New title

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**

Finalize this section.

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. 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 to name a few [1],[2],[3]. 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 promising applications in research and industry. In a typical target tracking formation a number of robots referred to as followers apply local control actions to follow a target in a specific predefined path such as cyclic, circular motion and time varying communication topologies. 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 avoiding 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.

II. Problem Setting

Fig. 1 shows the setup of the problem that we address in this manuscript. The robot is characterized by its 2D position

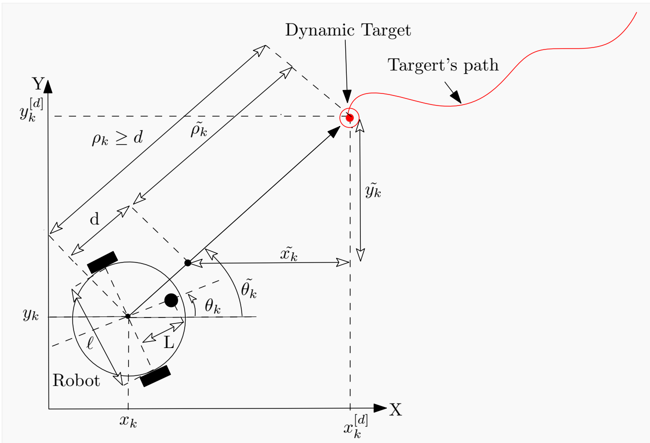


Fig. 1. Mobile robot and its dynamic target tracking problem setup.

(x_k, y_k) and orientation $\theta_k \in [-\pi, \pi)$ with respect to global coordinate frame X-Y at discrete time index $k = 0, 1, \ldots,$ where time $t = kT$ with $T > 0$ being the sampling time. The time-varying position of the target is denoted by $\mathbf{p}_k^{[d]} = [x_k^{[d]}, y_k^{[d]}]^T \in \mathbb{R}^2$. Let the position and orientation (pose) of the robot is denoted by the vector $\mathbf{q}_k^T \equiv [x_k, y_k, \theta_k] \in \mathbb{R}^2 \times \mathbb{S}^1$. Without loss of generality, assume that the robot tracks the target's trajectory while maintaining line-of-sight distance (also known as safe distance between the robot and the target) $d > 0$. We emphasize that the mathematical models of both robot and the target are not required in the proposed control technology. For illustration, let us assume that the robot (rear-wheel drive) follows the discrete-time kinematic model described by

$$x_{k+1} = x_k + T\nu_k \cos\theta_k \tag{1a}$$
$$y_{k+1} = y_k + T\nu_k \sin\theta_k \tag{1b}$$
$$\theta_{k+1} = \theta_k + T\frac{\nu_k}{L}\tan\gamma_k \tag{1c}$$

where $\gamma_k \in (-\frac{\pi}{2}, \frac{\pi}{2})$ is the steering angle of the font castor of the robot (see Fig. 1), and $\nu_k \in \mathbb{R}$ is the linear speed. Without loss of generality, this work considers a target modeled by an integrator and its discrete-time kinematic model is described by

$$\mathbf{p}_{k+1}^{[d]} = \mathbf{p}_k^{[d]} + T_s\, \mathbf{u}_k^{[d]}, \tag{2}$$

with $\mathbf{u}_k^{[d]} \in \mathbb{R}^2$ being the control input vector (velocity) that defines the trajectory (considered random in this work) of the target. In case, the robot modeled by (1a) is described by a differential drive mobile robot, then its left-wheel velocity $\nu_{L,k}$ and right-wheel velocity $(\nu_{R,k})$ are related as

$$\nu_k = \frac{1}{2}(\nu_{R,k} + \nu_{L,k}), \tag{3a}$$
$$\omega_k = \frac{1}{\ell}(\nu_{R,k} - \nu_{L,k}) = \frac{\nu_k}{L}\tan(\gamma_k) \tag{3b}$$

where $\omega_k = \frac{\nu_k}{L}\tan\gamma_k$ is the angular velocity of the robot's center (x_k, y_k). Let us define the error vector as

$$\mathbf{e}_k = [\tilde{\rho}_k, \tilde{\theta}_k]^T = [\sqrt{\tilde{x}_k^2 + \tilde{y}_k^2}, \tilde{\theta}_k]^T \tag{4}$$

where $\theta_k^{'} = \text{atan2}\left(y_k^{[d]} - y_k, x_k^{[d]} - x_k\right)$ and $\rho_k = \sqrt{\tilde{x}_k^2 + \tilde{y}_k^2}$ is the Euclidean distance at time instant k as shown in Fig. 1 with $\tilde{x}_k = x_k^{[d]} - x_k - d\cos\theta_k^{'}$, $\tilde{y}_k = y_k^{[d]} - y_k - d\sin\theta_k^{'}$, and $\tilde{\theta}_k = \theta_k^{'} - \theta_k$. 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 (1a) and (2).

III. Model-Free Reinforcement Learning Approach

Reinforcement learning is a promising machine learning technique that is widely used in solving problems of many cyber-physical systems that pertain to the goal-directed learning from interaction [4]. In this paper, tracking a mobile target using a wheeled mobile robot is formulated as a reinforcement learning problem, where the goal of the robot is to track the trajectory of the target while maintaining the safe-distance $d > 0$. The RL solution employed in the current problem leverages Bellman's principle of optimality to decide on the appropriate control actions (actuator commands) for the robot to track the position trajectory of the target. The relative importance of the states in the error vector $\mathbf{e}_k$ and the control decisions (linear velocity ν_k and steering angle γ_k) of the robot are evaluated using the performance (cost) index:

$$J = \sum_{k=0}^{\infty} \frac{1}{2}\left[\mathbf{e}_k^T\, \mathbf{Q}\, \mathbf{e}_k + \mathbf{u}_k^T\, \mathbf{R}\, \mathbf{u_k}\right], \tag{5}$$

where $\mathbf{Q} \in \mathbb{R}^{2\times2}$ and $\mathbf{R} \in \mathbb{R}^{2\times2}$ are symmetric positive definite weighting matrices, and $\mathbf{u}_k^T = [\nu_k, \gamma_k]$ is the control input vector. The objective of the optimization problem, following [5], 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 (2) and (1a). Motivated by the structure of the convex quadratic cost functional (5), 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

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

Following Bellman's optimality principle yields the optimal control policies $\mathbf{u}_k^*$, $k \geq 0$, such that [6]

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

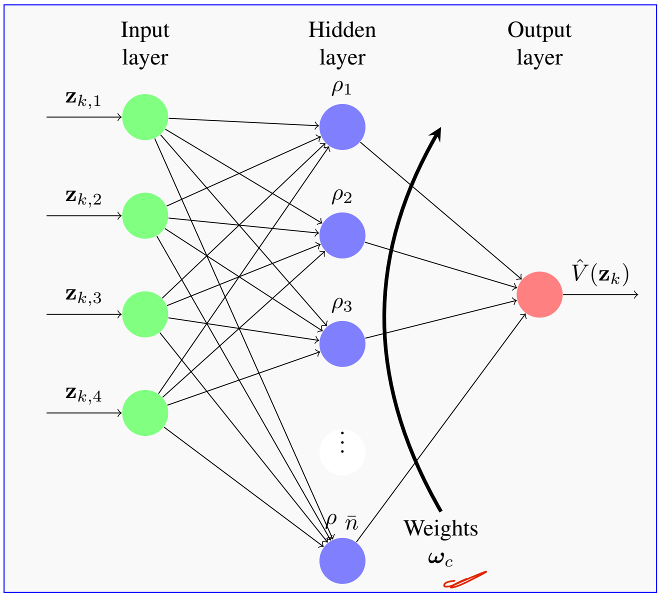


Fig. 2. Multi-layer critic neural network for approximating value function $V(\mathbf{z}_k)$.

Alternatively, this optimal policy form is equivalent to $\mathbf{u}_k^* = \text{argmin}_{\mathbf{u}_k} [V(\mathbf{e}_k, \mathbf{u}_k)]$. 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

$$V(\mathbf{z}_k) = \frac{1}{2}\mathbf{z}_k^T\,\bar{\mathbf{P}}\,\mathbf{z}_k + V(\mathbf{z}_{k+1}), \tag{6}$$

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 $(\mathbf{Q}, \mathbf{R})$, *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 $\hat{V}(\mathbf{z}_k) = \frac{1}{2}\mathbf{z}_k^T\,\mathbf{P}\,\mathbf{z}_k$, for some positive definite matrix $\mathbf{P} \in \mathbb{R}^{4\times4}$. The optimal control input $\mathbf{u}_k*$ is then determined by setting $\frac{\partial\hat{V}(\mathbf{z}_k)}{\partial\mathbf{u}_k} = 0$, which yields

$$\mathbf{u}_k^* = -\,\mathbf{P}_{uu}^{-1}\,\mathbf{P}_{ue}\,\mathbf{e}_k, \tag{7}$$

where $\mathbf{P}_{uu}$ and $\mathbf{P}_{ue}$ are sub-blocks of symmetric matrix $\mathbf{P}$. The proposed approach approximates the block matrix $\mathbf{P}$ by following a two-step solution mechanism based on policy iteration method [4], which in turn approximates the value function $\hat{V}(\cdot)$ given by (6). First, a multi-layer critic neural network shown in Fig. 2 is used to estimate the value function, $\hat{V}(\mathbf{z}_k)$. Second, the policy evaluation step that follows is utilized to update the critic weights $\boldsymbol{\omega}_c$ in real-time regardless of the mathematical models of the robot and the target. This step is aimed to search for a strictly better policy using the updated critic weights $\boldsymbol{\omega}_c$. Note that $\hat{V}(\mathbf{z}_k)$ can also be written using temporal difference form as $\hat{V}(\mathbf{z}_k) = \mathbf{z}_k^T\,\bar{\mathbf{P}}\,\mathbf{z}_k + \hat{V}(\mathbf{z}_{k+1})$, which is described by

$$\mathbf{z}_k^T\,\mathbf{P}\,\mathbf{z}_k - \mathbf{z}_{k+1}^T\,\mathbf{P}\,\mathbf{z}_{k+1} = \mathbf{z}_k^T\,\bar{\mathbf{P}}\,\mathbf{z}_k. \tag{8}$$

Equation (8) is evaluated repeatedly to collect data online for at least $\eta \geq \bar{n}, \bar{n} = (2+2)(2+2+1)/2 = 10$ evaluation steps due to the fact that the matrix $\mathbf{P}$ is positive definite and symmetric. The strictly better policy $\mathbf{P}$, is determined every η time/evaluation steps. To find the better policy in terms of matrix $\mathbf{P}$, let us define the vector $\boldsymbol{\omega} = \text{vec}(\mathbf{P}) \in \mathbb{R}^{16}$, which is formed by stacking columns of $\mathbf{P}$ matrix. However, since $\mathbf{P}$ is symmetric, we only need determine the weight $\boldsymbol{\omega}_c \in \mathbb{R}^{\bar{n}}$ using the critic neural network shown in Fig. (2). The left hand side of (8) is expressed using the following critic approximation form

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

where $\tilde{\boldsymbol{\rho}}(\mathbf{z}_{k,k+1}) = \boldsymbol{\rho}(\mathbf{z}_k) - \boldsymbol{\rho}(\mathbf{z}_{k+1}) \in \mathbb{R}^{10\times1}$, $\boldsymbol{\rho}(\mathbf{z}_k) = (\mathbf{z}_k \otimes \mathbf{z}_k)$, which is a Kronecker product that generates the basis function vector with components $z_{k,1}^2$, $z_{k,1}z_{k,2}$, $z_{k,1}z_{k,3}$, $z_{k,1}z_{k,4}$, $z_{k,2}^2$, $z_{k,2}z_{k,3}$, $z_{k,2}z_{k,4}$, $z_{k,3}^2$, $z_{k,3}z_{k,4}$, and $z_{k,4}^2$. Note that the critic weight $\boldsymbol{\omega}_c$ is related to the policy matrix $\mathbf{P}$ as $\boldsymbol{\omega}_c^T = [0.5\,P^{11}, P^{12}, P^{13}, P^{14}, 0.5\,P^{22}, P^{23}, P^{24}, 0.5\,P^{33}, P^{34}, 0.5\;P^{44}]^T \in \mathbb{R}^{1\times10}$ with P^{ij} being the ij^{th} entry of matrix $\mathbf{P}$. Here, the critic weights $\boldsymbol{\omega}_c$ are updated using a gradient descent update rule. The tuning error ε_k at each computational instance k is given by $\varepsilon_k = \boldsymbol{\omega}_c^T\,\tilde{\boldsymbol{\rho}}(\mathbf{z}_{k,k+1}) - v_k$, where $v_\kappa = \frac{1}{2}\mathbf{z}_k^T\,\bar{\mathbf{P}}\,\mathbf{z}_k$. The error data ε_k is collected online for at least $\eta \geq \bar{n}$ evaluation steps before updating the critic weights $\boldsymbol{\omega}_c$ that is used to determine the improved policy $\mathbf{P}$. The sum of squared tuning errors is given by

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

where $\boldsymbol{\Lambda} = [\mathbf{o}_0, \mathbf{o}_1, \ldots, \mathbf{o}_{\eta-1}]^T \in \mathbb{R}^{\eta\times10}$ with $\mathbf{o}_\kappa = \tilde{\boldsymbol{\rho}}^T(\mathbf{z}_{k+\kappa,k+\kappa+1}) \in \mathbb{R}^{1\times10}$ 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 $\kappa = 0, 1, \ldots, \eta-1$. The sum of squared error δ_c can be minimized by adapting the critic weight using the gradient decent approach for at least $\bar{n}$ data samples as

$$\begin{aligned}\boldsymbol{\omega}_c^{[r+1]} &= \boldsymbol{\omega}_c^{[r]} - \ell_c\frac{\partial\delta_c}{\partial\boldsymbol{\omega}} = \boldsymbol{\omega}_c^{[r]} - \ell_c\left(-\boldsymbol{\Lambda}^T\mathbf{v} + \boldsymbol{\Lambda}^T\boldsymbol{\Lambda}\boldsymbol{\omega}_c^{[r]}\right) \\ &= \boldsymbol{\omega}_c^{[r]} - \ell_c\boldsymbol{\Lambda}^T\left(\boldsymbol{\Lambda}\boldsymbol{\omega}_c^{[r]} - \mathbf{v}\right),\end{aligned} \tag{9}$$

where $0 < \ell_c < 1$ is a critic learning rate and r is known as iteration index for updating the policy. The updated critic weights can be used to reconstruct the policy matrix $\mathbf{P}$ as

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

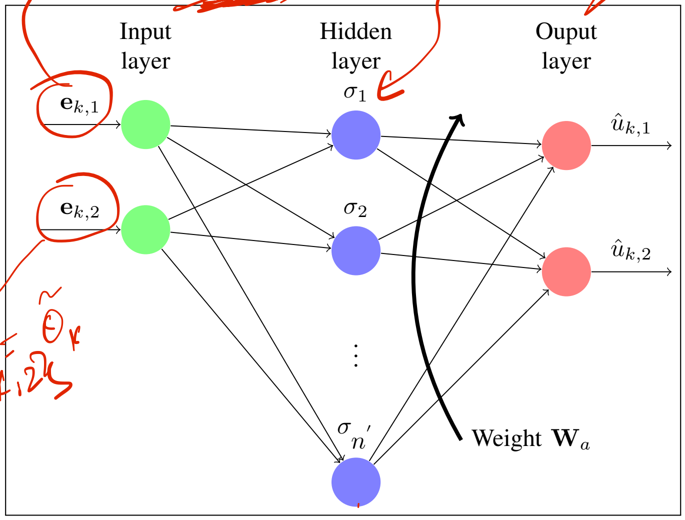
Update with ω_c

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

$e_{\{k,1\}} = \tilde{\rho}_k$

$\sigma_1 = e_1$, $\sigma_2 = e_2$

$\sigma(e_k) = e_1^2,\ e_1 e_2,\ e_2^2$



$e_{\{k,2\}} = \tilde{\theta}_k$

Fig. 3. Actor neural network structure for approximating control input.

The critic weights are then used to update the actor weights $W_a\mathbb{R}^{n\times m}$ (here $n = 2$ and $m = 2$) which maps the state error to the desired policy (control actions) as at every time step t with the following equation:

$$\hat{\mathbf{u}}_k = \mathbf{W}_a^T \boldsymbol{\sigma}(\mathbf{e}_k), \tag{10}$$

where $\mathbf{W}_a \in \mathbb{R}^{2\times n'}$ is the actor weight matrix and $\boldsymbol{\sigma} : \mathbb{R}^2 \rightarrow \mathbb{R}^{n'}$ is the basis function vector. Equation (10) is realized using a multi-layer actor neural network shown in Fig. 3. The elements of the vector $\boldsymbol{\sigma}$ are the activation functions of the neurons in the hidden layer shown in Fig. 3. We will need to determine the actor weight matrix $\mathbf{W}_a$ that minimizes the error between the estimated control inputs determined by model (10) and the that determined by critic neural network approximation model (7). For that, let us define the error

$$\delta_a = \frac{1}{2}\|\hat{\mathbf{u}}_k - \mathbf{u}_k^*\|^2, \tag{11}$$

which can be written as

$$\delta_a = \frac{1}{2}\left(\mathbf{W}_a^T\boldsymbol{\sigma}(\mathbf{e}_k) - \mathbf{u}_k^*\right)^T\left(\mathbf{W}_a^T\boldsymbol{\sigma}(\mathbf{e}_k) - \mathbf{u}_k^*\right).$$

Taking partial derivative of the above expression δ_a with respect to $\mathbf{W}_a$ gives

$$\frac{\partial\delta_a}{\partial\mathbf{W}_a} = \mathbf{W}_a^T\boldsymbol{\sigma}(\mathbf{e}_k)\boldsymbol{\sigma}^T(\mathbf{e}_k) - \mathbf{u}_k^*\boldsymbol{\sigma}^T(\mathbf{e}_k).$$

Therefore, actor weight matrix $\mathbf{W}_a$ in every policy iteration $r = 01, 2, \ldots$ can be updated using the learning rule

$$\begin{aligned}\mathbf{W}_a^{[r+1]} &= \mathbf{W}_a^{[r]} - \ell_a\frac{\partial\delta_a}{\partial\mathbf{W}_a}\\ &= \mathbf{W}_a^{[r]} - \ell_a\left(\mathbf{W}_a^T\boldsymbol{\sigma}(\mathbf{e}_k) - \mathbf{u}_k^*\right)\boldsymbol{\sigma}^T(\mathbf{e}_k),\end{aligned} \tag{12}$$

where ℓ_a is the actor learning rate. The complete policy iteration solution process for the leader-follower problem is detailed out in Algorithm 1. For clarity, number of hidden neurons chosen in the actor neural network shown in Fig. 3 is

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 */
    η = (n + m)(n + m + 1)/2
    Initialize P^[0] /* Positive definite */
    Set offset distance d
    Given approximate initial poses of leader and
      follower, compute e_0 using error model (4)
    Compute follower's input u_0^[0] using policy (10)
    repeat /* Main timing loop */
        Find e_{k+1} using (4)
        Compute policy u_{k+1}^[r] using (10)
        if [(k + 1) modulo η] == 0 then
            r ← r + 1 /* Evaluate policy */ using (7)
            Solve for the critic-weights ω_c
              using (9)eights

        Solve for the actor-weights ω_a using (12)

        Construct matrix P^[r] using vector ω_c
    until Tracking errors are zero
    k ← k + 1
```

$n' = 2$ and the basis function vector is simply $\boldsymbol{\sigma}(\mathbf{e}_k) = \mathbf{e}_k$. In the following, we provide the computer simulation results to highlight the performance of the proposed model-free trajectory tracking approach for a mobile robot in tracking a random target while maintaining a geometric safe distance $d > 0$.

IV. Computer Experiments and Results

review

This section adopts the theoritical results discussed in the previous section by simulating the Algorithm using the Pioneer 3-DX robot. The proposed Actor-Critic RL algorithm is tested using the commercial robot simulator CoppeliaSim in integration with Matlab software as a prelminary step to implement the Algorithm experimantally in real world.Here we present the performance of the proposed method 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,ℓ_a are set to 0.0001 and 0.01 respectively. The sampling time T_s is set to 0.001 sec. The desired distance offset between the leader and the follower is set to $d = 0.2$ [m].

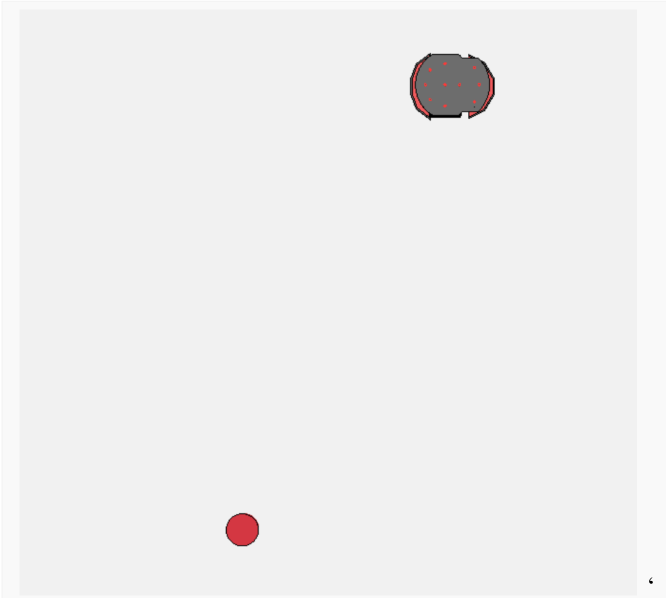


Fig. 4. Simulation Setup

During the simulation CoppeliaSim simulator accuratly mimics what would happen in a realworld scenario following idustry standatrds and guidlines.The setup of the virtual expirement is illustrated in figure 4 where pioneer robot model was initally placed at position $(x, y) = (-1, -2.5)$ [m] with an orientation $\theta = 0°$ .The Target which is modeled as a cylinder object was initially located at a random position around $(x, y) = (0, 0)$ [m]. The simulation was run for the duration of 15 seconds. The Target is set to move on a completely random trajectory starting from a random position. We intensify that the follower robot has no previous knoweledge of the dynamic model of the moving target nor it's initial position. The results obtained by the simulation are shown in figure 5. the control actions of the pioneer robot are determined by the random actor weights .we clearly see tracking error converges to the desired value of d = 0.2 [m] along with control signal converging to values that allows target tracking. the results reveal the ability of the proposed algorithm to sucessfully track the dynamic moving target and adapt to the changes in the target's path in an online manner.

V. Conclusion

review

A model free actor-critic reinfoncement learning algorithm for dynamic target tracking has been validated using virtual robot experimantation platform where a differenitial drive mobile robot and a randomly moving target are utilized in the simulation.Opposed to previous approaches in the litrature this Algorithm possess the advantage of being completlety model free and doesn't rely on any dynamic model to be known in priori. The actor weights which drives the control actions of the robot sucessfully converge to values that enable the robot to track the moving target through unplanned trajectory. To the authors knowlege this work is the first of its kind to employ a model-free actor-critic RL approach in the context of dynamic target tracking. In future work the proposed approach will be applied to multi-robot scenarios where a number of robots are to achieve a certain task in unknown enviroments.

References

[1] A. Kolling and S. Carpin, "Multirobot cooperation for surveillance of multiple moving targets - a new behavioral approach," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, 2006, pp. 1311–1316.

[2] P. Encarnacao and A. Pascoal, "Combined trajectory tracking and path following: an application to the coordinated control of autonomous marine craft," in *IEEE Conference on Decision and Control*, vol. 1, 2001, pp. 964–969.

[3] D.-Y. Ju and S. Kushida, "Intelligent control of mobile robot during autonomous inspection of welding damage based on genetic algorithm," in *International conference on Industrial and engineering applications of artificial intelligence and expert systems: engineering of intelligent system*, Budapest, Hungary, June 2001, pp. 661–669.

[4] R. Sutton, *Reinforcement learning : an introduction*. Cambridge, Massachusetts London, England: The MIT Press, 2020.

[5] F. L. Lewis and D. Liu, *Reinforcement learning and approximate dynamic programming for feedback control*. John Wiley & Sons, 2013, vol. 17.

[6] F. L. Lewis, D. L. Vrabie, and V. L. Syrmos, *Optimal Control*, 3rd ed. Hoboken, New Jersey: John Wiley and Sons, 2012.

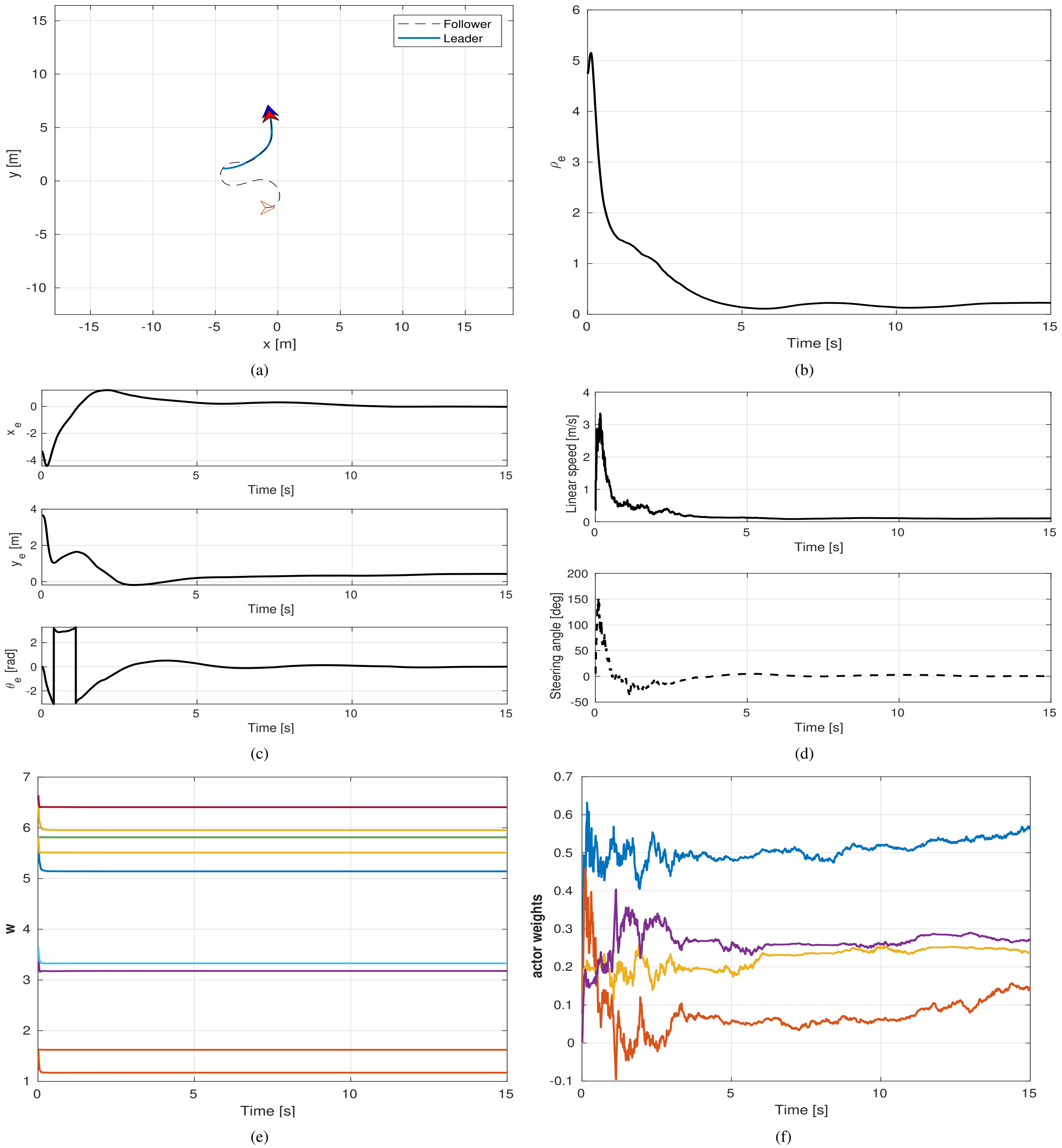


Fig. 5. Performance in tracking random trajectory: (a) leader-follower trajectories, (c) state error, (d) linear speed and steering angle of the follower; and (e) learning weights.