Model-free Reinforcement Learning Approach for Area Coverage Control using Heterogeneous Mobile Robots

Amr Elhussein and Md Suruz Miah
Electrical & Computer Eng.
Bradley University, Peoria, Illinois, USA
aelhussein@mail.bradley.edu and smiah@bradley.edu

***Abstract*—In this paper, we present a novel model-free reinforcement learning approach for solving a conventional leader-follower problem using autonomous wheeled mobile robots. Specifically, the proposed learning approach decides the desired linear velocity and the steering angle (control actions) of a follower robot to follow the time-varying motion trajectory of a leader robot. The setup of the online adaptive learning mechanism does not rely on any dynamical or kinematic information, *i.e.,* "model-free", of the car-like robots used in this work. Bellman's principle of optimality is employed to approximate the reward of the control actions determined by the proposed model-free adaptive learning algorithm. A set of computer experiments has been conducted to validate the performance of the proposed algorithm under various unplanned leader-trajectories.**

***Index Terms*—Leader-follower formation, mobile robots, reinforcement learning, policy iteration, trajectory tracking**

I. Introduction

The field of multi-agent systems is of a paramount importance in recent years due to its extensive applications in perimeter surveillance, search and rescue missions, cooperative localization, and target tracking tasks [**?**], [**?**], [**?**]. Multi-robot formation is one of many active research topics within the realm of multi-agent systems and cooperative control theory [**?**]. In particular, the leader-follower problem has received thorough attention to the robotics community, where mobile robots follow leader robot(s) satisfying certain geometric constraints, such as cyclic pursuit [**?**], [**?**], circular motion [**?**], and time-varying communication topologies [**?**], [**?**]. However, many formation control strategies which are proposed in the literature to date rely on mathematical models of various mobile robots, such as underactuated autonomous surface vehicles [**?**], nonholonomic wheeled mobile robots [**?**], unmanned underwater vehicles [**?**], [**?**], and unmanned helicopters [**?**]. In some cases, tools of computational intelligence are used to solve leader-follower problems, see [**?**], [**?**], for example, and some references therein, however, their learning mechanisms still rely on the underlying system dynamics. Recently, Miah *et al.* developed several motion control strategies [**?**], [**?**], [**?**], [**?**], [**?**] for nonholonomic mobile robots and fixed-base helicopters, where partial or full dynamical models are required for determining the desired control actions.

The current manuscript overcomes some shortcomings associated with many leader-follower control approaches proposed in the literature by introducing an online model-free adaptive learning control mechanism. The proposed learning approach determines the actuator commands of a follower robot to follow an independent virtual leader. Note that the actuator commands (control actions) are the linear speed and steering angle of the follower robot. The learning mechanism relies on collecting the state information (i.e., position and orientation) of motion trajectories generated from both the leader and follower robots online over a finite time period. Bellman's principle of optimality is then formulated as a model-free reinforcement learning strategy to determine the optimal linear speed and the steering angle of the follower robot so that it follows the motion trajectories of the leader while maintaining a pre-defined safe distance between them. We emphasize that model parameters of both the leader and follower robots are unknown. Furthermore, the effectiveness of the proposed model-free reinforcement learning approach is validated by taking into consideration that the leader robot is a physical robot model as opposed to a simple point robot model, that is considered in many formation control, trajectory tracking, and leader-follower problems in the literature [**?**].

II. Problem Setup

Suppose that a car-like robot (follower) is deployed in a planar (2D) environment. Therefore, the coordinate (x, y) is the position of the follower robot with an orientation of $\theta \in [-\pi, \pi)$ rad with respect to the global X-Y coordinate frame. Without loss of generality, the action commands of the robot are considered to be its linear speed ν and the steering angle γ. Let $\mathbf{q}_k \equiv [x_k, y_k, \theta_k]^T$ denote the pose (position and orientation) vector of the robot at time $t \geq 0$ with $t = k\,T_s$, $k \in \mathbb{N}_0$, and $T_s > 0$ being the sampling time. The robot follows a leader trajectory defined by $(\mathbf{p}_k^{[\ell]})^T = [x_k^{[\ell]}, y_k^{[\ell]}(t)]$ with the dynamics given in discrete-time as:

$$\mathbf{p}_{k+1}^{[\ell]} = \mathbf{p}_k^{[\ell]} + T_s\,\mathbf{u}_k^{[\ell]}, \tag{1}$$

where $\mathbf{u}_k^{[\ell]} \in \mathbb{R}^2$ is the control input vector of the leader at time instant k, $k = 0, 1, \ldots$. A standard setup of a leader-follower problem using car-like mobile robots is shown in Fig. 1. The leader is supposed to maintain a constant safe distance $d > 0$. The robot's discrete-time model is approximated by the first-

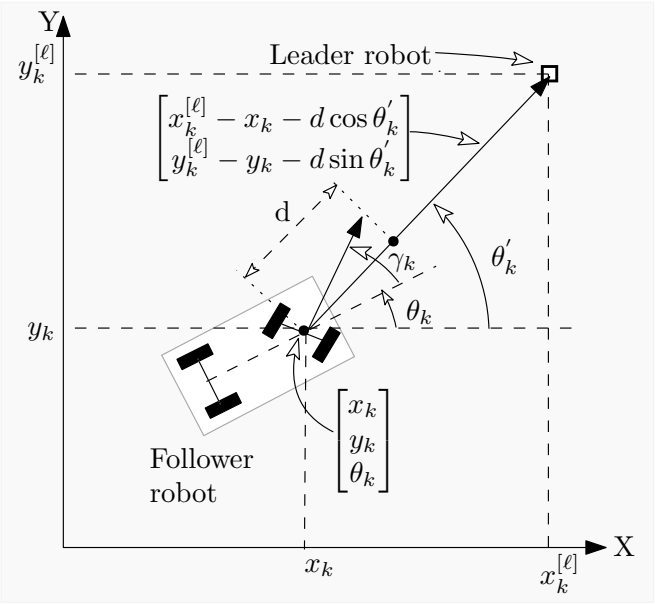

Fig. 1. Mobile robot and its leader-follower problem setup.

order Euler integration law given as:

$$x_{k+1} = x_k + T_s \nu_k \cos(\theta_k + \gamma_k) + \zeta_1, \tag{2a}$$

$$y_{k+1} = y_k + T_s \nu_k \sin(\theta_k + \gamma_k) + \zeta_2, \tag{2b}$$

$$\theta_{k+1} = \theta_k + T_s \nu_k \frac{\sin(\gamma_k)}{l} + \zeta_3, \tag{2c}$$

where $\gamma_k \in (-\frac{\pi}{2}, \frac{\pi}{2})$ is the front wheel steering angle with respect to the robot's orientation $\theta_k \in [-\pi, \pi)$, ν_k is the linear speed, l is the distance between the drive wheels of the robot, and $\zeta_1, \zeta_2, \zeta_3 \in \mathbb{R}$ are the model uncertainties. It is assumed that, the left and right wheels of the robot steer together under a no-slip condition [**?**]. The state error between the follower and the leader is defined such that

$$\mathbf{e}_k^T = [x_e, y_e, \theta_e]^T = \left[x_k^{[\ell]} - x_k - d\cos\theta_k^{'}, y_k^{[\ell]} - y_k - d\sin\theta_k^{'}, \theta_k^{'} - \theta_k\right], \tag{3}$$

where $\theta_k^{'} = \text{atan2}\left(y_k^{[\ell]} - y_k, x_k^{[\ell]} - x_k\right)$. The control problem can then be formally stated as follows: Find ν_k and γ_k such that $\mathbf{e}_k \to \mathbf{0}$ as $k \to \infty$ subject to (1) and (2). The optimization goal is to let the follower robot asymptotically follow the leader robot while maintaining a safe distance $d > 0$ regardless of motion trajectories with various complexities generated by the independent leader. It is emphasized that, the current model-free reinforcement learning mechanism determines the control actions ν_k and γ_k based on the error vectors $\mathbf{e}_k$ (data collected over a finite-time interval) as $k \to \infty$. Before illustrating the proposed learning algorithm, let us briefly revisit the preliminaries of reinforcement learning in the next section.

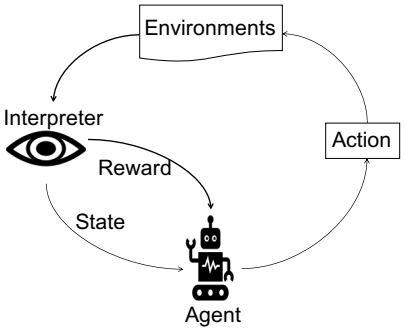

Fig. 2. A typical framing of a reinforcement learning scenario.

III. Preliminaries of Reinforcement Learning

Reinforcement learning (RL) highlights a class of problems in the field of multi-agent systems, where an agent determines sequential decisions (control actions) while learning from its associated environment. That is, an agent takes into account its learning experience from the environment and the cumulative reward to optimize decision [**?**]. The RL technology is described by four finite state tuples $(\mathcal{S}, \Omega, \mathcal{R}, \mathcal{A})$, that are environment $(s \in \mathcal{S})$, agent $(w \in \Omega)$, reward $(r \in \mathcal{R})$, and action $(a \in \mathcal{A})$ as illustrated in Fig. 2. As pioneered by Barto *et al.* in [**?**], an agent determines its action in environment $s_0 \in \mathcal{S}$ by collecting information via agent $w_0 \in \Omega$. Since the learning process is a sequential time based process, therefore, at each time step k, an agent determines the action $a_k \in \mathcal{A}$ following three cascaded phases: (i) obtaining a reward $r_k \in \mathcal{R}$, (ii) state transitions from agent state $s_k \in \mathcal{S}$ to $s_{k+1} \in \mathcal{S}$, and (iii) the agent obtains an observation $w_{k+1} \in \Omega$. The leader-follower problem addressed in this paper is formulated using a model-free reinforcement learning approach as follows:

1) **Environment**: It is a 2D planar space where agents (leader and follower) interact with each other.
2) **Agent**: Agents considered in this work are two robots (leader robot and follower robot). The robot motion behaviors are modeled using models (1) and (2). The follower robot mimics the behavior of the leader robot.
3) **Action**: The actions are the linear velocity and steering angles of the follower robot. Robots' state-transition task is performed using the control actions applied to their actuators. The main parameters that agents use to interact with the operating environment are the current position and orientation, speed, steering angle, and distance between the robots.
4) **Reward**: The goal here is to determine how follower robot can learn the leader robot's behavior to keep the minimum distance at any time and location to adjust its speed and distance. Therefore, follower robot will obtain reward $r_k \in \mathcal{R}$ at time instant k if the distance between the follower and the leader is close to a safe distance d. Herein, robots' state error is formulated as in (2). Hence, once the error $\|\mathbf{e}_k\|$ is minimized at time instant k, the

follower robot will obtain reward V, which is a modeled solving value function. This will be detailed out in the following section.

IV. Proposed Model-Free RL Approach

The solution of the leader-follower formation problem is realized using a reinforcement learning approach. It employs model-free strategies for solving a temporal difference equation developed herein. This solution is equivalent to solving the underlying Bellman optimality equation for the dynamical error model (3). The relative importance of the states in the error vector $\mathbf{e}_k$ and the control decisions (linear velocity and steering angle) of the follower-robot are evaluated using the performance (cost) index:

$$J = \sum_{k=0}^{\infty} \frac{1}{2} \left[\mathbf{e}_k^T \, \mathbf{Q} \, \mathbf{e}_k + \mathbf{u}_k^T \, \mathbf{R} \, \mathbf{u_k} \right], \tag{4}$$

where $\mathbf{Q} \in \mathbb{R}^{3\times 3}$ and $\mathbf{R} \in \mathbb{R}^{2\times 2}$ are symmetric positive definite weighting matrices. The objective of the optimization problem, following [**?**], is to find an optimal sequence of control polices $\{\mathbf{u}_k^*\}_{k=0}^{\infty}$ that minimizes the cost index J along the state-trajectories (1) and (2). Motivated by the structure of the convex quadratic cost functional (4), let the solution of the tracking control problem employ the value function $V(\mathbf{e}_k, \mathbf{u}_k)$ defined by

$$V(\mathbf{e}_k, \mathbf{u}_k) = \sum_{\kappa=k}^{\infty} \frac{1}{2} \left(\mathbf{e}_\kappa^T \, \mathbf{Q} \, \mathbf{e}_\kappa + \mathbf{u}_\kappa^T \, \mathbf{R} \, \mathbf{u}_\kappa \right).$$

This structure yields a temporal difference form (i.e., Bellman equation) as follows

$$V(\mathbf{e}_k, \mathbf{u}_k) = \frac{1}{2} \left[\mathbf{e}_k^T \, \mathbf{Q} \, \mathbf{e}_k + \mathbf{u}_k^T \, \mathbf{R} \, \mathbf{u_k} \right] + V(\mathbf{e}_{k+1}, \mathbf{u}_{k+1}).$$

Applying Bellman's optimality principle yields the optimal control policies $\mathbf{u}_k^*,\ k \geq 0$, such that [**?**]

$$\mathbf{u}_k^* = \underset{\mathbf{u}_k}{\operatorname{argmin}} \left[\frac{1}{2} \left[\mathbf{e}_k^T \, \mathbf{Q} \, \mathbf{e}_k + \mathbf{u}_k^T \, \mathbf{R} \, \mathbf{u}_k \right] + V(\mathbf{e}_{k+1}, \mathbf{u}_{k+1}) \right].$$

Alternatively, this optimal policy form is equivalent to $\mathbf{u}_k^* = \operatorname{argmin}_{\mathbf{u}_k} \left[V(\mathbf{e}_k, \mathbf{u}_k) \right]$. Therefore, the underlying Bellman optimality equation follows

$$V^*(\mathbf{e}_k, \mathbf{u}_k^*) = \frac{1}{2} \left[\mathbf{e}_k^T \, \mathbf{Q} \, \mathbf{e}_k + \mathbf{u}_k^{*T} \, \mathbf{R} \, \mathbf{u}_k^* \right] + V^*(\mathbf{e}_{k+1}, \mathbf{u}_{k+1}^*),$$

where $V^*(\cdot, \cdot)$ is the optimal solution for Bellman optimality equation. This temporal difference equation is utilized by reinforcement learning process which solves the following temporal difference approximation form

$$\hat{V}(\mathbf{z}_k) = \frac{1}{2} \mathbf{z}_k^T \, \bar{\mathbf{P}} \, \mathbf{z}_k + \hat{V}(\mathbf{z}_{k+1}), \tag{5}$$

where $\mathbf{z}_k = [\mathbf{e}_k, \mathbf{u}_k]^T \in \mathbb{R}^5$, $V(\mathbf{e}_k, \mathbf{u}_k) \approx \hat{V}(\mathbf{z}_k)$, and $\bar{\mathbf{P}}$ is a symmetric block-diagonal matrix formed using $(\mathbf{Q}, \mathbf{R})$, *i.e.,* $\bar{\mathbf{P}} = \operatorname{blockdiag}(\mathbf{Q}, \mathbf{R})$. The approximation of the solving value function $\hat{V}(\mathbf{z}_k)$ employs a quadratic form so that $\hat{V}(\mathbf{z}_k) = \frac{1}{2} \mathbf{z}_k^T \, \mathbf{P} \, \mathbf{z}_k$, where $\mathbf{P} \in \mathbb{R}^{5\times 5}$ is a positive definite matrix. Hence, the optimal control strategy $\mathbf{u}_k^*$ can be expressed as follows

$$\mathbf{u}_k^* = \underset{\mathbf{u}_k}{\operatorname{argmin}} \left[\hat{V}(\mathbf{z}_k) \right] = -\mathbf{P}_{uu}^{-1} \, \mathbf{P}_{ue} \, \mathbf{e}_k, \tag{6}$$

where $\mathbf{P}_{uu}$ and $\mathbf{P}_{ue}$ are sub-blocks of symmetric matrix $\mathbf{P}$.

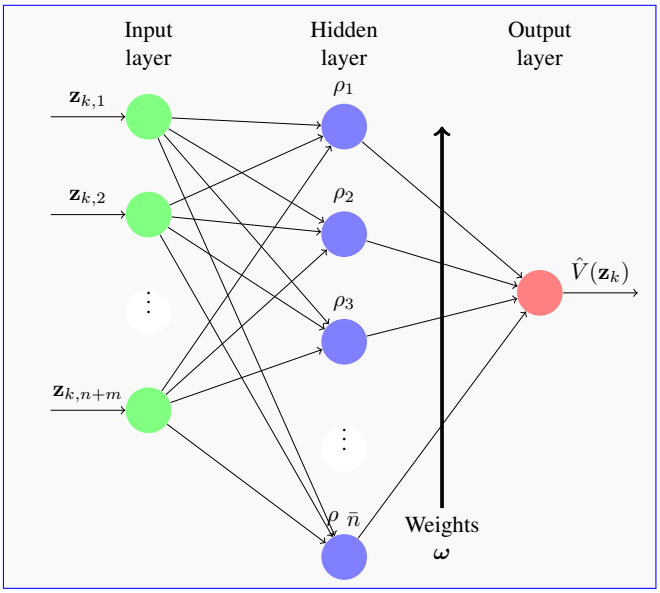

Fig. 3. Critic neural network structure for approximating value function.

A two-step solution mechanism that is based on policy iteration is employed to solve the temporal difference equation (5) using the policy (6). First, the adaptive critics are used to approximate the solving value function $\hat{V}(\cdot)$ using a multi-layer critic neural network as shown in Fig. 3. Second, the policy evaluation step of this process updates the critic weights $\boldsymbol{\omega}$ in real-time without acquiring any formation about the dynamics of the leader or follower dynamical systems (the calculation mechanism of the critic weighs $\boldsymbol{\omega}$ is explained later on). This is done to search for a strictly better policy.

Note that, the policy iteration computational setup rearranges the temporal difference expression (5) such that

$$\mathbf{z}_k^T \, \mathbf{P} \, \mathbf{z}_k - \mathbf{z}_{k+1}^T \, \mathbf{P} \, \mathbf{z}_{k+1} = \mathbf{z}_k^T \, \bar{\mathbf{P}} \, \mathbf{z}_k. \tag{7}$$

This equation is utilized repeatedly in order to evaluate a certain policy during at least $\eta \geq \bar{n}, \bar{n} = (3+2)(3+2+1)/2$ evaluation steps (i.e., the lowest evaluation interval spans k to $k + \bar{n}$ calculation samples) in order to update the critic weights vector $\boldsymbol{\omega} = \operatorname{vec}(\mathbf{P})$, which consists of connection weights between the neurons of the hidden layer and the output layer of the critic neural network shown in Fig. (3). The operator $\operatorname{vec}(\mathbf{P})$ forms the columns of the $\mathbf{P}$ matrix into a column vector ω of dimension $\bar{n} = 15$ since the matrix $\mathbf{P}$ is a symmetric matrix. The left hand side of (7) is expressed using the following critic approximation form

$$\hat{V}(\mathbf{z}_k) - \hat{V}(\mathbf{z}_{k+1}) = \boldsymbol{\omega}^T \tilde{\boldsymbol{\rho}}(\mathbf{z}_{k,k+1}),$$

where $\tilde{\boldsymbol{\rho}}(\mathbf{z}_{k,k+1}) = \boldsymbol{\rho}(\mathbf{z}_k) - \boldsymbol{\rho}(\mathbf{z}_{k+1}) \in \mathbb{R}^{15\times 1}$, $\boldsymbol{\rho}(\mathbf{z}_k) = \left(\mathbf{z}_k^q \otimes \mathbf{z}_k^h\right)$ $(q = 1, \ldots, 5,\ h = q, \ldots, 5)$, and

$\boldsymbol{\omega}^T = [0.5\,P^{11}, P^{12}, P^{13}, P^{14}, P^{15}, 0.5\,P^{22}, P^{23}, P^{24}, P^{25}, 0.5\,P^{33}, P^{34}, P^{35}, 0.5\,P^{44}, P^{45}, 0.5\,P^{55}]^T \in \mathbb{R}^{1\times 15}$ (P^{ij} is the ij^{th} entry of matrix $\mathbf{P}$). The critic weights $\boldsymbol{\omega}$ are updated using a gradient descent approach, where the tuning error ε_k at each computational instance k follows $\varepsilon_k = \boldsymbol{\omega}^T\tilde{\boldsymbol{\rho}}(\mathbf{z}_{k,k+1}) - v_k$, where $v_\kappa = \frac{1}{2}\mathbf{z}_k^T\bar{\mathbf{P}}\,\mathbf{z}_k$. As detailed earlier, it is required to perform at least $\eta \geq \bar{n}$ evaluation steps before updating the critic weights $\boldsymbol{\omega}$ (i.e., finding the new improved policy). Hence, it is required to minimize the sum of square errors such that

$$\delta_c = \sum_{\kappa=0}^{\eta-1}\frac{1}{2}(\boldsymbol{\omega}^T\tilde{\boldsymbol{\rho}}(\mathbf{z}_{k+\kappa,k+\kappa+1}) - v_{k+\kappa})^2 = \frac{1}{2}\|\mathbf{v} - \boldsymbol{\Lambda}\boldsymbol{\omega}\|^2 = \frac{1}{2}\left(\mathbf{v} - \boldsymbol{\Lambda}\boldsymbol{\omega}\right)^T\left(\mathbf{v} - \boldsymbol{\Lambda}\boldsymbol{\omega}\right),$$

where $\boldsymbol{\Lambda} = [\mathbf{o}_0, \mathbf{o}_1, \ldots, \mathbf{o}_{\eta-1}]^T \in \mathbb{R}^{\eta\times 15}$ with $\mathbf{o}_\kappa = \tilde{\boldsymbol{\rho}}^T(\mathbf{z}_{k+\kappa,k+\kappa+1}) \in \mathbb{R}^{1\times 15}$ and $\mathbf{v} = [v_0, v_1, \ldots, v_{\eta-1}]^T \in \mathbb{R}^\eta$ with $v_\kappa = \frac{1}{2}\mathbf{z}_{k+\kappa}^T\bar{\mathbf{P}}\,\mathbf{z}_{k+\kappa}$ for $\kappa = 0, 1, \ldots, \eta-1$. Therefore, the update law of the critic weights using the gradient decent approach for at least $\bar{n}$ samples is given by

$$\boldsymbol{\omega}^{[r+1]} = \boldsymbol{\omega}^{[r]} - \ell_c\frac{\partial\delta_c}{\partial\boldsymbol{\omega}} = \boldsymbol{\omega}^{[r]} - \ell_c\left(-\boldsymbol{\Lambda}^T\mathbf{v} + \boldsymbol{\Lambda}^T\boldsymbol{\Lambda}\boldsymbol{\omega}^{[r]}\right) = \boldsymbol{\omega}^{[r]} - \ell_c\boldsymbol{\Lambda}^T\left(\boldsymbol{\Lambda}\boldsymbol{\omega}^{[r]} - \mathbf{v}\right), \quad (8)$$

where $0 < \ell_c < 1$ is a critic learning rate and r is the update index of the critic weights.

The newly computed critic weights $\boldsymbol{\omega}$ are used to reconstruct the matrix $\mathbf{P}$ (i.e., updating the solving value function and hence calculating the associated policy) such that

$$\mathbf{P} = \begin{bmatrix} 2\,\omega^1 & \omega^2 & \omega^3 & \omega^4 & \omega^5 \\ \omega^2 & 2\,\omega^6 & \omega^7 & \omega^8 & \omega^9 \\ \omega^3 & \omega^7 & 2\,\omega^{10} & \omega^{11} & \omega^{12} \\ \omega^4 & \omega^8 & \omega^{11} & 2\,\omega^{13} & \omega^{14} \\ \omega^5 & \omega^9 & \omega^{12} & \omega^{14} & 2\,\omega^{15} \end{bmatrix} \in \mathbb{R}^{5\times 5},$$

where ω^i is the i^{th} entry of the weight vector $\boldsymbol{\omega}$. The complete policy iteration solution process for the leader-follower problem is detailed out in Algorithm 1.

Algorithm 1: Model-free reinforcement learning using the policy iteration solution.

Input: Sampling-time T_s, $\mathbf{Q}$, and $\mathbf{R}$
Output: Error trajectory $\mathbf{e}_k$, for $k = 0, 1, \ldots$

begin
- $k = 0, r = 0$ /* Discrete time and policy indices */
- $\eta = (n+m)(n+m+1)/2$
- Initialize $\mathbf{P}^{[0]}$ /* Positive definite */
- Set offset distance d
- Given approximate initial poses of leader and follower, compute $\mathbf{e}_0$ using error model (3)
- Compute follower's input $\mathbf{u}_0^{[0]}$ using policy (6)
- **repeat** /* Main timing loop */
  - Find $\mathbf{e}_{k+1}$ using (3)
  - Compute policy $\mathbf{u}_{k+1}^{[r]}$ using (6)
  - **if** *[*$(k+1)$ modulo $\eta] == 0$ **then**
    - $r \leftarrow r+1$ /* Evaluate policy */
    - Solve for the critic-weights $\boldsymbol{\omega}$ using (8)
    - Construct matrix $\mathbf{P}^{[r]}$ using vector $\boldsymbol{\omega}$
  - $k \leftarrow k+1$
- **until** *Tracking errors are zero*

V. Computer Experiments and Results

In the sequel, computer experiments are conducted to validate the performance of the proposed model-free adaptive learning algorithm in real-time. The results of computer experiments highlight the dynamics of the tracking errors and the convergence characteristics of the proposed algorithm (i.e., updating the critic weights). This will judge the ability of the follower robot to track independent motion trajectory of the leader robot under two different independent leader-motion trajectories. The computer experiments are realized using MATLAB simulation environment. The weighting matrices are set to $\mathbf{Q} = \text{diag}[0.01, 0.01, 0.005]$ and $\mathbf{R} = \text{diag}[10^{-6}, 10^{-6}]$. The learning rate ℓ_c is set to 0.0001. The sampling time T_s is set to $0.08\,\text{sec}$. The desired distance offset between the leader and the follower is set to $d = 0.5$ [m] for all scenarios.

In the first scenario, the leader trajectory motion is described by a unicycle model so that $\dot{x}^{[\ell]}(t) = \nu^{[\ell]}(t)\cos\theta^{[\ell]}(t)$, $\dot{y}^{[\ell]}(t) = \nu^{[\ell]}(t)\sin\theta^{[\ell]}(t)$. Initially, the leader's position is set to be the same for a short period of time with $\nu^\ell = 0, \gamma^\ell = 0$. Then it starts to move on a horizontal line with $\nu^\ell = 0.1$ [m/s], eventually the leader starts to move on an inclined path with $\nu^\ell = 0.2$ [m/s], $\gamma^\ell = 45^\circ$. The leader is initially placed at $(x, y) = (5, 2)$ [m] with an orientation of $\theta = 0^\circ$, while the follower is initially placed at $(x, y) = (0, 0)$ [m] with an orientation of $\theta = 0^\circ$. Note that during this scenario, the critic weights converge rapidly and the tracking errors decay till the follower tracks the leader. The trajectory phase plan plot, tracking error states, control signals, and tuning of critic weights are shown in Fig. 4.

A relatively unplanned complex trajectory-tracking scenario is considered. The leader moves according to a sinusoidal trajectory so that $x^{[\ell]}(t) = \alpha(t) \in [0, 10\pi]$, $y^{[\ell]}(t) = 5\,\sin(\alpha t)$, and $\theta^{[\ell]}(t) = cos(\alpha t)$. The leader robot was initially placed at $(x, y) = (0, 0)$ [m] with an orientation of $\theta = 0^\circ$ while the follower robot was initially placed at a random position around $(x, y) = (0, 0)$ [m] with a random orientation. The simulation results are summarized in Figure 5. It is observed that the critic weights take more time to converge compared to the earlier scenario, which is reluctant to the complexity of the independent trajectory of the leader. This result emphasizes the adaptability of the proposed adaptive learning mechanism to different scenarios. Further, the follower starts to move away from the leader at the beginning of the simulation before it finally converges to the leader where the tracking errors are bounded by a safe distance d.

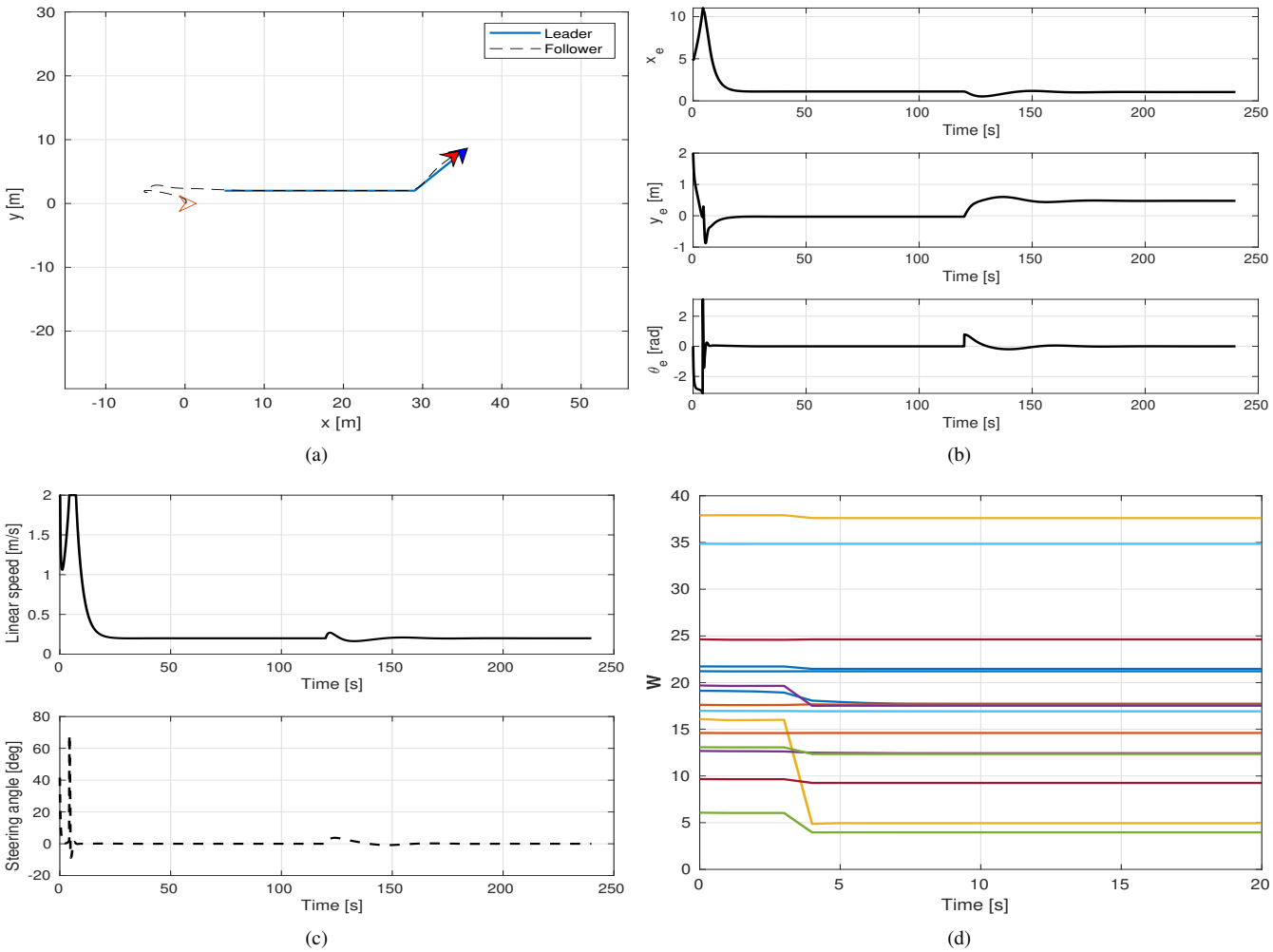

Fig. 4. First scenario (rectilinear trajectory): (a) leader-follower trajectories, (b) state tracking errors, (c) linear speed and steering angle of the follower, and (d) critic weights.

VI. Conclusion

A novel policy iteration mechanism based on a model-free reinforcement learning approach is presented for solving a conventional leader-follower formation problem using car-like mobile robots. The proposed approach does not rely on model parameters inherent in the mobile robots employed in this work. The follower robot is able to follow the leader robot that navigates along unplanned trajectories of various complexities while maintaining a nonzero safe distance between them. To the best of authors' knowledge, this is the first milestone of its kind where a model-free policy iteration based reinforcement learning approach is employed in a multi-robot formation control problem. The future work is going to extend the learning algorithm for solving a time-varying formation control problem using networked robots.

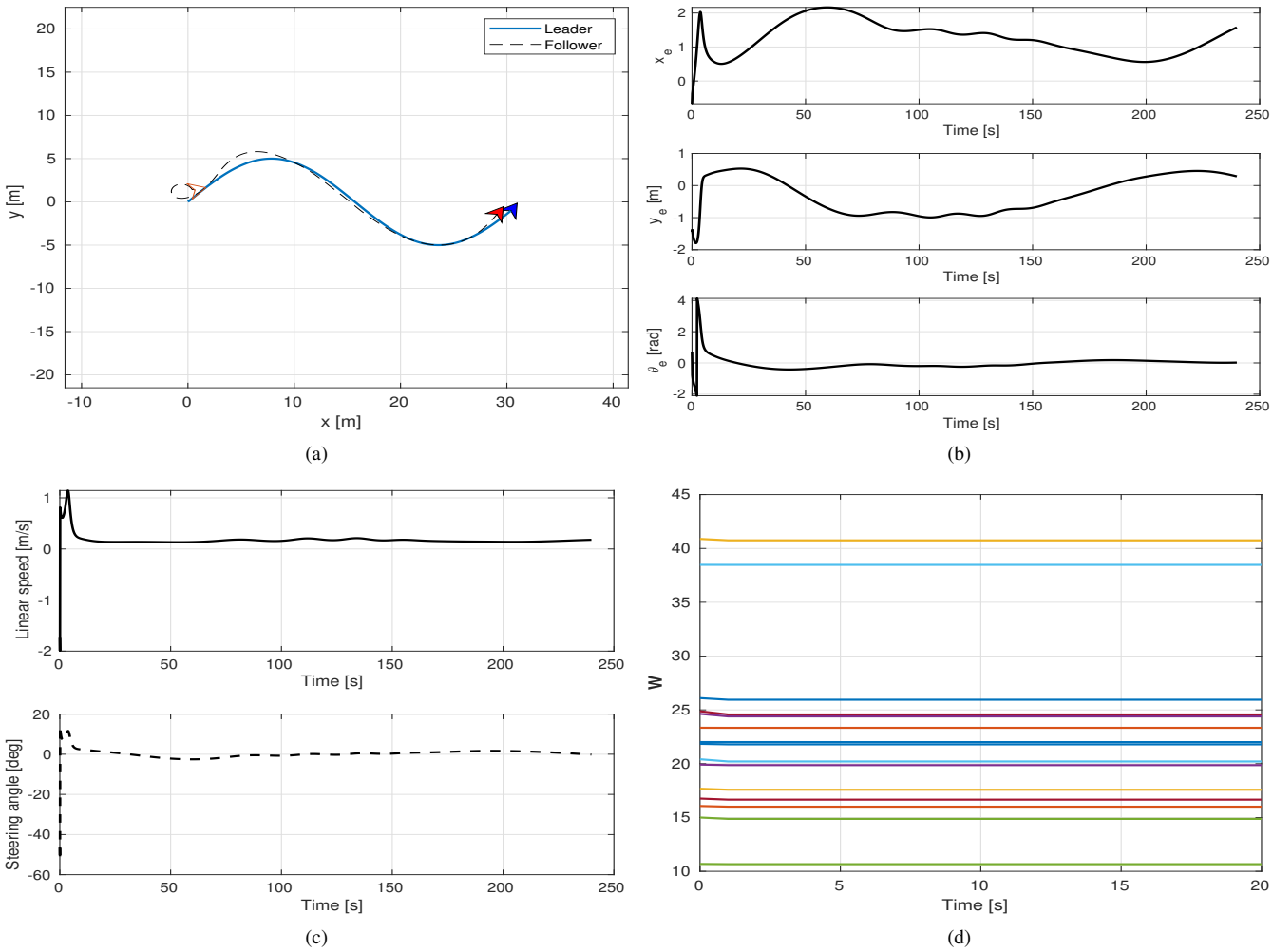

Fig. 5. Second scenario (sinusoidal trajectory): (a) leader-follower trajectories, (b) state tracking errors, (c) linear speed and steering angle of the follower, and (d) critic weights.