A Novel Model-Free Learning Approach for Dynamic Target Tracking Problem and its Validation using Virtual Robot Experimentation Platform

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

***Abstract*—Addressing the trajectory tracking problem of a mobile robot in tracking a dynamic target is still one of the challenging problems in the field of robotics. In this paper, we address the position tracking problem of a mobile robot where it is supposed to track the position of an mobile target whose dynamics is unknown a priori. This problem is even more challenging when the dynamics of the mobile robot is also assumed to be unknown, which is a basically a realistic assumption. Most of trajectory tracking problems proposed in the literature in the field of mobile robotics are either focused on algorithms that rely on mathematical models of the robots or driven by a overwhelming degree of computational complexity. This paper proposes a model-free actor-critic reinforcement learning strategy to determine appropriate actuator commands for the robot to track the position trajectory (unknown a priori) of the target. We emphasize that mathematical models of both mobile robot and the target are not required in the current approach. Moreover, Bellman's principle of optimality is utilized to minimize the energy required for the robot to track the target. The performance of the proposed actor-critic reinforcement learning approach is backed by a set of computer simulations with various complexities using a virtual circular-shaped mobile robot and a point target modeled by an integrator.**

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

I. Introduction

Need to discuss in details the limitations of previous work, we can also add RL applications to robotics navigation

Tracking a random moving target using a mobile robot, for instance, is a challenging task. This is mainly due to their inherent complex nonlinear dynamics dynamics. Most of the target tracking algorithms proposed in the literature either rely on A) complex mathematical models, B) simplified mathematical models, and C) nonlinear control techniques or driven by large amount of mostly offline data that leads to an overwhelming degree of computational complexity.

Over the past years mobile robots has been used in several applications in commercial and military sectors such as surveillance, search and rescue missions,coverage optimization, cooperative localization and dynamic target tracking tasks. In all of these mentioned applications a fleet of robot i.e agents interact with each other to achieve a certain goal. Leader-Follower or dynamic target tracking problem has recieved an extensive amount of study and research in the world of cooperative control theory due to it's wide promising applications such as search and rescure missions, wildlife monitoring and survillance to name a few . In a typical leader follower formation a number of robots referred to as followers apply local control actions to follow leader's robots in a specific predefined path such as cyclic, circular motion and time varying communication topologies. limitations of other methods many control methods were proposed however they have had the following limitations:
(1) a static leader/target is used.
(2) the leade/target's location or dynamic model is predetermined.
(3) expensive hardware platforms are used to achieve the mobile target tracking.
The main contribution of this work is the development of a model-free learning approach to control the follower robot by which it overcome many of the limitations mentiond above as it does not rely on any prior information of the mathmetical model or the dynamic model. The steering angle and the linear speed of the follower robot are determined by collecting the position and orientation of both the leader and the follower. This set of information is gathered online over a finite period of time. The optimal control actions are then generated by utilizing Bellman's principal of optimality which acts as model free-reinfocment learning approach that allows the follower robot to follow the path of the leader while oviding collision by maintaining a safe distance. In this paper the proposed algorithm is further validated using a commercially available robot simulator, CoppeliaSim. This paper acts as first milestone in generalizing the algorithm to solve more sophisticated problem such as coverage and mapping. cite area coverage papers
The rest of the paper is orgnized as follows. Section II lays down the problem setting of the leader follower problem and mathemetical models of the robots and the state error. The model free actor-critic reinforcment learning and its key steps are described in section III.Section IV illustrates computer simulations for different secnarios that reflects the effectivness of the proposed method followed by conlusion and future work presented in section V.

Take some references from the following paragraphs

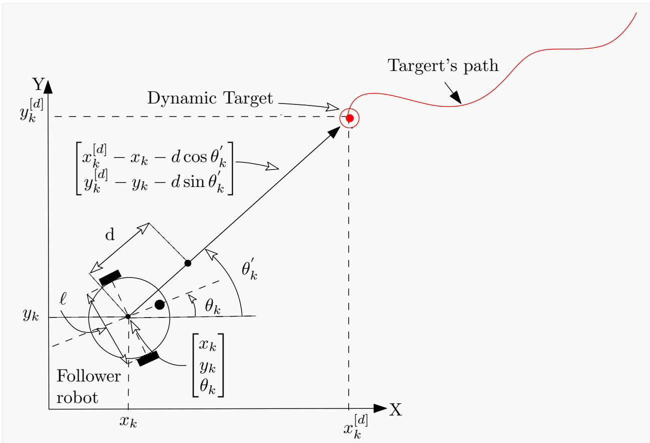

Fig. 1. Mobile robot and its dynamic target tracking problem setup.

Need to be rewritten some parts of [?]

II. Dynamic target tracking and problem setting

Needs to reviewed

Suppose that a wheeled mobile robot (follower) with coordinate (x, y) and orientation $\theta \in [-\pi, \pi)$ rad with respect to the global X-Y coordinate has a linear speed ν and steering angle γ which are considered as the two control actions. Let the posistion and orientation (pose) of the target i.e leader be $\mathbf{q}_k \equiv [x_k, y_k, \theta_k]$. The robot and the target are deployed in a 2D configuration in which the Follower is to follow the target trajectory defined by $(\mathbf{p}_k^{[d]})^T = [x_k^{[d]}, y_k^{[d]}(t)]$ at time $t \geq 0$ with $t = k\,T_s$, $k \in \mathbb{N}_0$, and $T_s > 0$ being the sampling time. Note that this trajectory is completley random. The target motion is modeled by the following discrete-time system:

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

with $\mathbf{u}_k^{[\ell]} \in \mathbb{R}^2$ being the control inputs of the target movment. The follower robot is to maintain a constant safe-distance $d > 0$ from target. The dynamic car-like model of the follower robot is defined as follows:

$$x_{k+1} = x_k + T_s \nu_k \cos(\theta_k + \gamma_k) + \zeta_1, \tag{2a}$$
$$y_{k+1} = y_k + T_s \nu_k \sin(\theta_k + \gamma_k) + \zeta_2, \tag{2b}$$
$$\theta_{k+1} = \theta_k + T_s \nu_k \frac{\sin(\gamma_k)}{l} + \zeta_3, \tag{2c}$$

Note that the follower dynamic model obeys Ackermann steering principle 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. We then define the error vector as:

$$\mathbf{e}_k^T = [\rho_e, \theta_e]^T = [\sqrt{x_e^2 + y_e^2}, \theta_e]^T = \left[\sqrt{(x_k^{[d]} - x_k - d\cos\theta_k^{'})^2 + (y_k^{[d]} - y_k - d\sin\theta_k^{'})^2}, \theta_k^{'} - \theta_k\right], \tag{3}$$

where $\theta_k^{'} = \operatorname{atan2}\left(y_k^{[d]} - y_k, x_k^{[d]} - x_k\right)$. and ρ_e being the euclidean distance.

The control problem can then be formally stated as follows: Find ν_k and γ_k such that $\mathbf{e}_k \rightarrow \mathbf{0}$ as $k \rightarrow \infty$ subject to (**??**) and (**??**).

III. Proposed actor-critic RL Approach

Needs to be rewritten

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 (**??**). 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 $\mathbf{Q} \in \mathbb{R}^{2\times 2}$ 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 (**??**) and (**??**). Motivated by the structure of the convex quadratic cost functional (**??**), 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 $\mathbf{z}_k = [\mathbf{e}_k, \mathbf{u}_k]^T \in \mathbb{R}^4$, $V(\mathbf{e}_k, \mathbf{u}_k) \approx \hat{V}(\mathbf{z}_k)$, and $\bar{\mathbf{P}}$ is a symmetric block-diagonal matrix formed using

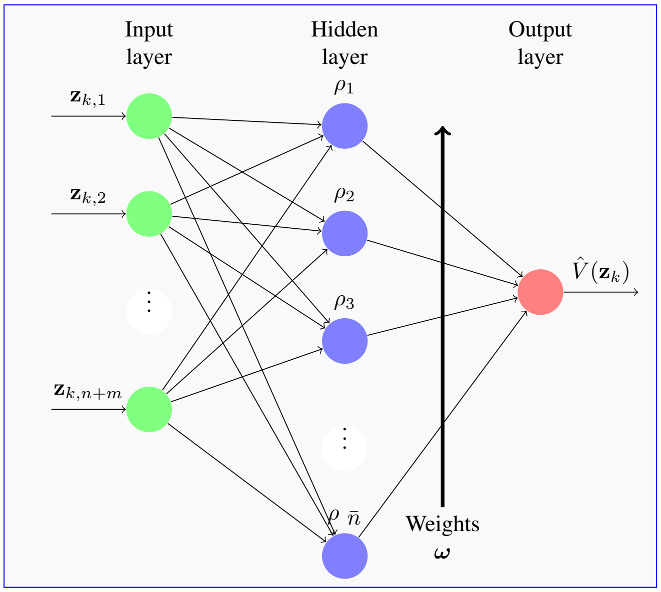

Fig. 2. Critic neural network structure for approximating value function.

$(\mathbf{Q}, \mathbf{R})$, *i.e.,* $\bar{\mathbf{P}} = \text{blockdiag}(\mathbf{Q}, \mathbf{R})$. 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 $\mathbf{P} \in \mathbb{R}^{4\times4}$ is a positive definite matrix. Hence, the optimal control strategy $\mathbf{u}_k^*$ can be expressed as follows

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

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

The following paragraph and the NN structure needs to be rewritten

A two-step solution mechanism that is based on policy iteration is employed to solve the temporal difference equation (**??**) using the policy (**??**). 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. **??**.

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.

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

This equation is utilized repeatedly in order to evaluate a certain policy during at least $\eta \geq \bar{n}, \bar{n} = (2+2)(2+2+1)/2$ evaluation steps (i.e., the lowest evaluation interval spans k to $k + \bar{n}$ calculation samples) in order to update the critic weights vector $\boldsymbol{\omega} = \text{vec}(\mathbf{P})$, which consists of connection weights between the neurons of the hidden layer and the output layer of the critic neural network shown in Fig. (**??**). The operator $\text{vec}(\mathbf{P})$ 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 (**??**) 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 $\tilde{\boldsymbol{\rho}}(\mathbf{z}_{k,k+1}) = \boldsymbol{\rho}(\mathbf{z}_k) - \boldsymbol{\rho}(\mathbf{z}_{k+1}) \in \mathbb{R}^{10\times1}$, $\boldsymbol{\rho}(\mathbf{z}_k) = (\mathbf{z}_k^q \otimes \mathbf{z}_k^h)$ $(q = 1,\ldots,5,\ h = q,\ldots,5)$, and $\boldsymbol{\omega}^T = [0.5\,P^{11}, P^{12}, P^{13}, P^{14},\ 0.5\,P^{22}, P^{23}, P^{24}, 0.5\,P^{33}, P^{34},\ 0.5\ \ P^{44},]^T \in \mathbb{R}^{1\times15}$ (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 $\boldsymbol{\Lambda} = [\mathbf{o}_0, \mathbf{o}_1, \ldots, \mathbf{o}_{\eta-1}]^T \in \mathbb{R}^{\eta\times15}$ with $\mathbf{o}_\kappa = \tilde{\boldsymbol{\rho}}^T(\mathbf{z}_{k+\kappa,k+\kappa+1}) \in \mathbb{R}^{1\times15}$ 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

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

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

$$\mathbf{P} = \begin{bmatrix} 2\,\omega^1 & \omega^2 & \omega^3 & \omega^4 \\ \omega^2 & 2\,\omega^5 & \omega^6 & \omega^7 \\ \omega^3 & \omega^6 & 2\,\omega^8 & \omega^9 \\ \omega^4 & \omega^7 & \omega^9 & 2\,\omega^{10} \end{bmatrix} \in \mathbb{R}^{4\times4},$$

where ω^i is the i^{th} entry of the weight vector $\boldsymbol{\omega}$.

merge the actor weights discussion with the rest

The critic wegihts are then used to update the actor weights which maps the state error to the desired plicy (control actions) at every time step t with the following equation:

$$\mathbf{u}_k = \omega_{\mathbf{a}}^{\mathbf{T}}\mathbf{e}_k \tag{9}$$

Applying a similar gradient descent approach as with the critic weights the actor weights are updated as follows:

$$\boldsymbol{\omega}_{\mathbf{a}}^{[r+1]} = \boldsymbol{\omega}^{[r]} - \ell_a(\mathbf{u}_k - \mathbf{u}_k^*)\mathbf{e}_k, \tag{10}$$

where ℓ is the actor learning rate.
The complete policy iteration solution process for the leader-follower problem is detailed out in Algorithm **??**.

Algorithm 1: Model-free actor-critic 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 (**??**)
  Compute follower's input $\mathbf{u}_0^{[0]}$ using policy (**??**)
  repeat `/* Main timing loop */`
    Find $\mathbf{e}_{k+1}$ using (**??**)
    Compute policy $\mathbf{u}_{k+1}^{[r]}$ using (**??**)
    if $[(k+1)$ modulo $\eta] == 0$ **then**
      $r \leftarrow r+1$ `/* Evaluate policy */` using (**??**)
      Solve for the critic-weights $\boldsymbol{\omega_c}$ using (**??**)eights

    Solve for the actor-weights $\boldsymbol{\omega_a}$ using (**??**)

    Construct matrix $\mathbf{P}^{[r]}$ using vector $\boldsymbol{\omega_c}$
  until *Tracking errors are zero*
  $k \leftarrow k+1$

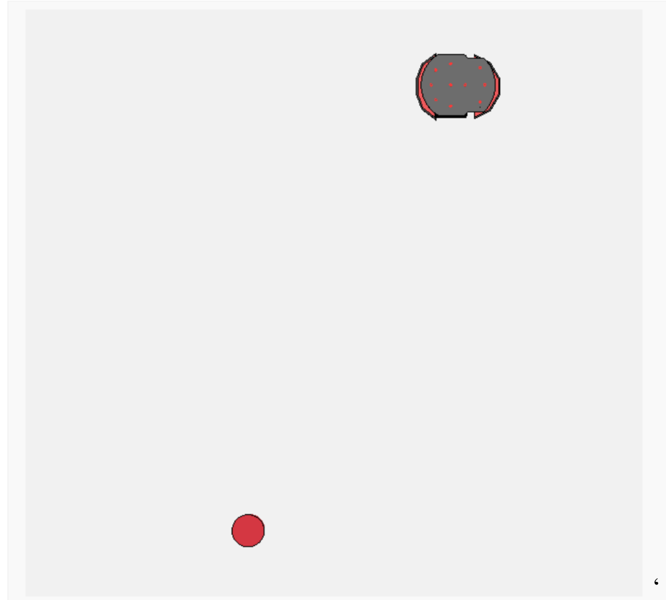

Fig. 3. CoppeliaSim Scene.

IV. Computer Experiments and Results

This section adopts the theoritical results discussed in the previous section by simulating the Algorithm using the Pioneer 3-DX in CoppeliaSim. The results obtained by the authors in cite flairs paper here are validated using the commercial robot simulator, CoppeliaSim as a prelminary step to implement the Algorithm experimantally in real world. we present the dynamics of the tracking error and the convergence characterisitcs of the actor and critic weights. The weighting matrices are set to $\mathbf{Q} = \mathrm{diag}[0.001, 0.001]$ and $\mathbf{R} = \mathrm{diag}[10^{-5}, 10^{-5}]$.

$$Q = 0.001 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, R = 0.00001 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

The actor and critic learning rates ℓ_c are set to 0.01 and 0.0001. The sampling time T_s is set to $0.001\,\mathrm{sec}$. The desired distance offset between the leader and the follower is set to $d = 0.5$ [m] for all scenarios. The CoppeliaSim simulator accuratly mimic what would happen in a realworld scenario following idustry standatrds and guidlines. The Pioneer robot model and the was initally placed at position (-1,-2.5) m.The Target modeled as a cylinder was initially positioned on a random position around the origin (0,0). The simulation was run for

V. Conclusion

an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks …". Instead, try "R. B. G. thanks…". Put sponsor acknowledgments in the unnumbered footnote on the first page.

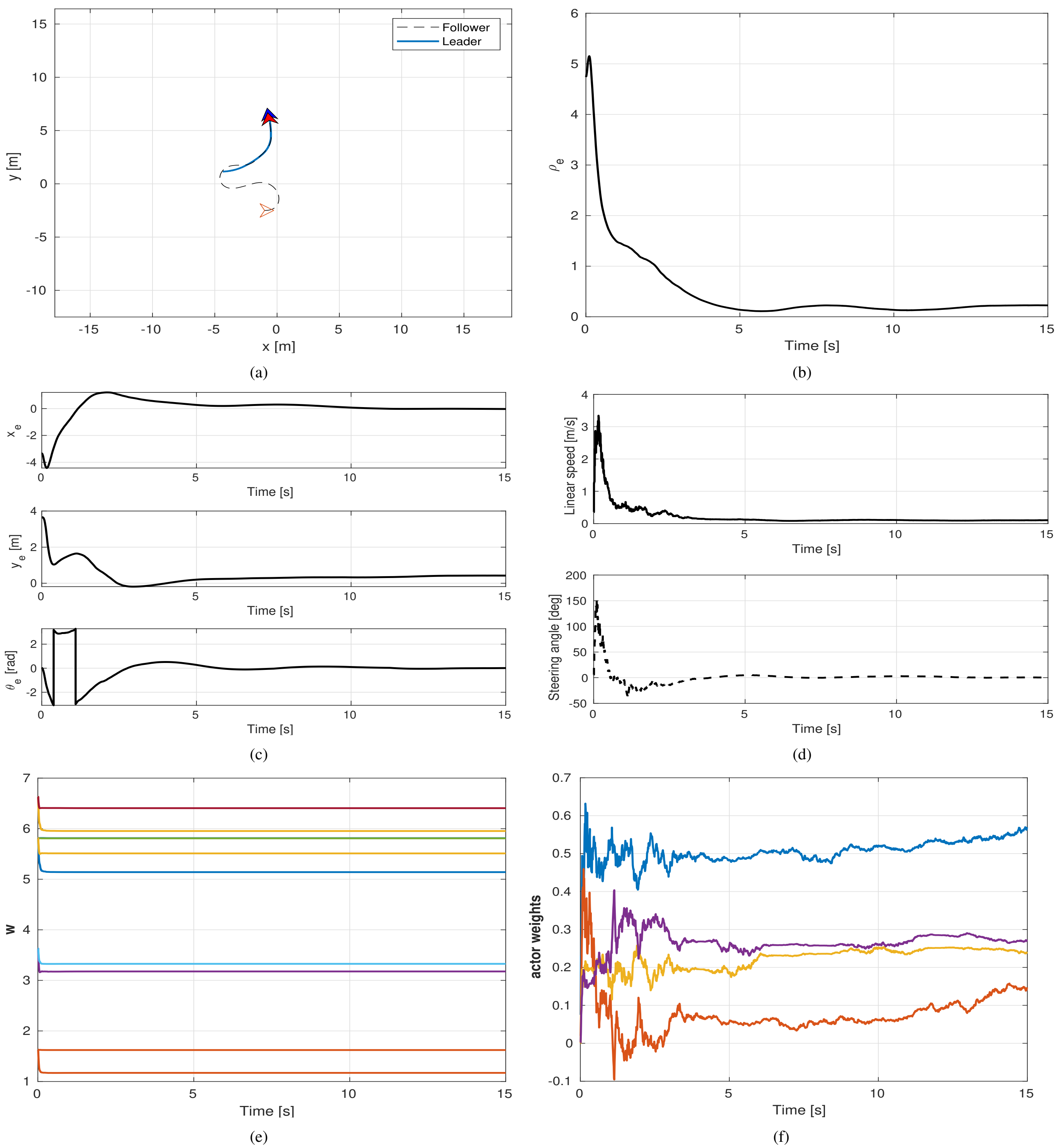

Fig. 4. Performance in tracking random trajectory: **??** leader-follower trajectories, **??** state error, **??** linear speed and steering angle of the follower; and **??** learning weights.