Journal of Robotics and Control (JRC)
Volume 5, Issue 5, 2024
ISSN: 2715-5072, DOI: 10.18196/jrc.v5i5.22560

Three-Dimensional Coordination Control of Multi-UAV for Partially Observable Multi-Target Tracking

Vincentius Charles Maynad [1], Yurid Eka Nugraha[2*], Abdullah Alkaff [3]
[1, 2, 3] Department of Electrical Engineering, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia
Email: [1] vincentiuscm@gmail.com, [2] yurid@its.ac.id, [3]alkaff@ee.its.ac.id
*Corresponding Author

***Abstract*—This research deals with multi-UAV systems to track partially observable multi-targets in noisy three-dimensional environments, which are commonly encountered in defense and surveillance systems. It is a far extension from previous research which focused mainly on two-dimensional, fully observable, and/or perfect measurement settings. The targets are modeled as linear time-invariant systems with Gaussian noise and the pursuers UAV are represented in a standard six-degree-of-freedom model. Necessary equations to describe the relationship between observations regarding the target and the pursuers states are derived and represented as the Gauss-Markov model. Partially observable targets require the pursuers to maintain belief values for target positions. In the presence of a noisy environment, an extended Kalman filter is used to estimate and update those beliefs. A Decentralized Multi-Agent Reinforcement Learning (MARL) algorithm known as soft Double Q-Learning is proposed to learn the coordination control among the pursuers. The algorithm is enriched with an entropy regulation to train a certain stochastic policy and enable interactions among pursuers to foster cooperative behavior. The enrichment encourages the algorithm to explore wider and unknown search areas which is important for multi-target tracking systems. The algorithm was trained before it was deployed to complete several scenarios. The experiments using various sensor capabilities showed that the proposed algorithm had higher success rates compared to the baseline algorithm. A description of the many distinctions between two-dimensional and three-dimensional settings is also provided.**

Keywords—Coordination Control; Extended Kalman Filter; Multi-Agent Reinforcement Learning; Multi-Target Tracking; Multi-UAV System.

I. INTRODUCTION

In recent years, cooperative control of multi-Unmanned Aerial Vehicle (multi-UAV) systems has become a hot research topic in the field of flight control [1]. The increasing complexity of environmental conditions and tasks makes multi-UAV systems very necessary to complete certain tasks [2], which cannot be done by only a single UAV. Using cooperation, multi-UAV systems can exhibit superior coordination, intelligence, and autonomy than ordinary UAV swarms [3]. The use of multi-UAV systems has been widely discussed and developed in various fields, such as logistics [4], [5], disaster response [6], [7], source seeking problem [8], [9].

Multi-UAV has been recognized as an important force through various proven instances in recent modern warfare [10]. Currently, most combat UAVs are remotely controlled. But the realization of offensive multi-UAV has become possible [11] with the advancements in multi-UAV control like formation control [12], [13] and supported by networking technology [14]. These developments raise potential public safety concerns regarding dangerous UAV attacks [15]. Therefore, research on how to overcome this is essential [16].

One of the solutions that is quite important to research is Multi-Target Tracking Guidance (MTTG) [3]. MTTG is a condition where a group of UAVs, that are called pursuers, is assigned to track several previously unknown targets. It also can be seen as a pursuit-evasion problem which is popular in many research areas [17]. The main purpose of MTTG is to identify the number of potential targets in the pursuer's field of view (FOV) [18]. Generally, the pursuers have limited observation and communication sensor range so that each pursuer can't see their environment globally because it's almost impossible to model the entire process due to the complexity and uncertainty of the environment [19]. Each pursuer can only receive information from the results of its own sensor readings [20] and from its neighbors via communication.

In the UAV decision-making process, both the analytical solution methods and the intelligent optimization techniques have specific limitations. The analytical solution method requires precision and a comprehensive description of the decision model and without it, it is unsuitable to be applied to air scenarios. On the other hand, intelligent optimization methods struggle to define an appropriate solution space, with the related research often confined to the two-dimensional plane [21], making direct application to real-world scenarios challenging. Elevation angles representing three-dimensional systems are rarely discussed in the literature [22]. Whereas, accurate three-dimensional target positioning is of paramount importance in the aviation industry and air defense application [23]. Additionally, this kind of method is really dependent on hyperparameter tuning as it can significantly influence the model performance [24]–[26].

To solve MTTG using intelligent optimization, the problem is generally modeled as a Markov Decision Process (MDP) model [27]. MDP refers to a discrete-time stochastic decision-making process based on the Markov Property principle. In fact, MDP is typically employed in scenarios

devoid of model uncertainty where information concerning the states and actions of all UAVs is accessible. Naturally, with wider communication, better solutions will be easier to obtain [28], a notion known as the centralized. However, the concept of centralization has its drawbacks like complexity and computational time required [29]. Moreover, this assumption does not match the description of the problems to be faced. Therefore, a more specific MDP model must be used, namely the Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [30]. Dec-POMDP is an extension of MDP and one way of formulating multi-agent decision-making under uncertainty with partial information.

Then, Reinforcement Learning (RL) serves as the mechanism for determining the optimal solution within the Dec-POMDP model. RL operates as a learning paradigm where an artificial intelligence (AI) agent engages with its environment through trial-and-error, acquiring an optimal behavioral strategy based on reward signals obtained from previous interactions [31]. This methodology overcomes the disadvantages of alternative techniques, such as intricate modeling, challenging sample labeling, and tedious problem-solving processes. RL facilitates the generation of sequences of decision-making with enduring consequences through self-interaction training, independent of human intervention. It has been used and achieved success across diverse domains, such as automatic driving [32]–[35], education [36], [37], economics [38], [39], logistic resource scheduling [40]–[43], industrial process [44], [45], medical [46], database parameter tuning [47], and robotics [48]–[53].

Current academic research primarily focuses on the deterministic and static environment characterized by predominantly discrete and fully observable states [54]. Hence, both academic and practical applications mainly focus on a model-free approach [55], [56]. On this basis, RL algorithms are further classified into value function-based [57], [58] and policy-learning-based methods [59]. The value function approach aids decision-making by evaluating the value of each action within specific environmental states [60]. For multi-agent cases, RL is extended to Multi-Agent Reinforcement Learning (MARL). It focuses on studying the behavior of multiple agents that coexist in a shared environment. It needs to manage not only one agent but a group of agents at the same time.

MARL has many alternative solutions. In [55] and [56], they utilized a method named Proximal Policy Optimization (PPO). Meanwhile, [57] and [58] using Multi-Agent Deep Deterministic Policy Gradient (MADDPG). Q-Learning is used in [59] and [60]. There is hierarchical and curriculum learning paradigm [67], and others [68], [69]. Above that, the Centralized Training with Decentralized paradigm Execution (CTDE) makes it possible to train UAVs centrally while in execution, the UAVs make decisions based only on local observation [70], [71]. This paradigm significantly shifts complexity to training and makes execution light.

The above studies have discussed the strategies for using MARL, that have certain guiding significance. In general, applying the MARL algorithm to multi-robot tracking is a feasible solution. However, there is still room for further exploration. Much of the MTTG-related literature still solve this problem in a two-dimensional environment. In [3] and [65] actor-critic is used. PPO and Q-Learning are utilized in [73] and [74], respectively. The reasons are varied, such as agents used are mobile robots or UAVs but move only on the x and y axes. Pursuers that move in three dimensions are used in [75]. However, the targets move in two dimensions and uncertainties like noise have not been taken into account. On the other hand, three-dimensional agents are widely used in air combat references. In [69] and [70], they trained aircraft maneuvers in a 1-on-1 scenario. But, as mentioned in [78]–[80], most of them assumed that all information was known. The agents in [64] and [74] was knowing the positions of their adversaries. In practical applications, information, such as the position of the targets, can be incorrect due to sensor limitations or noises. Some studies have been conducted on sensor errors [82], [83] but noise in the state value hasn't been considered.

In this paper, we design an approach to describe the MTTG problem between pursuer UAVs and target UAVs in a three-dimensional and uncertain environment. Then, we modeled the problem as Dec-POMDP. The goal is to optimize the UAV control decisions in a decentralized manner. We use the extended Kalman filter to assist observation, updating the belief regarding targets. Besides that, Q-Learning will solve the control problem and obtain the best policy for the model. Lastly, we perform simulation studies to validate our proposed method.

The research contributions of this work are:

- We formulate the multi-UAV MTTG problem as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) where pursuers only receive partial observations about the targets.
- We take the problem into a more complex three-dimensional space according to the target faced.
- For pursuers, the three-dimensional dynamic model of a UAV is used. Extended Kalman filter is used to update belief of target position. Based on the soft Double Q-Learning algorithm and a CTDE architecture, a stochastic tracking policy is trained.

The paper is organized as follows. Section 1 delves into the background and summarizes related works about MTTG and MARL. Section 2 describes the problem model, details the proposed approach, and designs the MARL algorithm. The results and discussion are presented in section 3. Lastly, section 4 concludes the paper with suggestions for further research.

II. METHOD

A. Problem Formulation

Consider a scenario involving multiple UAVs where the objective of a group of N homogeneous pursuer UAVs is to track M homogeneous target UAVs as seen in Fig. 1. Each pursuer, denoted by i, possesses a state x_t^i, and the target, denoted by j, with state y_t^j moving with a discrete UAV dynamics model and a double integrator model, respectively. The pursuer's initial state, x_0^i, target initial state, y_0^i, and horizon T are predefined. Then, each pursuer must choose

control actions u_t^i to maximize the mutual information between y_t^j and the history of measurement denoted by $\boldsymbol{o}_{1:t}$. The objective for localizing target states is [74].

$$\max_{u(\cdot)} \sum_{i=1}^{N} \sum_{t=0}^{T} \sum_{j=1}^{M} I\left(\boldsymbol{y}_t^j, \boldsymbol{o}_{i:t} | \boldsymbol{x}_{1:t}^i\right) \tag{1}$$

with $x_{t+1}^i = f\left(x_t^i, u_t^i\right)$, $y_{t+1}^j = g\left(y_t^j\right)$, $o_t = h\left(x_t^i, y_t^1, \ldots, y_t^M\right)$ and $u_t^i = u(\boldsymbol{o}_{1:t})$ for $t \in \{0, T\}$. The function $f(\cdot)$ and $g(\cdot)$ represent the pursuer's dynamical model and the targets, while $h(\cdot)$ denote the model of observation. $u(\cdot)$ is the policy to determine the control action u_t^i, which will be explained later. The measurement history, $\boldsymbol{o}_{1:t}$, is shared among all agents.

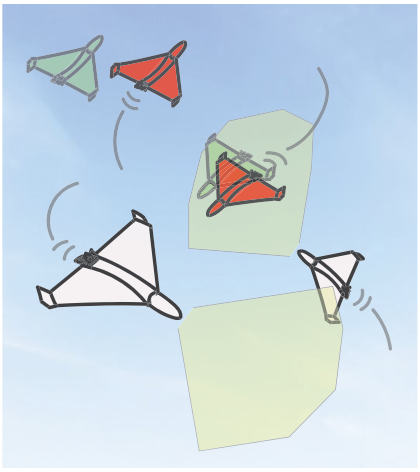

Fig. 1. Illustration of MTTG problem and scenario used in this paper. A group of pursuers with limited observation is trying to track multiple targets. White UAVs are the pursuers with limited observation ability. The red and green UAVs are the targets and their corresponding beliefs

Due to the limitation where each pursuer is only able to observe the environment partially, i.e., only receives observations solely of targets nearby, we define the task as a Dec-POMDP model. This model can be defined as a tuple $(\boldsymbol{N}, \boldsymbol{S}, \boldsymbol{A}_i, \boldsymbol{O}_i, \boldsymbol{P}, \boldsymbol{R}, \gamma)$. Here, $\boldsymbol{N}$ is a set of pursuers, $\boldsymbol{S}$ is the state space, in which $\boldsymbol{s} \in \boldsymbol{S}$. In this work, the state of the pursuer i is:

$$\boldsymbol{s}_i \coloneqq [x_i, y_i, z_i, v_i, \theta_i, \varphi_i] \tag{2}$$

Each variable will be explained in the next subsection. $\boldsymbol{A}_i$ represents a discrete set of actions available to each agent i, $\boldsymbol{Z}_i$ denotes the observations set for each agent i, $\boldsymbol{P}(s_{t+1}|s_t): \boldsymbol{S} \times \boldsymbol{A} \times \boldsymbol{S}$ indicates the transition probability model from one state to the next state after executing a particular action, $\boldsymbol{R}: \boldsymbol{S} \times \boldsymbol{A}$ stands for the global reward, and γ is the discount factor. Each pursuer makes individual decisions based on its local observations and the policy, π.

1) Observations: As an alternative to the limited observation of the UAV, the pursuer maintains a belief regarding target states. The belief distribution for the j^{th} target is defined as $B\left(y_{j,t}\right) = p\left(y_{j,t} | \boldsymbol{o}_{1:t}, \boldsymbol{x}_{1:t}\right)$ and its predicted distribution for the subsequent step as $\hat{B}\left(y_{j,t+1}\right) = p\left(y_{j,t+1} | \boldsymbol{o}_{1:t}, \boldsymbol{x}_{1:t}\right)$ [84]. We can simplify the optimization problem to minimize the cumulative differential entropy, $H(y_{t+1}|\boldsymbol{o}_{1:t}, \boldsymbol{x}_{1:t})$, under the assumption that y_{t+1} is independent of $\boldsymbol{x}_{1:t}$. If the belief is Gaussian,

$$B\left(y_{j,t}\right) = \mathcal{N}\left(\hat{y}_{j,t}, \boldsymbol{\Sigma}_{j,t}\right) \tag{3}$$

and

$$H(y_{t+1}|o_{1:t}, x_{1:t}) = \frac{1}{2} \log\left((2\pi e)^M \det\left(\boldsymbol{\Sigma}_{j,t}\right)\right) \tag{4}$$

Subsequently, the optimization function (1) becomes

$$\min_{\pi} \sum_{t=0}^{T} \log\det\left(\boldsymbol{\Sigma}_{j,t}\right) \tag{5}$$

At each time step $t \in T$, the pursuer with state x_t selects a control input u_t based on the policy π, relying on the predicted target belief $\hat{B}\left(y_{j,t+1}\right)$. So (1) can be maximized, measurement o_{t+1} is received from the sensor by the pursuer, while the actual target transitions from y_t to y_{t+1}. If the pursuer observes it, the corresponding belief distribution undergoes an update with the new measurement. This entire cycle is managed using centralized training and decentralized execution approach.

The belief regarding target states is continuously updated using an extended Kalman filter as new observations about the target are obtained. The dynamic of the targets remains unknown to the pursuers and importantly, the target states remain unaffected by the agent's control inputs. As a result, the value function and control policy of the i^{th} agent utilizes the predicted belief of the target j at time step $t+1$ as input at time t. Additionally, we adjust the belief of the j^{th} target within the pursuers body frame ($\hat{\boldsymbol{y}}^{(x_i)}$) to make sure there is no distribution shift between observations collected by different pursuers. The belief state is

$$\boldsymbol{s}_j \coloneqq \left[\hat{y}_{j,r}^{(x_i)}, \hat{y}_{j,\alpha}^{(x_i)}, \hat{y}_{j,\beta}^{(x_i)}, \hat{y}_{j,r}^{(x_i)}, \hat{y}_{j,\alpha}^{(x_i)}, \hat{y}_{j,\beta}^{(x_i)}, \log\det \boldsymbol{\Sigma}_j, \mathbb{I}\left(y_j \in \mathcal{O}(x_t)\right)\right] \tag{6}$$

where $\hat{y}_{j,r}^{(x_i)}, \hat{y}_{j,\alpha}^{(x_i)}, \hat{y}_{j,\beta}^{(x_i)}$ denote sphere coordinates of the mean of the target j's belief. The next three terms are their derivatives. $\boldsymbol{\Sigma}_j$ represent target belief covariance as in (5). As the pursuers have limited sensing range, a Boolean function $\mathbb{I}(\cdot)$ is used. If the actual target is in the vicinity, it returns 1, and 0 otherwise.

In the existing framework, we establish a centrally located extended Kalman filter accessible to each pursuer for updates. We ensure that when a pursuer detects a target, the belief associated with that target is accurately updated. When another pursuer accesses the Kalman filter, it retrieves the latest information from other agents. While this aspect of the system operates in a centralized manner, local observations and control actions remain unshared among the team. The framework as seen in Fig. 2, builds upon a single-agent strategy by employing parameter sharing across multiple agents.

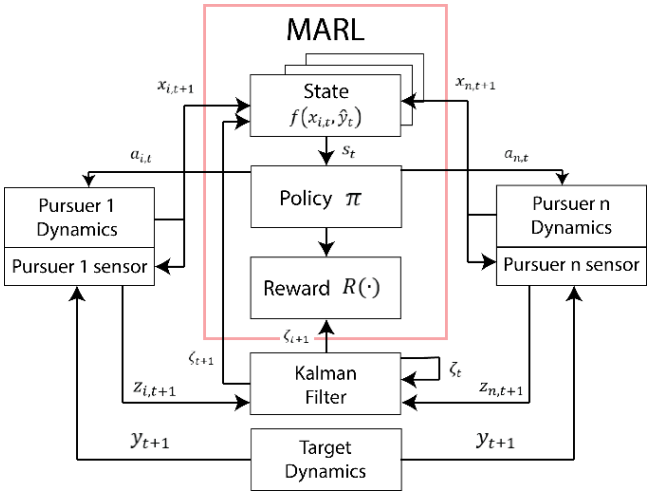

Fig. 2. Framework for multi-agent multi-target tracking. The MARL block consists of a variable number of pursuers with identical dynamics. Each pursuer receives a local observation state of target beliefs from the centrally stored extended Kalman filter

2) Rewards: Our aim is to discover the optimal policy π^* that maximizes the cumulative mutual information (1) between the i^{th} pursuer and the j^{th} target, effectively minimizing the objective (5). We simplify the objective by representing it as a discounted sum, where the reward is defined as the average uncertainty across all targets [74].

$$R(s_t, a_t) = -\frac{1}{M}\sum_{j=0}^{M} \log\det\left(\boldsymbol{\Sigma}_{t+1}^{j}\right) \tag{7}$$

so that the value function is

$$V^{\pi}(s) = -\mathbb{E}_{\pi}\left[\frac{1}{M}\sum_{t=0}^{T-1}\sum_{j=0}^{M}\gamma^t \log\det\left(\boldsymbol{\Sigma}_{t+1}^{j}\right)|s_0 = s\right] \tag{8}$$

Notice that the reward allocated to each agent is contingent on the performance of all other agents. This setup is advantageous as it ensures shared value functions and policies among agents. It also enables policy π^* to adapt based on how other agents, following the same policy, are tracking their respective targets. Intuitively, this strategy acknowledges that even when uncertainty is minimal for most targets, high uncertainty regarding the final target significantly impacts the overall average reward.

3) Action Space: Lastly, for the action space, we choose 12 discrete actions, as shown in Fig. 3 and Table I, that consist of a combination of tangential overload n_x, normal overload n_z, and roll angle ϕ. This approach reduces the complexity of UAV training. In contrast to the standard seven actions, it enables the UAV to execute consistent speed, acceleration, and deceleration control, mirroring the actual flight dynamics of the UAV more closely. The pursuer will choose an action $a \in \boldsymbol{A}_i$.

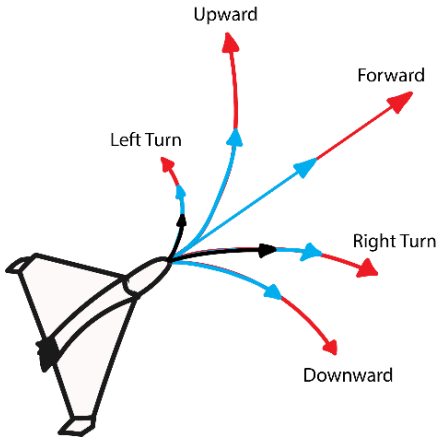

Fig. 3. Pursuer UAV maneuver library

TABLE I. MANEUVER LIBRARY

No.	Maneuver	Control Values		
		n_x	n_z	ϕ
a_1	Forward maintain	0	1	0
a_2	Forward accelerate	2	1	0
a_3	Left turn maintain	0	8	$-\arccos\left(\frac{1}{8}\right)$
a_4	Left turn accelerate	2	8	$-\arccos\left(\frac{1}{8}\right)$
a_5	Left turn decelerate	-1	8	$-\arccos\left(\frac{1}{8}\right)$
a_6	Right turn maintain	0	8	$\arccos\left(\frac{1}{8}\right)$
a_7	Right turn accelerate	2	8	$\arccos\left(\frac{1}{8}\right)$
a_8	Right turn decelerate	-1	8	$\arccos\left(\frac{1}{8}\right)$
a_9	Upward maintain	0	8	0
a_{10}	Upward accelerate	2	8	0
a_{11}	Downward maintain	0	8	π
a_{12}	Downward accelerate	2	8	π

B. *UAV Dynamic Model*

For the target UAV motion, we use a non-linear target model based on the double integrator with Gaussian noise [84].

$$\boldsymbol{y}_{t+1} = \boldsymbol{A}\boldsymbol{y}_t + w_t, \quad w_t \sim \mathcal{N}\left(0, \boldsymbol{W}(q)\right) \tag{9}$$

where $y_t = [x_t, y_t, z_t, \dot{x}_t, \dot{y}_t, \dot{z}_t]^T$ and

$$\boldsymbol{A} = \begin{bmatrix} I_3 & T_s I_3 \\ 0 & I_3 \end{bmatrix} \quad \boldsymbol{W}(q) = q\begin{bmatrix} \frac{T_s^3}{3} I_3 & \frac{T_s^2}{2} I_3 \\ \frac{T_s^2}{2} I_3 & T_s I_3 \end{bmatrix} \tag{10}$$

q is a noise constant, T_s is a given sampling time, and I_n is an $n \times n$ identity matrix.

According to Fig. 4, The motion model of the pursuer UAV is inspired by a fixed-wing UAV in [85] which has been modified to the description of the rotation angle used

$$\begin{cases} \dot{x} = v\sin\theta\cos\psi \\ \dot{y} = v\sin\theta\sin\psi \\ \dot{z} = v\cos\theta \end{cases} \tag{11}$$

where x, y, and z represent the position of the UAV within the coordinate system, v signifies the current speed direction

of the UAV, $\dot{x}$, $\dot{y}$, and $\dot{z}$ represent the change rate of v along the three coordinate axes, θ is the pitch angle, and ψ is the yaw angle.

In the same coordinate system, the dynamic model of UAV can be articulated as

$$\begin{cases} \dot{v} = g(n_x - \cos\theta) \\ \dot{\theta} = \dfrac{g}{v}(n_z \cos\phi - \sin\theta) \\ \dot{\psi} = \dfrac{g n_z \sin\phi}{v \sin\theta} \end{cases} \tag{12}$$

where g is the gravitational acceleration and ϕ is the roll angle. n_x and n_z represent tangential overload and normal overload, respectively. In this model, $[n_x, n_z, \phi]$ are the feasible control parameters for UAV maneuver control. So, the pursuer state is $\boldsymbol{x}_t = [x, y, z, v, \theta, \psi]^T$ as mentioned in (2). See [85], [86] for more details on the UAV maneuver model.

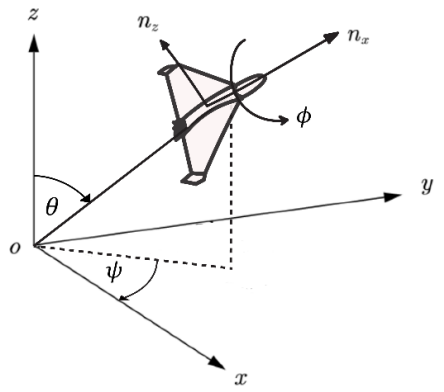

Fig. 4. Motion model of pursuer UAV

C. Frame Conversion and Belief Update

We work within a three-dimensional space, so we utilize a range (r)-bearing (α)-elevation (β) sensor depicted in Fig. 5 and written in (14). It's assumed that the pursuers can distinctly discern different targets. The measurement model of the sensor pertaining to each target is as follows:

$$\boldsymbol{z}_{i,t} = \boldsymbol{h}(x_t, y_{j,t}) + v_t, \quad v_t \sim \mathcal{N}(0, \boldsymbol{V}) \tag{13}$$

where $\boldsymbol{V}$ is an observation noise covariance matrix and the observation model is

$$\boldsymbol{h}(x_t, y_{j,t}) = \begin{bmatrix} r \\ \alpha \\ \beta \end{bmatrix} := \begin{bmatrix} \sqrt{(x_{tg} - x_o)^2 + (y_{tg} - y_o)^2 + (z_{tg} - z_o)^2} \\ \tan^{-1}\dfrac{y_{tg} - y_o}{x_{tg} - x_o} \\ \tan^{-1}\dfrac{\sqrt{(x_{tg} - x_o)^2 (y_{tg} - y_o)^2}}{(z_{tg} - z_o)} \end{bmatrix} \tag{14}$$

x_{tg}, y_{tg}, and z_{tg} represent the target position while x_o, y_o, and z_o are pursuer's position. Consequently, $(x_{tg} - x_o)$ means the distance between the pursuer and the observed target in the x-axis and so on.

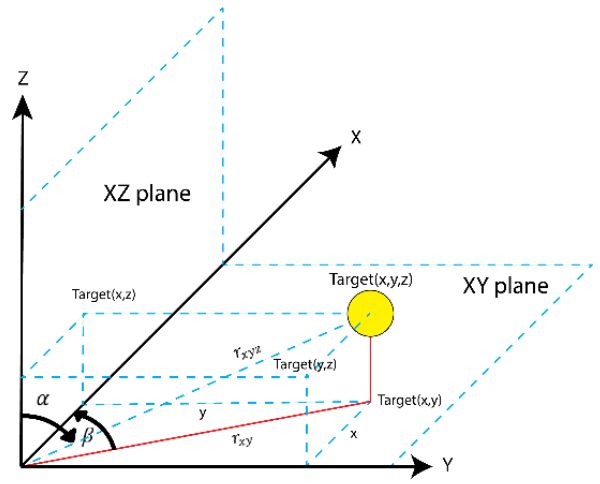

Fig. 5. Illustration of range-bearing-elevation measurement

If there is a target observed by the pursuer, the Kalman filter will update the beliefs regarding the target. The filter will estimate $\hat{r}$, $\hat{\alpha}$, and $\hat{\beta}$. The measurement model (14) is reduced to its linear form, with the Jacobian matrix of $\boldsymbol{h}(x, y)$

$$\mathbf{J}(\boldsymbol{h}(x,y)) = \begin{bmatrix} \frac{(\hat{x}_t - x_o)}{\hat{r}_{xyz}} & \frac{(\hat{y}_t - y_o)}{\hat{r}_{xyz}} & \frac{(\hat{z}_t - z_o)}{\hat{r}_{xyz}} & 0_{1\times3} \\ \frac{-\sin\hat{\alpha}}{\hat{r}_{xy}} & \frac{\cos\hat{\alpha}}{\hat{r}_{xy}} & 0 & 0_{1\times3} \\ \frac{\cos\hat{\alpha}\cos\hat{\beta}}{\hat{r}_{xyz}} & \frac{\sin\hat{\alpha}\cos\hat{\beta}}{\hat{r}_{xyz}} & \frac{-\sin\hat{\beta}}{\hat{r}_{xyz}} & 0_{1\times3} \end{bmatrix} \tag{15}$$

where $\hat{r}_{xy}$ indicates range which is calculated using the difference of x axis and y axis only. Meanwhile $\hat{r}_{xyz}$ also include z axis. We can get bearing and elevation in (15) in XYZ form if we substitute these variables according to the specification in Fig. 5, to become

$$\mathbf{J}(h(x,y)) = \begin{bmatrix} \frac{(\hat{x}_t - x_o)}{\hat{r}_{xyz}} & \frac{(\hat{y}_t - y_o)}{\hat{r}_{xyz}} & \frac{(\hat{z}_t - z_o)}{\hat{r}_{xyz}} & 0_{1\times3} \\ \frac{-(\hat{y}_t - y_o)}{\hat{r}_{xy}^2} & \frac{(\hat{x}_t - x_o)}{\hat{r}_{xy}^2} & 0 & 0_{1\times3} \\ \frac{(\hat{x}_t - x_o)(\hat{z}_t - z_o)}{\hat{r}_{xyz}^2 \hat{r}_{xy}} & \frac{(\hat{y}_t - y_o)(\hat{z}_t - z_o)}{\hat{r}_{xyz}^2 \hat{r}_{xy}} & \frac{-\hat{r}_{xy}}{\hat{r}_{xyz}^2} & 0_{1\times3} \end{bmatrix} \tag{16}$$

Insights into the concept of measurement, localization, and mapping can be found in [87].

How the Kalman filter works until data sent to the reward function is illustrated by Fig. 6. Initial state prediction, $\hat{\boldsymbol{x}}_0^-$, and initial prediction error covariance matrix, $\boldsymbol{P}_0^-$, are required before the filter calculates the Kalman gain using

$$\boldsymbol{K}_t = \boldsymbol{P}_t^- \boldsymbol{J}^T (\boldsymbol{J} \boldsymbol{P}_t^- \boldsymbol{J}^T + r)^{-1} \tag{17}$$

and estimation error covariance matrix, $\boldsymbol{P}_t$,

$$\boldsymbol{P}_t = (\boldsymbol{I} - \boldsymbol{K}_t \boldsymbol{J}) \boldsymbol{P}_t^- \tag{18}$$

as well as

$$\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}_t^- + \boldsymbol{K}_t (\boldsymbol{z}_t - \boldsymbol{J} \hat{\boldsymbol{x}}_t^-) \tag{19}$$

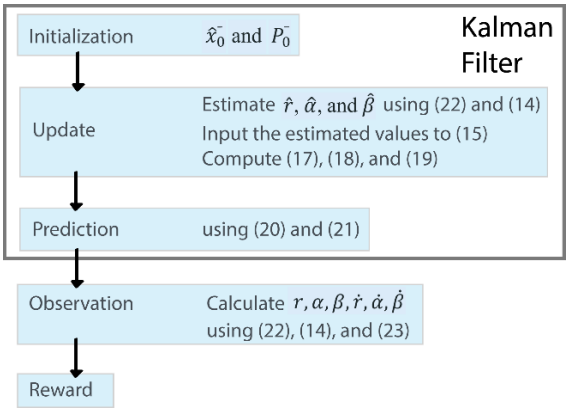

Fig. 6. Belief update flow in each time step. The initialization only occurs in the first step

We presume that these beliefs adhere to the same double integrator model as the target outlined in (9), although with differences in parameters $\boldsymbol{A}$ and $\boldsymbol{W}(q_b)$ where $q_b \neq q$. Essentially, this implies that the pursuer possesses only partial knowledge about the target. Then, the filter will predict the next state of the target for the next iteration, using

$$\hat{\boldsymbol{x}}_{t+1}^- = \boldsymbol{A}\hat{\boldsymbol{x}}_t \tag{20}$$

and calculate the prediction error variance

$$\boldsymbol{P}_{t+1}^- = \boldsymbol{A}\boldsymbol{P}_t\boldsymbol{A}^T + q_b \tag{21}$$

The belief is in Cartesian coordinates. Thus, we have to convert it to another, the Spherical coordinate. Additionally, as mentioned in II.A, we must adjust the belief of the target in the local frame to the pursuer's body frame. Using

$$\boldsymbol{C}_L^B = \begin{bmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ (-c\phi s\psi + s\phi s\theta c\psi) & (c\phi c\psi + s\phi s\theta s\psi) & s\phi c\theta \\ (s\phi s\psi + c\phi s\theta c\psi) & (-s\phi c\psi + c\phi s\theta s\psi) & c\phi c\theta \end{bmatrix} \tag{22}$$

and multiply it by the difference between the pursuer position and the target belief position then we will get the target relative position according to the pursuer in Cartesian.

Regardless of whether there is an update or not, beliefs about the target will be processed further to be used as observations in the MARL. To convert positions in Cartesian to Spherical, we can use the same equations as in (22) and (14). For the velocity, we need the target velocity in the pursuer's body frame. We can convert the target velocity from local to body frame using (22) and reduce it with the pursuer's velocity. Then, with the position and velocity of the target in the pursuer's body frame, we can calculate the target velocity in spherical coordinates using (23) and updating (6).

$$\begin{cases} \dot{r} = \dfrac{x\dot{x} + y\dot{y} + z\dot{z}}{r} \\ \dot{\alpha} = \dfrac{x\dot{y} - \dot{x}y}{x^2 + y^2} \\ \dot{\beta} = \dfrac{\dot{z} - \dot{r}\dfrac{z}{r}}{\sqrt{x^2 + y^2}} \end{cases} \tag{23}$$

D. Model Architecture

We need a policy that can adapt to varying target amounts visible in observation. Therefore, we view an observation as a set of elements processed by an embedding that is permutation-invariant and able to handle any number of elements within the observation set. The policy is constructed using an encoder-decoder model architecture inspired by DeepSets [88], augmented with self-attention layers. These self-attention layers ensure the property of permutation-invariance by emphasizing actionable information in the observation. The capability to handle an arbitrary number of elements is facilitated by the summation within the DeepSets architecture where additional elements are simply part of the linear combination of the embedding space.

According to [88], function $\phi(A)$ is invariant to the permutation of elements when acting on a set A if and only if it can be broken down into the form $\phi(A) \equiv \rho(\sum_{a \in A} \psi(A))$ for certain functions ψ and ρ. This concept is illustrated in Fig. 7. Both functions are adaptable and can be trained to incorporate invariance, while the summation allows $\phi(A)$ to accommodate sets of different sizes.

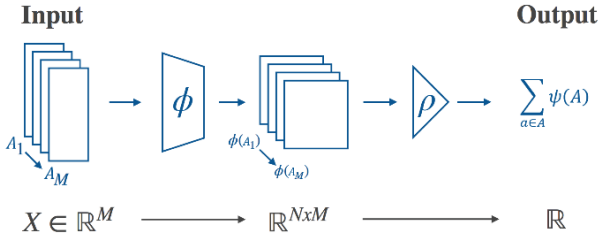

Fig. 7. Illustration of DeepSets concept

The self-attention mechanism is a potent tool for the value function to focus on the most relevant aspects of the input in relation to the output. It operates by embedding learning structures, where attention involves associating a query Q with a set of key-value pairs, K and V, to generate an output. This process essentially computes a weighted sum of the values, with attention being represented as a linear combination of V. The compatibility between the query and keys is gauged through the dot product QK^T. The attention module,

$$Attention(Q, K, V) = \omega\left(d_q^{-\frac{1}{2}} QK^T\right)V, \tag{24}$$

assigns greater importance to keys that exhibit a higher dot product with the query vector. Here, ω denotes an activation function like softmax, and $\frac{1}{\sqrt{d_q}}$ acts as a scaling factor. Additionally, enhancing the self-attention module involves exploring higher-order interactions between Q and K. This is achieved by projecting the inner product across multiple sub-spaces, resulting in the creation of the Multi-Head Attention Block (MAB).

MAB runs through an attention mechanism several times in parallel. Compared with the attention model, multi-head attention allows the model to jointly focus on information from different positions. The multi-head attention is computed as [70]

$$Multihead(Q,K,V) = Concat(head_1, \dots, head_i)W^o \quad (25)$$

where

$$head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right) \quad (26)$$

W^o, W^Q, W^K, and W^V are the parameters of the Concat function for Q, K, and V, respectively.

E. Maximum Entropy

Stochastic policy can be learned using the maximum entropy regularized objective [89] which aims to strike a balance between maximizing the anticipated return and entropy or in other words, to excel in the task while exhibiting as much randomness as feasible. The policy's entropy is modulated by a temperature parameter, α, which means that greater entropy corresponds to higher temperature and increased randomness, and the other way around. From distribution P across a random variable x, the entropy H is mathematically expressed as:

$$H(P) = \mathbb{E}_{x\sim P}[-\log P(x)] \quad (27)$$

At every time step, the pursuer receives an additional reward corresponding to the policy's entropy. Then, the RL problem can be reframed with a maximum entropy objective:

$$\mathcal{J}(\pi) = \sum_{t=0}^{T-1} \mathbb{E}_{(s_t,a_t)\sim\pi}[r(s_t,a_t) - \alpha \log \pi(a_t|s_t)] \quad (28)$$

This objective promotes stochastic policies by adding the equation to the policy. There are both theoretical and empirical advantages in stochastic policies. One, they encourage broader exploration while abandoning fewer promising paths. Additionally, the policies can encompass multiple modes of optimal behavior when faced with equally attractive actions, the policy assigns equal probability to each. Lastly, these policies have demonstrated significant improvements in exploration, making them valuable for cooperative tasks.

F. Soft Double Q-Learning

Q-Learning belongs to the category of off-policy RL methods. This approach has become the basis of many RL algorithms because, unlike other methods, Q-Learning is relatively simple and shows excellent learning capabilities in single-agent environments. It aims to minimize the expectation of the 1-step temporal difference (TD) error. Off-policy methods are called such because they maintain a dataset, denoted as $D = \{(s_t, a_t, r_t, s_{t+1})\}_{t=0,\dots}$, gathered using a behavior policy and used to estimate the current policy's value function. The value function will be represented using a Deep Neural Network (DNN) with parameters θ. Utilizing stochastic gradient descent, the optimal parameters θ^* can be determined to minimize the Huber loss applied to the TD error.

We use and enhance the advanced off-policy method known as Clipped Double Q-Learning [90], [91], serving as the framework for learning the stochastic policies. The original double Q-Learning algorithm trains two separate estimates of the actual Q value, represented by neural networks labeled as $Q_{\varphi 1}$ and $Q_{\varphi 2}$. This framework mitigates Q value overestimation by selecting the minimum value between the two independent estimates. This approach assigns higher values to states that have lower variance estimation errors, leading to more consistent learning. The update process for double Q-Learning is outlined below

$$Q^*(s,a) \leftarrow r + \gamma \min_{k=1,2} Q_{\phi_k'}\left(s', \arg\max_{a'} \sum_{k=1,2} Q_{\phi_k}(s',a')\right) \quad (29)$$

ϕ_k' denote target network k.

In practice, double Q-Learning learns a deterministic policy by selecting actions based on the $\arg\max\big(Q(s,a)\big)$. To acquire a stochastic policy, we modified the conventional double Q-Learning framework with an entropy-regularized objective. This algorithm referred to as soft Double Q-Learning, is designed to balance the anticipated return and entropy while exploring. The process for soft double Q-Learning is explained in Algorithm 1 and the update is [74]

$$Q^*(s,a) \leftarrow r + \gamma \min_{k=1,2} \mathbb{E}_{a'\sim\pi}\Big[Q_{\phi_k'}(s',a') - \alpha \log \pi(a'|s')\Big] \quad (30)$$

When using a stochastic policy, actions are determined through sampling from a multinomial distribution, which is derived from the Q function values log softmax. During exploration phases, higher entropy causes the policy to sample actions beyond those with the highest value. This approach aids the policy in exploring less-visited areas of the state space, therefore enhancing exploration.

Algorithm 1 Soft Double Q-Learning

Initialize environment with N pursuers and M targets
Initialize replay buffer $\mathbb{G}$
Initialize value networks $Q_{\phi 1}$ and $Q_{\phi 2}$
Initialize target networks ϕ_1', and ϕ_2'
for each episode **do**
  Random sample $n \in [1, N]$ and $m \in [1, M]$
  For each step $t = 0, T$ **do**
    for $i = 0, n$ **do**
      observe s_i and select $a_i \sim \pi(a_i|s_i)$
      do a_i, get reward r_t, observe s_i'
      store (s_i, a_i, r_t, s_i') in replay buffer $\mathbb{G}$
    end for
  end for
  for each update **do**
    select a batch of N: $g^t \sim \mathbb{G}$
    calculate $Q^*(s,a)$ using (30)
    perform clipped gradient descent step on $\sum_{k=1,2} huber\left(Q_{\phi_k}(s,a) - Q_{target}(s,a)\right)$
    Update ϕ_1' and ϕ_2'
  end for
end for

The soft Double Q-Learning algorithm is modified with parameter sharing to offer centralized training with

decentralized execution for multi-agent learning. The algorithm starts with sampling the environment with each pursuer, feeding their own observations through shared networks to generate unique experiences. When every pursuer acts according to the present policy, the entire environment changes from t to $t+1$. The second half of the algorithm updates value functions using stochastic gradient descent, eliminating the non-stationarity issue regarding decentralized trained systems. Each pursuer is given a copy of the learned policy for decentralized execution upon completion of training. To improve generalization, n and m number of pursuers and targets are uniformly randomly sampled during training. This is done to create a different number of cooperative agents in the team. This flexibility and diverse training episodes expand the task space, resulting in a more robust policy. This is especially important for real-world implementations that are full of possibilities.

III. RESULTS AND DISCUSSION

A. Setup

The MTTG scenario is in a highly uncertain three-dimensional environment. We initialize agent, target, and belief locations randomly. Targets move unpredictably with their fixed double integrator model and are added by Gaussian noise. Moreover, both the belief and observation models include noise components, further enhancing the environmental randomness. But we implement a restricted random initialization for target locations and their associated beliefs to decrease training variance and focus on tracking rather than exploration.

All agent positions, $x_{0,\ldots,N}$, are randomly placed within the designated map. Target positional components, $y_{0,\ldots,M}$, are initialized 5-10 kilometers from a randomly chosen agent. Then, Gaussian belief distributions for target locations are initialized within 0-5 kilometers of their respective targets, with covariance Σ. The initial position of pursuers and evaders, as well as the belief of targets, will change in each experimental iteration.

Both pursuers and targets are constrained to a maximum velocity of $300\frac{m}{s}$. The initial velocity for all targets is set to $90\frac{m}{s}$. T is set to 200 steps. Each agent has an observation range of 20 kilometers, bearing and elevation for $\frac{\pi}{4}$ rad. These setup values are inspired by a fixed-wing aircraft common parameters used in research. Training, testing, and evaluation are carried out in a space measuring 50 kilometers × 50 kilometers × 50 kilometers without obstacles.

In multi-agent multi-target environments, maintaining a balance between the number of pursuers and targets is essential for effective behavior. We'll conduct testing on tasks involving n pursuers within the range of $[1, N]$, and m targets within $[1, M]$. Subsequently, we'll assess the method's performance, without requiring retraining, across different task configurations. The testing will be conducted for 16 scenarios with various amounts of pursuers and targets.

B. Agent Training, Testing, and Evaluating

In the MARL model, each layer contains 256 hidden units, and there are 4 attention heads. We employ Adam optimizer with a learning rate of 0.0001. Learning rate is a factor determining how much new information overrides old information. With that value, pursuers can learn quickly but not too fast so as not to miss important details. The discount factor γ is set to 0.99. This factor encourages long-term beneficial actions rather than short-term gains. The replay buffer size is 10^6, and the training batch size is 256. The temperature parameter α is 0.4, and ReLU serves as the activation function.

We train the model for 500000 steps that are divided into 20 episodes as shown in Fig. 8. Each training episode consists of training and testing, and they have their own average rewards. It was trained using 4 pursuers and 3 targets scenario. As we can see, our algorithm initially exhibits an upward trend for the initial 25000 steps, following which it gradually stabilizes. From the conclusion of the first episode, the average reward consistently maintains a positive value, signifying the pursuers' capability to track the targets. Convergence is achieved around the 50000 step mark, especially for the training. The final score stabilized at approximately 259 for the training and 302 for the testing.

Fig. 8. Average rewards of training and testing episodes

This result comes from the fact that during evaluation, the pursuers act with their learned policy. But, when training, they use a stochastic policy to explore the environment. This is a standard practice in MARL, despite the fact we want to achieve some form of continual learning. Another cause that also can be considered is noise in the environment.

To validate the result, we show how the Q value increases over time as seen in Fig. 9. There are two Q values because we used a double Q-Learning paradigm, but it looks like there is only one due to their identical values. Their value will converge around 380. Q-value represents the value of taking a specific action in a specific state. Therefore, this value can be said to estimate the rewards that can be obtained in the future. A higher Q-value means the potential reward obtained will be higher.

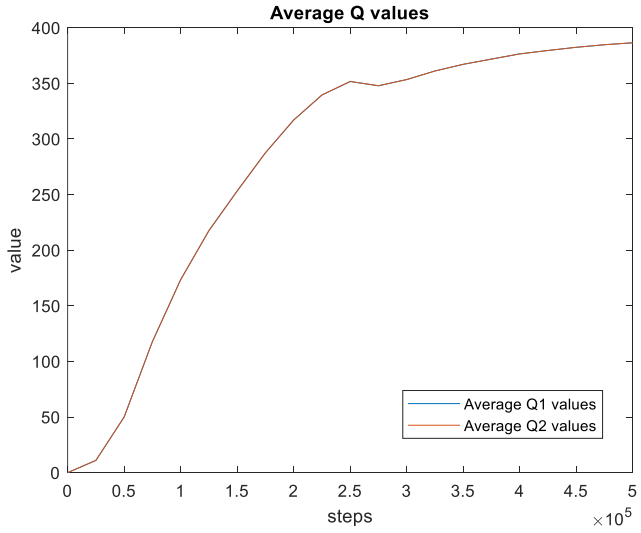

Fig. 9. Change in Q value during training

Then, for evaluation, we evaluate the policies resulting from the training process in sixteen scenarios based on the number of pursuers and targets on 50 episodes each with the same seed. In scenario 1, there is 1 pursuer and 1 target. The second scenario comes with 2 pursuers and 1 target, and so on, until there are 4 pursuers and 4 targets. The results can be seen from Fig. 10 up to Fig 12. We count the success rate of each scenario where a successful episode is indicated with a positive reward from that episode. If the episode reward comes with a negative value, it identifies the failure of the episode. We present the data in percentage form.

Even though training and testing show positive results, there is still the potential for negative rewards. This is caused by the stochastic nature of the environment and pursuer behavior. Apart from that, differences in initial conditions can also determine success. Episodes fail when stochastic policy pursuers delay excessively, causing targets to move beyond observable range, and compelling pursuers to resort to random searches.

As the number of pursuers increases, the percentage of success tends to increase. What is quite interesting is that when the target increases, it turns out that the possibility of success also becomes greater. This result is possible because of the reward function that we use where the reward calculation is done from the covariance matrix regarding the target. The more targets, the bigger the matrix, and the bigger the potential reward. One thing that is certain is that with a larger number, the pursuer has a higher potential for success even if tested with the same number of targets. But the success percentage of all scenarios is above 50%. This phenomenon arises as the policy operates with greater intelligence to reduce the uncertainty surrounding all targets, prioritizing long-term team rewards over immediate gains. Conversely, a policy-oriented towards instant rewards would lead pursuers to solely pursue the nearest target.

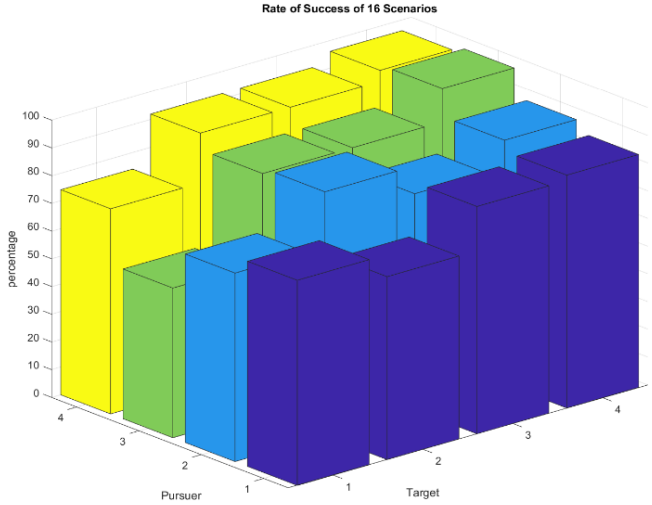

Fig. 10. Comparison of rate of success from sixteen evaluation scenarios

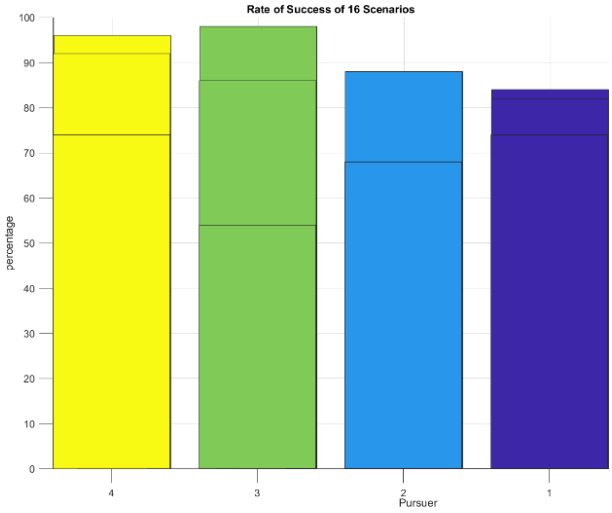

Fig. 11. Comparison of rate of success in percentage-pursuer view

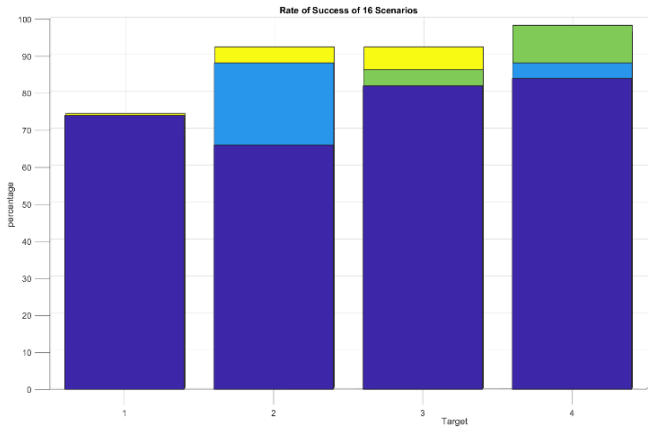

Fig. 12. Comparison of rate of success in percentage-target view

C. Cooperative Behaviour Analysis

Fig. 13 illustrates the stochastic policies' unique traits, cooperative-like behaviors.

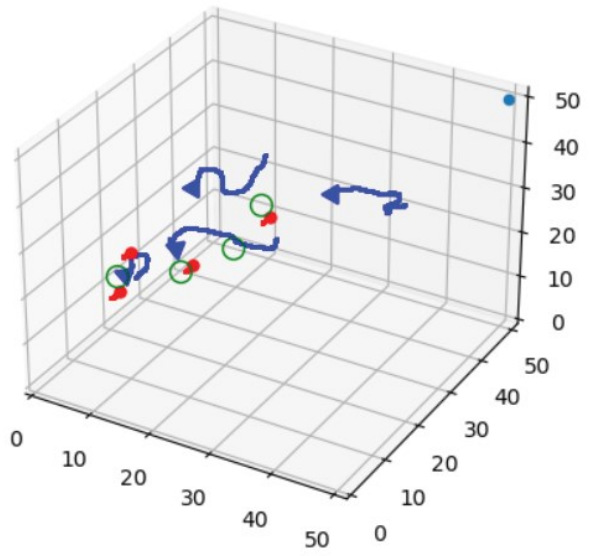

Fig. 13. Task allocation demonstrated by the pursuers

The blue arrows and blue lines are the pursuers and their respective trajectory. The red dots are the targets, and the green circle is the belief of the targets. Stochastic policies acquire insights beyond merely tracking the nearest target. They not only identify distant targets but also autonomously determine target assignments, indicating a degree of cooperation. The target's belief will always change at any time, especially when the target is not in the pursuers' field of view.

This behavior is very important in problems like MTTG where the number of targets is unknown. Because they want to maximize the rewards they get, pursuers will try to track all existing targets. Without cooperative behavior, each pursuer will focus more on targets that are visible even though they have been tracked by other pursuers. This can cause a decrease in the value of the reward.

D. Comparison of the Proposed Algorithm with Double Q-Learning

Subsection II.F explains soft double Q-Learning (soft-DQL) and its differences from basic double Q-Learning (DQL), which is the use of temperature parameter, α. The influence of alpha on the performance of the algorithm is to encourage exploration which is one of the important points in the MTTG problem. To see how big the effect is, we trained three other models that have the same parameter specifications as the previous model, except for the bearing angle and elevation angle which are π rad. In other words, now pursuers should have better observation skills. The three models are differentiated based on the temperature parameters used. The α values used were 0, 0.2, and 0.4 for the first, second, and third models, respectively. The first model is the model that uses basic double Q-Learning.

Then, we evaluate them in several scenarios and once again, calculate the rate of success for each of them through 100 evaluation episodes. The intended scenario is to make changes to the bearing angle and elevation angle values, thus decreasing the sensor Field of View (FOV). From there, the adaptability of each model is tested. The result is shown in Table II.

TABLE II. RATE OF SUCCESS OF BASIC DQL AND SOFT-DQL IN VARIOUS SENSOR FOV

	Model 1 DQL ($\alpha = 0$)	**Model 2 soft-DQL ($\alpha = 0.2$)**	**Model 3 soft-DQL ($\alpha = 0.4$)**
$\frac{\pi}{4}$	2%	6%	14%
$\frac{\pi}{2}$	86%	88%	92%
π	100%	100%	100%

In the first scenario where the observation angle decreases drastically to $\frac{\pi}{4}$, all models produce poor results. However, it can be seen that model 1 has the least success. On the other hand, model 3 is the highest and leaves far behind the other two models. This continues to the next scenario even though the difference is not as far as the first scenario. Meanwhile, all models succeed in the third scenario because they were trained in that scenario. α gives the pursuers the ability to be more flexible in dealing with various conditions. Model 3 has better exploration capabilities than model 1. Therefore, when the observations are limited, pursuers from model 3 can get more success because they move more actively in their environment.

Apart from the influence of α, it can also be seen how the pursuer performance is affected by changes in sensor specifications. The smaller sensor viewing distance certainly limits the pursuer's ability to track their target, especially if they were previously trained in a very different scenario. Different from the model in the previous experiment, that model has been trained on a scenario with an observation angle of $\frac{\pi}{4}$ so it can still provide better performance. Using piecewise cubic hermite interpolating polynomial (PCHIP) function in Matlab, the influence of the sensor FOV and α value on the rate of success is more clearly visible as seen in Fig. 14.

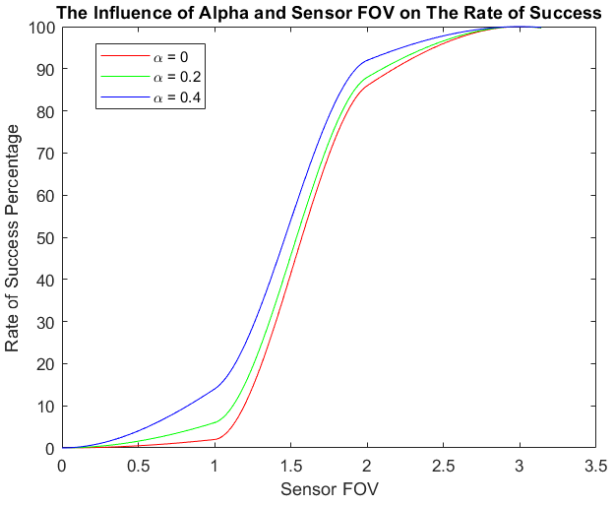

Fig. 14. Polynomial that describes the influence of FOV and α on the rate of success of the system

E. Comparison of Three-dimensional and Two-dimensional Environments

When compared to a two-dimensional environment, a three-dimensional environment has higher complexity. We summarize some of the differences between the two in Table III.

TABLE III. COMPARISON OF 3D AND 2D ENVIRONMENT

No.	Indicator	Three-Dimensional	Two-Dimensional
1.	Variables	22	13
2.	Rate of Success	76%	100%
3.	Environment Size	125000 units	2500 units
4.	Steps Needed to Converge	50000	50000
5.	Computation Time	28651 time unit	26709 time unit

We carry out 20 episodes of training by equating the values of all hyperparameters in a two-dimensional environment to those in a three-dimensional environment.

First, three-dimensional scenarios certainly have a greater number of variables, approximately two times the two-dimensional. For example, there is only one rotation angle that is considered in a two-dimensional environment, namely the yaw angle. Meanwhile, we must consider three angles in a three-dimensional environment, namely roll, pitch, and yaw. This carries over to the coordinate transformation where we have the addition of elevation. The rate of success is calculated from the highest and lowest percentage among sixteen scenarios average.

The rate of success is calculated from the averages of the highest and lowest percentages among sixteen scenarios. From Fig. 10, the 4 pursuers and 3 targets scenario is the highest with 98%. The lowest comes from 1 pursuer and 3 targets, which is 54%. Meanwhile, for the two environmental dimensions, both were successful.

The size of the three-dimensional environment is fifty times bigger than the two-dimensional one as it has the z-axis. This occurs because a three-dimensional environment is calculated in cubes instead of squares, as in a two-dimensional environment. This much larger size means that the number of available states also increases so that more space needs to be explored.

The time to reach convergence between the two scenarios is the same. However, two-dimensional scenarios generally have a larger average reward. Lastly, although the values of all hyperparameters are the same, the computing time to complete 20 episodes is different. Three-dimensional environments take longer. However, if this concept is to be brought to a real-world setting, there are many other factors that need to be considered. One of them is the increase in computational complexity connected to three-dimensional scenarios.

IV. CONCLUSION

This paper presented a method for multi-agent target tracking within a complex three-dimensional environment with high uncertainty. First, the problem is expressed as the Dec-POMDP model. We design and derive several necessary equations to describe the relationship between observations and the state of pursuers in the three-dimensional environment such as the Jacobian matrix of the measurement model and velocity in Spherical coordinates. Then, we use soft double Q-Learning which is a development of basic double Q-Learning to carry out coordination control for each pursuer. The experiment results proved the fact that the proposed method is capable of completing the task in various scenarios and is better than the baseline.

A significant challenge lies in striking a balance between decentralized control decisions, which is necessitated by communication constraints, and achieving cooperation or information sharing to effectively track multiple targets. This study delves into a heuristic to address this tradeoff. It trains a stochastic policy that enables pursuers to hedge their control actions when multiple targets are nearby. Although this form of cooperation is relatively weak, it needs no communication. However, this method may not be suitable for more demanding scenarios that inherently require more agents to track cooperatively, such as targets with rapidly increasing uncertainty or emergencies like accidents. Therefore, adding communication capabilities between pursuers can be a good development. Additionally, analysis of computational complexity in a three-dimensional environment also needs to be carried out to prepare this method for real-world scenarios, such as surveillance systems and public safety.

ACKNOWLEDGMENT

This work was funded by PT Infoglobal Teknologi Semesta, Indonesia.

REFERENCES

[1] K. Guo, X. Li, and L. Xie, "Simultaneous cooperative relative localization and distributed formation control for multiple UAVs," *Sci. China Inf. Sci.*, vol. 63, no. 1, pp. 2019–2021, 2020, doi: 10.1007/s11432-018-9603-y.

[2] Y. Cao and Y. Sun, "Necessary and sufficient conditions for consensus of third-order discrete-time multi-agent systems in directed networks," *J. Appl. Math. Comput.*, vol. 57, no. 1–2, pp. 199–210, 2018, doi: 10.1007/s12190-017-1101-8.

[3] W. Zhou, J. Li, Z. Liu, and L. Shen, "Improving multi-target cooperative tracking guidance for UAV swarms using multi-agent reinforcement learning," *Chinese J. Aeronaut.*, vol. 35, no. 7, pp. 100–112, 2022, doi: 10.1016/j.cja.2021.09.008.

[4] Z. Ma and J. Chen, "Multi-UAV Urban Logistics Task Allocation Method Based on MCTS," *Drones*, vol. 7, no. 11, 2023, doi: 10.3390/drones7110679.

[5] M. Zhang and C. Pan, "Hierarchical Optimization Scheduling Algorithm for Logistics Transport Vehicles Based on Multi-Agent Reinforcement Learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 3, pp. 3108–3117, 2023, doi: 10.1109/TITS.2023.3337334.

[6] H. R. Lee and T. Lee, "Multi-agent reinforcement learning algorithm to solve a partially-observable multi-agent problem in disaster response," *Eur. J. Oper. Res.*, vol. 291, no. 1, pp. 296–308, 2021, doi: 10.1016/j.ejor.2020.09.018.

[7] M. A. Blais and M. A. Akhloufi, "Drone Swarm Coordination Using Reinforcement Learning for Efficient Wildfires Fighting," *SN Comput. Sci.*, vol. 5, no. 3, 2024, doi: 10.1007/s42979-024-02650-6.

[8] T. A. Karaguzel, V. Retamal, N. Cambier, and E. Ferrante, "From Shadows to Light: A Swarm Robotics Approach with Onboard Control for Seeking Dynamic Sources in Constrained Environments," *IEEE Robot. Autom. Lett.*, vol. 9, no. 1, pp. 127–134, 2024, doi: 10.1109/LRA.2023.3331897.

[9] R. K. Lee, C. A. Kitts, M. A. Neumann, and R. T. McDonald, "Multiple UAV Adaptive Navigation for Three-Dimensional Scalar Fields," *IEEE Access*, vol. 9, pp. 122626–122654, 2021, doi: 10.1109/ACCESS.2021.3107854.

[10] M. Lee *et al.*, "A Study on the Advancement of Intelligent Military Drones: Focusing on Reconnaissance Operations," *IEEE Access*, vol. 12, pp. 55964–55975, 2024, doi: 10.1109/ACCESS.2024.3390035.

[11] Y. Liu, J. Liu, Z. He, Z. Li, Q. Zhang, and Z. Ding, "A Survey of Multi-Agent Systems on Distributed Formation Control," *Unmanned Syst.*, pp. 1–14, Mar. 2023, doi: 10.1142/S2301385024500274.

[12] Y. Liu, J. Hu, and Y. Li, "Quantized Formation Control of Heterogeneous Nonlinear Multi-Agent Systems with Switching Topology," *J. Syst. Sci. Complex.*, vol. 36, no. 6, pp. 2382–2397, 2023, doi: 10.1007/s11424-023-2387-2.

[13] J. Liao *et al.*, "UAV swarm formation reconfiguration control based on variable-stepsize MPC-APCMPIO algorithm," *Sci. China Inf. Sci.*, vol. 66, no. 11, p. 212207, 2023, doi: 10.1007/s11432-022-3735-5.

[14] Z. Pang, Y. Fu, H. Guo, and J. Sun, "Analysis of Stealthy False Data Injection Attacks Against Networked Control Systems: Three Case Studies," *J. Syst. Sci. Complex.*, vol. 36, no. 4, pp. 1407–1422, 2023, doi: 10.1007/s11424-022-2120-6.

[15] Z. Zhao, J. Chen, B. Xin, L. Li, K. Jiao, and Y. Zheng, "Learning Scalable Task Assignment with Imperative-Priori Conflict Resolution in Multi-UAV Adversarial Swarm Defense Problem," *J. Syst. Sci. Complex.*, vol. 37, no. 1, pp. 369–388, 2024, doi: 10.1007/s11424-024-4029-8.

[16] N. Li, Z. Su, H. Ling, M. Karatas, and Y. Zheng, "Optimization of Air Defense System Deployment Against Reconnaissance Drone Swarms," *Complex Syst. Model. Simul.*, vol. 3, no. 2, pp. 102–117, 2023, doi: 10.23919/CSMS.2023.0003.

[17] G. Wu, T. Xu, Y. Sun, and J. Zhang, "Review of multiple unmanned surface vessels collaborative search and hunting based on swarm intelligence," *Int. J. Adv. Robot. Syst.*, vol. 19, no. 2, pp. 1–20, 2022, doi: 10.1177/17298806221091885.

[18] S. Ishtiaq, X. Wang, S. Hassan, A. Mohammad, A. A. Alahmadi, and N. Ullah, "Three-Dimensional Multi-Target Tracking Using Dual-Orthogonal Baseline Interferometric Radar," *Sensors*, vol. 22, no. 19. 2022, doi: 10.3390/s22197549.

[19] X. Qu, W. Gan, D. Song, and L. Zhou, "Pursuit-evasion game strategy of USV based on deep reinforcement learning in complex multi-obstacle environment," *Ocean Eng.*, vol. 273, p. 114016, 2023, doi: 10.1016/j.oceaneng.2023.114016.

[20] A. A. Firmansyah and A. Alkaff, "Enhancing Autonomous Ground Vehicle Positioning and Navigation through Cascaded Multi-Sensor Integration," *2023 Int. Conf. Adv. Mechatronics, Intell. Manuf. Ind. Autom. ICAMIMIA 2023 - Proc.*, pp. 436–441, 2023, doi: 10.1109/ICAMIMIA60881.2023.10427562.

[21] T. Zhang, L. Yu, Z.-L. Zhou, and L. Wang, "Decision-making for air combat maneuvering based on hybrid algorithm," *Xi Tong Gong Cheng Yu Dian Zi Ji Shu/Systems Eng. Electron.*, vol. 35, pp. 1445–1450, Jul. 2013, doi: 10.3969/j.issn.1001-506X.2013.07.15.

[22] J. M. Gongora-Torres, C. Vargas-Rosales, A. Aragón-Zavala, and R. Villalpando-Hernandez, "Elevation Angle Characterization for LEO Satellites: First and Second Order Statistics," *Applied Sciences*, vol. 13, no. 7. 2023, doi: 10.3390/app13074405.

[23] R. G. Raju and S. K. Kashyap, "3D Localisation of Target using Elevation Angle Algorithm with the use of Ground Radars," *Def. Sci. J.*, vol. 70, pp. 260–271, 2020.

[24] K. E. Hoque and H. Aljamaan, "Impact of hyperparameter tuning on machine learning models in stock price forecasting," *IEEE Access*, vol. 9, pp. 163815–163830, 2021, doi: 10.1109/ACCESS.2021.3134138.

[25] N. Sutarna, C. Tjahyadi, P. Oktivasari, M. Dwiyaniti, and T. Tohazen, "Hyperparameter Tuning Impact on Deep Learning Bi-LSTM for Photovoltaic Power Forecasting," *J. Robot. Control (JRC)*, vol. 5, no. 3, 2024, doi: 10.18196/jrc.v5i3.21120.

[26] D. C. E. Saputra, A. Ma'arif, and K. Sunat, "Optimizing Predictive Performance: Hyperparameter Tuning in Stacked Multi-Kernel Support Vector Machine Random Forest Models for Diabetes Identification," *J. Robot. Control*, vol. 4, no. 6, pp. 896–904, 2023, doi: 10.18196/jrc.v4i6.20898.

[27] M. N. A. Al-Hamadani, M. A. Fadhel, L. Alzubaidi, and H. Balazs, "Reinforcement Learning Algorithms and Applications in Healthcare and Robotics: A Comprehensive and Systematic Review," *Sensors*, vol. 24, no. 8, p. 2461, 2024, doi: 10.3390/s24082461.

[28] S. Mukhopadhyay and B. Jain, "Multi-agent markov decision processes with limited agent communication," *IEEE Int. Symp. Intell. Control - Proc.*, pp. 7–12, 2001, doi: 10.1109/isic.2001.971476.

[29] S. Huang, H. Zhang, and Z. Huang, "Multi-UAV Collision Avoidance using Multi-Agent Reinforcement Learning with Counterfactual Credit Assignment," *arXiv preprint arXiv:2204.08594*, 2022.

[30] M. Kayaalp, F. Ghadieh, and A. H. Sayed, "Policy Evaluation in Decentralized POMDPs With Belief Sharing," *IEEE Open J. Control Syst.*, vol. 2, pp. 125–145, 2023, doi: 10.1109/ojcsys.2023.3277760.

[31] A. K. Shakya, G. Pillai, and S. Chakrabarty, "Reinforcement learning algorithms: A brief survey," *Expert Syst. Appl.*, vol. 231, p. 120495, 2023, doi: 10.1016/j.eswa.2023.120495.

[32] M. Yuan, J. Shan, and K. Mi, "Deep Reinforcement Learning Based Game-Theoretic Decision-Making for Autonomous Vehicles," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 818–825, 2022, doi: 10.1109/LRA.2021.3134249.

[33] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *J. F. Robot.*, vol. 37, no. 3, pp. 362–386, 2020, doi: 10.1002/rob.21918.

[34] Ó. Pérez-Gil *et al.*, "Deep reinforcement learning based control for Autonomous Vehicles in CARLA," *Multimed. Tools Appl.*, vol. 81, no. 3, pp. 3553–3576, 2022, doi: 10.1007/s11042-021-11437-3.

[35] S. Han *et al.*, "A Multi-Agent Reinforcement Learning Approach for Safe and Efficient Behavior Planning of Connected Autonomous Vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 5, pp. 3654–3670, 2023, doi: 10.1109/TITS.2023.3336670.

[36] B. Memarian and T. Doleck, "A scoping review of reinforcement learning in education," *Comput. Educ. Open*, vol. 6, p. 100175, 2024, doi: 10.1016/j.caeo.2024.100175.

[37] B. Fahad Mon, A. Wasfi, M. Hayajneh, A. Slim, and N. Abu Ali, "Reinforcement Learning in Education: A Literature Review," *Informatics*, vol. 10, no. 3, pp. 1–22, 2023, doi: 10.3390/informatics10030074.

[38] A. Charpentier, R. Élie, and C. Remlinger, "Reinforcement Learning in Economics and Finance," *Comput. Econ.*, vol. 62, no. 1, pp. 425–462, 2023, doi: 10.1007/s10614-021-10119-4.

[39] F. Soleymani and E. Paquet, "Deep graph convolutional reinforcement learning for financial portfolio management – DeepPocket," *Expert Syst. Appl.*, vol. 182, 2021, doi: 10.1016/j.eswa.2021.115127.

[40] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Performance optimization for blockchain-enabled industrial internet of things (iiot) systems: A deep reinforcement learning approach," *IEEE Trans. Ind. Informatics*, vol. 15, no. 6, pp. 3559–3570, 2019, doi: 10.1109/TII.2019.2897805.

[41] X. He, K. Wang, H. Huang, T. Miyazaki, Y. Wang, and S. Guo, "Green Resource Allocation Based on Deep Reinforcement Learning in Content-Centric IoT," *IEEE Trans. Emerg. Top. Comput.*, vol. 8, no. 3, pp. 781–796, 2020, doi: 10.1109/TETC.2018.2805718.

[42] Y. Dai, D. Xu, S. Maharjan, Z. Chen, Q. He, and Y. Zhang, "Blockchain and Deep Reinforcement Learning Empowered Intelligent 5G beyond," *IEEE Netw.*, vol. 33, no. 3, pp. 10–17, 2019, doi: 10.1109/MNET.2019.1800376.

[43] C. H. Liu, Q. Lin, and S. Wen, "Blockchain-enabled data collection and sharing for industrial iot with deep reinforcement learning," *IEEE Trans. Ind. Informatics*, vol. 15, no. 6, pp. 3516–3526, 2019, doi: 10.1109/TII.2018.2890203.

[44] R. Nian, J. Liu, and B. Huang, "A review On reinforcement learning: Introduction and applications in industrial process control," *Comput. Chem. Eng.*, vol. 139, p. 106886, 2020, doi: 10.1016/j.compchemeng.2020.106886.

[45] X. Fang, J. Wang, G. Song, Y. Han, Q. Zhao, and Z. Cao, "Multi-Agent Reinforcement Learning Approach for Residential Microgrid Energy Scheduling," *Energies*, vol. 13, no. 1. 2020, doi: 10.3390/en13010123.

[46] A. Subramanian, S. Chitlangia, and V. Baths, "Reinforcement learning and its connections with neuroscience and psychology," *Neural Networks*, vol. 145, pp. 271–287, 2022, doi: 10.1016/j.neunet.2021.10.003.

[47] F. Ye, Y. Li, X. Wang, N. Nedjah, P. Zhang, and H. Shi, "Parameters tuning of multi-model database based on deep reinforcement learning," *J. Intell. Inf. Syst.*, vol. 61, no. 1, pp. 167–190, 2023, doi: 10.1007/s10844-022-00762-0.

[48] J. García and D. Shafie, "Teaching a humanoid robot to walk faster through Safe Reinforcement Learning," *Eng. Appl. Artif. Intell.*, vol. 88, p. 103360, 2020, doi: 10.1016/j.engappai.2019.103360.

[49] T. Kobayashi and T. Sugino, "Reinforcement learning for quadrupedal locomotion with design of continual–hierarchical curriculum," *Eng.*

Appl. Artif. Intell., vol. 95, no. August, p. 103869, 2020, doi: 10.1016/j.engappai.2020.103869.

[50] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: lessons we have learned," *Int. J. Rob. Res.*, vol. 40, no. 4–5, pp. 698–721, Jan. 2021, doi: 10.1177/0278364920987859.

[51] M. D. Al-Masrur Khan *et al.*, "A systematic review on reinforcement learning-based robotics within the last decade," *IEEE Access*, vol. 8, pp. 176598–176623, 2020, doi: 10.1109/ACCESS.2020.3027152.

[52] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Sci. Technol.*, vol. 26, no. 5, pp. 674–691, 2021, doi: 10.26599/TST.2021.9010012.

[53] Y. Zhu, Z. Wang, C. Chen, and D. Dong, "Rule-Based Reinforcement Learning for Efficient Robot Navigation With Space Reduction," *IEEE/ASME Trans. Mechatronics*, vol. 27, no. 2, pp. 846–857, 2022, doi: 10.1109/TMECH.2021.3072675.

[54] X. Wang *et al.*, "Deep Reinforcement Learning: A Survey," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 35, no. 4, pp. 5064–5078, 2024, doi: 10.1109/TNNLS.2022.3207346.

[55] L. Canese *et al.*, "Multi-Agent Reinforcement Learning: A Review of Challenges and Applications," *Applied Sciences*, vol. 11, no. 11. 2021, doi: 10.3390/app11114948.

[56] J. Castellini, F. A. Oliehoek, R. Savani, and S. Whiteson, "Analysing factorizations of action-value networks for cooperative multi-agent reinforcement learning," *Auton. Agent. Multi. Agent. Syst.*, vol. 35, no. 2, p. 25, 2021, doi: 10.1007/s10458-021-09506-w.

[57] B. Liu, Y. Xie, L. Feng, and P. Fu, "Correcting biased value estimation in mixing value-based multi-agent reinforcement learning by multiple choice learning," *Eng. Appl. Artif. Intell.*, vol. 116, p. 105329, 2022, doi: 10.1016/j.engappai.2022.105329.

[58] Y. Deng, Z. Wang, X. Chen, and Y. Zhang, "Boosting Multi-agent Reinforcement Learning via Contextual Prompting," *J. Mach. Learn. Res.*, vol. 24, no. 399, pp. 1–34, 2023.

[59] J.-Y. Lee, A. Rahman, S. Huang, A. D. Smith, and S. Katipamula, "On-policy learning-based deep reinforcement learning assessment for building control efficiency and stability," *Sci. Technol. Built Environ.*, vol. 28, no. 9, pp. 1150–1165, Sep. 2022, doi: 10.1080/23744731.2022.2094729.

[60] N. Wang, Z. Li, X. Liang, Y. Hou, and A. Yang, "A Review of Deep Reinforcement Learning Methods and Military Application Research," *Math. Probl. Eng.*, vol. 2023, p. 7678382, 2023, doi: 10.1155/2023/7678382.

[61] X. Zhuang, D. Li, Y. Wang, X. Liu, and H. Li, "Optimization of high-speed fixed-wing UAV penetration strategy based on deep reinforcement learning," *Aerosp. Sci. Technol.*, vol. 148, p. 109089, 2024, doi: 10.1016/j.ast.2024.109089.

[62] J. Fan, D. Dou, and Y. Ji, "Impact-Angle Constraint Guidance and Control Strategies Based on Deep Reinforcement Learning," *Aerospace*, vol. 10, no. 11, 2023, doi: 10.3390/aerospace10110954.

[63] A. Garg and S. S. Jha, "Deep deterministic policy gradient based multi-UAV control for moving convoy tracking," *Eng. Appl. Artif. Intell.*, vol. 126, p. 107099, 2023, doi: 10.1016/j.engappai.2023.107099.

[64] J. Kim, D. Jang, and H. J. Kim, "Distributed Multi-agent Target Search and Tracking With Gaussian Process and Reinforcement Learning," *Int. J. Control. Autom. Syst.*, vol. 21, no. 9, pp. 3057–3067, 2023, doi: 10.1007/s12555-022-0555-0.

[65] A. Puente-Castro, D. Rivero, E. Pedrosa, A. Pereira, N. Lau, and E. Fernandez-Blanco, "Q-Learning based system for Path Planning with Unmanned Aerial Vehicles swarms in obstacle environments[Formula presented]," *Expert Syst. Appl.*, vol. 235, p. 121240, 2024, doi: 10.1016/j.eswa.2023.121240.

[66] Z. Qu, X. Zhao, H. Xu, H. Tang, J. Wang, and B. Li, "An Improved Q-Learning-Based Sensor-Scheduling Algorithm for Multi-Target Tracking," *Sensors*, vol. 22, no. 18, 2022, doi: 10.3390/s22186972.

[67] Q. Long, Z. Zhou, A. Gupta, F. Fang, Y. Wu, and X. Wang, "Evolutionary Population Curriculum for Scaling Multi-Agent Reinforcement Learning," *arXiv preprint arXiv:2003.10423*, 2020.

[68] X. Wang and X. Fang, "A multi-agent reinforcement learning algorithm with the action preference selection strategy for massive target cooperative search mission planning," *Expert Syst. Appl.*, vol. 231, p. 120643, 2023, doi: 10.1016/j.eswa.2023.120643.

[69] X. Li, J. Ren, and Y. Li, "Multi-mode filter target tracking method for mobile robot using multi-agent reinforcement learning," *Eng. Appl. Artif. Intell.*, vol. 127, p. 107398, 2024, doi: 10.1016/j.engappai.2023.107398.

[70] S. Li, Y. Jia, F. Yang, Q. Qin, H. Gao, and Y. Zhou, "Collaborative Decision-Making Method for Multi-UAV Based on Multiagent Reinforcement Learning," *IEEE Access*, vol. 10, pp. 91385–91396, 2022, doi: 10.1109/ACCESS.2022.3199070.

[71] R. Su, Z. Gong, D. Zhang, C. Li, Y. Chen, and R. Venkatesan, "An Adaptive Asynchronous Wake-Up Scheme for Underwater Acoustic Sensor Networks Using Deep Reinforcement Learning," *IEEE Trans. Veh. Technol.*, vol. 70, no. 2, pp. 1851–1865, 2021, doi: 10.1109/TVT.2021.3055065.

[72] L. Yue, R. Yang, J. Zuo, M. Yan, X. Zhao, and M. Lv, "Factored Multi-Agent Soft Actor-Critic for Cooperative Multi-Target Tracking of UAV Swarms," *Drones*, vol. 7, no. 3, p. 150, 2023, doi: 10.3390/drones7030150.

[73] K. Su and F. Qian, "Multi-UAV Cooperative Searching and Tracking for Moving Targets Based on Multi-Agent Reinforcement Learning," *Appl. Sci.*, vol. 13, no. 21, p. 11905, 2023, doi: 10.3390/app132111905.

[74] C. D. Hsu, H. Jeong, G. J. Pappas, and P. Chaudhari, "Scalable Reinforcement Learning Policies for Multi-Agent Control," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 4785–4791, 2021, doi: 10.1109/IROS51168.2021.9636344.

[75] X. Zhang, W. Yue, and W. Tang, "Research on Scheme Design and Decision of Multiple Unmanned Aerial Vehicle Cooperation Anti-Submarine Based on Knowledge-Driven Soft Actor-Critic," *Appl. Sci.*, vol. 13, no. 20, p. 11527, 2023, doi: 10.3390/app132011527.

[76] Z. Fan, Y. Xu, Y. Kang, and D. Luo, "Air Combat Maneuver Decision Method Based on A3C Deep Reinforcement Learning," *Machines*, vol. 10, no. 11, pp. 1–18, 2022, doi: 10.3390/machines10111033.

[77] Y. Cao, Y.-X. Kou, Z.-W. Li, and A. Xu, "Autonomous Maneuver Decision of UCAV Air Combat Based on Double Deep Q Network Algorithm and Stochastic Game Theory," *Int. J. Aerosp. Eng.*, vol. 2023, p. 3657814, 2023, doi: 10.1155/2023/3657814.

[78] J. H. Bae, H. Jung, S. Kim, S. Kim, and Y.-D. Kim, "Deep Reinforcement Learning-Based Air-to-Air Combat Maneuver Generation in a Realistic Environment," *IEEE Access*, vol. 11, pp. 26427–26440, 2023, doi: 10.1109/ACCESS.2023.3257849.

[79] J. Xianyong, M. Hou, G. Wu, Z. Ma, and Z. Tao, "Research on Maneuvering Decision Algorithm Based on Improved Deep Deterministic Policy Gradient," *IEEE Access*, vol. 10, pp. 92426–92445, 2022, doi: 10.1109/ACCESS.2022.3202918.

[80] D. Hu, R. Yang, J. Zuo, Z. Zhang, J. Wu, and Y. Wang, "Application of Deep Reinforcement Learning in Maneuver Planning of Beyond-Visual-Range Air Combat," *IEEE Access*, vol. 9, pp. 32282–32297, 2021, doi: 10.1109/ACCESS.2021.3060426.

[81] S. Li *et al.*, "Multi-UAV Cooperative Air Combat Decision-Making Based on Multi-Agent Double-Soft Actor-Critic," *Aerospace*, vol. 10, no. 7. 2023, doi: 10.3390/aerospace10070574.

[82] W. Kong, D. Zhou, Z. Yang, Y. Zhao, and K. Zhang, "UAV Autonomous Aerial Combat Maneuver Strategy Generation with Observation Error Based on State-Adversarial Deep Deterministic Policy Gradient and Inverse Reinforcement Learning," *Electronics*, vol. 9, no. 7. 2020, doi: 10.3390/electronics9071121.

[83] A. White and A. Karimoddini, "Event-based diagnosis of flight maneuvers of a fixed-wing aircraft," *Reliab. Eng. Syst. Saf.*, vol. 193, p. 106609, 2020, doi: 10.1016/j.ress.2019.106609.

[84] H. Jeong, B. Schlotfeldt, H. Hassani, M. Morari, D. D. Lee, and G. J. Pappas, "Learning Q-network for Active Information Acquisition," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 6822–6827, 2019, doi: 10.1109/IROS40897.2019.8968173.

[85] J. Hu, L. Wang, T. Hu, C. Guo, and Y. Wang, "Autonomous Maneuver Decision Making of Dual-UAV Cooperative Air Combat Based on Deep Reinforcement Learning," *Electronics*, vol. 11, no. 3. 2022, doi: 10.3390/electronics11030467.

[86] Y. Li, J. Shi, W. Jiang, W. Zhang, and Y. Lyu, "Autonomous maneuver decision-making for a UCAV in short-range aerial combat based on an MS-DDQN algorithm," *Def. Technol.*, vol. 18, no. 9, pp. 1697–1714, 2022, doi: 10.1016/j.dt.2021.09.014.

[87] N. Sadeghzadeh-Nokhodberiz, A. Can, R. Stolkin, and A. Montazeri, "Dynamics-Based Modified Fast Simultaneous Localization and

Mapping for Unmanned Aerial Vehicles with Joint Inertial Sensor Bias and Drift Estimation," *IEEE Access*, vol. 9, pp. 120247–120260, 2021, doi: 10.1109/ACCESS.2021.3106864.

[88] M. Zaheer, S. Kottur, S. Ravanbhakhsh, B. Póczos, R. Salakhutdinov, and A. J. Smola, "Deep sets," *Adv. Neural Inf. Process. Syst.*, pp. 3392–3402, 2017.

[89] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum Entropy Inverse Reinforcement Learning," *Proc. 23rd AAAI Conf. Artif. Intell. AAAI 2008*, pp. 1433–1438, 2008.

[90] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "Q-Learning Algorithms: A Comprehensive Classification and Applications," *IEEE Access*, vol. 7, pp. 133653–133667, 2019, doi: 10.1109/ACCESS.2019.2941229.

[91] H. Jiang, G. Li, J. Xie, and J. Yang, "Action Candidate Driven Clipped Double Q-Learning for Discrete and Continuous Action Tasks," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 35, no. 4, pp. 5269–5279, 2024, doi: 10.1109/TNNLS.2022.3203024.