Learning Safe Cooperative Policies in Autonomous Multi-UAV Navigation

Arshdeep Singh
Computer Science and Engineering
Indian Institute of Technology Ropar, INDIA
2019csm1001@iitrpr.ac.in

Shashi Shekhar Jha
Computer Science and Engineering
Indian Institute of Technology Ropar, INDIA
shashi@iitrpr.ac.in

***Abstract*—The deployment of multiple Unmanned Aerial Vehicles (UAV) in constrained environments has various challenges concerning trajectory optimization with the target(s) reachability and collisions. In this paper, we formulate multi-UAV navigation in constrained environments as a multi-agent learning problem. Further, we propose a reinforcement learning based Safe-MADDPG method to learn safe and cooperative multi-UAV navigation policies in a constrained environment. The safety constraints to handle inter-UAV collisions during navigation are modeled through action corrections of the learned autonomous navigation policies using an additional safety layer. We have implemented our proposed approach on the Webots Simulator and provided a detailed analysis of the proposed solution. The results demonstrate that the proposed Safe-MADDPG approach is effective in learning safe actions for multi-UAV navigation in constrained environments.**

***Index Terms*—UAV, Reinforcement Learning, Policy Gradient, Safe Navigation, Multi-agent System, Webots**

I. Introduction

Unmanned Aerial Vehicles (UAVs) are being deployed increasingly in various application domains such as entertainment, search and rescue, surveying, remote sensing, military, etc. The low cost hardware and off-the-selves flight controllers are driving the extensive use of UAVs with a strong push for adoption in e-commerce delivery business like Amazon & Zomato [1]. Although UAVs provide advantages such as invariance to road traffic, higher delivery rate under a specific timeline, easy deployment, etc., however to deploy a fleet of UAVs in e-commerce applications is still a far fry. The basic challenge lies in the autonomous navigation of multiple UAVs in a constrained environments with multiple target locations. Learning effective control policies in multi-UAV settings is tedious due to the huge joint state-space along with the continuous action spaces of the UAVs.

In this paper, we consider each UAV as an individual agent and formulate multi-UAV navigation in constrained environment as a multi-agent learning problem. We develop a reinforcement learning based solution to learn safe and cooperative navigation policies for the multi-agent system comprising UAVs. Safe navigation involves constraint satisfaction while the agents interact with the environment, thereby maximizing the total rewards. Learning with safety constraints has been explored through various approaches. Reward shaping is one such approach which encodes the safety constraints into the reward structure. Other approaches deal with formulating an additional controller that ensures safe exploration in the environment. We have modelled the safety constraints to handle inter-UAV collisions during navigation through action corrections of the learned autonomous navigation policies of the UAVs using an additional safety layer. This layer integrates the closed-loop solution provided by [2] for action correction with minimal perturbation whenever there is a constraint violation. We test our proposed algorithm using Webots Simulator [3] that provides a 3D environment with a real-world physics engine.

The rest of the paper is organized as follows. Section II describes the related work. Section III provides the preliminaries for our algorithm, and Section IV describes our proposed algorithm. Section V and Section VI cover the environmental design and evaluation results while Section VII concludes the paper.

II. Related Work

Multi-agent reinforcement learning (MARL) is gaining momentum recently. MARL has been researched in various areas such as joint action learning, cooperation and competitiveness, coordination, security, self-play, learning to communicate, transfer learning, imitation learning, and meta-learning [4]. It is also finding its application in game theory with respect to extensive form games. Recent works in MARL is following the approach of *centralized training* and *decentralized execution* for learning multi-agent policies. It has significantly improved the convergence of MARL systems. The same approach is followed in [5]. It can display complex emergent behavior between the agents with respect to their objectives. The authors in [5] used off-policy data for centralized training.

Recently safety constraints are being explored for the reinforcement learning algorithms taking real-world deployment criteria into considerations. [2] and [6] apply safety to the policy gradient algorithm namely Deep Deterministic Policy Gradient(DDPG) [7]. The former applies a closed-loop solution thereby finding optimal lagrange multipliers for finding the optimal action correction. In contrast, the latter applies a Control Barrier Function (CBF) to the existing policy of

978-1-6654-4175-9/21/$31.00 ©2021 IEEE

the agent. However, both works only consider the single-agent environment.

The authors in [8] apply a two-step approach to handle the inter-agent collisions. First, the algorithm is trained on the Optimal Reciprocal Collision Avoidance (ORCA) framework for collision avoidance. After that, the algorithm is trained on policy gradient till convergence. However, this approach becomes highly restricted in 3D environments where the agents are susceptible to falling into local minima. The authors in [9] provide a decentralized approach for UAV collision avoidance and path planning. However, their algorithm learns collision avoidance during learning which will not be practical for training real-world UAVs.

Our proposed approach utilizes both the methods proposed in [5] and [2] to learn a cooperative policy for collision avoidance while multiple UAVs are navigating to single or multi-target locations in a constrained environment. Our proposed method extends the safe policy method in [2] to multi-agent settings with a two-step process, first tuning the safety layer and then training the agents using MADDPG with the safe actions through the safety layer.

III. Preliminaries

In this section, we describe the building blocks of the methods employed to learn the cooperative policies for the safe navigation of multi-UAV system.

A. Markov Decision Processes (MDP)

The markov decision processes (MDP) provides a mathematical framework for modelling sequential decision making control problems. An MDP is a quintuple (S, A, T, R, s_0) where S is the state-space, A is the set of all possible actions, T denotes the model of the environment, R is the reward function and s_0 is the start state. The states in an MDP follow the markovian property [10].

We consider N UAVs (agents) in our system. Each agent makes an observations about the environment i.e. for N agents we have $O_t = \{o_1, o_2, o_3, ...o_N\}$ observations where t is the time index. Based on the observation space O_t, the agents execute the actions $A_t = \{a_1, a_2, a_3, ...a_N\}$. The observation o_i of each agent i includes a concatenation of agent's position in the global coordinates along with the relative positions of other agents with respect to agent i, i.e. $o_i = \{s_1, s_2, ..s_i, ..., s_N\}$ where s_k denotes the position of the k^{th} agent with respect to agent i. The action space A_i of each agent i consists of the pitch and yaw values (see [11] for details) responsible for the motion of the UAV.

Each agent follows a parameterized policy $\pi_{\theta_i} : o_i \rightarrow a_i$. The transition function is denoted as $T : O_t \times A_t \rightarrow O'_{t+1}$. Each agent i receives a reward conditioned upon the observation o_i and the action a_i i.e. $r_i^t : o_i^t \times a_i^t \rightarrow \mathbb{R}$, $\mathbb{R}$ is the set of real numbers. The goal (G_i) of each agent is to maximize its sum of discounted rewards i.e. $G_i = \sum_t \gamma^t {r^t}_i$, where γ is a constant called discount factor. The objective is to learn the parameters θ_i of the policy π_{θ_i} such the overall sum of discounted rewards of all the agents is maximized i.e. $\max \sum_i G_i$.

B. Multi-Agent Reinforcement Learning

Actor-Critic algorithms [10] are model-free methods to solve reinforcement learning problems where the agents don't require the model (T) of the environment. These methods optimizes a parameterized *policy* through policy gradients [12] and reduces bias in the policy parameters using a parameterized value function [13]. Lowe et al. [5] provide an actor-critic method for multi-agent domains called Multi-Agent Deep Deterministic Policy Gradient (MADDPG). The *non-stationarity* of the environment is a major issue in multi-agent systems as multiple agents learn and modify their policies simultaneously. MADDPG tackles this issues by making the critic network learn *centralized* action-value function $(Q(s, a))$ while the decentralized actor networks learn their policies individually.

The centralized critic in MADDPG is represented as ${Q_i}^{\mu}(o_i, A_t)$. Each agent learns its Q-function ${Q_i}^{\mu}$ separately. This leads to agents having separate individual rewards. As a consequence, the agents can be modeled in both cooperative and competitive settings. The decentralized actor networks for each agent i update their parameters using the following gradients [5]:

$$\nabla_{\theta_i} J \approx \mathbb{E}_{o,a \backsim D}[\nabla_{\theta_i} \mu_i(o_i) \nabla_{a_i} {Q_i}^{\mu}(o_i, a_1, ..., a_N)|_{a_i = \mu_i(o_i)}] \tag{1}$$

where D is the replay buffer used for drawing samples for training. The critic network uses the following loss function for parameters updates [5]:

$$L(\theta_i) = \mathbb{E}_{o,a,r,o'}[(y - {Q_i}^{\mu}(o_i, a_1, ..., a_N))^2] \tag{2}$$

where the target y is set as:

$$y = r_i + \gamma {Q_i}^{\mu'}(o'_i, a'_1, ..., a'_N) \tag{3}$$

where μ' indicates the target network and is updated using soft parametric update.

C. Incorporating Safe Actions

For collision free navigation of UAVs, the learnt policies must allow only safe actions to be executed. Safe exploration in continuous action spaces can be achieved by incorporating safety constraints into the policy. The authors in [2] applied a closed-loop solution to incorporate safety constraints into the policy in the form of a safety layer. The constrained policy optimization equation is given as:

$$\max_{\theta} E[\sum\nolimits_t \gamma^t R(o_i, \mu_{i|\theta}(o_i))] \tag{4}$$

s.t.

$$c_k(o_i) \leq C_k, \forall k \in K \tag{5}$$

where K is the number of safety constraints. The constraints are applied over each position of the UAVs with respect to the closest UAV. The policy from the actor is passed through a safety layer $g(s, w)$. The safety layer weights are learned through the following quadratic program :

$$a_i{}^* = arg\min_{a|i} \frac{1}{2} \| \ a_i - \mu_{i|\theta}(o) \ \|^2 \tag{6}$$

s.t.

$$c_k(o_i) + g(o,w)^T a_i \leq C_k, \forall k \in K \tag{7}$$

The optimal solution exists which is to find the optimal lagrange multipliers λ^* by solving the KKT conditions of the optimization problem. The above equation solves how little the safety layer has to perturb the original action output in the euclidean space to satisfy the constraints.

In the next section, we present our proposed approach to learn safe multi-UAV navigation policies using reinforcement learning.

IV. Proposed Method

In this paper, we propose *Safe-MADDPG* to learn safe cooperative policies for autonomous multi-UAV navigation. The proposed Safe-MADDPG learns a continuous domain deterministic policy for each agent with a shared reward structure. Our model makes the assumption that the location of the UAVs along with their navigation targets are available that can be encoded in the observation space.

The proposed method divides the learning process in two phases. In the first phase, the safety layer $g(.,.)$ is trained to estimate the constraint values of the actions generated by the current policy. Initially a random policy is taken as input and samples $((a_i, o_i, c_i, {c_i}^{next}))$ from the environment is generated having the current constraint values (c_i) and the next constraint value (${c_i}^{next}$) observed after the execution of the action a_i generated by the current policy. These samples are stored in a constraint replay buffer. To update the parameters of the safety layer $g(.,.)$, a mini-batch of the constraint samples are extracted from the constraint replay buffer. For each tuple in the mini-batch, the safety layer $g(.,.)$ weights are used to estimate the next constraint value ${c_i}^{next.predicted}$ for the input (a_i, o_i, c_i). The gradient over squared loss for each predicted constraint value $\| \ {c_i}^{next} - {c_i}^{next.predicted} \ \|^2$ is used to update the parameters of the safety layer. Each action a_i comprises *pitch* and *yaw* values and the constraints are calculated over. The action tuple $< pitch_i, yaw_i >$ indicates whether the values of the action a_i are within allowed ranges that encodes the constraints c_i. Hence, for 2-element action space, there are a total of four constraints for the allowable maximum and minimum values of *pitch* and *yaw*.

In the second phase, the MADDPG actor-critic network is trained by sampling actions from the current deep policy $\mu_{i|\theta}$ with an exploration noise. An optimal lagrange multiplier ${\lambda_i}^*$ is calculated (based on [2]) for each constraint by passing the sampled action a_i to the safety layer $g(.,.)$ given as:

$${\lambda_i}^* = \frac{g(o_i,w_i)^T \mu_{i|\theta}(o_i) + c_i - C_i}{g(o_i,w_i)^T g(o_i,w_i)} \tag{8}$$

The adjusted/corrected safe action ${a_i}^*$ is then calculated as:

$${a_i}^* = \mu_{i|\theta}(o_i) - {\lambda_i}^* g(o_i, w_i) \tag{9}$$

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
  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
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;
    $o_i \leftarrow o_i'$;
    while agent i = 1 to N **do**
      Sampling of a randomized minibatch of M samples $(o_i^j, a_i^j, r_i^j, o'^j_i)$ from D;
      set ${y_i}^j = {r_i}^j + \gamma {Q_i}^{\mu'}(o'^j_i, a'_1, ..., a'_N)|_{a'_i = \mu'_{i|\theta}(o_i{}^j)}$;
      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; $\nabla_{\theta_i} J \approx \frac{1}{M} \sum_j \nabla_{\theta_i} \mu_i(o_i{}^j) \nabla_{a_i} {Q_i}^{\mu}(x_i{}^j, {a_1}^j, .., {a_N}^j)$
    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

The safe action ${a_i}^*$ is then executed in the environment and the resulting sample $(o_i, {a_i}^*, r_i, o_i')$ is stored in the replay buffer. The mean square error between the current critic's value and the target is used as the loss function to update the critic parameters. Finally each actors' policy parameters θ_i are updated using the deterministic policy gradient over the critic's action value. Algorithm 1 depicts the proposed Safe-MADDPG algorithm.

Hence, the Safe-MADDPG incorporates the safety navigational constraints for learning cooperative policies for the multi-UAV system. In the next section, we evaluate the efficacy of our proposed Safe-MADDPG method using multi-UAV simulations with real-world physics engine.

V. Experimentation

In order to evaluate the proposed approach, we employ a 3D robot simulator called Webots [14]. For our experimentation, we have used DJI Mavic UAV model provided by Webots. The observation space of each agent/UAV consists of location and velocity information. The position of each UAV i (P_i) is a 3-element tuple describing the 3D coordinates represented as $P_i : [{p_i}^x, {p_i}^y, {p_i}^z]$. Velocity ($V_i$) is 6-dimensional tuple which represents linear and angular velocities along the three dimensions represented as $V_i : [{l_i}^x, {l_i}^y, {l_i}^z, {v_i}^x, {v_i}^y, {v_i}^z]$ where l_i and v_i are linear and angular velocities along the three dimensions respectively. The observation space o_i of a UAV i is the concatenated space of $\{P^i, V^i, P^i_{Target}, \bigcup_{\forall j \in N; j \neq i} P^j\}$.

In the simulations, the UAVs were tasked to reach an assigned target location within a specified space (10m x 10m) without colliding with each other. During the training, UAVs were first given thrust to reach a height of 1m for vertical take-off. After the take-off, the learning cycle (Algorithm 1) begins to learn a safe policy. The start positions of the UAVs and their targets were randomized in every episode for generalization during the training. The episodes end when either the UAVs crashed with each other or all the UAVs reached their targets.

As mentioned earlier, the action space is a 2-dimensional tuple consisting of *Pitch* and *Yaw* disturbances i.e $a_i : \{pitch_i, yaw_i\}$. *Pitch* is the rotation of the UAV along the horizontal axis while *Yaw* is the rotation of the UAV around the vertical axis. In our setup, the *Roll* value is fixed through out the simulation and its value is set to *1.0*. The *Pitch* and *Yaw* disturbances as generated by the learnt policy are fed to the UAV through the robot controller of Webots.

The reward function used in the experiments is conditioned upon the linear distance & angular distance to the target and the collisions. The angle between the UAV's heading and its target affects the *yaw* angle as the UAV tries to position itself towards the target. Linear distance to the target affects the *pitch* as it is necessary for driving the UAV towards the target. If during the navigation, the UAVs collide en-route then a collision penalty is imposed. Table I lists all the rewards used to define the reward function.

Further, the actor network for each agent i is modeled with a 3-layer multilayer perceptrons in PyTorch. The input to the actor network is the observation o_i. It has a 30-element tensor for three agents environment. The number of neurons of the hidden layer is set to 64. The output from each actor network is 2-dimensional UAV action. The centralized critic takes the joint action space for training. For non-linearity, *Relu* activation is applied to the hidden layers and *LeakyRelu* for the output layer of the network. Batch normalization is done before training with batch size of *1024*. In addition, OU noise [5] has been used to provide exploration in the environment.

TABLE I
Reward Structure for the Multi-UAV Navigation

Reward Value	Conditions
-4	if the angle between agent 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

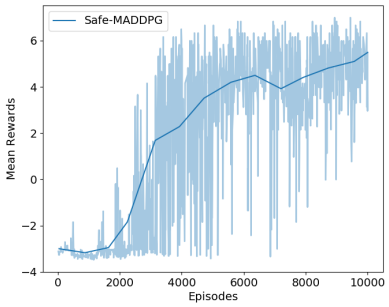

Fig. 1. Mean Training Rewards

VI. Results and Discussions

We trained the proposed Safe-MADDPG algorithm in the Webots environment using three UAVs. The locations of the UAVs and their targets were randomized in each episode for generalization.

A. Mean Rewards

Figure 1 shows the mean rewards per episode collected during the training along with a smoothed trend line. As can be observed, the mean rewards shows an upward trend indicating the learning progress as the UAVs are able to discovers safe actions in order to reach their destinations. Similar configurations with hyper-parameter tuning have been tested and the mean reward graphs show a similar pattern. The hyper-parameter tuning involves adjusting the batch size for training, setting the optimal learning rate, the value of τ parameter, number of exploration episodes, and noise exploration percentage.

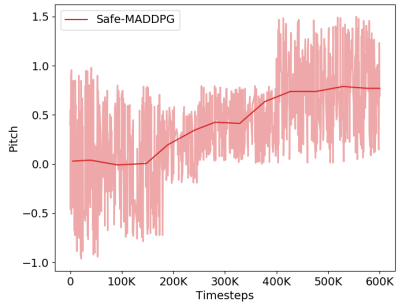

Fig. 2. Pitch value received from the Actor Network

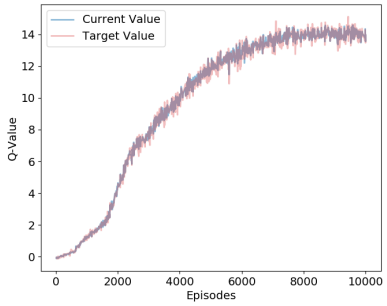

Fig. 4. Current v.s. Target Q-value of the Critic

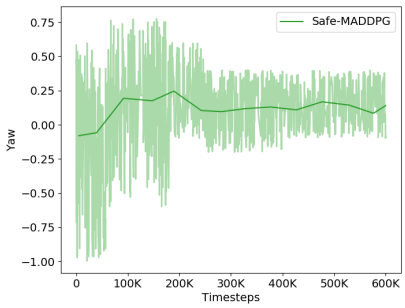

Fig. 3. Yaw value received from the Actor Network

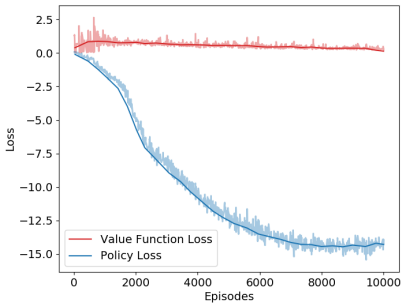

Fig. 5. Policy and Value function loss

The dependency on pitch and yaw disturbances is leading up to the increase in the mean rewards. These disturbances have a considerable effect on the motion of the UAV as these are the driving parameters for configuring the UAV in the direction of the target.

B. Variations in Pitch and Yaw

The graphs in Figures 2 and 3 depict the value of *pitch* and *yaw* disturbances along with smoothed trend lines for each time-step during the whole training phase as received from the actor network. The graph in Figure 2 shows that during the initial training phase, the pitch values are highly unstable. However, they start becoming positive mid-way and are mostly positive at the end of the learning showing positive movement of the UAVs towards the target. The graph in Figure 3 also indicates initial instability and then becomes centered around zero Yaw disturbance. This makes sense as the pitch is the primary driver of the *forward* motion of the UAV and yaw is used to align the UAV in the direction of the target. During the evaluation, we found that the learnt policy first makes the UAVs to align themselves towards the target focusing mainly on yaw (low positive pitch). After the alignment, there is a high positive pitch (low yaw angle) for the movement towards the target.

C. Critic's Learning

The graph in Figure 4 depicts the *critic's current* value against its *target* value. The value of the critic indicates how good is the policy in the current state. As can be observed, the graph shows an upward trend with respect to the episodes. This shows that the UAVs are able to find themselves in good states as the training progresses towards completion resulting in a better cooperative policy.

D. Policy and Critic Losses

The graph in Figure 5 depict the *policy loss* and *critic's value loss*. As can be observed, the value loss remains very low (close to 0.0) throughout the training. Our previous graph shows that the current q-value remains close to the target q-value, leading to a low value loss. The policy loss keeps getting low with more training episodes indicating that the agents learning the optimal behavior and reaching the destination using safe actions. It also aligns with the mean reward graph in Figure 1; as the latter gets higher, the policy loss starts getting low. Policy loss is with respect to the actor network whereas value loss is the loss in the critic network.

E. UAV Trajectories

Figure 6 depicts the trajectory of the three UAVs (shown in different colours) of a sample evaluation episode. The UAVs are tasked to take off from their start positions, reach their targets and hover. As can be seen in the 3D plot, initially all the UAVs rise to their configured altitude of 1m. After that, the UAVs start moving towards the target. It can be noted that the trajectories of the UAVs are intermingled hence the UAVs

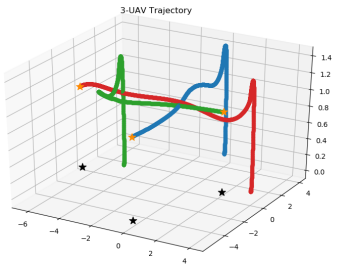

Fig. 6. Multi-UAV Trajectories

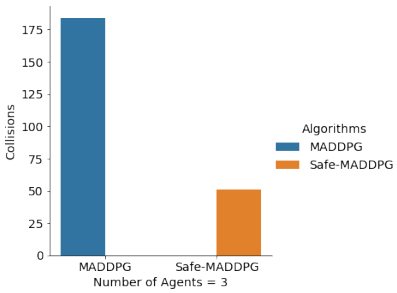

Fig. 7. Number of collisions over 1000 test episodes

would need to have a cooperative policy with safe actions to avoid collisions en-route to reach their targets. The trajectories taken by the UAVs clearly show that the UAVs are able to achieve their targets due to the learnt cooperative safe policy using the proposed Safe-MADDPG algorithm[1]. Interestingly, we noted that whenever the UAVs were about to enter in a situation that can result in a collision, one of the UAVs would reduce its pitch value thereby slowing down while the other UAV crosses the point of collision. This result captures an apt example of learning a safe cooperative policy by the UAVs.

F. Number of Collisions

Figure 7 compares the number of instances of collisions with the converged policy for baseline MADDPG and Safe-MADDPG over 1000 test episodes. As can be observed, the number of collisions found in case of Safe-MADDPG is manifold less than that of baseline MADPPG. These results clearly show that Safe-MADDPG helps the UAVs to learn a better policy as compared to the base MADDPG algorithm. However, there are still some states wherein the policy is not able to tackle the collisions effectively hence resulting in some collisions over 1000 episodes. As the Webots simulator uses a real-world physics engine, such instances of collisions occur when the UAVs are under the effect of inertia even though safe actions were provided.

[1]A video of this sample simulation episode can be accessed at the following link:with Safe-MADDPG https://youtu.be/xjZdsxv9Jd0 MADDPG only https://youtu.be/sMVtAhavFDE

VII. Conclusions

Multi-UAV navigation in constrained environment is a challenging task. This paper proposes a reinforcement learning algorithm named Safe-MADDPG to learn safe and cooperative policies for multi-UAV navigation in constrained environments. The proposed approach is implemented in the Webots simulator that uses a real-world physics engine. The results demonstrate that the proposed approach is able to learn safe and cooperative behaviours for multiple UAV navigation. Further, it has been shown that due to the safety layer, the UAVs are able to avoid collisions by adapting their pitch values. In future, we would like to test the proposed approach in more complex environments with occlusions, static and moving obstacles, etc.

References

[1] CNET. Ups, amazon delivery drones a step closer to reality with new us rules. [Online]. Available: https://www.cnet.com/news/ups-amazon-delivery-drones-a-step-closer-to-reality-with-new-us-rules/

[2] G. Dalal, K. Dvijotham, M. Vecerík, T. Hester, C. Paduraru, and Y. Tassa, "Safe exploration in continuous action spaces," *CoRR*, vol. abs/1801.08757, 2018. [Online]. Available: http://arxiv.org/abs/1801.08757

[3] Webots, "http://www.cyberbotics.com," open-source Mobile Robot Simulation Software. [Online]. Available: http://www.cyberbotics.com

[4] K. Zhang, Z. Yang, and T. Basar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *CoRR*, vol. abs/1911.10635, 2019. [Online]. Available: http://arxiv.org/abs/1911.10635

[5] 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. [Online]. Available: http://arxiv.org/abs/1706.02275

[6] 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, pp. 3387–3395.

[7] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[8] D. Wang, T. Fan, T. Han, and J. Pan, "A two-stage reinforcement learning approach for multi-uav collision avoidance under imperfect sensing," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3098–3105, 2020.

[9] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," 2018.

[10] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[11] M. A. Lukmana and H. Nurhadi, "Preliminary study on unmanned aerial vehicle (uav) quadcopter using pid controller," in *2015 International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (ICAMIMIA)*, 2015, pp. 34–37.

[12] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, ser. NIPS'99. Cambridge, MA, USA: MIT Press, 1999, p. 1057–1063.

[13] N. Heess, G. Wayne, D. Silver, T. P. Lillicrap, Y. Tassa, and T. Erez, "Learning continuous control policies by stochastic value gradients," *CoRR*, vol. abs/1510.09142, 2015. [Online]. Available: http://arxiv.org/abs/1510.09142

[14] 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, pp. 64–75.