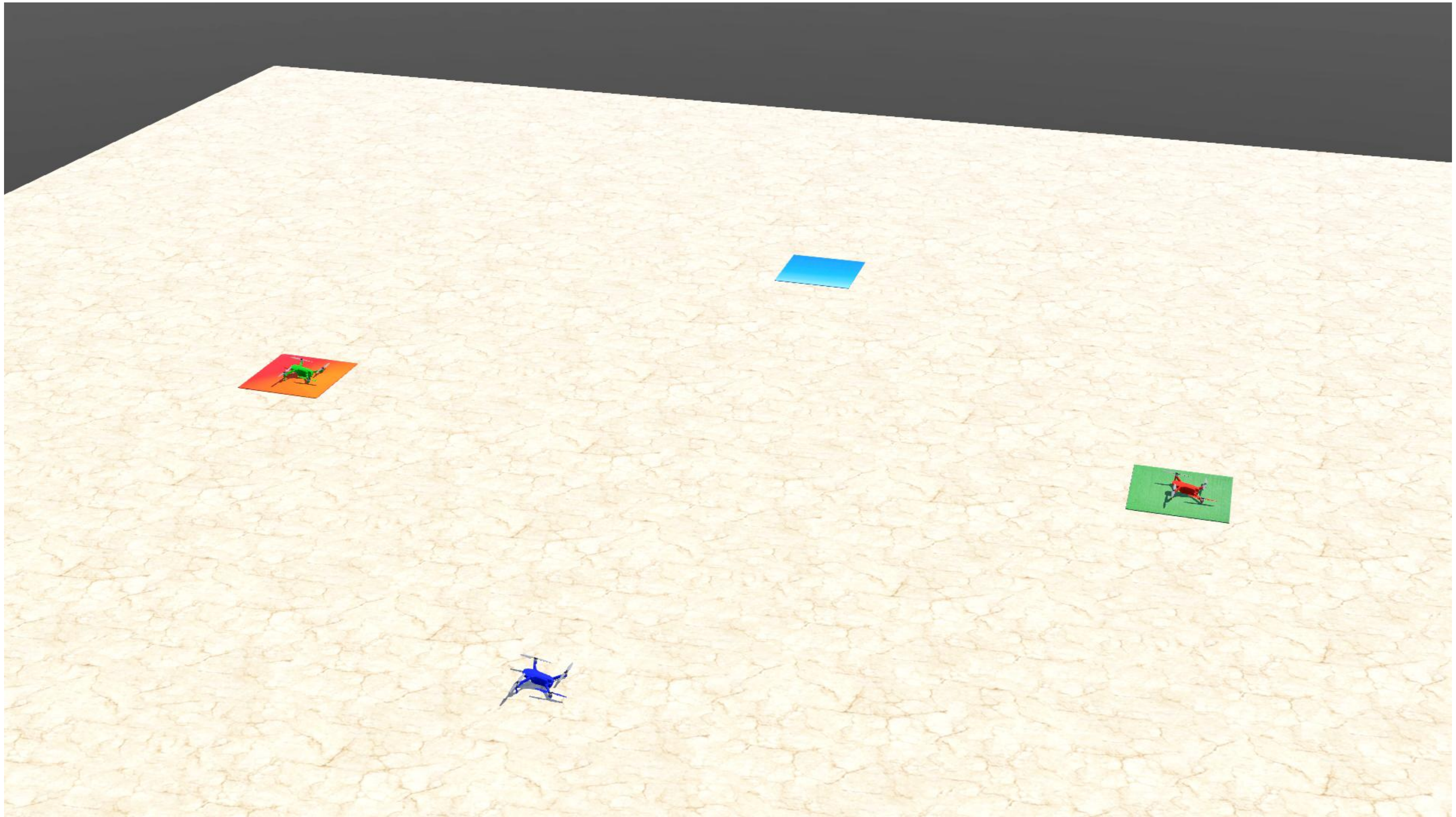
System Settings

- UAV Observation Space
  - Position of each entity (UAV, Target location) : P_i : $[p_i^x,\ p_i^y,\ p_i^z]$
  - Velocity of each UAV : V_i : $[l_i^x,\ l_i^y,\ l_i^z,\ v_i^x,\ v_i^y,\ v_i^z]$
  - Concatenated tuple : $\{P^i,\ V^i,\ P^i_{Target},\ \cup_{\forall j \in N;\, j \neq i} P^j\}$
- UAV Action Space
  - For each UAV : a_i : $\{pitch_i,\ yaw_i\}$
  - Constraints (K=4) : $\{pitch_{max}, pitch_{min}, yaw_{max}, yaw_{min}\}$

Reward Structure

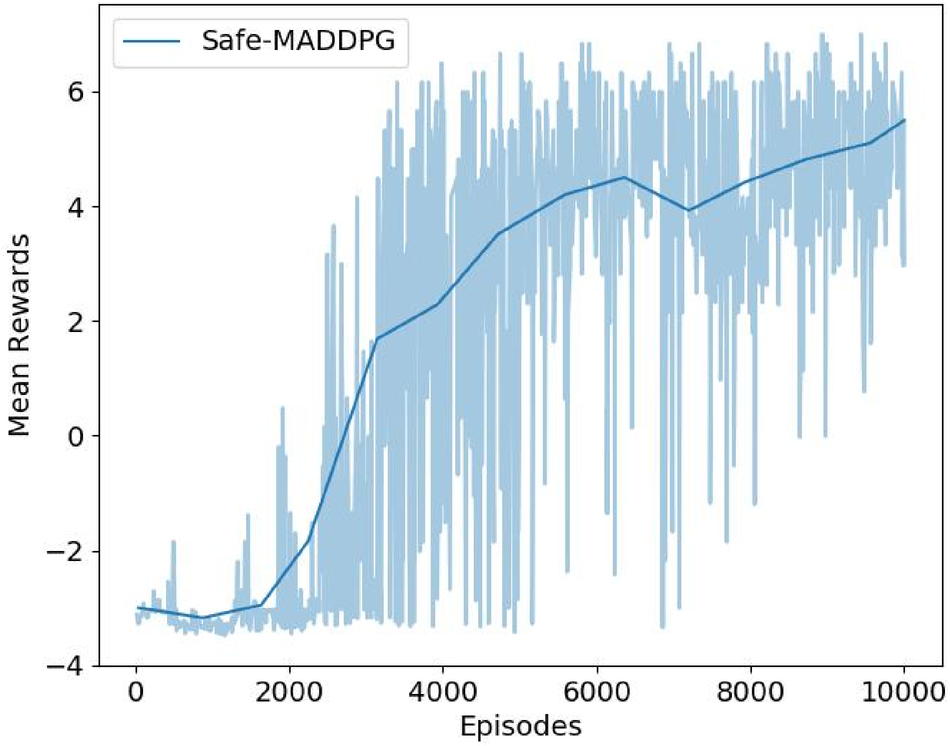
Reward Value	Conditions
-4	if the angle between UAV and its target is > 1.5 Radians
-2	if angle < 1.5 Radians and curr_dist > prev_dist
-2	if angle < 0.1 Radians and curr_dist > prev_dist
+5	if angle < 1.5 Radians and curr_dist < prev_dist
+10	if angle < 0.1 Radians and curr_dist < prev_dist
-10	if there is a collision

Results (3 UAV scenario)

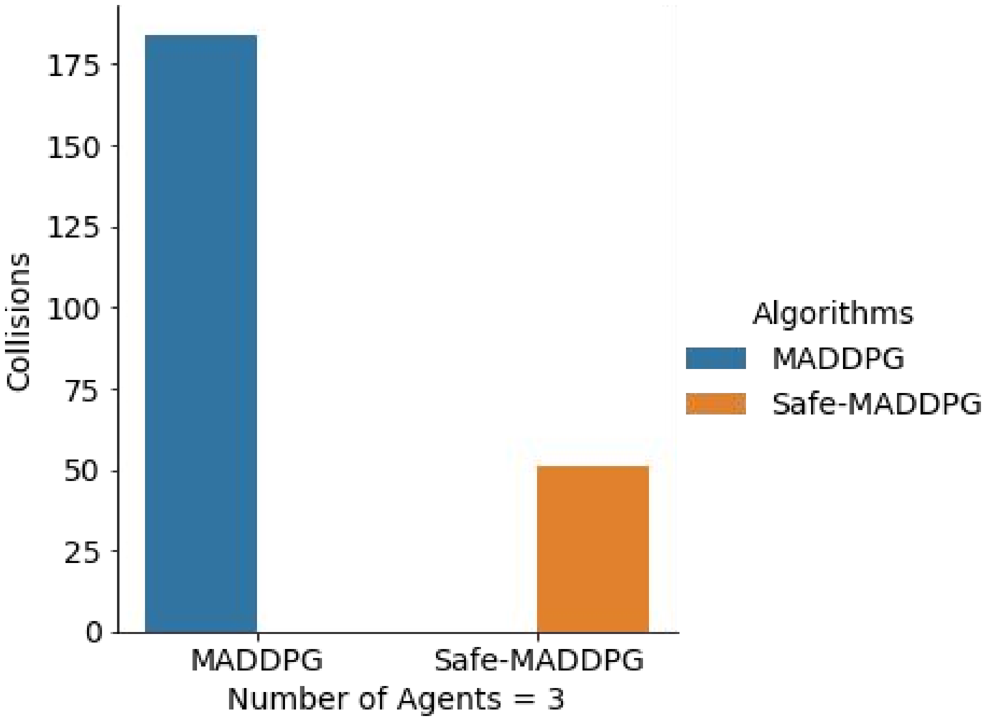


Evaluation

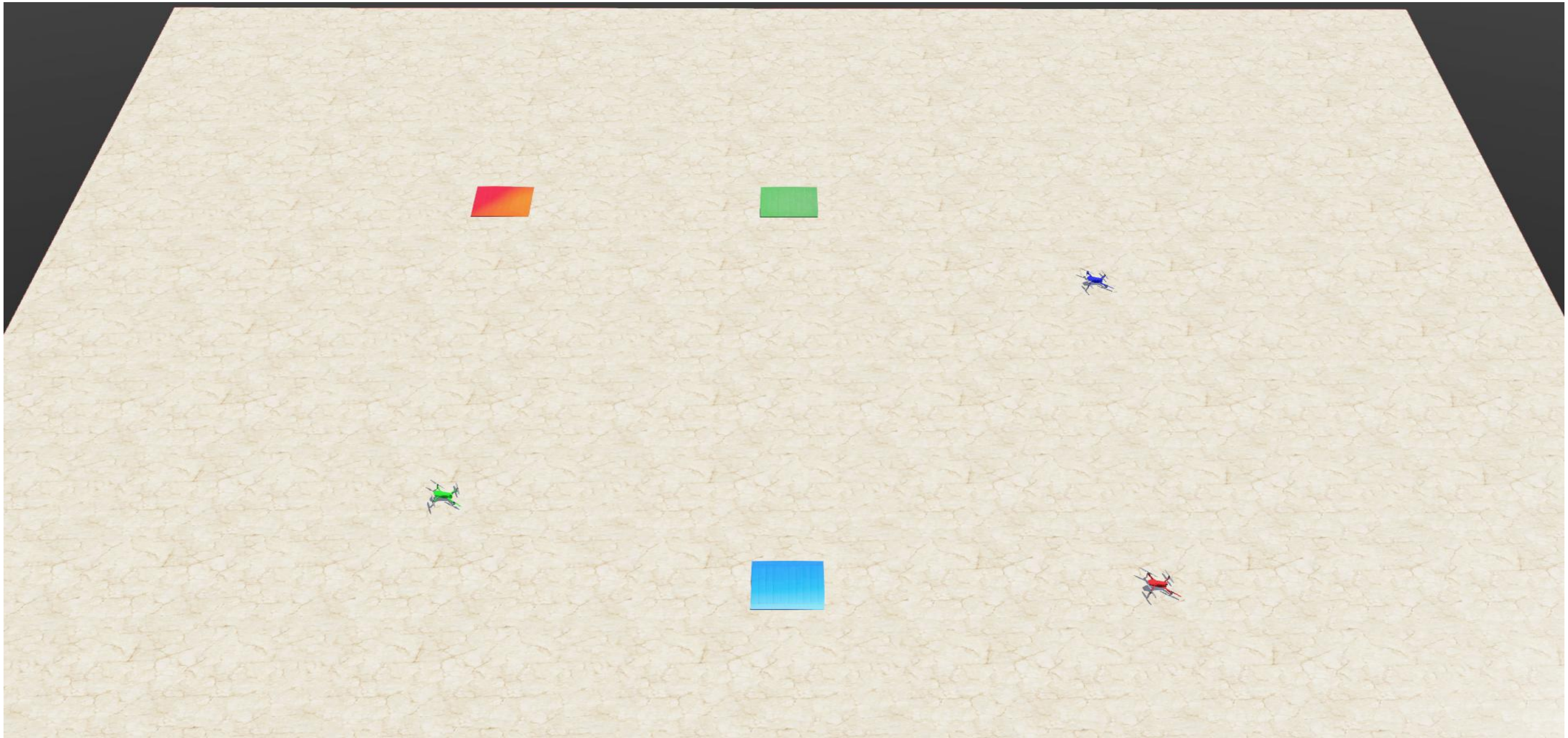
- Mean Rewards



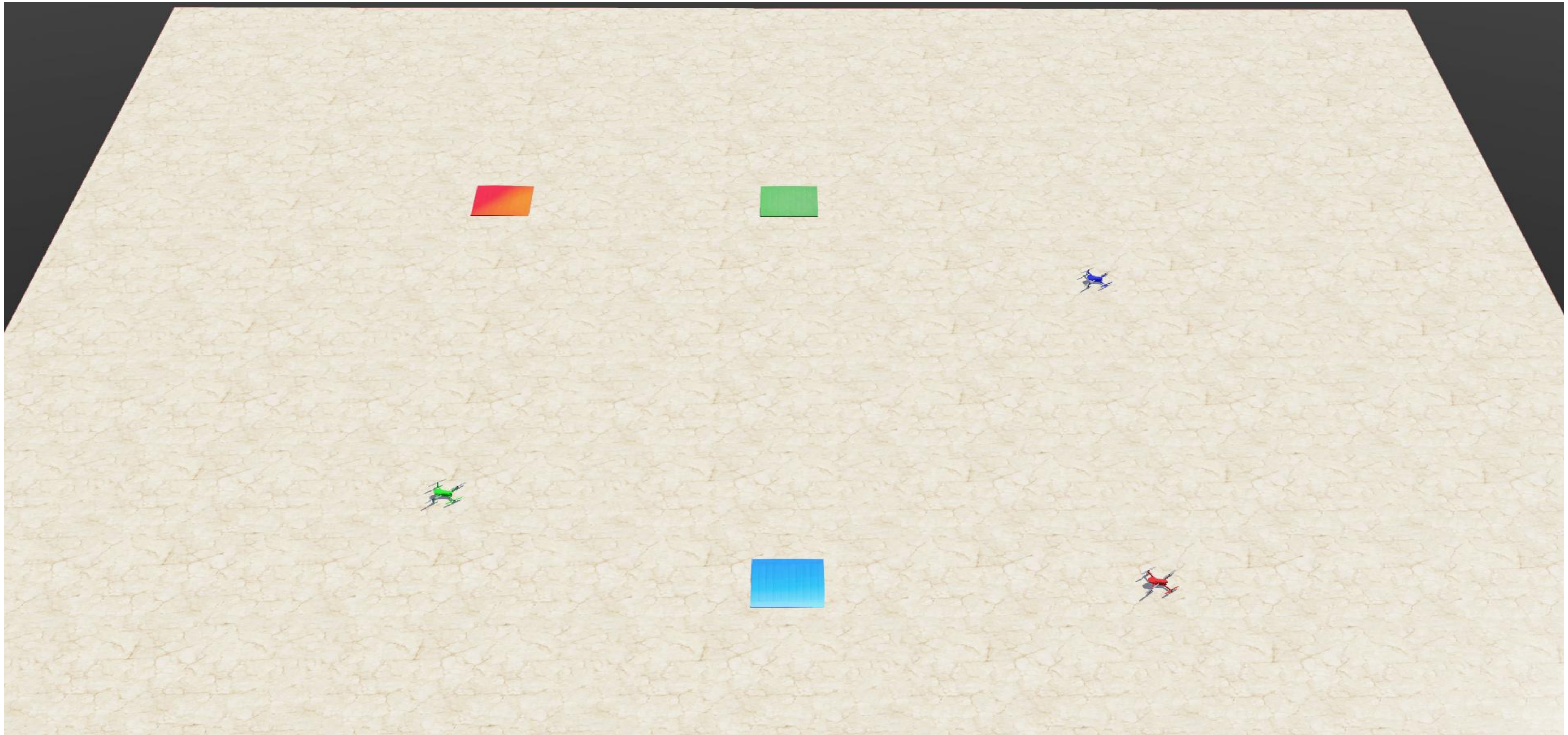
- Testing final policies



Multi-UAV behavior using MADDPG

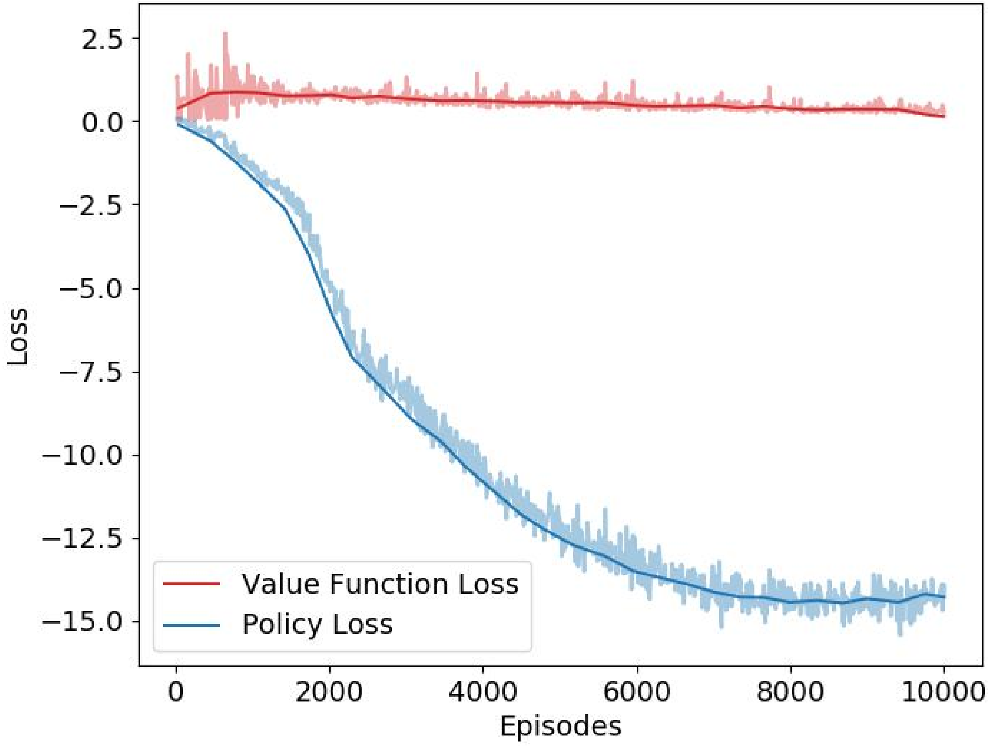


Multi-UAV behavior using Safe-MADDPG

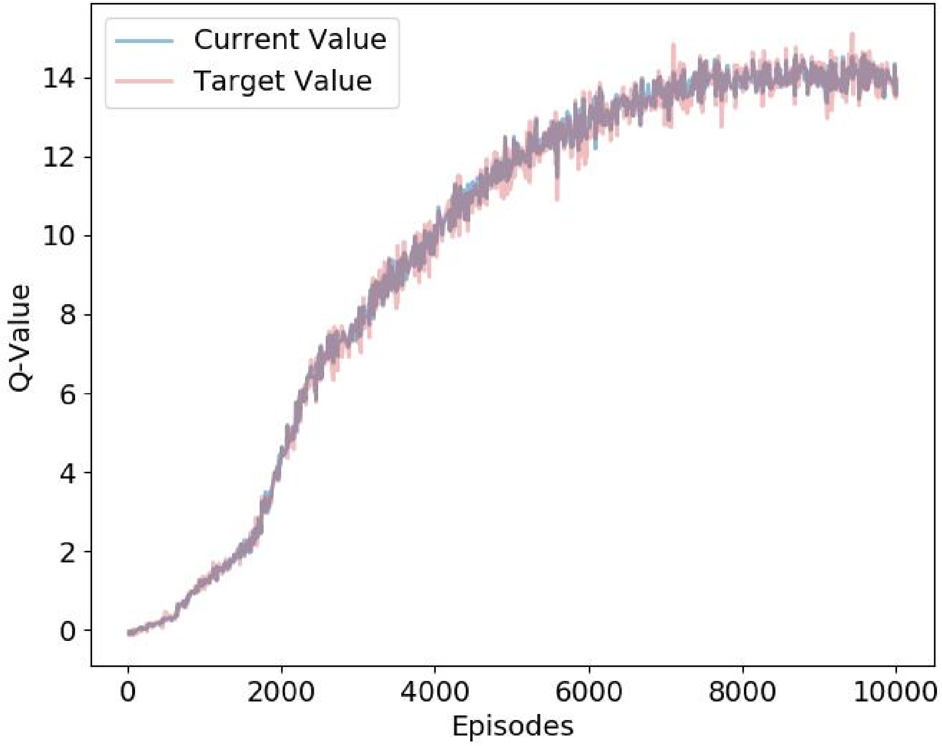


Evaluation

- Value & Policy Loss

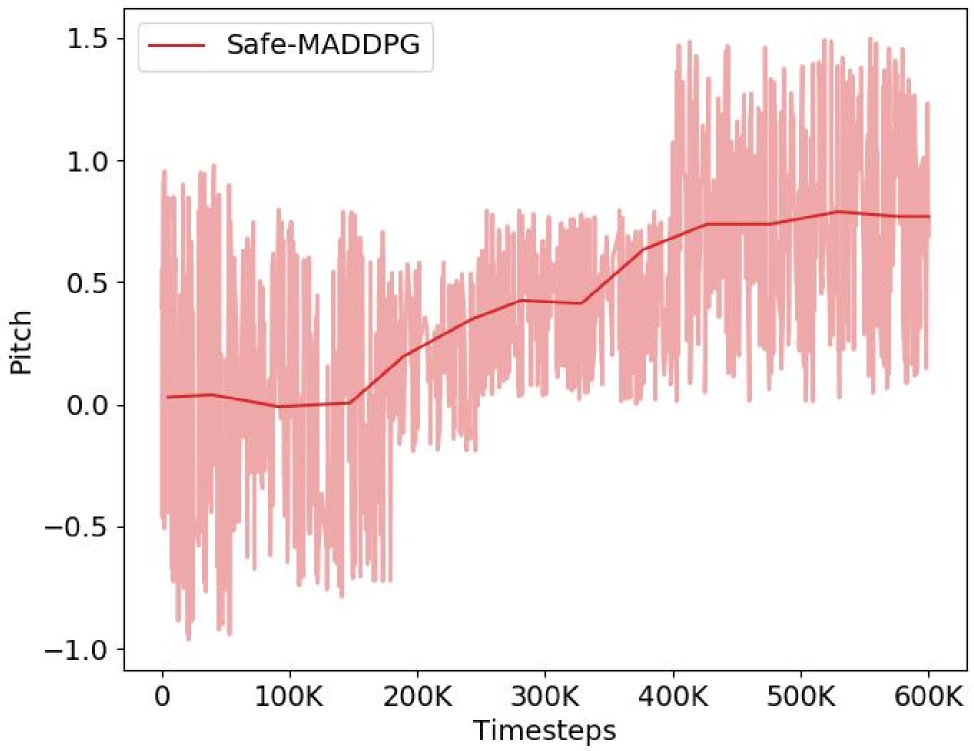


- Current – Target Q-Value

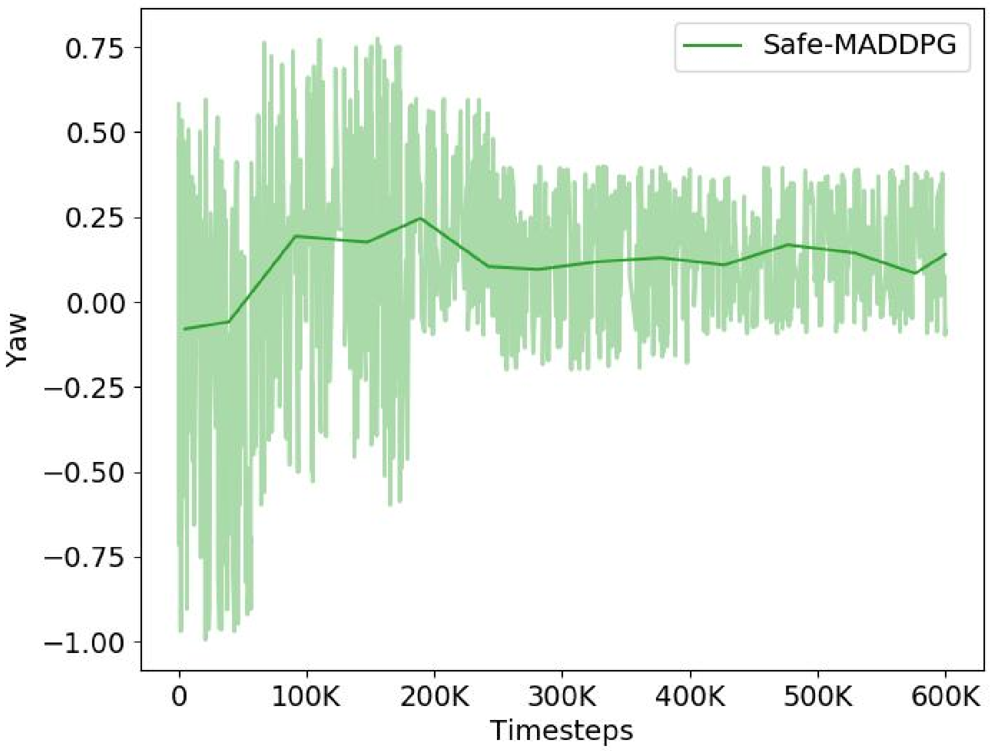


Evaluation

- Pitch variation



- Yaw variation



Conclusions

- Safe-MADDPG for learning safe cooperative policies for Multi-UAV navigation via safety layer
- The proposed Safe-MADDPG is better able to handle collisions than MADDPG
- Future work to evaluate with more complex environments with static and dynamic obstacles, etc.

References

1. [Deep Reinforcement Learning (researchgate.net)]
2. R.Lowe, Y.Wu, A.Tamar, J.Harb, P.Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” *CoRR*, vol. abs/1706.02275, 2017.
3. G.Dalal, K.Dvijotham, M. Vecerık, T. Hester, C.Paduraru, and Y.Tassa, “Safe exploration in continuous action spaces,” *CoRR*, vol. abs/1801.08757, 2018.
4. R.Cheng, G.Orosz, R.M.Murray, and J.W.Burdick, “End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019.
5. Webots, “http://www.cyberbotics.com,” open-source Mobile Robot Simulation Software. [Online]. Available: http://www.cyberbotics.com.
6. M.Kirtas, K.Tsampazis, N.Passalis, and A.Tefas, “Deepbots: A Webots-based deep reinforcement learning framework for robotics,” in *Artificial Intelligence Applications and Innovations*. Cham: Springer International Publishing, 2020.

Thank You!