ITC 3/50
Information Technology and Control
Vol. 50 / No. 3 / 2021
pp. 507-521
DOI 10.5755/j01.itc.50.3.25979

Application of Deep Reinforcement Learning Tracking Control of 3WD Omnidirectional Mobile Robot

Received 2020/05/02 | Accepted after revision 2021/05/20

http://dx.doi.org/10.5755/j01.itc.50.3.25979

HOW TO CITE: Mehmood, A., Shaikh, I. U. H., Ali, A. (2021). Application of Deep Reinforcement Learning for Tracking Control of 3WD Omnidirectional Mobile Robot. *Information Technology and Control*, 50(3), 507-521. https://doi.org/10.5755/j01.itc.50.3.25979

Application of Deep Reinforcement Learning Tracking Control of 3WD Omnidirectional Mobile Robot

Atif Mehmood
Department of Electrical Engineering; University of Engineering and Technology Taxila; Student, UET Taxila, Rawalpindi, Pakistan; e-mail: atif.mehmood@students.uettaxila.edu.pk

Inam ul Hasan Shaikh, Ahsan Ali
Faculty of Electrical Engineering; University of Engineering and Technology Taxila; Faculty: UET Taxila, Rawalpindi, Pakistan; e-mails: inam.hassan@uettaxila.edu.pk, ahsan.ali@uettaxila.edu.pk

Corresponding author: atif.mehmood@students.uettaxila.edu.pk

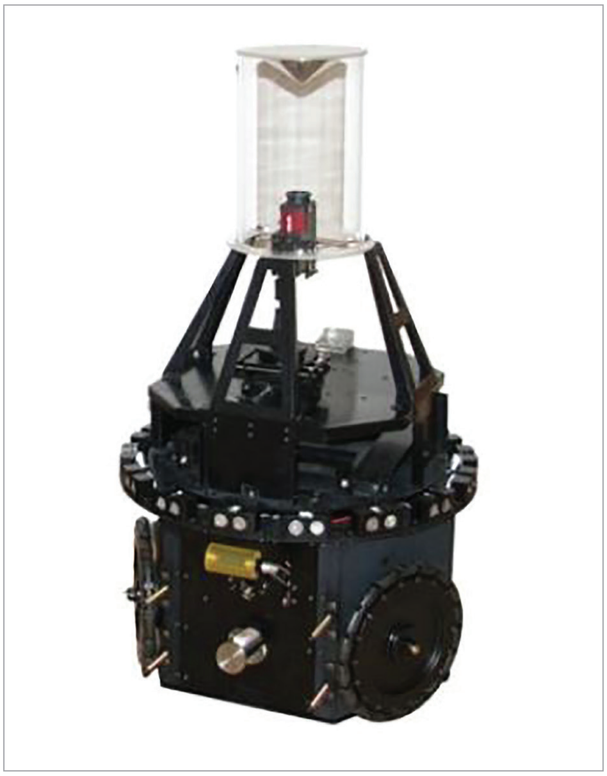
Deep reinforcement learning, the fastest growing technique, to solve real-world complex problems by creating a simple mathematical framework. It includes an agent, action, environment, and a reward. An agent will interact with the environment, takes an optimal action aiming to maximize the total reward. This paper proposes the compelling technique of deep deterministic policy gradient for solving the complex continuous action space of 3-wheeled omnidirectional mobile robots. Three-wheeled Omnidirectional mobile robots tracking is a difficult task because of the orientation of the wheels which makes it rotate around its own axis rather to follow the trajectory. A deep deterministic policy gradient (DDPG) algorithm has been designed to train in environments with continuous action space to follow the trajectory by training the neural networks defined for the policy and value function to maximize the reward function defined for the tracking of the trajectory. DDPG agent environment is created in the Reinforcement learning toolbox in MATLAB 2019 while for Actor and critic network design deep neural network designer is used. Results are shown to illustrate the effectiveness of the technique with a convergence of error approximately to zero.

KEYWORDS: 3WD-Omnidirectional mobile robot, Deep Reinforcement Learning (DRL), Deep Deterministic Policy Gradient (DDPG), Reinforcement Learning Toolbox (RL toolbox).

1. Introduction

Wheeled mobile robots have many advantages compared to their legged counterparts such as structural simplicity, energy efficiency, high locomotion speed, and low cost of manufacturing. One of the types of a wheeled mobile robot is holonomic wheeled mobile robots which can be designed to move in any direction without changing its orientation. These omnidirectional robots are made up of three or more Swedish wheels which can move not just forward or backward but sideways also A 3-wheel mobile robot is shown in Figure 1.

Figure 1
3WD-Omnidirectional Mobile Robot



The desired capability of an advanced robotic system is that of an adaptation of effective behavior while interacting with the dynamic environment [19]. The control hierarchy of wheeled mobile robots is often categorized as high-level and low-level. In high-level control, one of the three major control paradigms, (e.g. hierarchical, reactive and hybrid) are applied to undertake a motion task such as path following, point to point tracking, trajectory tracking, wall following, and obstacle avoidance [14]. The hierarchical control architecture requires a complete world model to plan an action based on sensor data. Due to its high computational requirements, the hierarchical control scheme is, however, slower to respond. The reactive control architecture does not have a planning stage. It executes an action based on the sensor data and hence it is quick in producing a response. The traditional method to control the movement of these robots is to apply classic controllers like PID using mathematical modeling of these robots and their inverse kinematics. But now reinforcement learning, artificial intelligence, and even deep learning are being used very commonly instead of the previous methods. As the robots experience many uncertainties in the real world, the traditional controllers experience difficulties. These uncertainties include fluctuations in the environment and goals. Reinforcement learning can be combined with deep learning to solve such complex problems with ease. Analogies between temporal difference (TD) reinforcement learning algorithms and dopaminergic neurons of the brain have demonstrated by recent studies in cognitive science. Despite nature-derived inspiration, many effective implementations of reinforcement learning (RL) for self-governing drive and movement controlling of dynamic robotic systems manipulation have proven the real-time application of previously theoretical concepts for the control of physical systems [3, 6-7]. Many of these methods use specific policy structures to represent policies to put a limitation on the number of iterations which is necessary for optimizing the results. Though efficient, but by adopting this approach there is a loss of generalization as it tightens the policy space to some specific trends [10]. To overcome these non-linear function approximators, neural networks are used for policy parameterization. This eliminates the need for handwritten specific policy representation and human supplied demonstrations to adjust them. Furthermore, usage of parameters in higher numbers also theoretically ensures learning of those complex behaviors that would not have been conceivable with linear handwritten policies.

In [4], partial reinforcement learning is used along with a neural network-based algorithm for the tracking of wheeled mobile robots to overcome the complexity of time-varying advance angle. Both actor-critic adaptive laws are defined by the gradient descent method and the Critic network was defined to maximize the long-term reward while actor-network is defined to minimize a long-term cost function. In [12], the problem of performance analysis of visual servo control of the robot is considered with measurements and modeling errors. A solution is proposed by coupling Q-learning and SARSA with the neural network. In [15], an actor-critic algorithm for PeopleBot robot is used to find and reach the table so it can pick up the things from it by using a camera mounted on it. The network is trained from random wandering to finding a table. In [23], reinforcement learning is used to learn the walking of an omnidirectional humanoid robot and design a controller for high-level push recovery. In [28], a deep reinforcement learning algorithm DDPG is implemented in continuous action space for a Mobile robot that uses a single network structure to learn all three skills: go to the ball, turn and shoot. The main drawback of this technique is that if the opponent learns to block the shot, then this will fail.

A reinforcement learning algorithm SARSA and Q learning are applied in [1] for robot navigation by discretizing the continuous state and actions. Discretization determines the performance of the algorithm applied. Q-Values in the algorithm are represented in tabular form which requires large memory spaces and difficult mathematical calculations. A deep reinforcement method is implemented in [8] for collision avoidance for an indoor service robot. The controller is parameterized using the neural network while DDPG is used to train the agent. It is proved in [9] that the decentralized planning outperforms its centralized counterpart in-terms of computational assets. The technique is confirmed on two problems: a lengthy version of the 3-dimensional mount car, and a ball-pushing act performed with a differential-drive robot, which is also verified on a physical setup.

In the last few years or so, deep learning made a great impact maybe this is due to the improvement in the computer technologies which are used to train these deep neural networks. For extracting useful information from visual data object detection and object classification techniques are used these techniques are convolutional neural networks (CNN) based. CNN is a subclass of deep learning, in which meaningful data is used to train models to learn patterns and make decisions. CNN based models are better able to detect and extract information from images, but there is a limitation of data and greater computation cost is required. In CNN some models are pre-trained and need to be trained, in pre-trained models the model is already trained on specific data. Small models that are pre-trained yield better results but in cases where models are huge a lot of computation is not focused on the original task, extra parameters are involved. To reduce the computation cost pruning parameter is proposed by Zheng et al. [25], the pruning method in CNN, reduces model parameters, accelerating its computation. Paper proposed a PAC-Bayesian framework that is based on drop-path, it works by identifying the important paths in the CNN model, it can work on multi-layer and multi-branch models resulting in improved performance and speed of the network.

CNN requires a large amount of data to learn features and due to the non-availability of large data techniques like data augmentation are used. Data augmentation is a process that increases the diversity of data without increasing the size of data using techniques like transformation, overfitting, underfitting and it helps to minimize overfitting problems in CNN. Data augmentation on joint training and testing stages can help in optimizing network performance. In CNN overfitting problems exists, to solve this Zheng et al. [26] proposed a full stage data augmentation framework, which can reduce model training cost, the framework has been tested on CIFAR-10 and CIFAR-100 and gives improved generalization. [27] introduced a novel approach of a two-stage method for the training of deep convolution neural networks to improve the generalized ability of CNN by ensuring robustness to the selection of hyperparameter and optimizing feature boundary while initialization hardly affecting the ability of classification of the convergent network model. Further Zheng et al. [24] introduced a technique called layer-wise learning-based stochastic gradient descent method for the gradient-based optimization of the objective function which is a computationally effective and simple technique. The practical performance of the learned model is improved and the training process accelerates. The Generalness and robustness of these methods make it insensitive to hyperparameters which makes this technique more vastly applicable to other datasets and architecture networks.

In recent times most astonishing achievement in the field of DRL is the designing of the algorithm which can learn to play 2600 Atari games at the superhuman level directly from pixels of images [13].

In the case of three-wheeled Omnidirectional mobile robots tracking is a difficult task because of the orientation of the wheels which makes it rotate around its axis rather than follow the trajectory. Motivation to use the DRL algorithm is that in traditional reinforcement learning algorithm bellman equation is used which itself mathematically complex to solve and find the optimal solution on a particular state and action. But in DRL this equation is replaced by a neural network that can iterate and describes the best result according to the action and state. We use a neural network to define an actor and critic network to maximize the long-term reward while DDPG is used to train the agent using the reward function which is developed based on the difference between the actual and desired value of the output. DDPG is used because we are considering continuous observation and continuous actions.

The rest of the paper is organized in the following way, in Section 2 introduction to reinforcement learning, deep reinforcement learning and deep deterministic policy gradient are discussed then in Section 3 Dynamic Modeling of the 3WD-Omnidirectional Mobile Robot is derived, in Section 4 DDPG algorithm is described with reward function, environment, and actor-critic networks. Section 5 and 6 describe the results/simulation and conclusion respectively.

2. Background

Reinforcement learning is a recent and much powerful approach that can be used for wheeled mobile robots, as it enables us to find an optimal solution to a problem with the help of a trial-and-error approach. This technique is based on a neuropsychological cognitive science perspective [2]. Inspired by the behavior of animals where animals, learn to do some specific task to get a reward or to avoid punishment, this technique has the ability to solve many recent complex problems with ease [20]. It is becoming famous among the control enthusiastic because of its model less approach, also known as a black-box approach in which Reinforcement learning can find an optimal solution to a problem for the systems with very complex or high dimensional systems those systems whose modeling itself consider a problem in control system field. A generalized scheme for reinforcement learning and feedback control system is shown in Figures 2-3, respectively.

Figure 2
Reinforcement learning Scheme

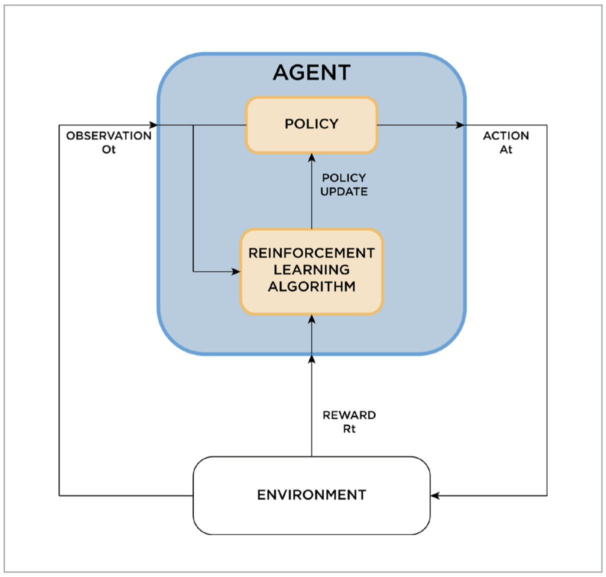
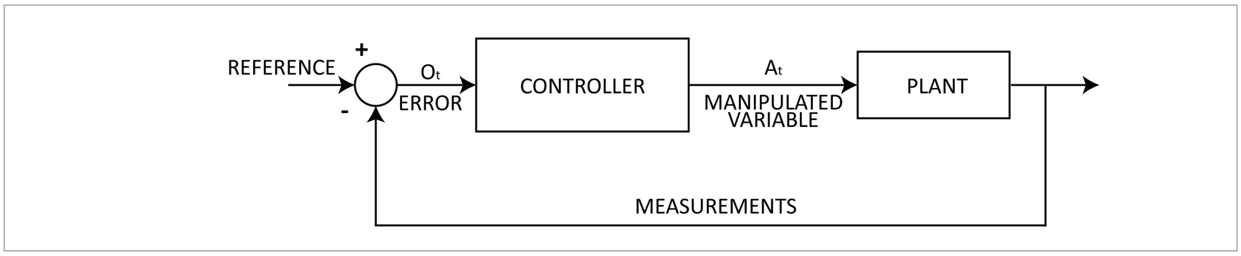


Figure 3
Feedback Control Scheme



The mapping of reinforcement learning terms for the control system is given below.

Policy – Policy in a control system is a controller

Environment – Everything in a control system excluding controller is the environment. It shows in Figure 3, that the environment contains the plant, the desired reference, and the error. In general, the environment contains everything elements like disturbance, analog-digital, digital-analog converters, filters, measurement noise, etc.

Observation – Any value that can be measure and visible to an agent. In Figures 2-3, the controller can see the error signal from the environment. We can also develop an agent that can observe outputs, reference signal measurement signals, and rate of change of these signals.

Actions – Actions that can be taken by an actuator in a control system to control a plant.

Reward – Reward is a function of signals which can evaluate the performance of the system according to the requirements. It can include sensors output, error, or some performance metric. For example; we can implement a reward function to minimize the control effort while minimizing the error of a control system.

Learning Algorithm – Learning algorithm is an adaptation mechanism of adaptive control of a system.

2.1. Deep Reinforcement Learning

In this paper, Deep Deterministic Policy Gradient (DDPG) as proposed in [11], is used. In DDPG as a baseline of deep reinforcement learning, the actor-critic network is used. Deep reinforcement learning is a blend of deep learning and reinforcement learning. It makes an agent capable of learning to behave in an environment based on feedback rewards or cost function. The main attribute of deep reinforcement learning is that deep neural networks can autonomously explore compact low-dimensional representations (features) of high-dimensional inputs (e.g., text, observations, images, and audio). This field of research has had the option to tackle a wide scope of complex decision-making errands that were already distant for a machine. Along these lines, DRL opens up numerous new applications in spaces, for example, social insurance, mechanical autonomy, Robotics, savvy lattices, and some more.

2.2. Deep Deterministic Policy Gradient (DDPG)

For the problems of high dimensionality, complex task, and the environment with continuous action space, only DDPG is used. The deterministic policy gradient algorithm which simultaneously learns Q-Value (max. reward) and a policy. For finding the max. Q-function, the Bellman equation is used. For solving the Bellman equation, there are two methods i.e. Value-based (deterministic policy)

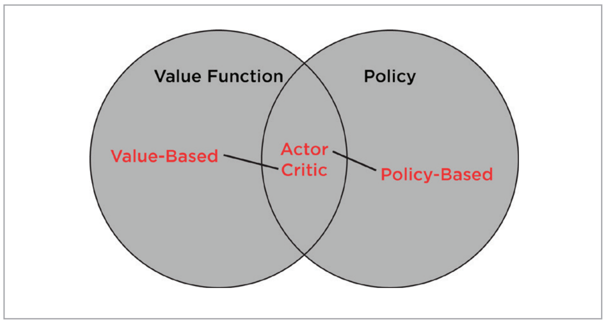
$$\forall s \in S : \hat{V}^{\pi}(s) := R(s) + \gamma \sum_{s \in S} P(s' \mid s, \pi(s)) \hat{V}^{\pi}(s')$$

and Policy-based (stochastic policy) [17]

$$\forall s \in S : \hat{V}^{*}(s) := R(s) + \gamma \max_{a \in A} \sum_{s \in S} P(s' \mid s, a) \hat{V}^{*}(s'),$$

where S = set of states, A=set of actions, a= particular action belongs to A, P = probability, R(s) = reward at s state and γ = discount factor. In Value-based, the output is an action while in Policy-based actions are vague. There is always a probability of every possible action. When the action space is confined Q-function is computed using value iteration. In a continuous action space, we cannot evaluate reward every step, quite a time consuming and exhausting. The Q function becomes differentiable concerning the action for every continuous action space. So instead of using the Value iteration, Policy evaluation is used. A deep deterministic policy gradient used the Actor critic algorithm which is in between the Value-based and policy-based shown in Figure 4.

Figure 4
Actor-Critic Approach



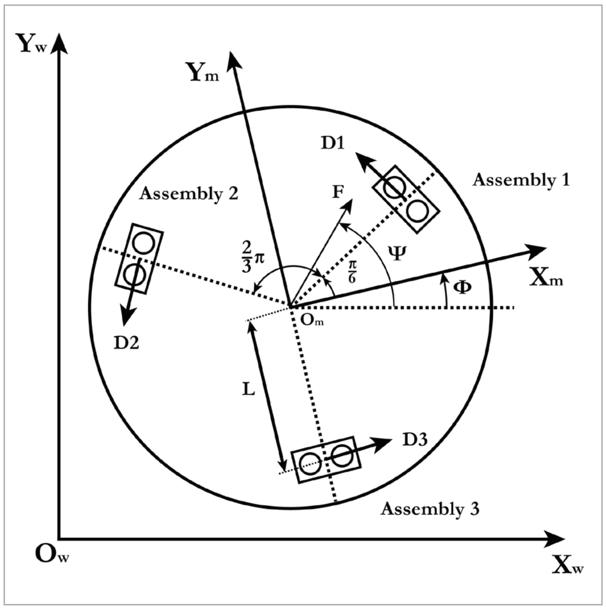
The actor uses the policy-based approach in which it learns how to act by directly evaluating the optimal policy. Gradient ascent is used to maximizing the reward. While Critic used the value-based approach. It directly maps the action i.e., the different states.

3. Dynamic Modeling of the 3WD-Omnidirectional Mobile Robot

This section describes a dynamic model of three-wheeled omnidirectional robot. This robot has three Swedish wheel assemblies. The mathematical modeling of the robot is central to controller design.

Figure 5

Model of 3WD-Omnidirectional Mobile Robot



Consider a 3-wheel omnidirectional mobile robot moving on a solid surface. Real-world coordinates system can be assumed as $O_R : X_R\, Y_R$ whereas the robot coordinates system is $O_r : X_r\, Y_r$ is static on the center of gravity(cog) for the mobile robot as in Figure 5. While describing the position vector of the center of gravity for a 3-wheel omnidirectional mobile robot-like $S_R[x_R\, y_R]^T$, we have

$$M\ddot{S}_R = F_R\,, \tag{1}$$

where real-world coordinates system $F_R[F_x\, F_y]^T$ is the force vector applied to the center of gravity of the robot and M is mass matrix.

Let's take the difference of angle between the real-world coordinates X_R and moving coordinates X_r as φ, i.e., the rotational angle of the robot coordinate system with respect to the real-world coordinate system [21]. The transformation matrix to convert robot coordinates to real-world coordinates system is

$${}^{w}R_c = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \tag{2}$$

it follows that

$$\dot{S}_R = {}^{R}R_r\, \dot{s}_r \tag{3}$$

$$F_R = {}^{R}R_r\, f_r\,, \tag{4}$$

where the position vector and force vector of center of gravity are $s_r[x_r\, y_r]^T$ and $f_r[f_x f_y]$ in the robot coordinate system. Therefore, the following equation is obtained after solving Eq (1) for the robot coordinates system provides

$$M({}^{R}R_r\ {}^{R}\dot{R}_r \dot{s}_r + \ddot{s}_r)\,. \tag{5}$$

Then, the three-wheeled omnidirectional mobile robot dynamic properties can be described as [5,18].

$$M(\ddot{x}_r - \dot{y}_r\dot{\varphi}) = f_x \tag{6}$$

$$M(\ddot{y}_r - \dot{x}_r\dot{\varphi}) = f_y \tag{7}$$

$$I_v\ddot{\varphi} = M_I\,, \tag{8}$$

where I_v is robot's moment of inertia, M_I is the moment around the center of gravity of the robot, and f_x, f_y, M_I are following:

$$f_x = -\frac{1}{2}D_1 - \frac{1}{2}D_2 + D_3 \tag{9}$$

$$f_y = \frac{\sqrt{3}}{2}D_1 - \frac{\sqrt{3}}{2}D_2 \tag{10}$$

$$M_I = (D_1 + D_2 + D_3)L\,. \tag{11}$$

In addition, the property of driving system [22], [16] for each assembly is taken as

$$I_R\dot{\omega}_i + c\omega_i = ku_i - rD_i\,, \quad i = 1,2,3\ldots\,, \tag{12}$$

where L is the distance from any wheel and the center of gravity of the robot; k is the driving gain factor; D_i is the driving force of robot wheel; r is the radius of each wheel of robot; c is the viscid resistance factor of the wheel; ω_i is the rate of change of angle of the robot; I_R is the moment of inertia of the wheel of robot around the driving shaft; and u_i is driving input torque. The geometrical relationships between variables $\dot{\varphi}$, $\dot{x}_r$, $\dot{y}_r$ and ω_i i.e., the inverse kinematics can be written as:

$$r\omega_1 = -\frac{1}{2}\dot{x}_r + \frac{\sqrt{3}}{2}\dot{y}_r + L\dot{\varphi} \tag{13}$$

$$r\omega_2 = -\frac{1}{2}\dot{x}_r - \frac{\sqrt{3}}{2}\dot{y}_r + L\dot{\varphi} \tag{14}$$

$$r\omega_3 = \dot{x}_r + L\dot{\varphi}\,. \tag{15}$$

Using Equations (6) to (15) gives:

$$\ddot{y}_r = a_1\dot{y}_r - a_2'\dot{x}_r\dot{\varphi} + \sqrt{3}b_1(u_1 - u_2) \tag{16}$$

$$\ddot{x}_r = a_1\dot{x}_r + a_2'\dot{y}_r\dot{\varphi} - b_1(u_1 + u_2 - 2u_3) \tag{17}$$

where

$$\ddot{\varphi} = a_3\dot{\varphi} + b_2(u_1 + u_2 + u_3)\,, \tag{18}$$

$$a_1 = -\frac{3c}{3I_R + 2Mr^2}$$

$$a_2' = \frac{2Mr^2}{3I_R + 2Mr^2}$$

$$a_3 = -\frac{3cL^2}{3I_RL^2 + I_vr^2}$$

$$b_1 = \frac{kr}{3I_R + 2Mr^2}$$

$$b_2 = \frac{krL}{3I_RL^2 + I_vr^2}\,.$$

Model parameter used for the simulation are given in Table 1.

Table 1
Model Parameters of 3WD-Omnidirectionl Mobile Robot

Parameters	Description	Value
I_v	Robot Moment of Inertia	11.25 kgm^2
M	Robot Mass	9.4 kg
L	Distance Between	0.178 m
k	Driving Gain Factor	0.448
c	Viscous Friction Factor	0.1889 kgm^2s^{-1}
I_w	Moment of Inertia of Wheel	0.02108 kgm^2
r	Radius of Wheel	0.0245 m

4. Deep Deterministic Policy Gradient (DDPG)

The DDPG algorithm is an off-policy, online, model-free reinforcement learning method. A DDPG agent is based on an actor-critic reinforcement learning agent that maximizes the long-term reward by computing an optimal policy. The main difference between the actor-critic approach and DDPG is that the action space of DDPG is a continuous while for actor-critic approach have discrete action space. DDPG agents can be trained in environments with continuous or discrete observations and continuous action spaces. In [11], the working and algorithm of DDPG used in this paper. While training, a DDPG agent do the following things:

1 Agent updates critic and actor properties at every time step during training.

2 Using a circular experience buffer, it stores past experiences. The agent updates the critic and actor using a mini-batch of experiences randomly sampled from the buffer.

3 Use noise models to perturbs the action chosen by the policy at every training step.

The following four functions are maintained by a DDPG agent to estimate a value and policy function.

approximators:

– Actor μ(S): The actor takes observation S and outputs the corresponding action that maximizes the long-term reward.

- Target actor $\mu'(S)$: To improve the stability of the optimization, the agent periodically updates the target actor based on the latest actor parameter values.
- Critic $Q(S, A)$: It takes inputs as action A and observations S and provides the output of corresponding expectation of long-term rewards
- Target critic $Q'(S, A)$: To increase the stability of optimization the agent updates the target critic periodically based on the newest critic parameter values.

$Q(S, A)$ and $Q'(S, A)$ both have the similar parameterization and structure, and both $\mu(S)$ and $\mu'(S)$ have the similar parameterization and structure. When training is complete, the trained optimal policy is stored in actor $\mu(S)$.

4.1. DDPG Algorithm

DDPG agents use the following training algorithm, in which they update their actor and critic models at each time step.

- Randomly Initialize the critic $Q(S, A)$ with some parameter values θ_Q, and initialize the target critic with the same random parameter values: $\theta_Q = \theta_{Q'}$.
- Randomly Initialize the actor $\mu(S)$ with some parameter values θ_μ, and initialize the target actor with the same parameter values: $\theta_\mu = \theta_{\mu'}$.
- For each training time step:

1 Select action A = $\mu(S) + N$ for the current observation S, where N is noise belongs to a noise model.

2 Execute action A. See the reward R and the next observation is S'.

3 Store the experience (S, A, R, S') in the experience buffer.

4 Sample a random mini-batch of M experiences (S_i, A_i, R_i, S_i') from the experience buffer.

5 If S_i' is a terminal state, set the value function target y_i to R_i. Otherwise, set it to

$$y_i = R_i + \gamma Q'(S_i', \mu(S_i' \mid \theta_\mu) \mid \theta_Q') .$$

The value function target is the sum of the experience reward R_i and the discounted future reward.

To compute the cumulative reward, the agent first calculates the successor action bypassing the successor observation S_i' from the sampled experience to the target actor. The agent finds the cumulative reward bypassing the successor action to the target critic.

6 Update the critic parameters by minimizing the loss function $f(Loss)$ across all sampled experiences.

$$f(Loss) = \frac{1}{M} \sum_{i=1}^{M} (y_i - Q(S_i, A_i \mid \theta_Q)) .$$

7 Update the actor parameters using the following sampled policy gradient to maximize the expected discounted reward.

$$\nabla_{\theta_\mu} J \approx \frac{1}{M} \sum_{i=1}^{M} G_{ai} G_{\mu i}$$

$$G_{ai} = \nabla_A Q(S_i, A \mid \theta_Q) ,$$

where

$$A = \mu(S_i \mid \theta_\mu)$$

$$G_{\mu i} = \nabla_{\theta_\mu} \mu(S_i \mid \theta_\mu) .$$

Here, G_{ai} is the gradient of the critic output with respect to the action computed by the actor-network, and $G_{\mu i}$ is the gradient of the actor output with respect to the actor parameters. Both gradients are evaluated for observation S_i.

8 Update the target actor and critic depending on the target update method

(Smoothing or periodic).

For smoothing:

$$\theta_{Q'} = \tau\theta_Q + (1-\tau)\theta_Q$$

$$\theta_{\mu'} = \tau\theta_\mu + (1-\tau)\theta_{\mu'} .$$

For Periodic:

$$\theta_{Q'} = \theta_Q$$

$$\theta_{\mu'} = \theta_\mu .$$

The reinforcement learning toolbox of MATLAB 19 is used to create a DDPG agent and the parameters used for the creation of the DDPG agent are as follows.

Discount Factor = 0.99, Mini batch size = 128, Experience buffer length = 1×10^6, Target smooth factor = 1×10^{-3}, Noise mean attraction constant = 1, Noise variance = 0.1.

Parameters used for the training of DDPG agents are sampling time = 1, Discount Factor = 0.99, Mini Batch size = 128, Experience buffer length = 1×10^6, Target smooth factor = 1×10^{-3}.

4.2. Actor and Critic Network

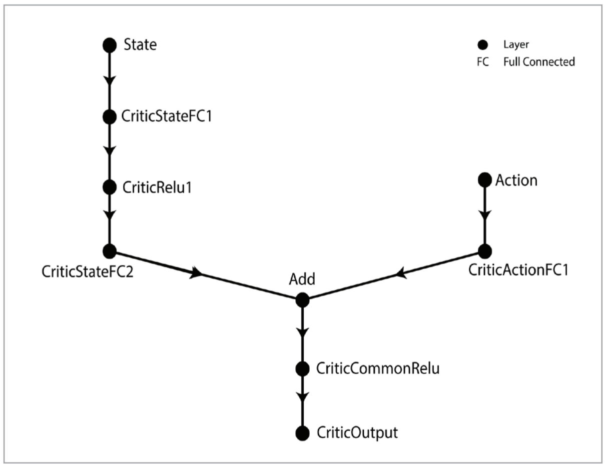
The actor and critic network are defined by the help of deep neural network toolbox and design to create actor-network which intakes observation and outputs action which in case of a 3WD-Omnidirectional mobile robot is the motor speed of three Swedish wheels. Observations that are used for this system are x, y, θ, $\dot{x}$, $\dot{y}$, $\dot{\theta}$, x_e, y_e, $\dot{x}_e$, $\dot{y}_e$ and motor speeds from previous agent. The steps are as follows to create a good actor and critic network.

1 Start with the smallest possible network and a high learning rate (0.01). Train this initial network to see if the agent converges quickly to a poor policy or acts randomly. If either of these issues occurs, rescale the network by adding more layers or more outputs on each layer. The goal is to find a network structure that is just big enough, does not learn too fast, and shows signs of learning (an improving trajectory of the reward graph) after an initial training period.

2 Initially configure the agent to learn slowly by setting a low learning rate. By learning slowly, it can be checked to see if the agent is on the right track, which can help verify whether the network architecture is satisfactory for the problem. For difficult problems, tuning parameters is much easier once we settle on good network architecture. Figure 6, shows the graphical representation of the critic neural network.

Figure 6

Critic Neural Network



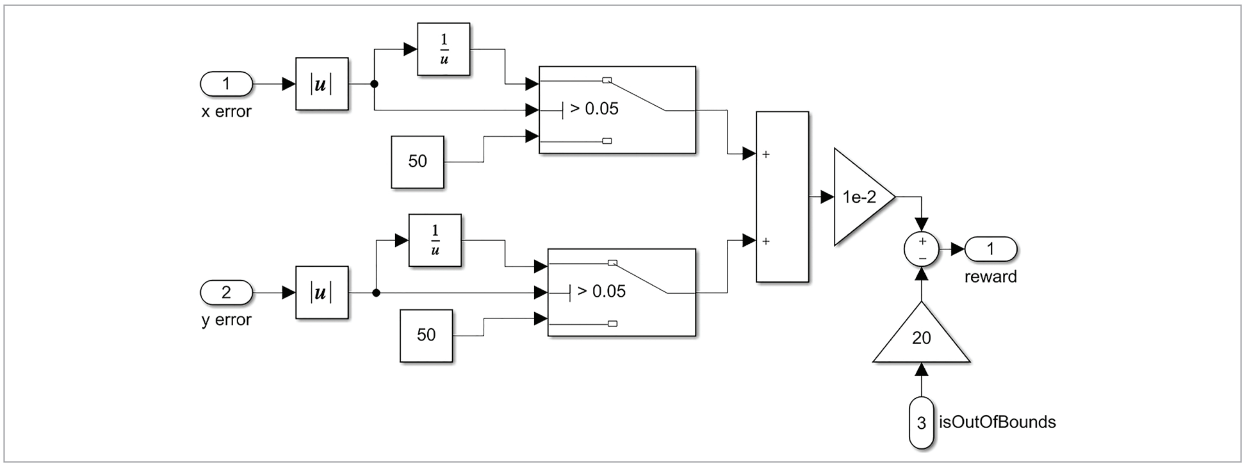
Setting for NN actor-critic networks are optimizer = adam, learn rate= 1×10^{-3}, Gradient threshold = 1, Regularization factor = 1×10^{-5}.

4.3. Reward Function

The main purpose of the paper is to track a reference trajectory, where the main task is to minimize the error function so one can design a reward function based on an error signal. The error signal used for the simulations is as follows. Simulink representation of total reward function $R1$ is shown in Figure 7.

Figure 7

Reward Function Simulink Representation



$$r_1 - 2(x_e^2 + y_e^2) + 2(0.9 < x < 1.1) + 2(0.9 < y < 1.1) + 2(\theta < 2)$$

$$r_2 = (-20 \ if \ \ x, y \geq \pm 5)$$

$$R1 = r_1 + r_2 \,.$$

Whereas the second reward function $R2$ is defined which is simple but reward increases as error decrease

$$r_3 = 1 \times 10^{-2} \left[\begin{cases} \dfrac{1}{|x_e|}, & |x_e| > 0.05 \\ 50, & else \end{cases} + \begin{cases} \dfrac{1}{|y_e|}, & |y_e| > 0.05 \\ 50, & else \end{cases} \right]$$

$$r_2 = (-20 \ if \ \ x, y \geq \pm 5)$$

$$R2 = r_3 + r_2 \,.$$

4.4. Environment

In terms of reinforcement, the learning environment is everything except the agent. The environment includes the plant, the desired reference, and the error. In general, the environment also contains some other elements like disturbance, analog-digital, digital-analog converters, filters, measurement noise, etc.

In the case of 3 wheeled omnidirectional mobile robot environment block is created in Simulink which includes reward function, exceed bound limits, observations. Figure 8, shows the dynamic model of the system Environment created for this paper then this block is integrated with the RL agent which learns the policy and implemented it on the dynamic model of the system Figure 9.

Figure 8
Environment for DDPG Agent

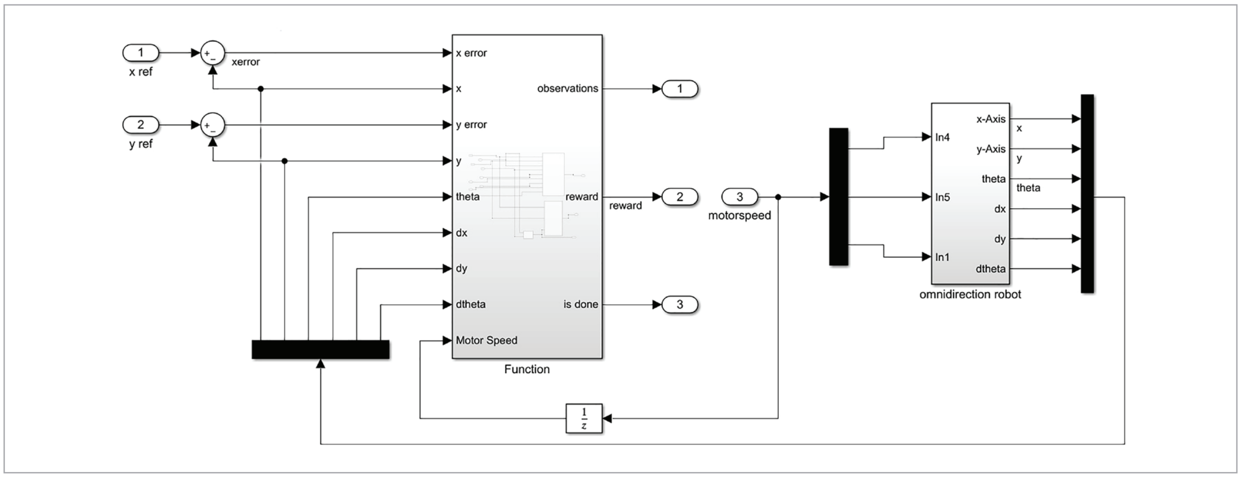
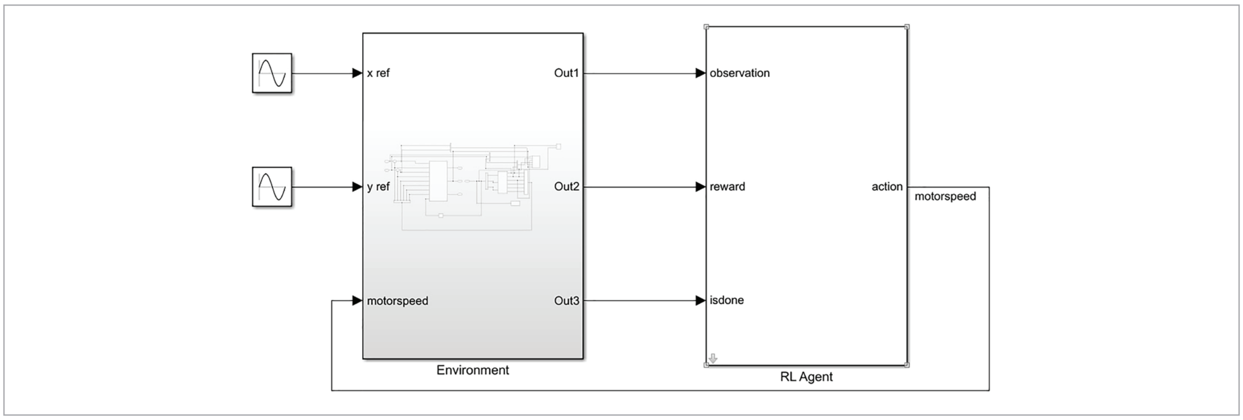


Figure 9
Integrated Environment with DDPG Agent



RL agent takes an observation, reward function, and flag function which shows if the simulation is done or not as an input and outputs the motor speeds of 3-wheeled omnidirectional mobile robots.

5. Results and Simulations

Simulation for the validation of the results has been done in MATLAB 19 and the Reinforcement learning toolbox is used for environment creation, actor-critic networks, agent, and training of that agent. To reduce complexity, the simulation range of the motor's inputs is selected [0 –∞] where two motors M1 and M2 are set as positive while third M3 is set to move opposite to the first two motors. This is done to limit the rotation of the 3WD-Omnidirectional mobile robot along its axis. Two different scenarios for the trajectories are used to validate the results first scenario is to track a point to point with a straight line this is the simplest scenario because as the robot advance forwards its angle φ remain constant. While in the second scenario tracking of circular trajectory is used because it is a complex trajectory for the 3WD omnidirectional robot because φ changes at each point of the circle.

5.1. Scenario 1

For initial training reference is given as a point-to-point tracking, Simulation results are given in Figure 10, shows the control inputs to the motors (M1, M2, M3). Figure 11, Shows the no of iteration and different rewards on each iteration which includes episode reward, average reward, and expected reward while Figure 12, shows the results of point-to-point tracking of the 3-wheel omnidirectional mobile robot.

The simulation stops when the average reward reaches to 1000. Iteration's graph shows that for about 100 iterations there is nothing special happen then suddenly Neural networks of actor and critics start to predict the inputs where reward function maximizes. Stopping criteria are selected by monitoring the average reward. It is because each episode reward is very random and can go to the maximum value and minimum value at any time.

Figure 10
Control Inputs

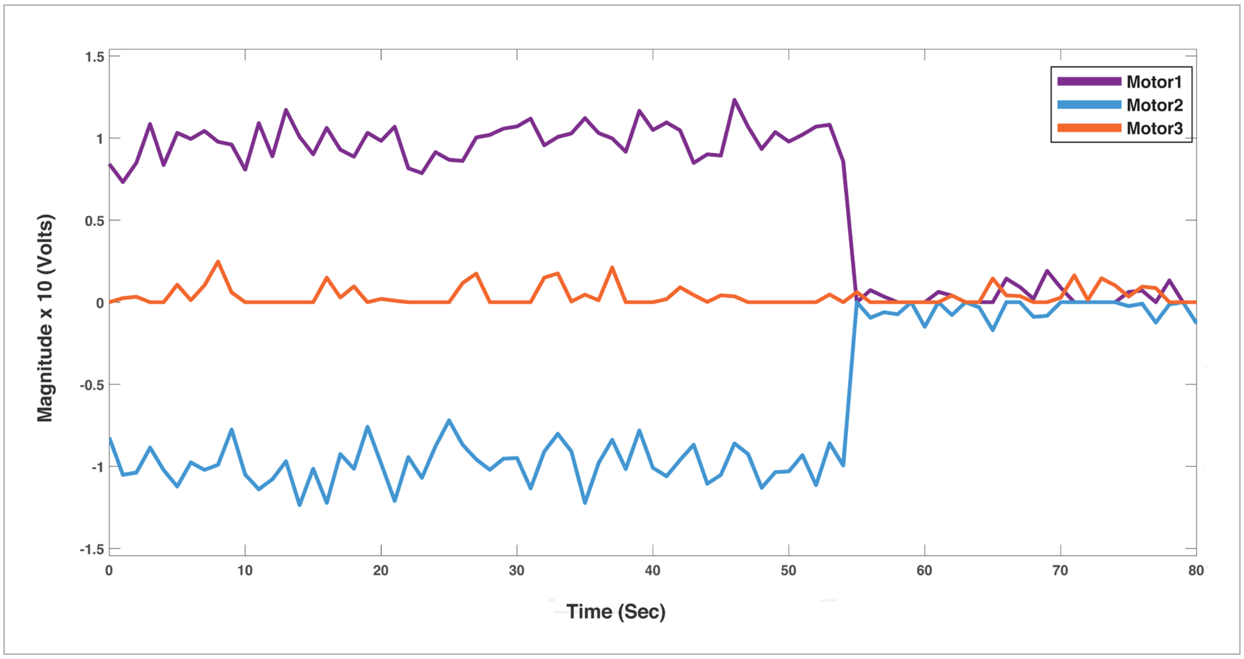


Figure 11
Point Tracking Iterations

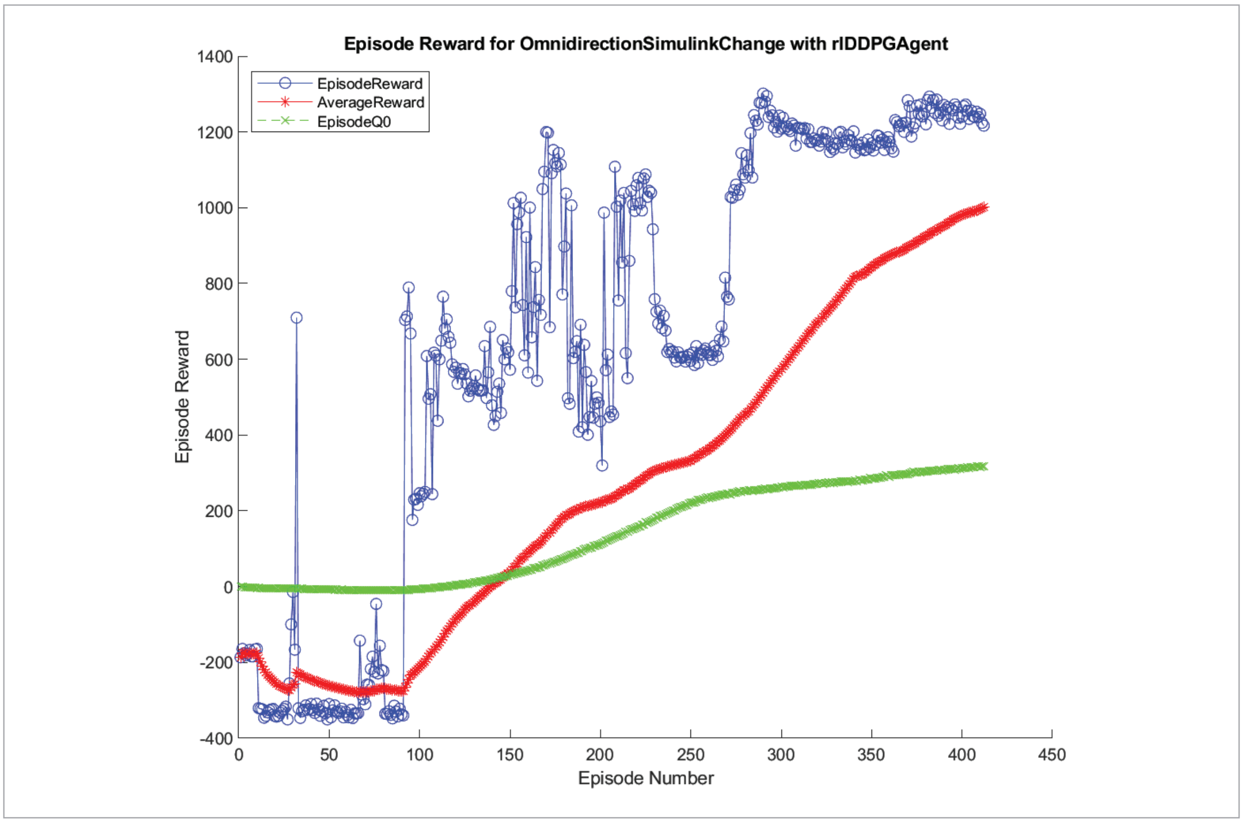
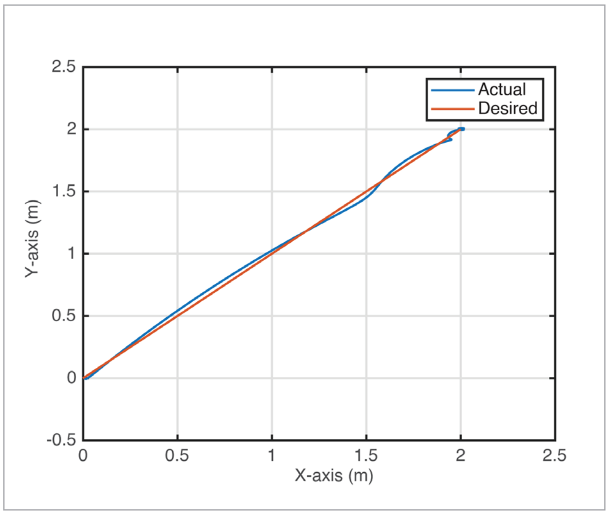


Figure 12
Point to Point Tracking



5.2. Scenario 2

In this scenario, a sine wave is applied as a reference of the x-axis while the cosine wave is applied as a reference of the y-axis. Which combines to make a circle to the trajectory for reference. Simulation results are given in Figure 13, shows the iteration for the tracking of circular trajectory simulations stops when the average reward approaches 1900.

Figure 14, shows the error signal of the x-y axis for the tracking of circular trajectory error starts at maximum because robot initial position is at the origin then it starts follows the circle and error become zero.

Figure 15, shows the result for circular trajectory tracking of 3 wheels omnidirectional mobile robots. While the reset function is set to come to the origin when every iteration ended.

Figure 13
Circle Tracking Iterations

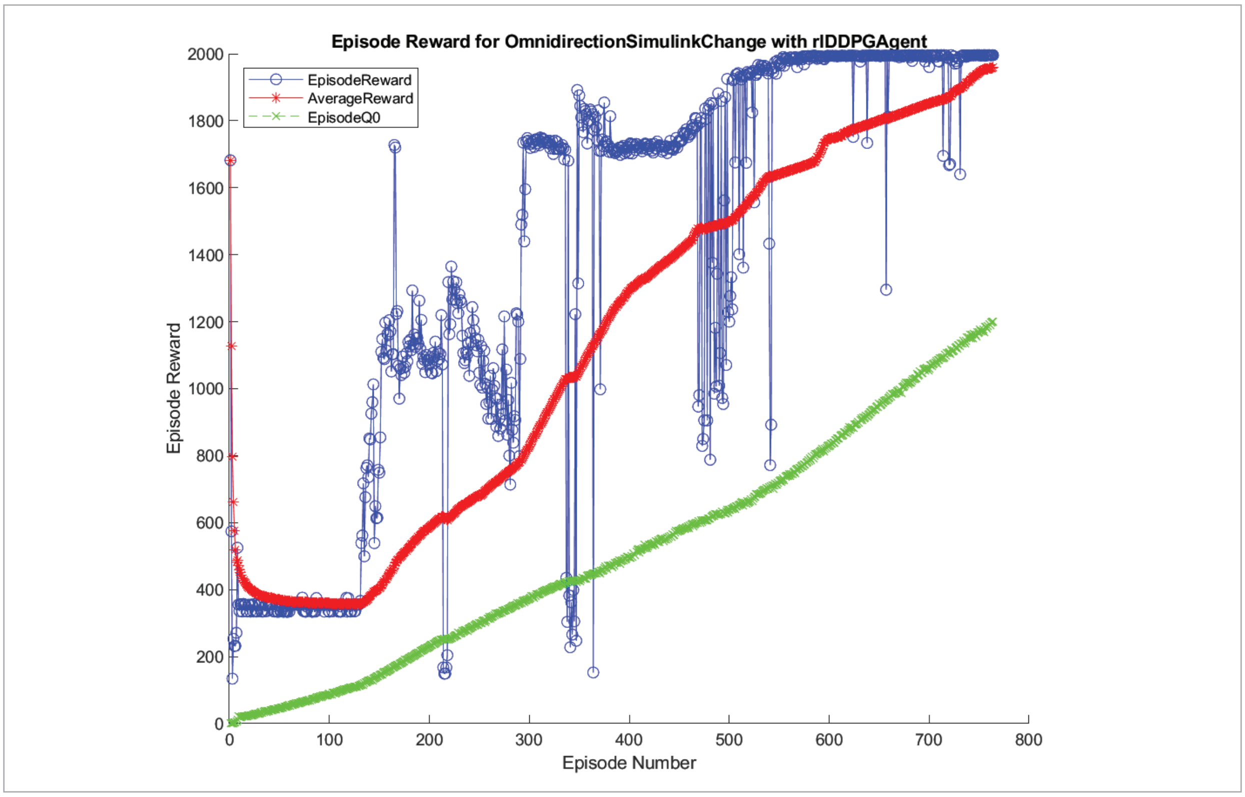


Figure 14
X-Y Error

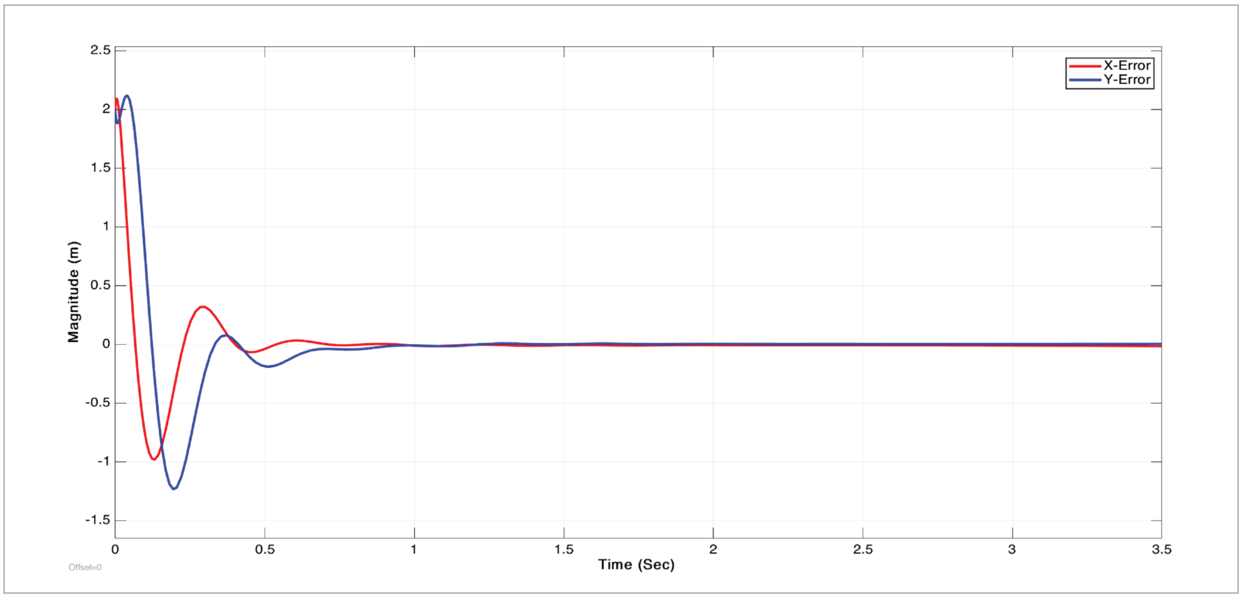
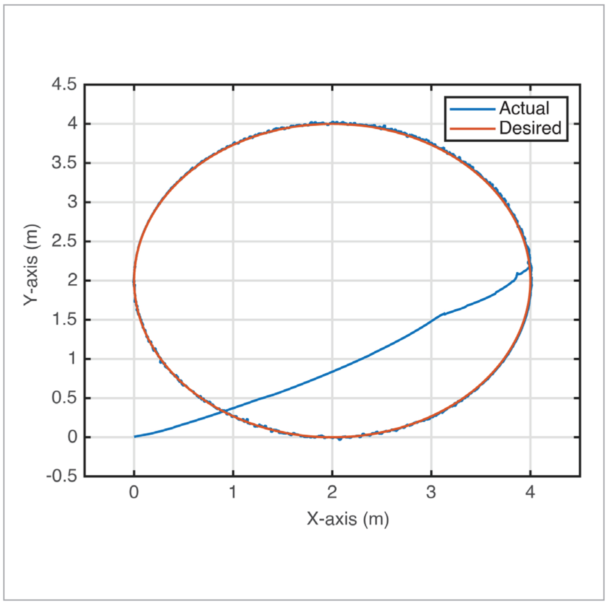


Figure 15

Circle Tracking



6. Conclusion

To achieve tracking of 3 wheels omnidirectional mobile robot (deep reinforcement learning) DRL algorithm (deep deterministic policy gradient) DDPG is used which allows us to achieve our goal by taking continuous actions and states. To attain a control objective, less calculation is needed compare to the full optimal control algorithm, and we always got more accuracy, compared to a typical control method. MATLAB R2019a is used for the simulation and the reinforcement learning toolbox makes the whole work very easy. The best part of this technique is that we can achieve a goal with no or very less knowledge of the dynamic model and it will work on that too. This research is very useful where a robot has to do a task repeatedly millions of times like automatic mobile assembly, automatic sorting a book in the library, robots working in congested areas, planetary exploration, etc. Further research can be carried out by attaching a traditional feedback controller with reinforcement learning to achieve more fast and better results.

References

1. Altuntas, N., Imal, E., Emanet, N., Öztürk, C. N. Reinforcement Learning-Based Mobile Robot Navigation. Turkish Journal of Electrical Engineering & Computer Sciences, 2016, 24(3), 1747-1767.https://doi.org/10.3906/elk-1311-129
2. Amarjyoti, S. Deep Reinforcement Learning for Robotic Manipulation-The State of The Art. arXiv Preprint arXiv:1701.08878, 2017.
3. Deisenroth, M. P., Neumann, G., Peters, J. A Survey on Policy Search for Robotics. Foundations and Trends in Robotics, 2013, 2(1-2), 388-403. https://doi.org/10.1561/2300000021
4. Ding, L., Li, S., Gao, H., Chen, C., Deng, Z. Adaptive Partial Reinforcement Learning Neural Network-Based Tracking Control for Wheeled Mobile Robotic Systems. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2018, 50(7), 2512-2523. https://doi.org/10.1109/TSMC.2018.2819191
5. Iwatsuki, M., Nakano, K., Ohuchi, T. Target Point Tracking Control of Robot Vehicle by Fuzzy Reasoning. Transactions of the Society of Instrument and Control Engineers, 1991, 27(1), 70-76. https://doi.org/10.9746/sicetr1965.27.70
6. Kober, J., Bagnell, J. A., Peters, J. Reinforcement Learning in Robotics: A Survey. The International Journal of Robotics Research, 2013, 32(11), 1238-1274. https://doi.org/10.1177/0278364913495721
7. Kober, J., Peters, J. Policy Search for Motor Primitives in Robotics. Learning Motor Skills, Springer, Cham, 2014, 83-117. https://doi.org/10.1007/978-3-319-03194-1_4
8. Leiva, F., Lobos-Tsunekawa, K., Ruiz-del-Solar, J. Collision Avoidance for Indoor Service Robots Through Multimodal Deep Reinforcement Learning. Robot World Cup, Springer, Cham, 2019, 140-153. https://doi.org/10.1007/978-3-030-35699-6_11
9. Leottau, D. L., Vatsyayan, A., Ruiz-del-Solar, J., Babuška, R. Decentralized Reinforcement Learning Applied to Mobile Robots. Robot World Cup, Springer, Cham, 2016, 368-379. https://doi.org/10.1007/978-3-319-68792-6_31
10. Levine, S., Koltun, V. Guided Policy Search. In International Conference on Machine Learning, 2013, 1-9.
11. Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D. Continuous Control with Deep Reinforcement Learning. arXiv Preprint arXiv:1509.02971, 2015.

12. Miljković, Z., Mitić, M., Lazarević, M., Babić, B. Neural Network Reinforcement Learning for Visual Control of Robot Manipulators. Expert Systems with Applications, 2013, 40(5), 1721-1736. https://doi.org/10.1016/j.eswa.2012.09.010

13. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S. Human-Level Control Through Deep Reinforcement Learning. Nature, 2015, 518(7540), 529-533. https://doi.org/10.1038/nature14236

14. Murphy, R. R. Introduction to AI Robotics. MIT Press, 2019. https://doi.org/10.1108/ir.2001.28.3.266.1

15. Muse, D., Weber, C., Wermter, S. Robot Docking Based on Omnidirectional Vision and Reinforcement Learning. International Conference on Innovative Techniques and Applications of Artificial Intelligence, Springer, London, 2005, 12, 23-36. https://doi.org/10.1007/978-1-84628-226-3_3

16. Saito, M., Tsumura, T. Collision Avoidance Among Multiple Mobile Robots. Transactions of the Institute of Systems, Control and Information Engineers, 1990, 3(8), 252-260. https://doi.org/10.5687/iscie.3.252

17. Sutton, R. S., Barto, A. G. Reinforcement Learning: An Introduction. MIT Press, 2018.

18. Tang, J., Watanabe, K., Shiraishi, Y. Design and Traveling Experiment of an Omnidirectional Holonomic Mobile Robot. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems IROS'96, 1996, 1, 66-73. https://doi.org/10.1109/IROS.1996.570633

19. Tutsoy, O., Brown, M., Wang, H. Reinforcement Learning Algorithm Application and Multi-Body System Design by Using Maplesim and Modelica. The 2012 International Conference on Advanced Mechatronic Systems, 2012, 650-655.

20. Tutsoy, O., Brown, M. Convergence Analysis of Reinforcement Learning Approaches to Humanoid Locomotion. IET Conference Proceedings, 2010, 1130-1135. https://doi.org/10.1049/ic.2010.0439

21. Watanabe, K., Shiraishi, Y., Tang, J., Fukuda, T., Tzafestas, S. G. Autonomous Control for an Omnidirectional Mobile Robot with Feedback Control System. Advances in Intelligent Autonomous Systems, Springer, Dordrecht, 1999, 289-308. https://doi.org/10.1007/978-94-011-4790-3_13

22. Watanabe, K. Control of an Omnidirectional Mobile Robot. Second International Conference. Knowledge-Based Intelligent Electronic Systems. Proceedings KES'98 (Cat. No. 98EX111) IEEE, 1998, 1, 51-60. https://doi.org/10.1109/KES.1998.725827

23. Yi, S. J., Zhang, B. T., Hong, D., Lee, D. D. Online Learning of a Full Body Push Recovery Controller for Omnidirectional Walking. 11th RAS International Conference on Humanoid Robots IEEE, 2011, 1-6. https://doi.org/10.1109/Humanoids.2011.6100896

24. Zheng, Q., Tian, X., Jiang, N., Yang, M. Layer-Wise Learning Based Stochastic Gradient Descent Method for the Optimization of Deep Convolutional Neural Network. Journal of Intelligent and Fuzzy Systems, 2019, 37(4), 5641-5654. https://doi.org/10.3233/JIFS-190861

25. Zheng, Q., Tian, X., Yang, M., Wu, Y., Su, H. PAC-Bayesian Framework-Based Drop-Path Method for 2D Discriminative Convolutional Network Pruning. Multidimensional Systems and Signal Processing, 2019, 1-35. https://doi.org/10.1007/s11045-019-00686-z

26. Zheng, Q., Yang, M., Tian, X., Jiang, N., Wang, D. A Full Stage Data Augmentation Method in Deep Convolutional Neural Network for Natural Image Classification. Discrete Dynamics in Nature and Society, 2020. https://doi.org/10.1155/2020/4706576

27. Zheng, Q., Yang, M., Yang, J., Zhang, Q., Zhang, X. Improvement of Generalization Ability of Deep CNN Via Implicit Regularization in Two-Stage Training Process. IEEE Access, 2018, 6, 15844-15869.https://doi.org/10.1109/ACCESS.2018.2810849

28. Zhu, Y., Schwab, D., Veloso, M. Learning Primitive Skills for Mobile Robots. International Conference on Robotics and Automation (ICRA) IEEE, 2019, 7597-7603. https://doi.org/10.1109/ICRA.2019.8793688