Learning Deception Using Fuzzy Multi-Level Reinforcement Learning in a Multi-Defender One-Invader Differential Game

Authors:

Amirhossein Asgharnia[1]
ORCID: https://orcid.org/0000-0003-0240-7560
amirhosseinasgharnia@cmail.carleton.ca
Department of Systems and Computer Engineering, Carleton University
1125 Colonel By Drive, Ottawa, ON, K1S 5B6, Canada

Howard Schwartz
ORCID: https://orcid.org/0000-0002-4489-5892
schwartz@sce.carleton.ca
Department of Systems and Computer Engineering, Carleton University
1125 Colonel By Drive, Ottawa, ON, K1S 5B6, Canada

Mohamed Atia
ORCID: https://orcid.org/0000-0002-4272-7019
mohamed.atia@carleton.ca
Department of Systems and Computer Engineering, Carleton University
1125 Colonel By Drive, Ottawa, ON, K1S 5B6, Canada

[1]Corresponding Author

Abstract

Differential games are a class of game theory problems governed by differential equations. Differential games are often defined in the continuous domain and solved by the calculus of variations. However, modelling and solving these games are not straightforward tasks. Differential games, like game theory, are often involved with social dilemmas and social behaviours. Modelling these social phenomena with mathematical tools is often problematic. In this paper, we modelled deception to increase the pay-off in differential games. Deception is modelled as a bi-level policy system, and each level is modelled with a fuzzy controller. Fuzzy controllers are trained using a novel hierarchical fuzzy actor-critic learning algorithm. A deceitful player plays against multiple opponents. Although there is one ultimate goal for the player, it can choose multiple fake goals as well. The intention is to find a strategy to switch between the fake goals and the true goal to fool the opponents. The simulation platform is the game of guarding territories, a specific form of the pursuit-evasion games. We propose a method to easily increase the number of defenders with minimum changes in the policies. We create a universal structure that is not affected by the curse of dimensionality. We show that a discerning invader capable of using deception can improve its performance against the defenders by increasing the chance of invasion. We investigate the single-invader single-defender game and the single-invader multi-defender game. We study the superior invader and agents with the same speed. In all mentioned situations, the invader increases its pay-off by using deception versus being honest. A two-level policy system is used in this paper to model deception. The lower-level policy controls each goal's invasion actions, while the higher-level policy controls deception where a successful game is not initially possible.

Keywords: Multi-Agent Reinforcement Learning, Hierarchical Reinforcement Learning, Hierarchical Fuzzy Actor-Critic Learning, Pursuit-Evasion Game, Deception

1 Introduction

This paper proposes a method for learning deception in adversarial games. Reinforcement learning (RL) is implemented as the learning mechanism. Deception has multiple definitions provided by researchers from different fields. All definitions address an action or a sequence of actions to manipulate beliefs [1]. *Delibration* in the deceiver's action is the keyword to distinguish the difference between definitions. Bond and Robinson present deception as any false communication intended to benefit the communicator [2]. Their definition addresses intentional as well as unintentional forms of deception, such as mimicry and camouflage. However, the papers on deliberate deception outnumber the papers on unintentional deception. Skyrms recognizes deception as a systematically transmitted signal for the sender's benefit [3], whereas he denotes a signal without intention as *misinformation*. The same argument was previously noted in [4]. Whaley defines deception as intentional communication to manipulating some other agent [5]. In addition, Ettinger and Jahiel define deception as a process where actions are taken deliberately to manipulate beliefs and take advantage of the other agent [6]. Unintentional forms of deception such as camouflage are the intrinsic characteristics of an animal or, more generally, an agent. Animals are equipped with tools to show deception unintentionally, and they cannot learn these during their lives. Whereas, it is believed that intentional forms of deception, such as lying, can be learned and improved by practicing and receiving feedback [7]. In this paper, we adopt the definitions that highlight the deliberate action to demonstrate deceptive behaviour. In our study, the deceiver and the deceived agents use the same dynamical models. Thus, the deceiver does not have any intrinsic property to enhance its deceptive behaviours. The only advantage of the deceiver is its policy, which makes the agent capable of deceiving with delibration, and only can be learned from an external source.

Deception is studied in many fields, such as robotics [1, 8], cyber-physical systems [9], optimal control [10], and supervisory control [11].

Interdependance theory and game theory are investigated to explore the area of deception from a robotic perspective in [1]. The authors sought for an algorithm to determine the social situation, where deception is warranted. They showed that by having the situation's location in interdepandance space, the robot can decide whether to act deceptively or not. They also demonstrate that if the deceiver has the deceived agent's model, it can perform better. Finally, they showed that learnt communication can be used as false signals to deceive an other agent.

In [8], the researchers sought for deceptive strategies in robots, based on squirrel's cache protection. By observing the squirrel's behaviour is caching and protecting strategies, they provided a model to implement on robots. Their simulation consists of two robots, one gaining and protecting some goods, while the adversary tries to steal from the first one. It is shown that if the gaining robot implements a deceptive strategy, it can decrease its chance of being pilfered.

An application of deception in the pursuit-evasion game is studied in [12]. The researchers studied robotic reception in the context of fuzzy signalling games and inspiration from nature. In [12], agents send a false signal (like sending pheromones among ants) to deceive the adversary. It is observed that the adversary can find out the trick after a few time steps. However, during the time when the adversary is wrong, the agent can take advantage and increase its pay-off.

A practical problem with applications in autonomous vehicles is guarding a territory. Guarding a territory and its general form, the pursuit-evasion (PE) game, are among the well-studied differential game (DG). In the game of guarding a territory, three agents are defined within a playing field. There is a target, which can be assumed static [13–15] or dynamic [16, 17]. There is a defender, which actively protects the target from threats, and there is an invader, or sometimes referred to as an intruder, which is supposed to reach the target. Several methods are proposed to play this game based on geometric approaches [16, 18], solving Hamiltonian-Jacobi-Isaacs (HJI) equations [17, 19], game theoretical control methods [20], and machine learning [21–23].

In [19], a geometric approach is proposed to solve the pursuit-evasion differential game. The researcher solved a two-defender single-evader game (called the cutters and fugitive ship differential game) using the Apollonius circle. In this game, two pursuers cooperate to capture a single evader, where the evader's speed is less than the pursuers' speed. The validaty of the geometric method was later proven in [24]. However, the proposed approach suffers from the curse of dimensionality [17].

The Apollonius circle is a powerful tool in cases that the pursuer's/defender's speed is more than evader's/invader's. However, it is argued that the Apollonius circle is the most conservative solution for finding the agent's dominant region where the evader is faster than the pursuer [25]. In these cases, the capture is not guaranteed unless a positive capture radius is assumed for the pursuer. In games with an inferior pursuer with a positive capture radius, using Cartesian ovals is suggested [26].

A multi-pursuer single-evader game is studied in [27]. They implemented a min-max solution over the game duration and solved the game via linear programming. To reduce the game's complexity, they replace linear programming with a geometrical solution based on the Voronoi diagram.

A two-pursuer one-evader case is addressed in [16]. In this paper, the researchers obtained a state feedback cooperative strategy to navigate the pursuers. In addition, a cooperative geometric approach is addressed in [20] for a multi-pursuer single-evader case. In both papers, the evader's speed is less than the pursuers'.

A two-defender one-invader game is studied in [28] for the game of guarding territory or reach-avoid game. The invader strives to reach the field's boundary, while the defenders try to capture the invader. In the first stage, a reachability investigation is performed for the game, and in the second stage, the policies to accomplish the task are studied. The authors implemented the Apollonius circle and the Voronoi diagram to form strategies in their winning region. The approach is shown to be accurate and has low computational effort. Thus, it can be used as an online method for evaluating the policies. In addition, a reach-avoid game for two evaders and one-defender is studied in [29] from the game of kind and the game of degree perspective. The paper addresses cooperation between two evaders to reach the target region.

In [30], a multiple-pursuer multiple-evader case is studied. The authors discussed the number of pursuers needed for each evader, the shortest time to capture and allocate pursuers for each evader. The researchers studied pure pursuit and constant bearing cases and used the Apollonius circle and the Apollonius curve to check if a pursuer is active or redundant.

Pursuit-evasion games are classical game theoric problems, and the majority of the proposed solutions are based on geometric or analytic approaches. The significant issue in this game is the intractable growth of state numbers as the number of players increases. In addition, there is not yet a comprehensive solution for the M-pursuer and N-evader cases. A significant shortcoming in all the mentioned papers is the agents' limitations and constraints. For instance, most of the papers on pursuit-evasion games address agents with pedestrian differential equations. In a pedestrian model, the control signal is the agent's heading. It is assumed that the agent adjusts its heading to the desired value as soon as it receives the control signal. However, in reality, a robot limits the heading rate. Thus, using learning methods has attracted much attention in pursuit-evasion games.

In [31], the cooperation mechanism of multiple pursuers against multiple evaders is studied. The authors focused on how they can cluster similar evaders using reinforcement learning techniques. They proposed a new reward function to improve their results. Their proposed algorithm can outperform similar techniques in improving the flexibility and decreasing the capture time.

Active target defence using reinforcement learning is studied in [21]. They compared the actor being trained with supervised and unsupervised learning. They mentioned the importance and difficulty of obtaining a suitable reward function in the active target defence game. Thus, the machine learning methods may not increase the performance under all circumstances.

The Apollonius circle usage in building a reward function for reinforcement learning is addressed in [22]. The researchers studied training a superior invader and multiple defenders with the residual gradient fuzzy actor-critic learning (RGFACL) algorithm. They also employed the Apollonius circle to derive a reward function so that the defenders can make a formation without colliding. They showed that the RGFACL algorithm could train the agent to learn the optimal policy.

In [32] a new reinforcement learning algorithm is proposed in the continuous state-action domain. The algorithm is implemented to solve a pursuit-evasion with two agents that have a conflict of interests. Unlike many algorithms proposed in the literature, where only the output parameters are being updated, the proposed residual gradient fuzzy actor–critic learning (RGFACL) algorithm is able to update the input parameters as well, with a residual gradient value iteration approach.

Another application of reinforcement learning in differential games in presented in [33]. The study tackles the difficulty of using reinforcement learning approaches in multi-agent problems. The authors proposed a reward shaping method to increase the efficiency of joint policy training.

An extension to the game of guarding a territory is addressed in [14, 15], by adding more than one target to the game. In [14] deception is modelled for a grid-world game. The Q-tables were derived via the minimax Q-learning algorithm and the single-agent Q-learning algorithm. In addition, in [15], a deceitful invader confronts two defenders to reach a particular target among several targets. Although all targets are in the defenders' capture region, the invader tries to take advantage of the defenders' insufficient information. This game is created as a testbed for modelling deception. In [15], deception is modelled as a two-level hierarchical policy system. The lower-level policy contains the policy to invade each target, while the higher-level policy contains the proper switching instances between different goals. The lower-level policy is trained via the fuzzy actor-critic algorithm, whereas the higher-level policy is derived using the genetic algorithm. Although finding a fitness function for an optimization algorithm is a more straightforward task, the optimization algorithms such as the genetic algorithm limit the deception flexibility. For instance, the agents' initial positions must remain within a single or a few points in the field. In addition, the approach in [15] will be hit by the curse of dimensionality if the number of defenders increases.

The motivation of this paper is twofold. On the one hand, the paper investigates the possibility of learning deception using reinforcement learning, where the only difference between the deceiver and the deceived agent is the policy they use. We use a fuzzy inference system to accomplish this task. On the other hand, by using a deceptive strategy, we show an invader can improve its performance in the game of guarding a territory where the agents have constraints.

In this paper, we address a deceptive one-invader multiple-defender game in a continuous domain. We also investigate the performance of a superior invader case and the agents with equal speed. Similar to [15], we developed a bi-level hierarchical policy system. The lower-level policy is trained via the fuzzy actor-critic algorithm in the continuous domain. However, unlike [15], where the higher-level policy was trained with the genetic algorithm, in this paper, the higher-level policy is trained via the fuzzy actor-critic algorithm in the discrete domain. In other words, instead of evaluating the consequence at the terminal state with a cost function, we try to assess each action with a reward function. The contributions of this paper are as follows,

- Learning of deception with a hierarchical reinforcement learning,
- Providing a new approach to construct the reward function for the game of guarding a territory, which does not need any external policy,
- Proposing a new hierarchical fuzzy actor-critic learning algorithm to train both levels,
- Implementing deception in a single-invader multiple-defender case, where the invasion without deception is not possible,
- Comparing two speed scenarios, and studying difficult situations for using deception.

The paper addresses solving a particular class of differential games. Differential games are a form of game theory problems that are played in the continuous domain. In order to increase the invader's pay-off, we proposed using deception. Although the chosen game was guarding several territories, the nature of the game and the proposed method allow modelling deception in sports and economics [34]. There are different methods of using deception games. Many of them are only applicable to specific problems. Thus, comparing them is not a straightforward task. Some of the most recent publications on using deception in games are shown in Table 1. In comparison to the methods in Table 1, our proposed method does not need to know the detailed dynamics of the game since it uses reinforcement learning. In addition, the agent learns to deliberately use deception many times while playing the game by observation.

This paper is organized into six sections. In section 2, we present the preliminaries, such as the game of guarding several territories and the fuzzy actor-critic learning (FACL) algorithm. Section 3 is dedicated to the proposed method in modelling deception and training the policies. In section 4 we demonstrate the simulations and results. Section 5 is the conclusion.

Table 1 Recent publications on deception

Ref	Year	Application	Approach	Algorithms	Domain
[35]	2005	Decision making	Game theory	Not learning	Discrete
[1]	2011	Decision making	Game theory	Not learning	Discrete
[8]	2012	Decision making	Biomimetics	Not learning	Continuous
[36]	2015	Decision making	Pobability	Gradient decsend	Continuous
[37]	2019	Video games	RL	A2C	Discrete
[12]	2019	Pursuit-evasion	Game theory	Gradient decsend	Continuous
[38]	2020	Guarding a territory	RL	Deep RL	Continuous
[39]	2020	Video games	RL	Deep RL	Discrete
[14]	2020	Guarding territories	HRL	Q-learning	Discrete
[15]	2020	Guarding territories	RL	FACL+GA	Continuous
[11]	2021	Guarding territories	Optimal control	ADMM	Discrete
[40]	2021	Decision making	Game theory	Not learning	Continuous
Current work	2021	Guarding territories	HRL	FACL	Continuous

2 Preliminaries and Problem Definition

We take a brief look at the simulation platform of our learning strategy, which is the game of guarding several territories. The game of guarding several territories is similar to the classical game of guarding a territory [41] with more than one territory. The overall idea of our approach relies on the agents' ability to pretend to be chasing a goal, which is not the true goal. An agent tries to confuse its opponent by repeatedly changing its intended territory. The robot controller returns a control signal from a continuous interval to invade or protect one particular target. Thus, we use the fuzzy actor-critic learning (FACL) algorithm in the continuous-action domain to train the lower-level controller. However, the goal number is selected from a discrete set. Thus, we use the FACL algorithm for the discrete-action domain to train the goal selection mechanism.

2.1 The Game of Guarding several territories

This paper's simulation platform is similar to the game, which is defined in [15]. In this game, two kinds of agents are defined. The first agent is the invader, and the second agent is the defender. The invader(s) have to reach the target before being captured by the defender(s). Although the target may be moving, in this study, we assume the target is stationary. As mentioned in [15], there can be more than one agent of each kind playing the game. The invader and the defender are defined as bicycle robots with the following differential equation

[42]:

$$\begin{cases} \dot{x} = v.cos(\theta) \\ \dot{y} = v.sin(\theta) \\ \dot{\theta} = \frac{u \cdot v}{L} \end{cases}, \tag{1}$$

where, the tuple (x,y) is the agent's location in the field, and the term θ is the agent's heading with respect to x-axis. In addition, u is the robot's steering angle, L is the axle length, and v is the agent's speed. The robot's position in the game is fully defined in the field by knowing the location (x,y) and the orientation (θ). In addition, u is being used as the control input. We do not need to consider v as a control input and set v to the maximum [27].

In the deceptive version of the game, we add more targets into the game. The invader knows its true target, while the defender does not know the invader's true intention. The defender must have the policies regarding defending each goal individually. On the hand, the invader must know the policy to invade every desired goal.

Deception is a sociologic phenomenon. It is argued that deception can be learnt through trial and error and practice [7]. Although there are papers on analyzing deception via probability analysis, optimal control, or IF-THEN rules, the phenomenon is regarded as a black box in the majority of literature in Table 1. Thus, we utilized a reinforcement learning approach to learn deception as it is learnt naturally.

2.2 The Fuzzy Actor-Critic Learning (FACL) Algorithm

2.2.1 Continuous Action Space

Different learning algorithms can be implemented to find the agents' policies in the game of section 2.1. However, all of them must be in a continuous state domain. Thus, we need an approximator to map the states into an action. A comparison between the fuzzy Q-learning (FQL) and the FACL algorithms was conducted in [43] for a pursuit-evasion game. It is shown that the FACL algorithm can achieve a more significant pay-off in comparison to the FQL algorithm. Furthermore, it is shown that a fuzzy critic outperforms a neural network critic in [44]. In addition, a fuzzy network has greater explainability than neural networks. The FACL algorithm is a well-known learning method that can learn the optimal policy of the game [15]. The FACL algorithm uses two fuzzy inference systems (FISs); one fuzzy logic controller (FLC) as the actor and one FIS as the critic. In the algorithm, the actor stores the fuzzy controller's values, while the critic stores the value function. The continuous action space FACL is adopted from [45].

The output of the fuzzy controller is defined as follows:

$$u_t = \sum_{l=1}^{M} \Phi^l \omega_t^l, \tag{2}$$

where u_t is the control signal in time t, l is the rule number, M is the total number of rules, Φ^l is the firing strength of rule l, and ω_t^l is the output parameter of the actor's lth rule in time t. It should be mentioned that to increase exploration, Gaussian random noise is added to the actor's output ($u_t' = u_t + \sigma(0, \nu)$). The firing strength is defined as follows:

$$\Phi^l = \frac{\partial u}{\partial \omega^l} = \frac{\prod_{i=1}^{n} \mu^{F_j^l}(x_i)}{\sum_{l=1}^{M}(\prod_{i=1}^{n} \mu^{F_j^l}(x_i))}. \tag{3}$$

In (3), the term $\mu^{F_j^l}(x_i)$ is the membership degree of the fuzzy set jth membership function of the lth rule. The critic stores an evaluation of the value function in the form of a FIS as follows:

$$\hat{V}_t = \sum_{l=1}^{M} \Phi^l \zeta_t^l, \tag{4}$$

where, the term $\hat{V}_t$ is an approximation of the state-value function at time t. The term ζ_t^l is the output parameter of the lth critic's rule. In (4), the term Φ^l is similar to (3), and can be defined as follows:

$$\Phi^l = \frac{\partial \hat{V}}{\partial \zeta^l} = \frac{\prod_{i=1}^{n} \mu^{F_j^l}(x_i)}{\sum_{l=1}^{M}(\prod_{i=1}^{n} \mu^{F_j^l}(x_i))}. \tag{5}$$

The critic's task is to give feedback on the quality of the state of the agent. The state-value function is the expected value of the accumulated future reward and is shown as follows:

$$V_t = E\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\}. \tag{6}$$

In (6), the term V_t is the state-value function at time t, γ is the discount factor, and r_t is the received reward at time t. We define the temporal difference (TD) as follows:

$$\Delta = r_{t+1} + \gamma \hat{V}_{t+1} - \hat{V}_t. \tag{7}$$

The temporal difference is implemented to train both the critic and the actor. In each time step, the control signal is calculated using (2). To enhance exploration, random Gaussian

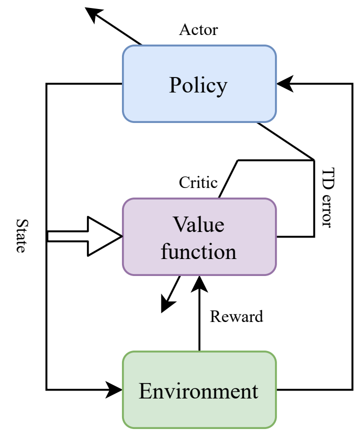


Fig. 1 The FACL algorithm structure [45].

noise is added with u_t. The agent takes action and receives the reward. The reward is used to calculate the temporal difference. Then, the output parameters of the actor and the critic are updated as follows:

$$\begin{aligned} \omega_{t+1}^{l} &= \omega_{t}^{l} + \beta_L \Delta (u_t' - u_t) \frac{\partial u}{\partial \omega^l} \\ \zeta_{t+1}^{l} &= \zeta_{t}^{l} + \alpha_L \Delta \frac{\partial \hat{V}}{\partial \omega^l}. \end{aligned} \tag{8}$$

In (8), ω_t^l and ζ_t^l are the lth rule's output parameters of the actor and the critic at time t, respectively. The term α_L and β_L are the critic's and the actor's learning rate. The term $\frac{\partial u}{\partial \omega^l}$ is given by (3). The term $\frac{\partial \hat{V}}{\partial \omega^l}$ is given by (5). The FACL algorithm structure is depicted in Fig. 1.

2.2.2 Discrete Action Space

In the FACL algorithm in continuous domain, the actions are real numbers in a defined interval. However, in the discrete action space algorithm, the actor's output parameters are selected from a designated set of actions. Thus, a value function is assigned to each state-action pair. In this section, we present a modified version of the algorithm proposed in [46].

In the discrete domain form, the actor's output is calculated as follows:

$$u_t = \sum_{l=1}^{M} \Psi_t^l c^l, \tag{9}$$

where, u_t is the actor's output, M is the total number of the rules, and c^l is the output parameter of lth rule. The term c^l is selected from an action set ($c^l \in A = \{a_1, a_2, ..., a_n\}$), with the softmax function. The term, Ψ^l is the firing strength of lth rule and is calculated as:

$$\Psi^l_t = \frac{\partial u_t}{\partial c^l} = \frac{\prod\limits_{i=1}^{n} \mu^{F^l_j}(x_i)}{\sum\limits_{l=1}^{M}(\prod\limits_{i=1}^{n} \mu^{F^l_j}(x_i))}. \tag{10}$$

In (9), c^l is chosen with an exploration-exploitation strategy. A possible exploration-exploitation approach is using the ε-greedy method, which is implemented for the fuzzy Q-learning algorithm in [45]. However, we use the softmax function to select the action in the training stage. The softmax function policy is defined such that the player chooses an action from the action set ($A = \{a_1, a_2, ..., a_n\}$). The softmax function selects a random action with probabilities proportional to their associated q-function.

$$Q_l = \frac{\exp(\tau q(l,a))}{\sum\limits_{a \in A} \exp(\tau q(l,a))}. \tag{11}$$

In (11), Q_l is a vector of normalized Q-values associated to all possible actions for lth rule. The term $q(l,a)$ is the associated Q-value given the rule l and the action $a \in A$. Finally, τ is the temperature constant. After selecting the output parameters and calculating the output using (9), the agent takes the action and receives a reward. The critic output is calculated as:

$$\hat{V}_t = \sum_{l=1}^{M} \Psi^l \zeta^l_t, \tag{12}$$

where, ζ^l_t is the lth critic's fuzzy output parameter at time step t. Temporal difference is calculated as:

$$\Delta = r_{t+1} + \gamma \hat{V}_{t+1} - \hat{V}_t. \tag{13}$$

Finally, the actor and critic are updated as follows:

$$\begin{aligned} q_{t+1}(l,a) &= q_t(l,a) + \beta_L \Delta \frac{\partial u_t}{\partial c^l} \\ \zeta^l_{t+1} &= \zeta^l_t + \alpha_L \Delta \frac{\partial \hat{V}}{\partial \zeta^l}. \end{aligned} \tag{14}$$

In addition, $\frac{\partial u_t}{\partial c^l}$ and $\frac{\partial \hat{V}}{\partial \zeta^l}$ are calculated by (3) and (5), respectively.

2.3 Dominant Regions

The dominant region of each agent is an area that the agent can reach before other agents. In games where the defender is faster than the invader, the defender can capture the invader. In these cases, the Apollonius circle (AC) can discriminate the agents' dominant regions.

A *circle* is defined as the loci of points with a constant distance (called the radius) from a fixed point, called the center. Apollonius gave another definition for a circle: A circle is a locus of points with a constant ratio of distances from two points, called foci. One of the foci is inside the circle, and one is outside the circle. The Apollonius definition gives a suitable tool to analyze the pursuit-evasion games [19]. The Apollonius circle determines the dominant region of each agent, which is the region that the agent can reach before its opponents. The Apollonius circle radius is defined as follows:

$$R = \frac{\lambda\sqrt{(x_d - x_i)^2 + (y_d - y_i)^2}}{1 - \lambda^2}, \tag{15}$$

where, (x_d, y_d) and (x_i, y_i) are the defender's and the invader's locations. The term, λ is the speed ratio ($\lambda = \frac{V_d}{V_i}$). The circle's center is as follows:

$$C = (\frac{x_d - \lambda^2 x_i}{1 - \lambda^2}, \frac{y_d - \lambda^2 y_i}{1 - \lambda^2}). \tag{16}$$

In games where the agents have the same speed, the dominant regions can be discriminated with a line, denoted as the *bisector* line. The bisector line is the bisector of the line that connects the invader to the defender.

In games where the invader is faster than the defender, the defender cannot capture the invader. Thus, a positive capture radius is assumed. A positive capture radius enables the defender to expand its dominant region in comparison to the results given by the Apollonius circle. In such a game, the dominant region can be discriminated by the Cartesian ovals (CO) [25, 26]. The Cartesian oval is determined as follows:

$$\begin{aligned} x_{co} &= x_i + R_{co}(\phi_{co})\cos(\eta + \phi_{co}) \\ y_{co} &= y_i + R_{co}(\phi_{co})\sin(\eta + \phi_{co}), \end{aligned} \tag{17}$$

where,

$$R_{co}(\phi_{co}) = \frac{\lambda\rho + d\cos(\phi_{co}) \pm \sqrt{\lambda\rho + d\cos(\phi_{co})2 - (1 - \lambda^2)(d^2 - \rho^2)}}{1 - \lambda^2}. \tag{18}$$

for $\phi_{co} \in [-\phi^i_{co}, \phi^i_{co}]$, where,

$$\phi^i_{co} = \arccos(\frac{\sqrt{(1 - \lambda^2)(d^2 - \rho^2)} - \lambda\rho}{d}). \tag{19}$$

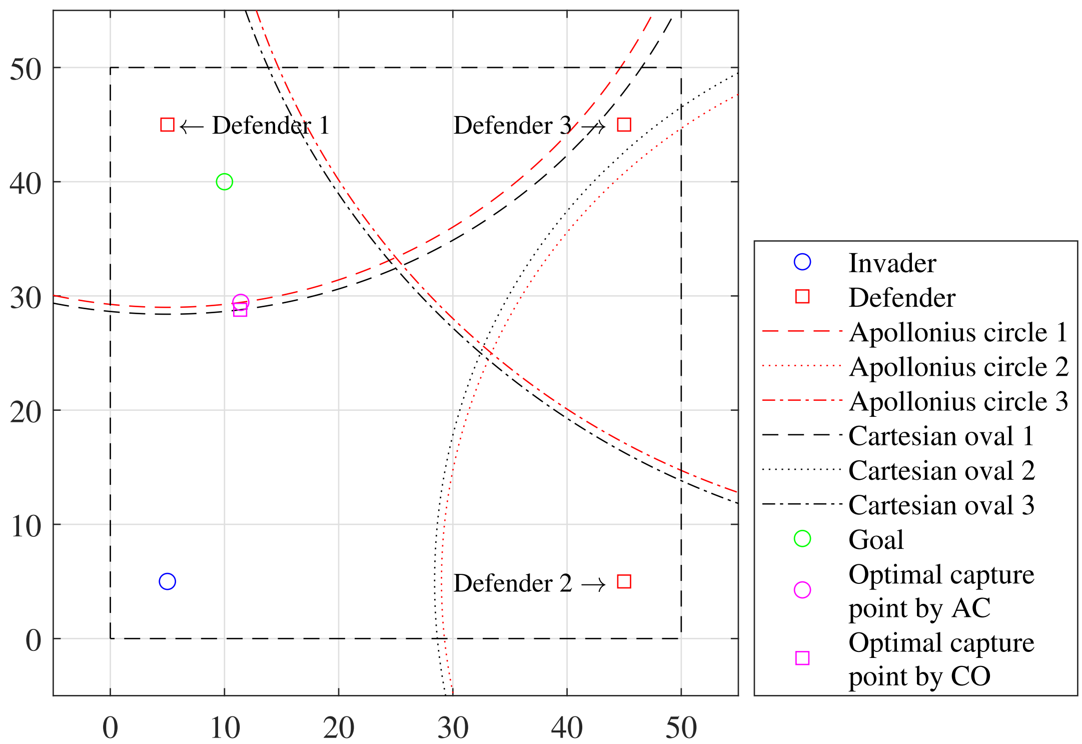


Fig. 2 The Apollonius circles and the Cartesian ovals of three defenders versus one superior invader.

In (17-19), d is the distance between the invader and the defender, η is the line-of-sight angle from the defender to the invader, and $\rho > 0$ is the capture radius.

The Apollonius circles and the Cartesian ovals are depicted in Figure 2. In the scenario in the figure, the capture radius is set to 1.0 unit, the defenders' speed is 1.0 units per second. The invader's speed is 1.1 units per second. It is observed that the optimal capture point given by the AC and the CO are very close to each other for the mentioned parameters.

3 Proposed method

In this paper, the goal is to create a deceptive invader that can deceive the defenders. We implement the game of guarding several territories as our testbed. The presented game in section 2.1 has several goals. In such a game, the invader is able to demonstrate a deceptive behaviour by consistently changing its goal. We utilize a hierarchical strategy, simillar to the strategy that was proposed in [15]. In this paper, we expand the idea to a game with one invader and several defenders. In this strategy, the invader initially knows its true goal; however, the defenders do not know the invader's intention. Since the defenders'

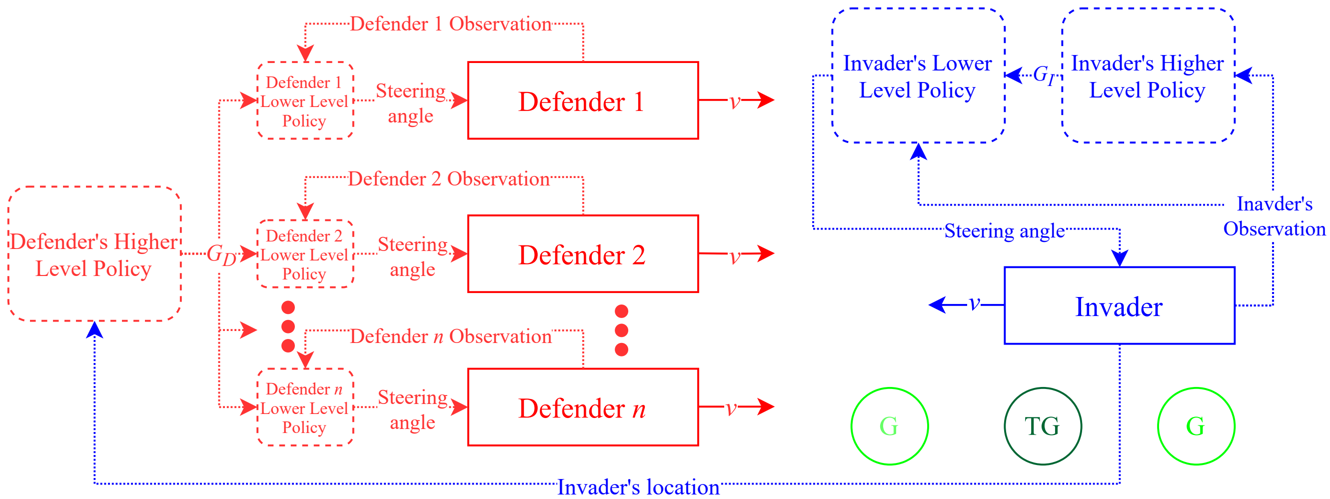


Fig. 3 The control structure

performance lies in knowing the goal, they must guess the goal by observing the invader's behaviour. In states where the invader cannot invade the goal, because it will be captured by the rational defenders, then the invader feigns toward a fake goal. Thus, the invader misleads the defenders. Although the defenders will soon find the invader's new intention, the invader can take advantage of changing the game's states. The idea leads us to two levels of policies, working simultaneously. We refer to the first level as the lower-level policy (LLP). The LLP leads the invader to seek a target and leads the defenders to defend a target. The LLP can be programmed as an artificial neural network, fuzzy inference system, or a hardcoded function such as the solution of the HJI equation or a series of IF-THEN rules. We call the second policy level the higher-level policy (HLP). Unlike the LLP, the HLP deals with a goal. It returns the goal that the invader should pretend as its true target. The HLP also returns the goal that the defender should protect. The control structure is depicted in Fig. 3. In following sections, we present how we trained the LLP and the HLP.

To train the structure depicted in Fig. 3, we implement a hierarchical FACL algorithm. The hierarchical FACL algorithm firstly trains the LLP and after obtaining the optimal policies, starts training the classifier of the HLP. The training phases are connected in an order that keeps the policies deterministic.

3.1 The Lower-Level Policy

The agents' lower-level policy is a controller and returns a suitable action, given the game's states at each time step. In this paper, we implement a fuzzy logic controller, which is trained via the FACL algorithm. To define the state, each agent will have four inputs. The defenders' input is as follows:

$$\text{Defender } i\text{th's input} = [d_{D_iI} \quad \beta_{D_i} \quad d_{D_iG} \quad \alpha_{D_i}]. \tag{20}$$

In (20), d_{D_iI} is the distance between the ith defender and the invader. The term β_{D_i} is the angle between the heading and a straight line from the ith defender to the invader. In addition, d_{D_iG} is the distance between the ith defender and the goal, whereas the term α_{D_i} is the angle between the heading and a straight line from the ith defender to the goal. The invader's input is as follows:

$$\text{Invader input} = [d_{ID} \quad \beta_I \quad d_{IG} \quad \alpha_I]. \tag{21}$$

In (21), d_{ID} is the distance between the closest defender and the invader. The term β_I is the angle between the heading and a straight line from the invader to the closest defender. In addition, d_{IG} is the distance between the invader and the goal, whereas the term α_I is the angle between the heading and a straight line from the invader to the goal.

In a one-invader one-defender game, the information provided in (20) and (21) are sufficient to define each agent with respect to the other agent and the goal. The agents' information is enough to take a decision to invade or defend the goal. It is possible to decrease the amount of information by some simplifications. For instance, in [41], the invader's inputs were the Manhattan distance from the defender and the heading to the defender. However, in [41], the goal was assumed to be fixed in one location during training and test, and the initial conditions were limited to specific regions in the field. Since in our study we are generalizing the initial conditions, we needed to add more information.

In [15], the invader sees all the defenders. In the beginning, this assumption looks helpful for the invader. However, by increasing the number of defenders, the rule base will be hit by the curse of dimensionality. In this paper, an invader with (21) as inputs only sees the closest defender. It means that unlike the method proposed in [19] and [15],the invader will have the same number of inputs, regardless of the number of the defenders. This selection has two advantages. Firstly, the invader overcomes the well-known curse of dimensionality by limiting the number of states [22]. For instance, if we consider all defenders' coordinates in the invader's inputs and use a fuzzy controller with five membership functions for each input, the number of rules is multiplied by 25 for each extra defender. Secondly, the invader might not know how many defenders are involved in the game. With (21), the invader can use the same policy, regardless of the number of the defenders. On the other hand, as in (20), the defenders only see the invader and the goal, but not each other. Thus, one may add as many defenders to the game without changing the policies. The inputs are depicted in Fig. 4 (a).

In games with a limited number of states and actions, such as grid worlds, it is more common to use a terminal reward function. In addition, if the game is complicated, it is possible to assign a reward to bottleneck states [47] or use the option-critic artichtecture [48]. However, another approach is to assign a reward for each action, referred to as an instantaneous reward function. This approach is suitable for problems with continuous state-space [41]. In this paper, we use an instantaneous reward function to train the LLP since it is observed that a terminal reward function cannot train the controller. The invader's task is to reach the goal while it is avoiding the defenders. Thus, we shape the reward function

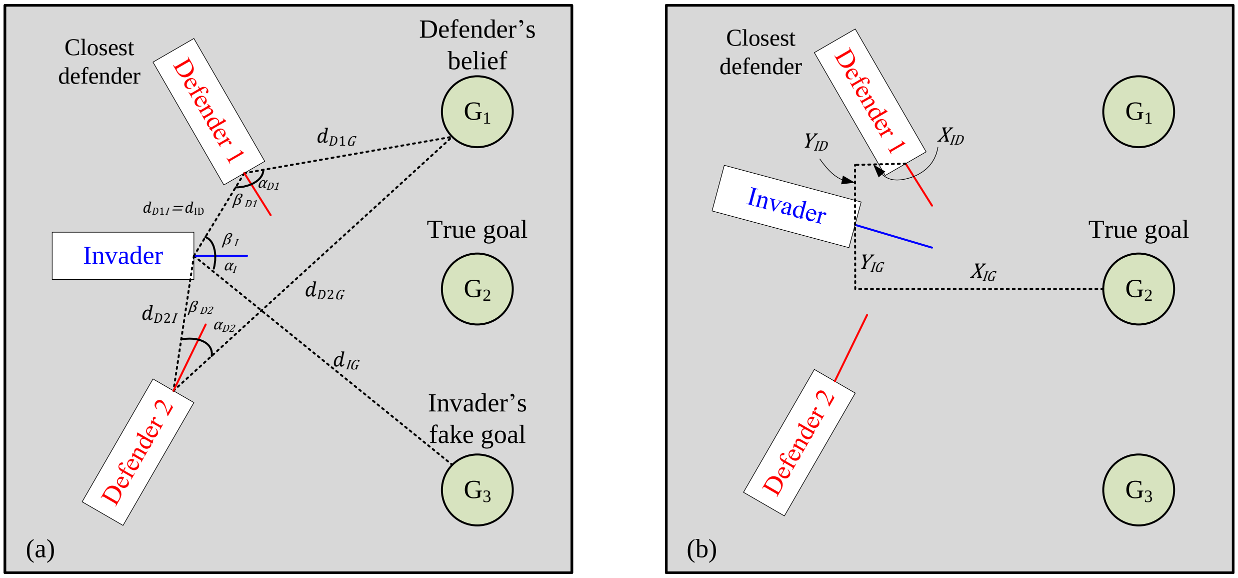


Fig. 4 The game inputs for an imaginary single-invader two-defender game: (a) The inavder's and the defender's LLP inputs. (b) The invader's HLP inputs.

as a summation of two components; getting closer to the goal at each time step and keeping distance from the defenders. Eq. (22) shows the invader's reward function:

$$\begin{aligned} R_{inv} =& \frac{W_I}{2v_{inv}\Delta t}[d_{IG}(t) - d_{IG}(t+1) + v_{inv}\Delta t] \\ &+ \frac{1-W_I}{2(v_{def}+v_{inv})\Delta t}[d_{ID}(t+1) - d_{ID}(t) + (v_{def}+v_{inv})\Delta t], \end{aligned} \tag{22}$$

where, R_{inv} is the invader's reward, $d_{IG}(t)$ is the distance between the invader and the goal at time t, while $d_{ID}(t)$ is the distance between the invader and the closest defender at time t. The term Δt is the time step. Finally, W_I is a weight in the interval of $[0, 1]$ and determines the importance of approaching the goal rather than keeping distance from the closest defender. The controller's performance relies on a suitable value for W_I. We propose a method to find the best W_I in section 4.

The defenders' task is to capture the invader. A simple strategy is to follow the invader at each time step. In this method, the defenders' heading is aligned along the defender-invader line of sight. This strategy is called pure pursuit and is studied in [30]. Another strategy is proposed in [22], where pursuers follow a capture point by estimating the invader's heading. In this paper, the defenders' reward function is defined for the invader's location and the goals' location. Thus, a reward function is defined to lead the defenders to simultaneously approach the goal and the invader. Eq. (23) shows the defenders' reward function:

$$
\begin{aligned}
R_{def_i} =& \frac{1-W_D}{2v_{def}\Delta t}[D_i G(t) - d_{D_i G}(t+1) + v_{def}\Delta t] \\
&+ \frac{W_D}{2(v_{def}+v_{inv})\Delta t}[d_{D_i I}(t) - d_{D_i I}(t+1) + (v_{inv}+v_{def})\Delta t],
\end{aligned}
\tag{23}
$$

where, R_{def_i} is the ith defender's reward, $d_{D_i G}(t)$ is the ith defender's distance to the goal, and $d_{D_i I}(t)$ is the ith defender's distance to the invader. The term W_D is a weight in the interval of $[0,1]$ and determines the importance of pure chasing versus moving toward the goal.

The learning process starts with the fuzzy controllers' initialization. The fuzzy controllers' output parameters will be set to zero. The training process starts, and each agent chooses an action based on its actor component. After taking the action and receiving the reward, the actor and the critic are updated via (8), respectively. The game continues until a terminal state is met or the game's maximum simulation time is over. In this situation, the game is restarted, while the actor and the critic will continue to adapt.

The process is decentralized since the defenders are learning separately [22]. However, it is predicted that the defenders' actor and critic would converge to a single array of output parameters after several epochs. The training process must be done for different values of W_I and W_D, to find the weights that lead to a terminal capture point that is the same as the Apollonius circle capture point.

3.2 The Higher-Level Policy

3.2.1 Invader's Higher-Level Policy

A genetic algorithm (GA) was used to train the HLP in [15]. In evolutionary algorithms, the policy is fixed at the beginning of a game, and the policy remains unchanged. The algorithm may change the policy only after the game is terminated. After the terminal time, a fitness function evaluates the consequence. As a result, all the actions in the policy are given credit, even if only a few actions have critical impacts. Even the actions that might not occur are given credit [49]. Whereas, in the methods that learn a value function, such as the FACL algorithm (section 2.2), each action is given credit by a reward function and the policy changes during the simulation.

Reinforcement learning strives to maximize the cumulative reward. On the other hand, the genetic algorithm tries to optimize a fitness function based on the result in the terminal state. The major disadvantage of using a genetic algorithm is the initial condition problem. The initial location of the agents profoundly changes the game result. For instance, if the invader's initial location is too close to the defender, the invader will be captured faster. The cost function defined in [15] can handle few initial locations, while in the RL method, a random initial location is visited at the beginning of each episode. The methods see deception from two different perspectives. The RL method can be used when the agents' initial location is unknown, whereas the GA method provides a suitable framework for studying multi-objective deception. In the latter case, by using an algorithm such as NSGA-II, a trade-off point with respect to many objectives can be achieved.

In contrast to [15], in this paper, we use the FACL algorithm in the discrete domain to train the invader's HLP. Thus, the HLP will be able to work properly for any initial location. The invader's HLP is modelled as a fuzzy classifier. The invader's HLP has four inputs as follows:

$$\text{Invader's input} = [X_{IG} \quad Y_{IG} \quad X_{ID} \quad Y_{ID}]. \tag{24}$$

In (24), (X_{IG}, Y_{IG}) is the Manhattan distance between the invader and the true goal. In addition, (X_{ID}, Y_{ID}) is the Manhattan distance between the invader and the closest defender. Fig. 4 (b) shows the invader's HLP inputs in the game.

The invader's HLP sees the true goal and the closest defender. The invader's HLP does not see the other goals. However, since the goals' locations are fixed, the invader's HLP learns how choosing a fake goal affects the invader's trajectory. The invader's HLP must be trained for each goal separately. For instance, the policy to invade goal one is different from the policy to invade goal two.

The difficulty of using the FACL algorithm for the higher-level policy is finding a proper reward function. We use a terminal reward to train the invader's HLP. The terminal reward function is assigned when the invader is captured or invades the true goal. The HLP's terminal reward function is as follows:

$$R_{inv}^{HLP} = \begin{cases} +100, & \text{True goal is invaded} \\ -100\sqrt{(x_{ter} - x_{TG})^2 + (y_{ter} - y_{TG})^2}, & \text{The invader is captured} \end{cases}, \tag{25}$$

where, R_{inv}^{HLP} is the invader's terminal HLP reward, (x_{ter}, y_{ter}) is the invader's capture point, and (x_{TG}, y_{TG}) is the true goal's location. Eq. (25) shows that the training prcess is only relies on the performance in the terminal state.

Remark 1: To avoid quick changes in the HLP output, the HLP only changes the goal once in several time steps. The invader's and defenders' selected goal remains unchanged for a few time steps after setting. The HLP updating time is reffered to t_r.

Remark 2: the actor's output is a real number in the interval of $[1, 3]$. However, the output must be mapped in the set $\{1, 2, 3\}$. We implemented the HLP as a fuzzy classifier. Each fuzzy rule will be as:

$$Rule\ l\ :\ \textbf{IF}\ x_1 \text{ is } F_1^l, ..., \text{ and } x_n \text{ is } F_n^l\ \textbf{THEN}\ x \text{ is } c_i, \tag{26}$$

where, $x = [x_1, x_2, ..., x_n]$ is the input, F_n^l is the nth membership function of lth rule. The term c_i is one of the actions and is selected from $\{1, 2, 3\}$. In our approach, first, the fired rules are grouped based on their output parameters. In each group, the fired rules have the same output parameter from $\{1, 2, 3\}$. For each group, the firing strength of each rule is calculated and summed. The output parameter of the rule corresponding to the highest summed firing strength is selected as the invader's HLP output. In the update section, only the rules in the selected group are updated.

We showed how we trained the LLP and the HLP. Training the LLP is done prior to training the HLP. In other words, we trained the HLP with the assumption of knowing the optimal LLP. We investigated training the LLP and the HLP simultaneously, which is possible in certain applications [48]. However, in our application, training individually is more effective. The reason is the added noise in the LLP, which makes the policy non-deterministic during the training phase [50]. The whole training process is depicted in Fig. 5. Fig. 5 shows the cooperation between the two policy levels. Table 2 presents the parameters used in Fig. 5.

We chose the Takagi-Sugeno fuzzy inference system for our application due to its simplicity and fast learning. The authors previously utilized the FACL algorithm in [15], and it demonstrates a suitable performance. It should be mentioned that using different kinds of the artificial neural networks (ANN) may seem tempting. However, each type of ANN has its unique application. ANNs usually need a large number of parameters, which may be unnecessary. For the HLP, we chose the FACL again because it has a suitable performance. However, using a fuzzy classifier in the HLP using a fuzzy classifier in the HLP helps one to understand the intelligent deceptive actions due to the explainability of the underlying fuzzy system. For specific purposes, such as economic analysis, training the HLP with another classifier may result in excellent performance, but it is essential to know the knowledge behind the deception. In other words, one may want to understand in which situation using deception is beneficial and a fuzzy interpretation helps these applications.

3.2.2 Defenders' Higher-Level Policy

The defenders' HLP returns a number, which represents one of the goals. In our study, the defenders are not learning an HLP, instead the defenders use a hardcoded function to guess the goal by observing the invader's position. Thus, this paper studies deception and not counter-deception as mentioned in [51]. The defenders believe that the invader has approached the true goal in the past t_d seconds. In the early time steps, the defenders' belief function returns the closest goal to the invader. Eq. (27) shows the defenders' HLP:

$$\begin{aligned}
&\text{For } t > t_d \\
&D = \underset{j\in\{1,2,..,n\}}{\arg\min}\left(\sqrt{(x_I^t - x_{G_j})^2 + (y_I^t - y_{G_j})^2} - \sqrt{(x_I^{t-t_d} - x_{G_j})^2 + (y_I^{t-t_d} - y_{G_j})^2}\right) \\
&\text{For } t \le t_d \\
&D = \underset{j\in\{1,\dots,n\}}{\arg\min}\sqrt{(x_I^t - x_{G_j})^2 + (y_I^t - y_{G_j})^2},
\end{aligned} \tag{27}$$

where, D is the defenders' belief, (x_I^t, y_I^t) and (x_{G_j}, y_{G_j}) is the invader's location at time step t. The term (x_{G_i}, y_{G_i}) is the jth goal location.

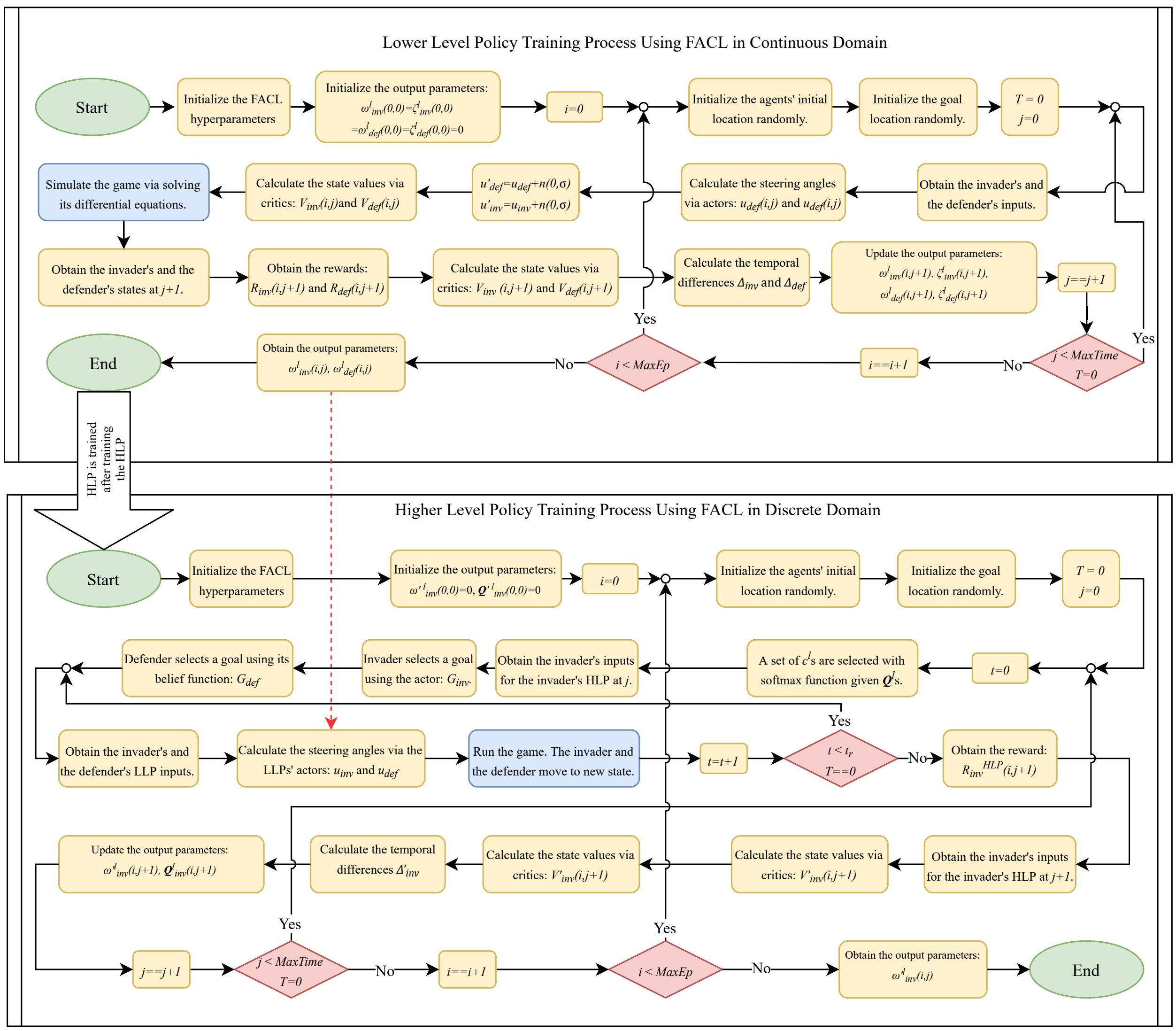


Fig. 5 The training process of the invader's and the defender's LLP and the invader's HLP

Table 2 The parameters used in Fig. 5

	Parameters	Description
Lower-level policy	$\omega^l_{inv}(i,j)$	The invader's actor output parameter given rule l at epoch i and time step j
	$\omega^l_{def}(i,j)$	The defender's actor output parameter given rule l at epoch i and time step j
	$\zeta^l_{inv}(i,j)$	The invader's critic output parameter given rule l at epoch i and time step j
	$\zeta^l_{def}(i,j)$	The defender's critic output parameter given rule l at epoch i and time step j
	i	Epoch counter
	T	Terminal state indicator. (0: non-terminal state, 1:terminal state)
	j	Time step counter
	u_{inv}	Invader's steering angle before added noise
	u_{def}	Defender' steering angle before added noise
	u'_{inv}	Invader's steering angle after added noise
	u'_{def}	Defender's steering angle after added noise
	$V_{inv}(i,j)$	Invader's state value at epoch i and time step j
	$V_{def}(i,j)$	Defender's state value at epoch i and time step j
	$R_{inv}(i,j)$	Invader's reward at epoch i and time step j
	$R_{def}(i,j)$	Defender's reward at epoch i and time step j
	Δ_{inv}	Invader's temporal difference at epoch i and time step j
	Δ_{def}	Defender's temporal difference at epoch i and time step j
	$MaxTime$	Maximum simulation steps in a single epoch
	$MaxEp$	Maximum epochs
Higher-level policy	$\omega'^l_{inv}(i,j)$	The invader's actor output parameter given rule l at epoch i and time step j
	$\mathbf{Q}^l_{inv}(i,j)$	The invader's critic output parameters given rule l at epoch i and time step j. This parameter is a vector. Each element represents the state-action value of a particular action
	i	Epoch counter
	T	Terminal state indicator. (0: non-terminal state, 1:terminal state)
	j	Time step counter
	u_{inv}	Invader's steering angle before added noise at epoch i and time step j
	u_{def}	Defender' steering angle before added noise at epoch i and time step j
	$V'_{inv}(i,j)$	Invader's state value at epoch i and time step j
	$R'^{HLP}_{inv}(i,j)$	Invader's reward at epoch i and time step j
	Δ'_{inv}	Invader's temporal difference at epoch i and time step j
	$MaxTime$	Maximum simulation steps in a single epoch
	$MaxEp$	Maximum epochs
	t_r	Updating time
	G_{inv}	Invader's selected goal
	G_{def}	Defender's selected goal
	c^l	Selected output parameter using softmax function given rule l

4 Simulation and Results

In this section, we implement the proposed method in section 3 for the game of guarding several territories as presented in section 2. The FACL algorithm is implemented to train both the LLP and the HLP.

4.1 Preliminaries

4.1.1 The Game

The game field can be in any shape and size; however, we set the field as 50×50 units square. We are going to investigate the effect of more agents in the game. We examine two scenarios: 1) single-invader single-defender case (SISD), 2) single-invader multi-defenders case (SIMD). We will also compare the effectiveness of deception under two speed scenarios: 1) superior invader 2) agents with equal speed. We set the agents' speed to 1.0 unit/sec in the equal speed case. We set the defenders' speed as 1.0 unit/sec and the invader's speed as 1.1 units/sec in the superior invader case. The agents are robots as modelled in (1), where their axle length (L in (1)) is 0.3 units. The positive capture radius of the defender is set to 1.0 unit. In addition, the target radius is also set to 1.0 unit. Finally, the time step is 0.1 seconds, and the maximum time for the game is 200 seconds. If the game does not finish in 200 seconds, no reward is given to the agents in training both the LLP and the HLP.

4.1.2 The Lower-Level Policy

To train the LLP, we set the discount factor (γ) to 0.7. With this choice, the agent tries to increase the current reward and the reward of a few steps ahead. In addition, we set α_L and β_L in (8) to 0.05 and 0.025, respectively. Finally, the maximum LLP training epoch is set to 50,000 and the initial σ is set to 1.0. It should be noted that α_L, and β_L are decayed by $10^{log_{10}(\frac{0.01}{MaxEp})}$ in each epoch, where $MaxEp$ is the maximum number of epochs. In other words, at the final epoch, α_L, β_L will be 0.01 of their initial values. The exploration rate (σ) is also decayed by $10^{log_{10}(\frac{0.1}{MaxEp})}$ in each epoch. The parameters $MaxEp$, α_L, β_L, and σ are selected with trial and error. After learning for half of the $MaxEp$, the policy has almost converged.

A Takagi-Sugeno FLC is used as the LLP's actor component, and a Takagi-Sugeno FIS is implemented as the critic component. The LLP has four inputs, as mentioned in section 3.1. The domain of each input is uniformly covered with five triangular membership functions. The distance inputs are in the interval of $[0, 50\sqrt{2}]$, and the angle inputs are in the interval of $[-\pi, \pi]$.

4.1.3 The Higher-Level Policy

To train the HLP, we set the discount factor (γ) to 0.9995. The reason behind this discount factor is the existence of a terminal reward function in the training process. We want to train the invader's deceptive behaviour to see the terminal reward from the first step of the game.

Thus, the invader can select the best deceptive actions from the beginning. In addition, we set α_L and β_L in (14), to 1.0 and 0.5, respectively. The softmax temperature in (11) is set to 2, and it is not changing during the training process. The domain of each input is uniformly covered with 20 triangular membership functions. The inputs are in the interval of $[-50, 50]$.

The term (t_d) in (27), and HLP updating time (t_r) in Fig. 5 are set to be 20 time steps $(t_d = t_r = 2$ seconds). It is observed that increasing t_d from 1 time step to 20 time steps decreases the defenders' chance to make a wrong belief. However, decreasing t_r causes to much changes in the invader's and the defenders' selected goal.

The coopration between the LLP and the HLP is depicted in Fig. 3.

4.2 Training the LLP

4.2.1 Single-Invader Single-Defender

The invader and the defender are trained via the FACL algorithm for continuous actions. The training process starts by initializing the agents' locations. The invader's and the defender's locations are initialized randomly in the field. The x coordinates and the y coordinates are selected from the interval of $[0, 50]$ via the uniform random number generator. In addition, the headings are selected from the interval of $(-\pi, \pi]$. The same as agents, the goal location is selected randomly. At each time step, an action is selected by the defender's and the invader's actors and is summed with Gaussian random noise. After taking the actions and receiving the rewards, the actor and the critic components are updated via (8). In this part, the agents' LLP input are (20) and (21) and the output is the steering angle given by (2).

In [15], the authors found the weights by trial and error. First, they derived the optimal capture point via the Apollonius circle. Then, they chose the reward weights that would reach the same capture point. In this paper, we proposed a min-max method to find the reward weights (W_I and W_D in (22) and (23)), which is an expanded and modified version of the method in [41]. In our approach, we used the terminal distance between the invader and the goal as the pay-off. The invader strives to minimize the pay-off, while the defender tries to maximize the pay-off. The terminal distance between the invader and the goal is as follows:

$$d = \sqrt{(x_{inv}^{ter} - x_G)^2 + (y_{inv}^{ter} - y_G)^2}, \tag{28}$$

where, $(x_{inv}^{ter}, y_{inv}^{ter})$ is the invader's terminal location, and (x_G, y_G) is the goal's location. Table 3 shows d for different W_I and W_D in the equal speed scenario. For the data in Table 3, the invader's initial location is (5,5), the defender's initial location is (30,30), and the goal's location is (10,40).

The defender is supposed to capture the invader at the furthest possible distance from the goal. Thus, we choose the weight, which leads the defender to have a relatively larger terminal distance. As shown in Table 3, for $W_D = 0.75$ the term d is at its maximum. In contrast, the invader must reach the closest possible distance to the goal. For $W_I = 0.50$, the term d is at its minimum. Thus, in training the LLP, the weights are set to $W_D = 0.75$ and $W_I = 0.50$.

Table 3 Terminal distance between the invader and the goal (d in (28)) for different reward weights in (22) and (23) in the equal speed scenario

W_D \ W_I	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80
0.700	Invaded	10.41	10.18	10.44	11.09	11.76	12.15	12.34	12.78
0.725	11.29	10.85	10.64	11.04	11.57	12.04	12.45	12.79	12.99
0.750	11.43	11.00	**10.69**	11.12	11.62	12.17	12.63	12.92	13.20
0.775	11.37	11.02	10.58	11.16	11.50	12.09	12.60	12.87	13.25

The same table is created for the superior invader scenario. Table 4 shows the terminal distance between the invader and the goal. It is shown that for $W_D = 0.75$, the invader is captured at a relatively long distance from the goal. The invader reaches a closer distance to the goal if $W_I = 0.50$. The term *Invaded* in Table 3 and 4 means the defender has failed to capture the invader, although the goal is in the capture zone of the defender. A W_D should not be selected if there is at least one successful invasion case in its row since the invader can find a policy to reach the goal.

Table 4 Terminal distance between the invader and the goal (d in (28)) for different reward weights in (22) and (23) in the superior invader scenario

W_D \ W_I	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80
0.700	8.51	8.53	8.26	8.68	9.51	10.00	10.60	11.08	11.22
0.725	8.81	8.66	**8.30**	8.65	9.51	10.19	10.66	11.21	11.39
0.750	8.55	8.30	7.68	Invaded	Invaded	Invaded	Invaded	Invaded	Invaded
0.775	8.42	8.20	7.23	Invaded	Invaded	Invaded	Invaded	Invaded	Invaded

Fig. 6 shows the LLP training outcome for both speed scenarios with the selected weights for each case. As shown in the Fig. 6, the capture point is very close to the optimal capture points in section 2.3. In Fig. 6, the bisector line is drawn for the equal speed scenario. In addition, the Cartesian oval is plotted for the superior invader case. In the LLP, we did not use the bisector nor the Cartesian oval to train the policy or find the trajectory. We used them to find the optimal capture point (the closest point on the discrimination line to the goal). Then, we showed that the learning algorithm could train the policy that has the same terminal point. Although we showed the performance for a few initial conditions in Fig. 6, the terminal point is almost the same as the optimal capture point for any initial condition.

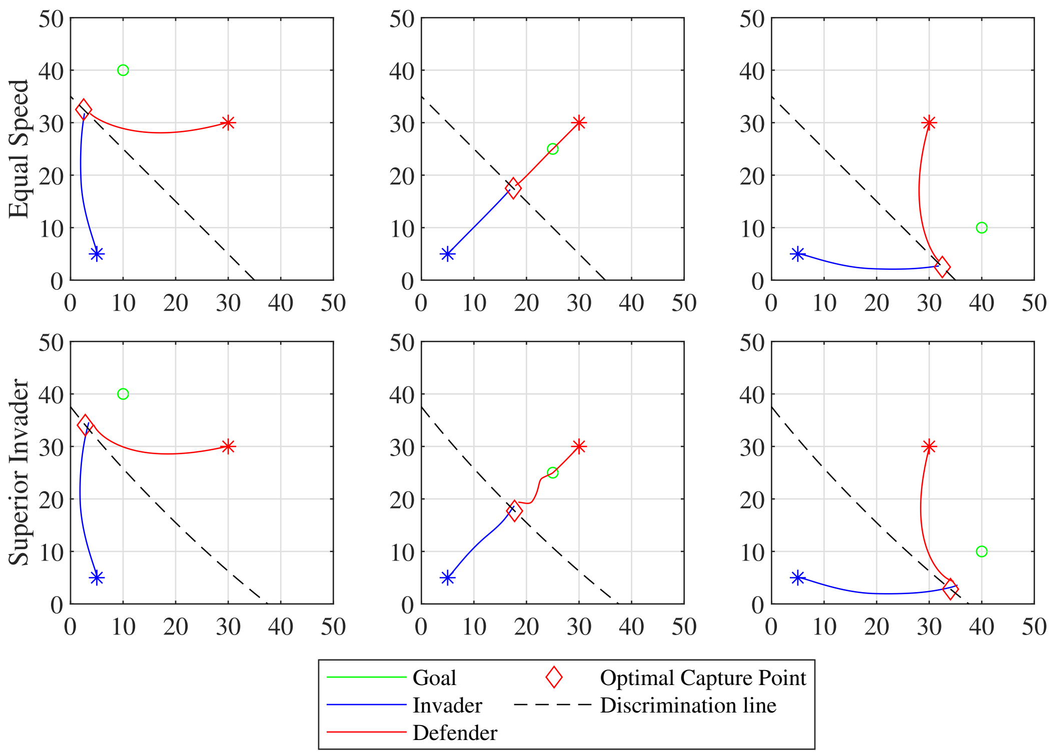


Fig. 6 The discrimination lines and the LLP training outcome. For the equal speed scenario $W_D = 0.75$ and $W_I = 0.50$. For the superior invader scenario $W_D = 0.725$ and $W_I = 0.50$.

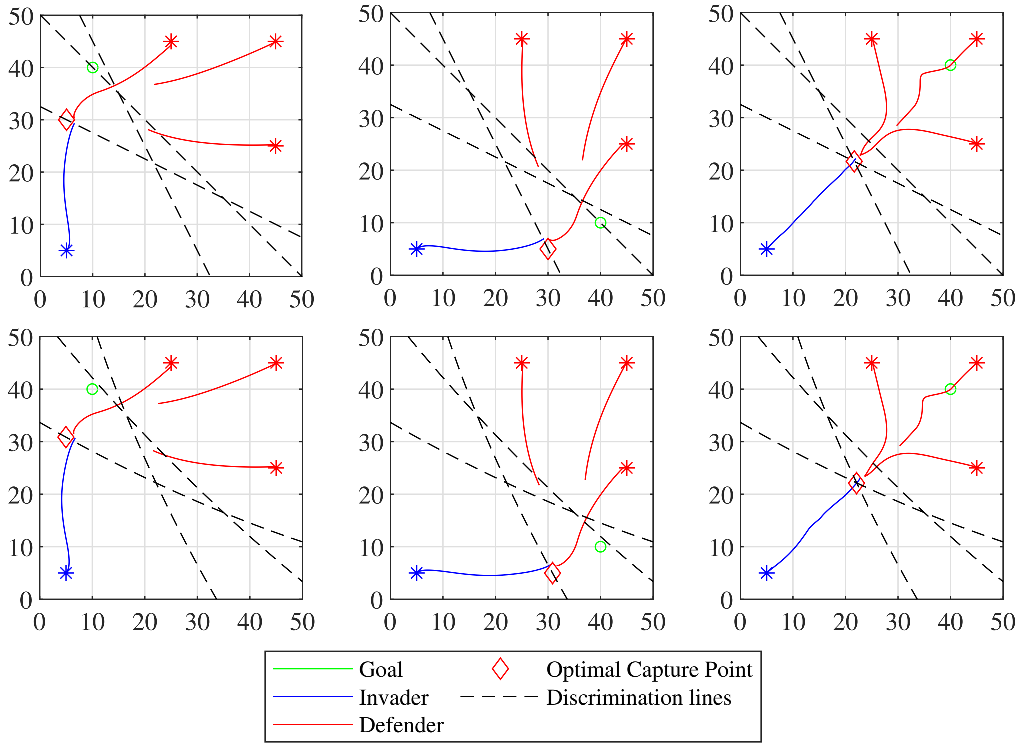


Fig. 7 The discrimination lines and the LLP training outcome. The same policy derived in section 4.2.1 is implemented. For the equal speed scenario $W_D = 0.75$ and $W_I = 0.50$. For the superior invader scenario $W_D = 0.725$ and $W_I = 0.50$.

4.2.2 Single-Invader Multi-Defender

We investigate the possibility of reusing the policy, derived in section 4.2.1 in a single-invader multi-defender (SIMD) case. As presented in section 3.1, the invader only sees the closest defender, and the reward function is calculated based on the location of the closest defender. On the other hand, the defenders only see the invader, and they do not communicate or share information. Thus, we use the same policy derived in section 4.2.1 in the SIMD case. Fig. 7 shows the trajectory of a single-invader three-defender case. It is shown that the capture point is close to the optimal capture point, derived via the bisector method and the Cartesian oval.

4.3 Training the invader's HLP

The LLP policies from section 4.2 are used in training the invader's HLP. As mentioned in section 4.1, there are three goals in the game. The primary task of the invader's HLP is to switch the invader's intended goal in the applicable states when the invader cannot reach the true goal. In all cases, the defenders' HLP are the same, calculated via (27). In cases with more than a single defender, all defenders have the same belief.

It should be noted that the invader's HLP only has the coordinates of the true goal and not the other goals. Meanwhile, the coordinates of the other goals are fixed with respect to the true goal. The invader's HLP can learn how switching between goals affects the invader's performance. Thus, the invader's HLP must be trained for each goal seperately [15]. As mentioned in section 3.2.1, the invader's HLP has four inputs as in (24). The invader only sees the closest defender. Table 5 shows some of the parameters to train the invader's HLP.

Table 5 Parameters used in the invader's HLP

Parameters	Values
Goals	G_1=(10,40), G_2=(40,10), G_3=(40,40)
Invader's initial location	Any point on the boundary
Invader's initial heading	Perpendicular to the boundary
Defender's initial location (SISD)	(25,25)
Defender's initial heading (SISD)	$(\frac{-3\pi}{4})$
Defenders' initial location (SIMD)	(25,50),(50,50),(50,25)
Defenders' initial heading (SIMD)	$(\frac{-\pi}{2})$,$(\frac{-3\pi}{4})$,(π)

Table 5 shows that the defenders' initial location is set to a fixed point. Although we fixed the defenders' location close to the goals in our simulations, the proposed algorithm can be implemented for any initial conditions. Based on the initial location of the agents, using deception might be more effective or might have little effect. If all the defenders start from the same location, the result will be identical to a game with one defender. The reason is the defenders use the same policies so that they will have the same trajectory. This situation makes deception more effective since the game will be the same as a one-invader one-defender game. Another example may happen if the defenders' initial locations are far from the invader and far from the goals. In such an example, deception will be more effective because the invader has enough time to play deceptively before the defenders get close. Another example may happen if all the defenders' initial conditions are located very close to the invader. In such a situation, deception will be less effective because the invader has little time to manipulate the defender. Thus, considering a random initial location for the defenders makes the presentation more complicated because of the numerous combinations. As such, we arrange the defenders close to the goals and highlight the strong effects of deception.

4.4 Results and Discussion

Figs. 8-11 show the performance of the non-deceptive cases. In these cases, the invader's HLP returns the true goal during the simulation. However, the defenders do not know the true goal, and they have to guess it via the defender's belief function in (27). Fig. 8 and 9 show the non-deceptive game for the SISD case. The first row shows the agents' trajectory,

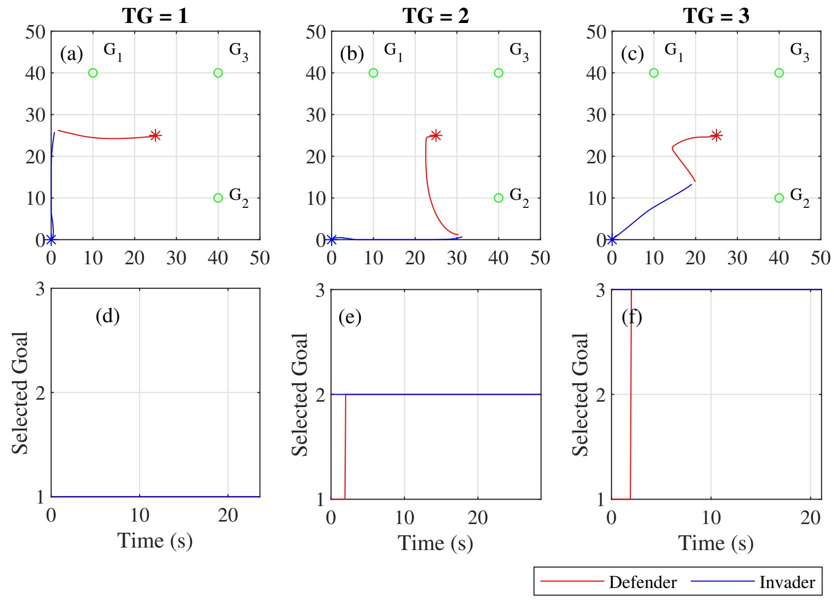


Fig. 8 The SISD game performance without using deception in the superior invader scenario

while the second row shows the HLP output. To keep the section concise, we describe the most challenging game in each figure, which is the game where $TG = 3$.

Fig. 8 shows three SISD examples with a superior invader. In case (c), the true goal is G_3. The invader takes G_3 from the beginning, and the defender's belief is G_1 as shown in Fig. 8 (d). At $t = 2$ seconds, the defender's belief is changed to 3. The invader is captured at (19.1,13.3) at $t = 21.1$ seconds.

Fig. 9 shows three SISD examples with equal speed agents. In case (c), the true goal is G_3. The invader takes G_3 from the beginning to the end, as shown in Fig. 9 (d). The defender's HLP returns G_1 at the beginning , and at $t = 2$ seconds, the defender chooses goal G_3. The game is terminated after 22.4 seconds, where the invader is at point (18.2,13.1).

Figures 10 and 11 show three examples of the non-deceptive SIMD game. In cases depicted in Figure 10, all defenders have an equal speed of 1.0 unit/sec, while the invader's speed is 1.1 units/sec. Since the game is non-deceptive, the invader takes true goals. We describe case (c). In case (c), the invader starts from (0,0) by taking G_3 as its true target. However, the defenders take G_1 from the beginning until $t = 2$ seconds. The game finishes at $t = 29.8$ seconds when two defenders capture the invader simultaneously at (23.7,22.6).

Figure 11 shows the SIMD game where the agents' speed is set to 1.0 unit/sec. In case (b), the invader goes toward G_3, the true goal. The defenders choose G_1 in the beginning. But, at $t = 2$ seconds, they switch their intended goal to G_3. The game finishes at $t = 31.6$ seconds when two defenders at (23.0,21.6) capture the invader.

Fig. 12 shows a deceptive SISD game. In this figure, the invader's speed is 1.1 units/sec, while the defender's speed is 1.0 unit/sec. We describe case (c), where the true goal is G_3. The invader's initial choice is heading toward G_1. The invader keeps choosing G_1 for 20

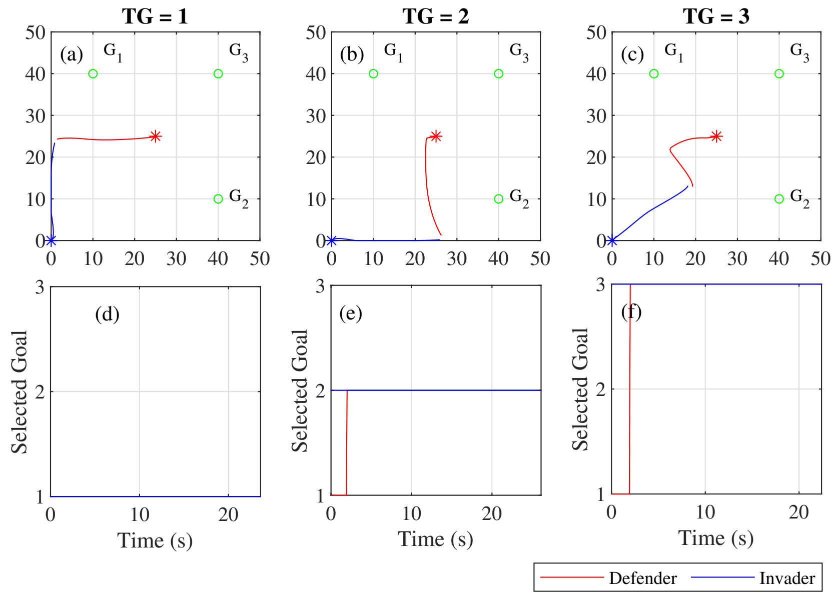


Fig. 9 The SISD game performance without using deception in the equal speed scenario

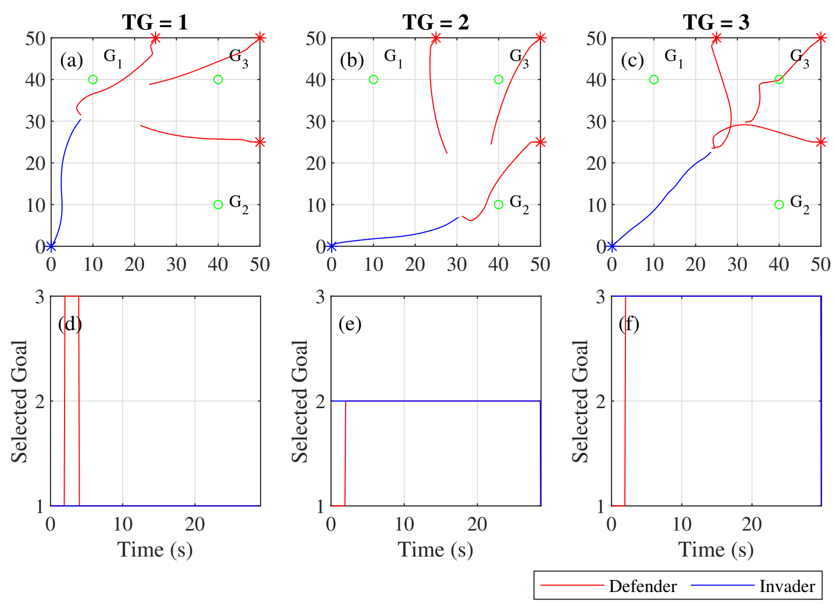


Fig. 10 The SIMD game performance without using deception the superior invader scenario

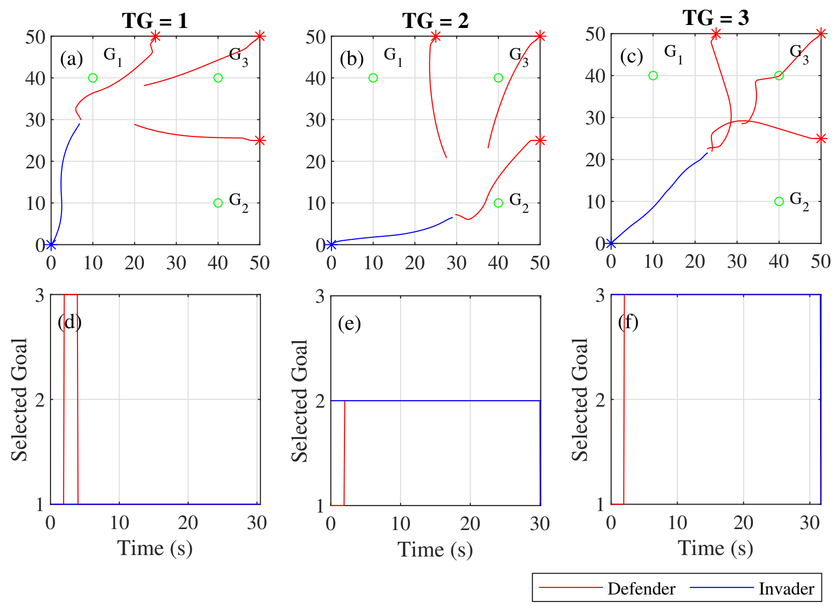


Fig. 11 The SIMD game performance without using deception in the equal speed scenario

seconds. The defender chooses G_1 in the beginning. At $t = 20$ seconds, the invader changes its goal to G_2 and at $t = 26$ seconds, the invader chooses the true goal, G_3. Meanwhile, the defender discovers the invader's intention and changes its goal to G_2 at $t = 22$ seconds. During this transition, the defender misses the invader at around (5.4,18.9). The defender chooses G_3 at $t = 30$ seconds when it is too late for capturing the invader. The game finishes at $t = 63.3$ seconds by a successful invasion. Almost the same process is carried out in cases (a) and (b).

Fig. 13 shows a deceptive SISD game for agents with equal speed. This game is more challenging than a superior invader game since the invader is weaker in comparison to the game in Fig. 12. In case (c) the true goal is G_3. The invader starts from (0,0). The invader chooses G_1 in the beginning. The defender chooses the same goal in the beginning. The invader changes its goal at $t = 20$ seconds. The invader's HLP returns G_2. The defender discovers the invader's new goal and chooses G_2 at $t = 22$ seconds. The invader changes its goal to G_3 at $t = 26$ seconds , and by this transition, it successfully dodges the defender at around point (5.1,17.1). The defender discovers the new goal at $t = 32$ seconds, when it is too late to capture the invader. The invader successfully invades G_3 at $t = 68.5$ seconds.

Figs. 14 and 15 show the SIMD deceptive cases. In these cases, the game is more complex, and the number of changes in the invader's HLP is significantly higher than the single-invader single-defender cases.

Fig. 14 shows the cases that the invader is faster than the defenders. We describe case (c). In this case, the invader starts from point (0,0) by choosing G_1 as its fake goal. The defenders' initial belief is G_1 as well. The defenders change their goal to G_3 at $t = 2$ seconds and keep it for two seconds. After two seconds, the defenders choose G_1 again. The invader

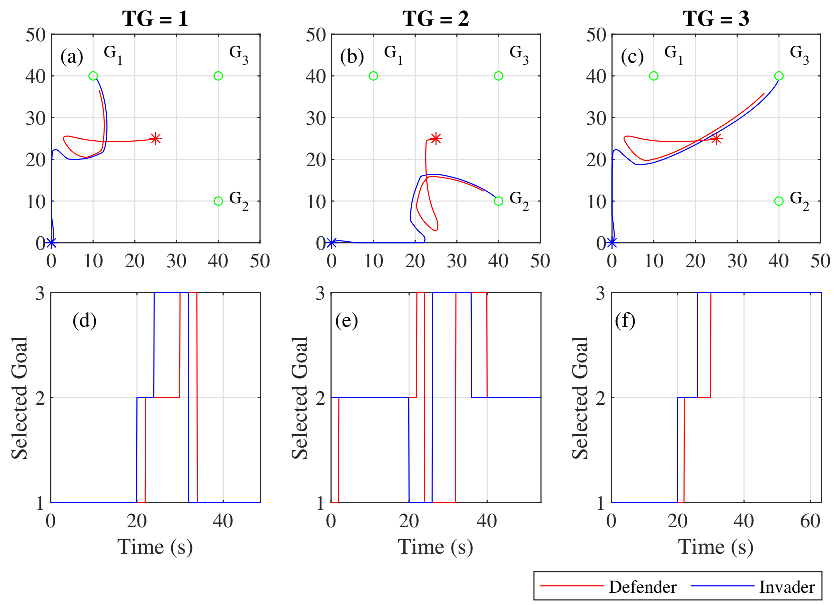


Fig. 12 The SISD game performance with using deception in the superior invader scenario

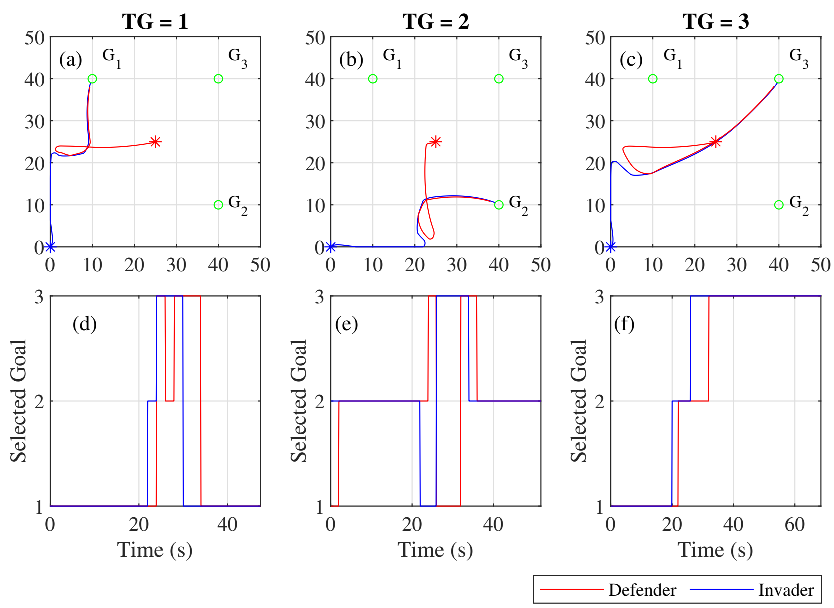


Fig. 13 The SISD game performance with using deception in the equal speed scenario

chooses G_3 at $t = 8$ seconds. After two seconds, the defenders choose G_3 as well. The invader chooses G_2 at $t = 18$ seconds. The defenders choose G_2 after two seconds. The invader chooses G_1 at $t = 24$ seconds, G_2 at $t = 26$ seconds, and G_1 at $t = 28$ seconds. During this time, the defenders keep their belief to G_2. However, at $t = 30$ seconds, the defenders choose G_3 and keep it for 2 seconds. At $t = 32$ seconds, the defenders choose G_1. To this point, the invader has successfully dodged the defenders. At $t = 54$ seconds, the invader chooses G_3. The invader dodges the defenders at around point (8.7,37.0). The defenders discover the true goal at $t = 56$ seconds when it is too late. The invader has successfully invaded the target at $t = 83.3$ seconds.

Fig. 15 shows a more challenging version of the game in Fig. 14, since the agents' speeds are equal. In this case, the number of changes in the HLP is higher than in the superior invader case. In the beginning, the invader chooses G_2, and the defenders choose G_1. At $t = 2$ seconds, the invader chooses G_1, and simultaneously, the defenders choose G_3. After two seconds, the invader chooses G_1 and keeps G_1 for four seconds. At $t = 8$, the invader chooses G_1. The invader chooses G_2 after 4 seconds and chooses G_1 after four seconds. Between $t = 20$ seconds to $t = 32$ seconds, the invader changes its goal from G_1 to G_2 and from G_2 to G_1 every two seconds. As shown in Fig. 15 (c), the invader is performing a Zig-Zag maneuver. In response, the defender changes its goal multiple times between $t = 8$ seconds to $t = 36$ seconds. After multiple changes in the HLP of both agents, the invader takes G_2 as its fake goal at $t = 32$ seconds. Meanwhile, the defenders choose G_2 at $t = 36$ seconds. The HLPs remain unchanged until $t = 60$ seconds. At $t = 60$ seconds, the invader's HLP selects G_3. After four seconds, the invader chooses G_1, and at $t = 68$ seconds, the invader chooses G_3. The defenders choose G_3 at $t = 62$ seconds, G_1 at $t = 66$ seconds, and finally G_3 at $t = 70$ seconds. The game finishes at $t = 90.7$ seconds by a successful invasion.

Multiple tests are done to challenge the robustness of the proposed method in cases with different initial conditions. In these tests, the invader's initial locations are different, and the comparison is made based on the number of successful invasions. In each test, the defenders' initial locations are fixed in the field as in Table 5. However, the invader's initial locations are points on the field's boundary. One hundred points on the boundary are selected with equal distance. The initial heading is perpendicular to the boundary. The number of successful invasions among those 100 games is reported in Table 6 for the non-deceptive case. Table 6 shows that in the SISD game with equal speeds, there are 32, 31, and 32 successful invasions out of 100, if the true goal is G_1, G_2, and G_3, respectively. In the SISD game, with a superior invader, the number of successful invasions is increased to 67, 68, and 74 successful invasions, respectively. In the SIMD game, with equal speeds, the number of successful invasions are 24, 25, and 20 successful invasions for G_1, G_2, and G_3, respectively. In the SIMD game, with a superior invader, the number of successful invasions increases to 29, 27, and 23 successful invasions for G_1, G_2, and G_3.

Table 7 shows the result of the same test as Table 6 for a deceptive game. Table 7 shows that the number of successful invasions in the SISD game and equal speeds has risen from 32, 31, and 32 to 68, 59, and 57 successful invasions for G_1, G_2, and G_3, respectively. The number of successful invasions for the SISD game and a superior invader has risen from 67,

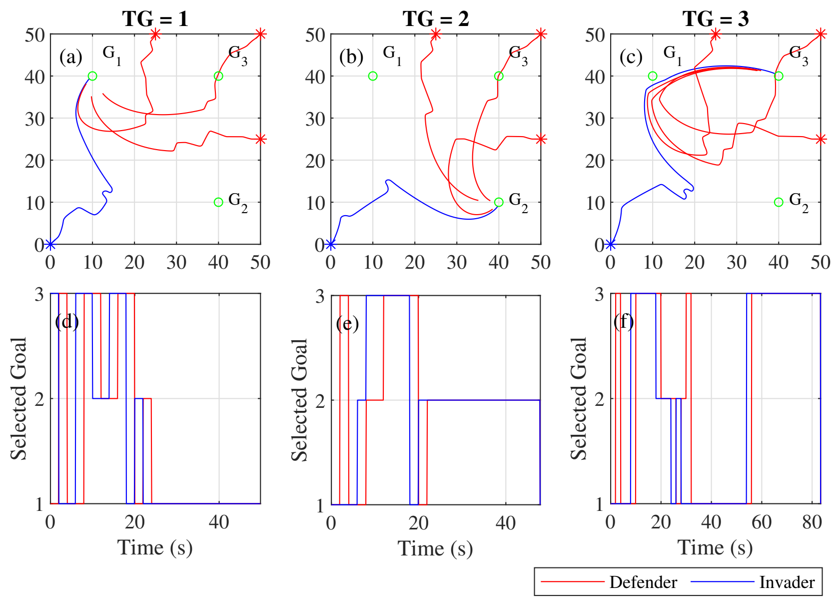


Fig. 14 The SIMD game performance with using deception in the superior invader speed scenario

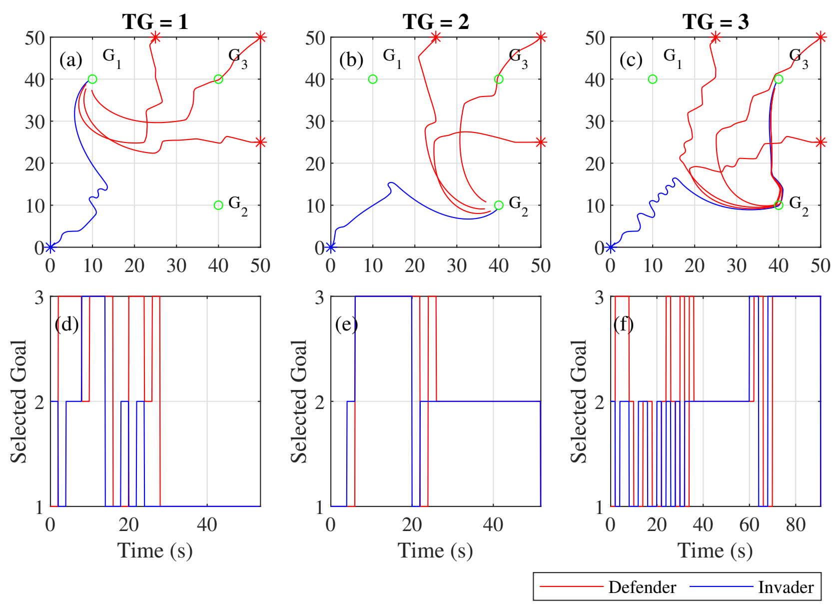


Fig. 15 The SIMD game performance with using deception in the equal speed scenario

Table 6 Number of successful invasions out of 100 games in the non-deceptive game

	Single-Invader Single-Defender (SISD)		Single-Invader Multi-Defender (SIMD)	
True Goal	Equal Speed	Superior Invader	Equal Speed	Superior Invader
1	32	67	24	29
2	31	68	25	27
3	32	74	20	23

68, and 74 successful invasions to 93, 93, and 91 successful invasions for G_1, G_2, and G_3, respectively. The performance improvement is also observed for the SIMD games. In the SIMD game with equal speeds, the number of successful invasions has risen from 24, 25, and 20 successful invasions to 30, 31, and 22 successful invasions for G_1, G_2, and G_3, respectively. Finally, the number of successful invasions in the SIMD case with a superior invader has risen from 29, 27, and 23 successful invasions to 53, 52, and 45 successful invasions, for G_1, G_2, and G_3, respectively.

Table 7 Number of successful invasions out of 100 games in the deceptive game

	Single-Invader Single-Defender (SISD)		Single-Invader Multi-Defender (SIMD)	
True Goal	Equal Speed	Superior Invader	Equal Speed	Superior Invader
1	68	93	30	53
2	59	93	31	52
3	57	91	22	45

5 Conclusion

This paper investigates a deceptive version of the game of guarding several territories. In the game, the single-invader single-defender case is studied and the single-invader multi-defender case. In addition, we investigated the agents with equal speeds as well as the superior invader cases. The deception was modelled using a hierarchical policy system. The FACL algorithm is used to train the policies hierarchically. The results show that using a hierarchical policy system to model deception can significantly improve the invader's performance in the example game. Deception and its application in real life have a long history. The results show that deception is beneficial in differential games. Although the simulation platform of this paper was a class of pursuit-evasion games, the nature of the differential games allows the designer to apply the model to any application which involves agents with conflict of interests. These applications justify using fuzzy controllers and fuzzy classifiers over artificial neural networks since the fuzzy explainability helps the human user understand the logic behind using deception in a particular state.

The proposed method cannot handle multiple invaders because of the assignment problem: when there are multiple invaders inside the game, which invader is assigned to each defender? Although the problem is not solved in this paper, the solution is proposed implicitly. A new policy level can be defined for the defender, so the defenders can actively select the best invader to follow. Last but not least, the defenders use a hardcoded function to guess the invader's intention. Defining a policy level to guess the invader's intention based on observing the invader's actions can be an exciting topic of further research. All the mentioned problems are also applicable to applications involving conflict of interest among multiple intelligent agents.

Declarations

Authors' contributions Amirhossein Asgharnia: Methodology, Software, Writing - original draft. Howard Schwartz: Supervision, Writing - review and editing. Mohamed Atia: Supervision, Writing - review and commenting.

Funding This research is funded by the Natural Sciences and Engineering Research Council of Canada (NSERC). (No. RGPIN-2017-06379 and No. RGPIN-2017-06261)

Code/Data availability The software and dataset are archived in the Machine Learning and Robotics Laboratory, Carleton University. They are available from the corresponding author on reasonable request.

Ethics approval Not applicable (this article does not contain any studies with human participants or animals performed by any of theauthors).

Consent to participate Not applicable (this article does not containany studies with human participants or animals performed by any ofthe authors).

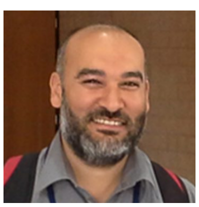
Consent for Publication All authors have approved the manuscript and agree with its publication on the Journal of Intelligent and Robotic Systems.

Competing interests The authors have no financial or proprietary interests in any material discussed in this article.

Biography

Amirhossein Asgharnia joined the Department of Systems and Computer Engineering at Carleton University as Ph.D. student in Fall 2019. He received his B.Sc. and M.Sc. degrees in Mechanical Engineering from the University of Guilan, Iran in 2015 and 2018, respectively. His research focuses on reinforcement learning and multiagent systems.

Howard Schwartz received the B.Eng. degree in Civil Engineering from McGill University, Montreal, QC, Canada in 1981, and the M.S. in Aeronautics and Astronautics in 1982 and the Ph.D. degree in Mechanical Engineering in 1987 from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. He is currently a Professor with the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON, Canada. His research interests include adaptive and intelligent systems, reinforcement learning, robotics, system modeling, and system identification. His most recent research is in multi-agent learning with applications to teams of mobile robots.



Mohamed Atia is an Assistant Professor in the Department of Systems and Computer Engineering. His research interests include real-time embedded systems, sensor-fusion, signal processing, estimation, machine learning, artificial intelligence, multi-sensors integrated navigation, guidance, and control, collaborative navigation, wireless positioning, global navigation satellite systems (GNSS), Inertial Sensors (INS), simultaneous localization and mapping, attitude and heading reference systems (AHRS), vision/radar/liDAR-aided navigation, localization and mapping.

References

[1] A. R. Wagner and R. C. Arkin, "Acting deceptively: Providing robots with the capacity for deception," *International Journal of Social Robotics*, vol. 3, no. 1, pp. 5–26, 2011.

[2] C. F. Bond and M. Robinson, "The evolution of deception," *Journal of Nonverbal Behavior*, vol. 12, no. 4, pp. 295–307, 1988.

[3] B. Skyrms, *Signals: Evolution, learning, and information.* Oxford University Press, 2010.

[4] I. Greenberg, "The Role of Deception in Decision Theory," *Journal of Conflict Resolution*, vol. 26, no. 1, pp. 139–156, 1982.

[5] B. Whaley, "Toward a General Theory of Deception," *Journal of Strategic Studies*, vol. 5, no. 1, pp. 178–192, 1982.

[6] D. Ettinger and P. Jehiel, "A theory of deception," *American Economic Journal: Microeconomics*, vol. 2, no. 1, pp. 1–20, 2010.

[7] C. F. Bond, K. N. Kahler, and L. M. Paolicelli, "The miscommunication of deception: An adaptive perspective," *Journal of Experimental Social Psychology*, vol. 21, no. 4, pp. 331–345, 1985.

[8] J. Shim and R. C. Arkin, "Biologically-inspired deceptive behavior for a robot," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7426 LNAI, pp. 401–411, 2012.

[9] R. Meira-Góes, E. Kang, R. H. Kwong, and S. Lafortune, "Synthesis of sensor deception attacks at the supervisory layer of Cyber–Physical Systems," *Automatica*, vol. 121, p. 109172, 2020.

[10] M. Ornik and U. Topcu, "Deception in Optimal Control," *2018 56th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2018*, pp. 821–828, 2019.

[11] M. O. Karabag, M. Ornik, and U. Topcu, "Deception in supervisory control," *IEEE Transactions on Automatic Control*, vol. 67, no. 2, pp. 738–753, 2022.

[12] M. Kouzehgar and M. A. Badamchizadeh, "Fuzzy signaling game of deception between ant-inspired deceptive robots with interactive learning," *Applied Soft Computing*, vol. 75, pp. 373–387, 2019.

[13] R. H. Venkatesan and N. K. Sinha, *The Target Guarding Problem Revisited: Some Interesting Revelations*, vol. 47. 2014. 19th IFAC World Congress.

[14] A. Asgharnia, H. M. Schwartz, and M. Atia, "Deception in the game of guarding multiple territories: A machine learning approach," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 381–388, 2020.

[15] A. Asgharnia, H. M. Schwartz, and M. Atia, "Deception in a multi-agent adversarial game: The game of guarding several territories," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1321–1327, 2020.

[16] E. Garcia, D. W. Casbeer, and M. Pachter, "Active target defence differential game: Fast defender case," *IET Control Theory and Applications*, vol. 11, no. 17, pp. 2985–2993, 2017.

[17] E. Garcia, D. W. Casbeer, and M. Pachter, "The complete differential game of active target defense," *arXiv*, 2020.

[18] E. Garcia, D. W. Casbeer, and M. Pachter, "Pursuit in the Presence of a Defender," *Dynamic Games and Applications*, vol. 9, no. 3, pp. 652–670, 2019.

[19] R. Isaacs, *Differential games: a mathematical theory with applications to warfare and pursuit, control and optimization*. Courier Corporation, 1999.

[20] E. P. Blasch, K. Pham, and D. Shen, "Orbital satellite pursuit-evasion game-theoretical control," *2012 11th International Conference on Information Science, Signal Processing and their Applications, ISSPA 2012*, pp. 1007–1012, 2012.

[21] M. Lau, M. Steffens, and D. Mavris, "Closed-loop control in active target defense using machine learning," *AIAA Scitech 2019 Forum*, no. January, 2019.

[22] M. D. Awheda and H. M. Schwartz, "A Decentralized Fuzzy Learning Algorithm for Pursuit-Evasion Differential Games with Superior Evaders," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 83, no. 1, pp. 35–53, 2016.

[23] H. Schwartz, "An object oriented approach to fuzzy actor-critic learning for multi-agent differential games," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 183–190, 2019.

[24] M. Pachter, "Isaacs' two-on-one pursuit-evasion game," in *Advances in Dynamic Games*, pp. 25–55, Springer, 2020.

[25] E. Garcia and S. D. Bopardikar, "Cooperative Containment of a High-speed Evader," *Proceedings of the American Control Conference*, vol. 2021-May, pp. 4698–4703, 2021.

[26] E. Garcia, "Cooperative target protection from a superior attacker," *Automatica*, vol. 131, p. 109696, 2021.

[27] A. Von Moll, D. Casbeer, E. Garcia, D. Milutinović, and M. Pachter, "The Multi-pursuer Single-Evader Game: A Geometric Approach," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 96, no. 2, pp. 193–207, 2019.

[28] R. Yan, Z. Shi, and Y. Zhong, "Reach-Avoid Games with Two Defenders and One Attacker: An Analytical Approach," *IEEE Transactions on Cybernetics*, vol. 49, no. 3, pp. 1035–1046, 2019.

[29] R. Yan, Z. Shi, and Y. Zhong, "Cooperative strategies for two-evader-one-pursuer reach-avoid differential games," *International Journal of Systems Science*, vol. 52, no. 9, pp. 1894–1912, 2021.

[30] V. R. Makkapati and P. Tsiotras, “Optimal Evading Strategies and Task Allocation in Multi-player Pursuit–Evasion Problems,” *Dynamic Games and Applications*, vol. 9, no. 4, pp. 1168–1187, 2019.

[31] M. Z. Qadir, S. Piao, H. Jiang, and M. E. H. Souidi, “A novel approach for multi-agent cooperative pursuit to capture grouped evaders,” *Journal of Supercomputing*, vol. 76, no. 5, pp. 3416–3426, 2020.

[32] M. D. Awheda and H. M. Schwartz, “A Residual Gradient Fuzzy Reinforcement Learning Algorithm for Differential Games,” *International Journal of Fuzzy Systems*, vol. 19, no. 4, pp. 1058–1076, 2017.

[33] L. Leng, J. Li, J. Zhu, K. S. Hwang, and H. Shi, “Multi-Agent Reward-Iteration Fuzzy Q-Learning,” *International Journal of Fuzzy Systems*, vol. 23, no. 6, pp. 1669–1679, 2021.

[34] U. Gneezy, “Deception: The role of consequences,” *American Economic Review*, vol. 95, no. 1, pp. 384–394, 2005.

[35] W. McEnenaey and R. Singh, “Deception in autonomous vehicle decision making in an adversarial environment,” *Collection of Technical Papers - AIAA Guidance, Navigation, and Control Conference*, vol. 4, no. August, pp. 3032–3043, 2005.

[36] A. Dragan, R. Holladay, and S. Srinivasa, “Deceptive robot motion: synthesis, analysis and experiments,” *Autonomous Robots*, vol. 39, no. 3, pp. 331–345, 2015.

[37] P. Bontrager, A. Khalifa, D. Anderson, M. Stephenson, C. Salge, and J. Togelius, “” superstition” in the network: deep reinforcement learning plays deceptive games,” in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 15, pp. 10–16, 2019.

[38] S. Ghiya and K. Sycara, “Learning complex multi-agent policies in presence of an adversary,” *arXiv preprint arXiv:2008.07698*, 2020.

[39] C. Li, X. Wei, Y. Zhao, and X. Geng, “An effective maximum entropy exploration approach for deceptive game in reinforcement learning R,” *Neurocomputing*, vol. 403, pp. 98–108, 2020.

[40] E. D. Oliveira, L. Donadoni, S. Boriero, and A. Bonarini, “Deceptive Actions to Improve the Attribution of Rationality to Playing Robotic Agents,” *International Journal of Social Robotics*, vol. 13, no. 2, pp. 391–405, 2021.

[41] H. Raslan, H. Schwartz, and S. Givigi, “A learning invader for the “guarding a territory” game,” *Journal of Intelligent & Robotic Systems*, vol. 83, no. 1, pp. 55–70, 2016.

[42] G. Klancar, A. Zdesar, S. Blazic, and I. Skrjanc, *Wheeled Mobile Robotics: From Fundamentals Towards Autonomous Systems*. USA: Butterworth-Heinemann, 1st ed., 2017.

[43] C. V. Analikwu and H. M. Schwartz, "Multi-agent learning in the game of guarding a territory," *International Journal of Innovative Computing Information and Control*, vol. 13, pp. 1855–1872, 2017.

[44] X. Dai, C. K. Li, and A. B. Rad, "An approach to tune fuzzy controllers based on reinforcement learning for autonomous vehicle control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 3, pp. 285–293, 2005.

[45] H. M. Schwartz, *Multi-agent machine learning: A reinforcement approach*. John Wiley and Sons, 2014.

[46] L. Jouffe, "Actor-critic learning based on fuzzy inference system," in *1996 IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems (Cat. No.96CH35929)*, vol. 1, pp. 339–344 vol.1, 1996.

[47] M. M. Botvinick, Y. Niv, and A. C. Barto, "Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective," *Cognition*, vol. 113, no. 3, pp. 262–280, 2009.

[48] P.-l. Bacon, J. Harb, and D. Precup, "The Option-Critic Architecture," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1726–1734.

[49] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[50] A. Levy, R. Platt, G. Konidaris, and K. Saenko, "Learning multi-level hierarchies with hindsight," *7th International Conference on Learning Representations, ICLR 2019*, pp. 1–16, 2019.

[51] S. Chen and R. C. Arkin, "Counter-misdirection in behavior-based multi-robot teams," *ISR 2021 - 2021 IEEE International Conference on Intelligence and Safety for Robotics*, pp. 268–275, 2021.