Received <day> <Month>, <year>; | Revised <day> <Month>, <year>; | Accepted <day> <Month>, <year>

DOI: xxx/xxxx

SPECIAL ISSUE ARTICLE

Deep-Q-Network Hybridization with Extended Kalman Filter for Accelerate Learning in Autonomous Navigation with the Auxiliary Security Module

Carlos D. S. Bezerra*[1] | Flávio H. T. Vieira[2] | Anderson da Silva Soares[1]

[1]Instituto de Informatica (INF), Federal University of Goias (UFG), Goiás, Brazil
[2]Escola de Engenharia Eletrica, Mecânica e de Computação (EMC), Federal University of Goiás (UFG), Goiás, Brazil

Correspondence
*Carlos D. Bezerra, Email: carlosdbez@discente.ufg.br

Present Address
Federal University of Goias - Campus Samambaia - Alameda Palmeiras, s/n - Chcaras California, CEP: 74690-900 Goiânia (GO), Brazil.

Abstract

This article proposes an algorithm for autonomous navigation of mobile robots that merges Reinforcement Learning with Extended Kalman Filter (EKF) as a localization technique, namely, EKF-DQN, aiming to accelerate learning and improve the reward values obtained in the process of apprenticeship. More specifically, Deep Neural Networks (DQN - *Deep-Q-Networks*) are used to control the trajectory of an autonomous vehicle in an indoor environment. Due to the ability of EKF to predict states, this algorithm is proposed to be used as a learning accelerator of the DQN network, predicting states ahead and inserting this information in the memory replay. Aiming at the safety of the navigation process, it is also proposed a visual safety system that avoids collisions of the mobile vehicle with people moving in the environment. The efficiency of the proposed algorithm is verified through computer simulations using the CoppeliaSIM simulator with code insertion in Python. The simulation results show that the EKF-DQN algorithm accelerates the maximization of rewards obtained and provides a higher success rate in fulfilling the proposed mobile robot mission compared to the DQN and Q-Learning algorithms.

KEYWORDS:
DQN, Reinforcement Learning, Autonomous Navigation, EKF.

1 | INTRODUCTION

The news involving autonomous mobile systems are current and frequent in the era we live in, of artificial intelligence and industry 4.0. These systems encompass both automotive vehicles and mobile robots, with different applications, and are directly impacting the lives of human beings. The day-to-day benefits that these technologies can provide are numerous. According to [1], autonomous vehicles can profoundly affect the economy, influencing and changing the way they are seen, for example, delivery, transport and manufacturing services just in time.

According to [1], autonomous driving is a complex task, composed of several sub-problems. Among these objections, we can mention the challenges of control and automation, artificial intelligence, communication processing systems and communication systems, mapping and location sensors, among others. The subject is interdisciplinary and to develop an extremely accurate, safe and condition-independent driving policy are challenges in the area of autonomous systems.

Machine Learning is an important field of research involving autonomous systems and advanced robotics. This science can be divided into three categories of problems: a) Supervised Learning, b) Unsupervised and c) Reinforcement Learning (RL). RL is

[0]**Abbreviations:** DQN, Deep-Q-Networks; EKF, Extended Kalman Filter.

a good machine learning tool for implementing autonomous robots. According to[2], this technique can be applied in sequential systems, where for example we have the agent (a robot) in a certain state (its position, for example), this agent performs actions and an environment (for example a track) . The actions of this agent generate feedback signals, which measure a quality of action in the environment. The objective is to maximize the expected reward value for a given state and action pair. In summary, the agent tends to learn by trial and error and always seeks to maximize the number of good rewards[3]. One of the advantages of RL is the possibility of navigating a mobile vehicle without route planning once the agent learns by iteration and it is able to generate adequate actions due to a change in the environment.

Concerning Reinforcement Learning techniques, the DQN algorithm (*Deep-Q-Networks*) is being used to different applications in the literature[4 5], and consists of a model based on deep artificial neural networks and Q-Learning. The authors in[4] show that this algorithm is able to perform control actions related to games that surpassed the performance of the human being. From this research, several variants of the DQN algorithm were created[6 7]. Mostly important, the DQN algorithm can be applied in several areas of knowledge, including autonomous systems[8 5].

Another important tool for autonomous navigation is the Kalman Filter (KF)[9]. This algorithm allows estimating the states that reflect the dynamics of a process or linear system. KF is able to correct information from uncertain or noisy sensory measurements, weighting them with the system's state transition model, thus establishing an estimate of the optimal state. The Extended Kalman Filter (EKF) is a sophisticated alternative to KF that can be applied to non-linear systems, thanks to the linearizations performed during the algorithm's update and prediction cycle. This tool had great application as a trajectory estimator for space modules during the Apollo mission, which aimed to take man to the moon. However, KF also has other numerous applications in control, most notably in modern control theory[10 11]. As a computationally fast algorithm, KF can be integrated with artificial intelligence methods to help solve a given problem that involves sensory uncertainties and state prediction, without excessively increasing the computational cost.

According to[12], one of the problems of reinforcement learning is the speed of convergence. Once the agent learns by iterations, if the exploration space, that is, the space in which the agent interacts with the environment is highly dimensional (there are several possible states), it is necessary to increase the number of training episodes and consequently make it more expensive. This situation can be critical for the agent that requires a quick learning process, or even needs to quickly adapt to new navigation conditions, such as route changes, new obstacles in the tracks, among others.

Considering the above mentioned factors, this paper has the **objective** to propose a learning acceleration method applied to DQN related to autonomous navigation of mobile robots. Therefore, the main contributions of this paper are:

- Proposal of a new control algorithm (EKF-DQN) based on DQN reinforcement learning for autonomous navigation using EKF;
- Proposal and evaluation of a suitable reward function to be used in the proposed algorithm for autonmous navigation;
- Comparison of the performance of the proposed algorithm with other algorithms in the literature.

We chose to consider the DQN network instead of Q-Learning in the proposed hybrid algorithm due to its better performance in problems involving a large number of possible states.

The development methodology is based on computer simulations, where a mission is proposed to an autonomous robot, which must learn to navigate the environment without a pre-planned route, guided by the reward function. The performance of the DQN and the DQN-EKF (proposed method) are compared using commonly used metrics[13 14]. Therefore, this work contributes to the investigation of learning techniques, providing enhancement to already established methods such as DQN, focusing on navigation of mobile vehicles that require fast learning and adaptability to new conditions.

In this paper, non-visual sensors such as GPS and Odometer are used and these measurements are combined with EKF to make state predictions. In this way, this research presents a novel learning and control algoritm (EKF-DQN), investigating if it is possible, through the use of EKF, to predict partial navigation states and to accelerate the convergence of the DQN through a proposal of inserting the prediction of states one or more steps ahead in the memory replay, thus establishing the optimal (or the best possible) navigation policy faster than the traditional algorithm.

The paper is organized as follows: Section 2 presents related works. Sections 3 and 4 briefly describes the theoretical background on control architectures, Reinforcement Learning and Kalman Filter, respectively. Section 5 presents the proposed EKF-DQN based learning and control algorithm and the implementation methodology. Section 6 comments about the results. Finally, Section 7 concludes and presents future researches.

2 | RELATED WORKS

The authors in [12] present a study on the acceleration of reinforcement learning using KF and were pioneers in this type of application. The authors' objective was to train a goalkeeper for robot soccer, in the championship known in the community as RoboCup. The reinforcement learning method used was the traditional Q-Learning, a tabular method [15]. The prediction of states with KF was applied to estimate the position of the ball that the goalkeeper should intercept, one step ahead of the current state. The results show that the proposed method was effective in accelerating the traditional Q-Learning (tabular method) for the observed application.

In [12], the dynamics, i.e., the physical equations of the ball motion have been simplified into linear motions, allowing for the use of traditional KF. In the present article, the proposed task is different, that is, the objective is related to autonomous navigation with a differential mobile robot. The physical equations of this type of system are nonlinear, which makes inappropriate to use the traditional KF, thus leading to the use of another estimation strategy, in this case the EKF. In addition, the DQN reinforcement learning technique is considered in the present work, that is more current and complete than Q-Learning. In this way, a new algorithm is proposed for autonomous navigation of a mobile robot, considering insertion of specific data in the DQN repetition memory.

The effectiveness of the EKF-DQN in the proposed task is observed through some metrics. Different from [12], in this work we present comparisons of metrics on the effectiveness of the algorithm, for example: how effective (average reward and success rate) was the proposed method in relation to traditional Q-Learning and DQN, how many tests were performed, etc.

The authors in [8] developed a research to build navigation systems applied to mobile robotic agents through the use of DQN. The research presents experimental and practical methodology, where the authors carried out simulations and after computational validation the model was embedded into a microcontroller. Images were used to represent the robot states, collected through an RGB camera. The authors use the DQN algorithm for locomotion and establish a certain mission to the agent. The experimental results show that the robot completed the suggested mission using the reinforcement learning based method without requiring previous construction of the environment map.

Recently, the authors in [16] published research involving the use of KF to improve reinforcement learning applied to the control of a two-wheeled robot, similar to the inverted pendulum problem. Different from our proposal, which uses EKF as a predictor of future robot states, the authors apply KF to filter out noisy information in sensory data before being used as a state in Q-Learning. This technique has been shown to improve the accuracy of states measured by a sensor, in this case the IMU (Inertial Measurement Unit), thus it increases the stabilization of the obtained rewards and the transient response of the control system. Therefore, from the results presented in [16], we are led to investigate if in addition to providing greater stabilization, KF can accelerate a newer or current reinforcement learning technique such as DQN.

In [17], it is investigated the automatic control of autonomous land vehicles. The authors propose to evaluate an algorithm for the control of the speed and position of a mobile robotic prototype. The algorithm used in the project is the classic Proportional Integral and Derivative (PID) control. The PID controller is used as a reactive method, or local control of actuators. However, planning algorithms are needed to deliberate routes and activities for the autonomous vehicle. The present work contributes in the sense of presenting a new navigation algorithm based on RL as well as practical information related to its implementation so that the robot learns adaptively to perform the desired path without requiring offline planning.

3 | CONTROL ARCHITECTURES, DQN AND EXTENDED KALMAN FILTER

This section presents a brief theoretical background on control architectures, reinforcement learning, extended Kalman filter and the relation of these techniques to autonomous navigation and accelerated learning.

3.1 | Control Architectures

This section presents a theoretical foundation on the control architectures applied to autonomous systems. It is also presented the architecture on which the present work is based, that is, a behavior-based architecture using Reinforcement Learning and Kalman filter.

The control architecture of an autonomous mobile robot is defined by [18] as a collection of structural components in which perception, reasoning and action occur together. According to [19] and [20], a control architecture is related to the software part of an

autonomous system, showing which modules must be used in the system and how the iteration of these modules occur during robot control.

Also according to[20], the basic modules present in a control system of a mobile robot can be classified into three groups: Modules of Perception and Sensing, Task Planning and Action. Perception means how the robot's sensors detect the environment, that is, how measurements of some necessary quantities are effectuated to locate, orient and control this robot. Planning tasks typically use computational algorithms to plan the route the robot will take during a proposed mission. Action modules are responsible for triggering or disabling, or even locally controlling robot actuators, including wheel motors and joints motors.

The classification of the control architecture is related to the use of reactivity or deliberation, and can be divided into: Purely Deliberative, Purely Reactive or Hybrid. In robotics, deliberation means decision making, action planning. In this way, deliberative architectures are largely composed of planning modules. The authors[20] cite an example of deliberation: a robot that transports clothes inside a house. This robot has knowledge of the house map and his planning algorithm plots a route for him to pick up clothes from a room and take them to the laundry room. It is observed that in this case the robot needs a knowledge map of the environment in which you will carry out your activities. Reactivity, on the other hand, is linked to the immediate response to a sensory reading obtained at a given time. For example the robot has no knowledge of the map of the environment, but it was programmed so that when it perceives an obstacle, it performs a deflection movement to avoid colliding. It is worth mentioning that the control architectures can be combined to a certain extent, forming the so-called hybrid architectures, which can combine reactive and deliberative algorithms. Figure 1 below illustrates the classic approach to control architectures in mobile robotics.

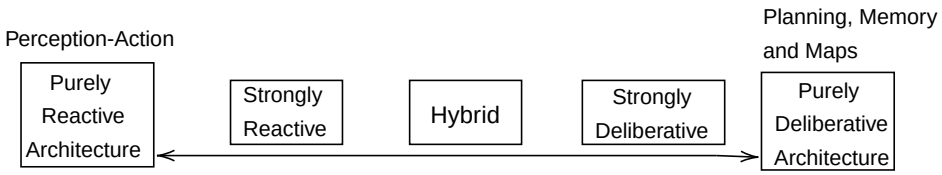

FIGURE 1 Classical Control Architectures

According to the authors[21,22] the evolution of Artificial Intelligence has led to new ways of understanding and developing architectures of control for mobile robotic systems, and one of the main concepts explored in his thesis is that of Behavior Based Systems. A behavior-based system is similar to a reactive architecture. Reactive systems provide fast, real-time responses using a collection of pre-programmed rules, and therefore these systems are unable to deliberate or learn new behaviors. On the other hand, the Behavior-based systems can store states in a distributed representation, i.e.: stimulus, behavior and robot response. This allows for a degree of deliberation and reaction at the same time. The robot learns different behaviors or different coordination methods that improve navigation performance. According to[23], Reinforcement Learning (RL) is one of the several techniques that can be used to implement architectures Behavior-based. This technique has been used to learn the internal structure of the behaviors by mapping the perceived states to control actions while maximizing the accumulated future rewards. The RL are very attractive for online learning as they do not use any knowledge base technique[22]. Figure 2 illustrates a behavior-based architecture.

3.2 | Reinforcement Learning and DQN Networks

For an autonomous mobile system to perform its proposed task, such as navigating in a track, flying over a certain environment, rescuing an object, among others, a control algorithm is required, where reinforcement learning can be a solution.

Basically, reinforcement learning is composed of the following elements: i) agent, ii) current state, iii) environment, iv) reinforcement signal and v) future state. The agent acts in the environment in an initial state, its action in the environment generates the reward signal and a state transition occurs. These states and their transitions, according to[2], can be mathematically modeled using Markov chains. The objective is to maximize the reward function using signals from the environment. This optimization problem can be decomposed and solved using the Bellman equation, known as the Temporal Differences (TD) method described in the equation below[2].

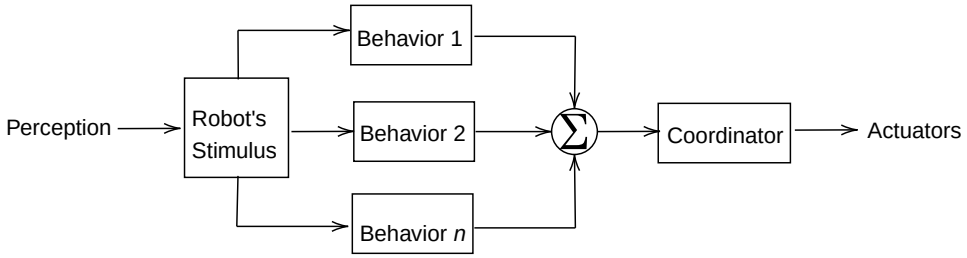

FIGURE 2 Behavior-Based Architecture

$$Q^{\pi}(s,a) = r + \gamma \ \max Q^{\pi}(s',a') \tag{1}$$

where r is the immediate reward obtained in the environment, γ the discount factor, s and a a current action state pair, respectively. s' and a' are a future state and future action pair.

The $Q(s,a)$ function is a value function, which evaluates the quality of an action state pair performed by the agent in the environment. According to Equation 1, it is possible to observe that the value of an action state pair (s,a) is measured by the immediate reward obtained with that action in the environment, as well as by the future reward that the future state (s') will provide.

Once $Q(s,a)$ is a function and neural networks are excellent universal approximants, this feature is used to approximate this optimal function Q through the time difference algorithm. Thus, in the DQN (Deep Q-Networks) algorithm, the deep neural network is used in a regression strategy. In the traditional Q-learning algorithm, the Q function is mapped through a table of visited states (tabular method). If the environment representation allows several states, the representation of these states through an array or table becomes unfeasible. Once the neural network directly learns the Q function, DQN becomes more attractive in environments with a large number of possible states. The estimation of a neural network $\hat{Q}$ is given by:

$$Q(s,a) \approx \hat{Q}(s,a,\theta) \tag{2}$$

where $Q(s,a)$ is the real value function, also called *target*, and $\hat{Q}(s,a,\theta)$ is the function approximated by synaptic weights θ.

The block diagram shown in Figure 3 illustrates a DQN neural network, where its inputs refer to the measurement of states in the environment and its output refers to the number of possible actions.

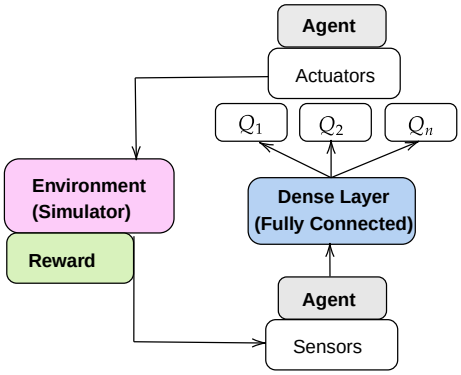

FIGURE 3 Block Diagram of a DQN Network

3.3 | Extended Kalman Filter

According to the state space control theory, a system can be described by a set of parameters or variables that characterize its physical aspects and behaviors. This set of variables are called state variables and are usually described mathematically by the vector **x**. Not every variable of interest can be measured directly in a process. Therefore, **x** can be estimated using a vector of measures **z**. KF is able to estimate an optimal state for the system, combining the measurement matrix **Z**, obtained with sensors, with the model in which the system evolves in time (instant k), that is, x_k. The Kalman prediction step is mathematically described by[24]:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}u \tag{3}$$

where **A** is the state transition matrix that describes how the system evolves from one state to another. If the system is controlled, the product of the control matrix **B** is added with the control vector u, elements that describe the external action of a control variable.

Each sensor has errors associated with the measurement process. As the KF is Gaussian in nature, that is, all its mathematical modeling is based on associations of Gaussian probability functions, the uncertainties are also modeled from this probability density function. The covariance matrix **P** represents the uncertainties of the model and its update equation is described by:

$$\mathbf{P}_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q} \tag{4}$$

where **Q** is a matrix that represents the noise external to the system, for example related to the lane where the autonomous vehicle will travel. For reasons of simplification, several authors adopt $\mathbf{Q} = 0$. Equations (3) and (4) are called the KF prediction step. They describe how the system gradually evolves over time. However, it is observed that there is no information about the sensory measurement of the process, that is, it is a simple model-based state transition. In this way, it is necessary to insert a KF step that correlates the observations measured by the sensors, called the correction step. The first equation of the correction step consists of calculating the Kalman gain, given by:

$$\mathbf{K}_k = \mathbf{P}_k\mathbf{H}^T(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R})^{-1} \tag{5}$$

where **H** is the sensory transfer matrix. This matrix is responsible for indicating which sensors will be used in the KF correction process. The sensory covariance matrix **R** presents the uncertainties related to the sensors used.
The state update with the measurements is given by:

$$\mathbf{x}_{k+1} = \mathbf{x}_{k+1} + \mathbf{K}_k(\mathbf{Z}_k - \mathbf{H}x_{k+1}) \tag{6}$$

The update of the covariance matrix is given by:

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_{k+1} \tag{7}$$

In EKF, the partial derivatives of the transition equation of states, called Jacobian matrix (**jF**), are calculated. The application of this matrix basically linearizes a nonlinear system over an operating point and is basically applied to the matrix **B**, which contains the nonlinearities of the system. The equation below presents the Jacobian matrix[25].

$$jF = \begin{bmatrix} \frac{\delta f1}{\delta x} & \frac{\delta f1}{\delta y} & \frac{\delta f1}{\delta z} \\ \frac{\delta f2}{\delta x} & \frac{\delta f2}{\delta y} & \frac{\delta f2}{\delta z} \\ \frac{\delta f3}{\delta x} & \frac{\delta f3}{\delta y} & \frac{\delta f1}{\delta z} \end{bmatrix} \tag{8}$$

4 | HYBRIDIZATION PROPOSAL: EKF-DQN ALGORITHM

In this section, we present the proposed algorithm for autonomous navigation of a mobile robot, as well as details about the simulated environment.

First, we present our proposal to adapt the EKF to the navigation problem. One of the main premises for using the KF (or EKF) is to know the dynamics of the system, that is, it is required to adopt a model of the system in state space. The dynamics of the differential robot is described through the following matricial equations:

$$\begin{bmatrix} x_k \\ y_k \\ \psi_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \psi_{k-1} \end{bmatrix} + \begin{bmatrix} \cos\theta\, d_k & 0 \\ \sin\theta\, d_k & 0 \\ 0 & d_k \end{bmatrix} \times \begin{bmatrix} v_{k-1} \\ \omega_{k-1} \end{bmatrix} \tag{9}$$

When comparing Equation 9 with Equation 3, we observe the presence of the matrices **A** and **B**, as well as the state vector and the control vector **x** and **u**. The states are basically the x, y coordinates of the Cartesian plane, and ψ is the yaw angle (Euller Angle) on the time variation d_k. The matrix **A** is an identity, while the control matrix **B** imposes the nonlinear dynamics of the system together with the control vector, which indicates that the vehicle can be controlled through its linear speeds and applied angles.

Now, imagine a vehicle moving and becoming closer to a curve. The DQN is expected to perceive the presence of this curve when this state is detected by a sensor and thus the best action will be taken for that particular state. Considering this situation, we propose the following reward function for the DQN learning algorithm:

$$R = \begin{cases} dist_{k-1} - dist_k & \text{if active} \\ -1, & \text{if collided} \\ +1, & \text{hit target} \end{cases} \tag{10}$$

$dist_k$ and $dist_{k-1}$ are respectively the distances from the robot to the target at instant k. This calculation allows you to reward actions that increasingly reduce the distance to the target. If the robot collides with a wall, it suffers a -1 point penalty. If the target is hit, the agent receives a 1 point reward. In the environment there are three targets: green, white and black circle and the main objective is to go through all three. Thus, for the proposed environment, the sum of immediate rewards exceeds a value of $\approx$ 10. As mentioned earlier, the DQN method needs to have the robot's state information. If only the Cartesian coordinates measured by the GPS and estimated by the Kalman filter are used, it would not be possible to obtain information about their orientation. For example, it would not be possible to identify a possible curvilinear movement. For this reason, the velocities, linear and angular (v_L, ω), of the agent are important components for the representation of states using the DQN method. Thus, it is proposed in this work that the state vector s of the DQN is composed as follows:

$$s = [x, y, v_L, \omega] \tag{11}$$

Note that the z coordinate is not applied to the state vector s as it has no relevant importance in the robot's movement. Its importance lies on the angular variation of ω (*yaw*), angle measured on this axis.

The proposed EKF-DQN hybrid algorithm allows the state (position information and robot speeds) to be perceived one or more steps before the sensory measurement, thanks to a prediction of the vehicle's movement. Thus, we propose that the DQN takes the EKF forecast as input. Considering a system with several states, this early perception can lead to an acceleration of learning. Algorithm 1 presents the steps of the proposed EKF-DQN algorithm.

The Block Diagram of Figure 4 visually presents the difference in the flow of sensory reading states and actions between the two algorithms. Note that we propose a modification to the Bellman's equation (see Equation 1) of the Q-Learning algorithm. It is possible to verify in Figure 4 that in the EKF-DQN algorithm there is a state prediction step before entering data to the neural network related to the DQN.Therefore, we propose that equation of updating the Q-value be given by:

$$Q(s, a) = R + \gamma \max Q(s_{k+p}, a) \tag{12}$$

Notice that in the second part of this equation, i.e., $\gamma \max Q^{\pi}(s_{k+p}, a)$, it is used the state s_{k+p} on $\max Q$ argument, that is, the p-steps ahead state estimated by the EKF. In the simulation section, we compare the performance of the control and learning algorithm with different values of p. Still in relation to this equation, the γ factor weights the value of the reward obtained in the future state (s_{k+p}) whose value is usually set between 0.95 and 0.99. Details of the exploration step, through the epsilon probability ϵ, as well as the choice of this hyperparameter will be discussed in Section 6. In the next section, details about the computer simulation and the modeling of the robot used as a mobile vehicle are presented.

4.1 | Simulation and Test Environment

For the simulation of the autonomous mobile vehicle, a differential robot, equipped with two wheels, model *Pioneer 3dx* was used. Figure 5 presents the frontal image of the robot, as well as the description of each element, such as the location of sensors and actuators. The dynamics of the movement of the differential robot happens from the difference in angular speed applied to its wheels, being able to turn left, right or forward if the speeds are the same.

Algorithm 1 EKF-DQN Algorithm

initialize **agent** DQN
define $EPISODES$ ▷ number of training episodes
for e in $EPISODES$ **do**
  start robot on state (s)
  initialize covariance matrix (P)
  $done$ = False ▷ non terminal state
  while $done$ = False **do**
    sensor state read (s)
    estimate EKF state (s_{k+1})
    with the probability ϵ select a random action a, otherwise; ▷ exploration step
    agent predict action $a =$ argmax$(\hat{Q}(s_{k+1}, a, \theta)$ ▷ DQN action prediction
    execute action a on simulator, observe reward R and estimate EKF next state (s_{k+2})
    agent memorize $(s_{k+1}, a, s_{k+2}, done)$ ▷ experience replay
    estimate a target $Q(s, a) = R + \gamma \ \max Q(s_{k+2}, a)$
    do $s = s_{k+1}$ and update covariance matrix (P)
    if $done$ is True **then**
      end episode e ▷ if a terminal state
    else
      continue e
    end if
  end while
  perform a Mean Square Error and gradient descent $(Q - \hat{Q}(s_{k+1}, a, \theta)^2)$ ▷ θ is a neural network parameter
end for

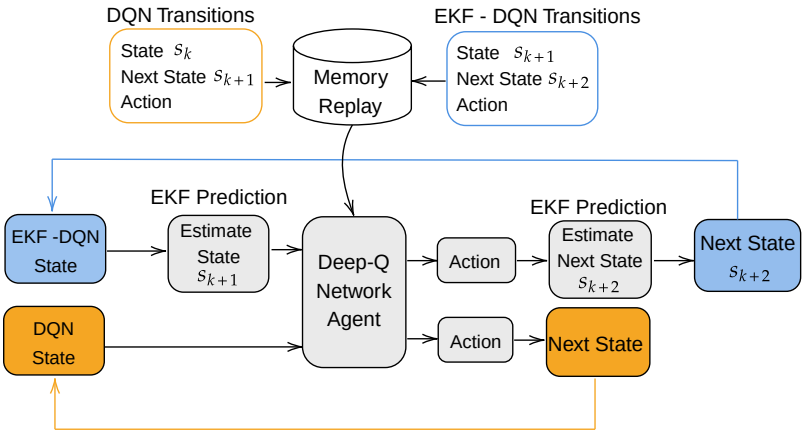

FIGURE 4 DQN and EKF-DQN Operation Flow.

To measure the robot's states, ultrasonic sensors, GPS (Global Position System) and IMU (Inertial Measurement Unit) are used. The function of the ultrasonic sensor is to identify robot collisions during the execution of the mission. Collisions are handled with terminal states that generate a penalty for the agent. GPS measurements directly enter the **x** state vector, as they provide the robot's coordinates in the plane. The IMU measures the angular and linear speed of the robot. It is worth mentioning that the angle of interest in measuring the IMU is the movement in relation to the z axis that indicates the robot's position. This Euler angle is known as yaw[20].

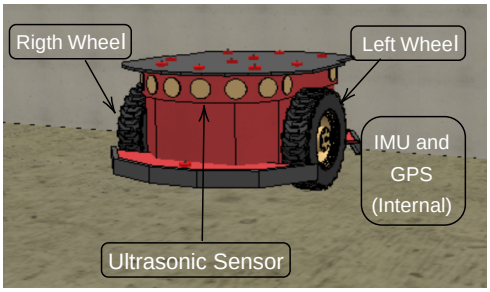

FIGURE 5 Pioneer Differential Robot

The simulation environment is illustrated in Figure 6 . There are three colored circles in the environment (green, white and black). The proposed mission for the robot is to navigate autonomously through the three circles without colliding with walls or objects. The black circle, next to a male avatar, is the final position in the quest.

FIGURE 6 Simulation Environment.

The simulator was programmed so that when the robot collides with an object, the simulation is interrupted, ending the episode. Thus, the agent is penalized and returns to its initial starting point again.

The actuators of the mobile robot are the motors present in its two wheels. Specifically in the Coppelia simulator, to act on a differential robot it is necessary to determine the angular speed of each wheel (left and right wheel). In this way, the actions projected for the DQN are basically a set of discrete actions w_l and w_r (rad/s), that are respectively the angular velocities of the left and right wheels. The action space is determined with the tuples of three possible actions. The size of the action space is a hyperparameter of the algorithm and can grow depending on the complexities of the environment and the movement required. For example, for a basic operation of the environment it is possible to use the following space of actions:($w_l = 3$, $w_r = 3$), ($w_l = 1.5$, $w_r = 1$), ($w_l = 1$, $w_r = 1.5$). These are examples of actions that could solve the problem in question, that is, they were enough to navigate the mobile robot in the given environment.

5 | SECURITY MODULE

The main purpose for using the safety module is to avoid collisions with people in motion during the navigation process. In this way, this algorithm is aggregated in the autonomous navigation architecture to help the main controller that uses the EKF-DQN algorithm. This security module is based on a supervised learning process, specifically, object classification using computer vision. In this way, the RGB-D camera type sensor is used.

The idea is that every time a person approaches the robot, a braking process takes place to avoid the collision. If the person moves away, or leaves the camera's field of view, the EKF-DQN main controller will work again. The safety module has priority to perform braking over commands from the main controller.

The security module uses the ResNet-50 convolutional neural network to detect the *shape* of human persons in the scene. Basically, a binary classifier is designed, which checks for the existence of people near or far (or without people) in the scene. As it is necessary to have the notion of depth in the scene, the RGB-D camera is used as a visual sensor. The idea is to perform the training from a *dataset* of RGB-D images. It was decided to combine images of real people in this dataset[26] and also to add synthetic images of people collected in the Coppelia simulator itself.

The[26] dataset contains more than 3,000 RGB-D frames acquired in a university hall from three vertically mounted *Kinect* sensors. The data mainly contain people walking upright and standing seen from different orientations and with different levels of occlusions. From these data, images were manually selected with people relatively close to the sensor and far from the sensor. Figure 7 presents an image with synthetic people collected in the Coppelia simulator.

FIGURE 7 Images of Synthetic People on the Scene.

Figure 8 presents an image of real people obtained through[26] experiments.

The training of the ResNet50 network is based on a fine tuning process, that is, the weights trained in ImageNet were used and tuned along the training epochs in all layers of the network. Only the final classifier of ResNet50 was set to a binary output.

6 | SIMULATIONS AND RESULTS

This section presents the results obtained via computer simulations using Coppelia VREP.

To evaluate the performance of the DQN and EKF-DQN algorithms in the environment, the following metrics are considered: **i) Average of obtained rewards** and **ii)** ***Success rate***, which is the number of completions of the proposed mission in a set of episodes. Through these metrics it is possible to assess whether or not the EKF-DQN accelerated the DQN method and conclude the hypothesis raised in the introduction of this article about performance enhancement. It is worth mentioning that in RL training the *Success rate* does not reach 100%, since the beginning of learning is composed of exploration phases (*exploration*),

FIGURE 8 Images of Real People[26].

where the actions are random. It is common to observe the convergence of the algorithm during training with the decrease in the exploration rate (ϵ), a phase called *exploitation*.

The same hyperparameter setting to the fully connected ANN (Artificial Neural Network) (see Figure 3) was used for the DQN and EKF-DQN algorithms. This fully connected network was parameterized with four layers, namely: two hidden layers of 200 neurons, an input with the number of neurons corresponding to the agent state space length and an output layer with the number of neurons corresponding to the size of the agent action space length.

The agent's training occurs through the memory replay technique, a strategy used in *off policy* reinforcement algorithms[27]. In this strategy, the agent stores state transitions that occur in episodes on a deck and samples them as training data. The efficiency of this technique depends on the exploitation of the agent that decays one ϵ step in each episode. The size considered for this memory (hyperparameter) is 6000 (six thousand) positions. Table 1 below shows the main parameters mentioned of the fully connected ANN, such as: i) learning rate (α), ii) exploitation rate (ϵ), iii) number of layers and iv) *batch size*.

TABLE 1 Parameters of the Artificial Neural Network.

α	N° Layers	*batch size*	*memory*	ϵ
1×10^{-3}	4	64	6×10^{3}	0,995

The algorithms were simulated in a seperated manner, that is, tests were performed for the DQN and later, for the EKF-DQN. The test track (environment), illustrated in Figure 6 , was used for both algorithms. Figure 9 below shows the first result of the reinforcement learning algorithms through a graph ($Score \times Episodes$) using the EKF-DQN algorithm. In this graph, the light blue line represents the sum of rewards obtained in each training episode and the dark blue line represents the moving average of the rewards, calculated over an interval of twenty episodes ($N = 20$).

It is possible to observe the tendency to maximize rewards over the episodes, so that stabilization occurs close to episode 70. It is also verified that the maximum reward obtained in an episode was $\approx$ 12 (twelve) points, where the vehicle mobile did not suffer any collision and reached the proposed objective. In this simulation, 150 training episodes were considered.

In Figure 10 , the graphs of the training moving averages by reinforcement learning are highlighted, for both the DQN and EKF-DQN algorithms ($Score \times Episodes$). It is observed that in the region close to episode 60 it is possible to verify the effect of acceleration, where the blue line (EKF-DQN) presented the highest average reward. In this region, the moving average of rewards obtained is $\approx$ 5 for the DQN (orange line), while for the EKF-DQN the average is $\approx$ 7. In this case, for the EKF-DQN, the mobile vehicle practically learned how to carry out the mission, frequently reaching the proposed target, while for the DQN, 70 more training episodes are required to reach this level. In addition, the stabilization of the DQN presented sudden oscillations throughout the 100th episode, where it is possible to verify an abrupt drop in the moving average.

In order to compare the DQN and EKF-DQN algorithms with a computationally simpler method and also *off-policy*, thus confirming the advantages of the proposed approach, the traditional Tabular Q-Learning was also implemented. The results of the learning evolution of this algorithm are shown in Figure 11 . It was necessary to increase the number of simulated episodes

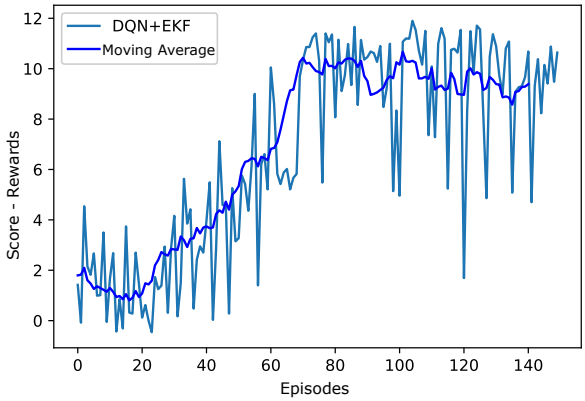

FIGURE 9 Rewards Obtained for the EKF-DQN Algorithm.

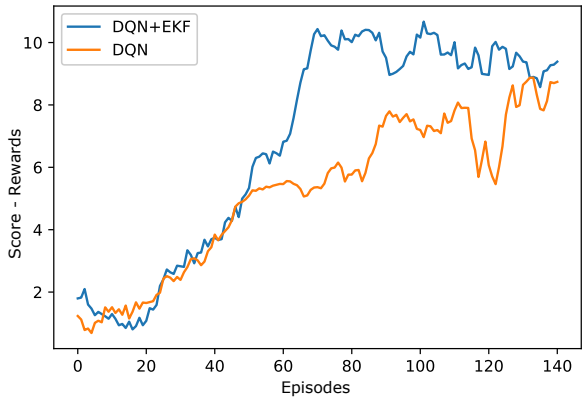

FIGURE 10 Comparison of DQN with EKF-DQN.

to observe the maximization tendency of the algorithm, since the matrix **Q** is relatively large and convergence is computationally expensive.

A relatively low average reward is observed in relation to the results obtained with DQN and EKF-DQN. The agent (robot) navigation is free through the environment, which makes the environment's state space ($s = [x, y, v_L]$) huge and wide for a tabular method. This factor makes it difficult for the Q function to converge to an optimal policy in traditional Q-Learning. It is also verified that in traditional Q-Learning the graph of rewards versus episodes stabilized at approximately $\approx$ 4.5. This result indicates that the robot learned only to reach the white circle in the simulation (Figure 6).

6.1 | Performance Metrics

Table 2 presents a summary of the performance between the algorithms considered in this article (DQN and the proposed EKF-DQN) based on the following metrics: *success rate* and *reward average*. We have evaluated the performance of the algorithm for different values of p. However, we noted that worse performance was obatined for values of p equal or greater than 3. Therefore, to be concise, two different values for the p-step are presented, one using s_{k+1} and other, s_{k+2}. The first consists of passing the one-step ahead prediction of the state given by the EKF to the DQN's memory replay. On the other hand, s_{k+2} represents an

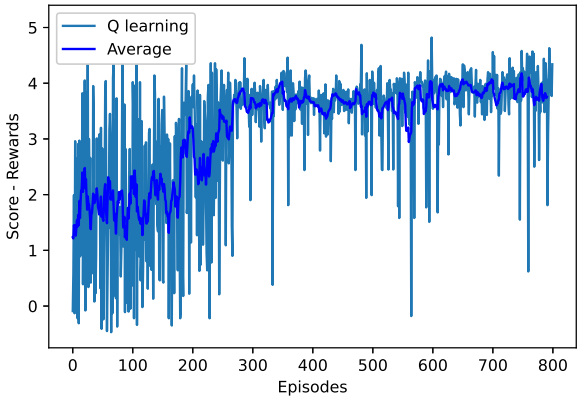

FIGURE 11 Tabular Q Learning.

attempt to speed up the process even more, passing the two-steps ahead prediction of the given by the EKF algorithm to the DQN's memory replay.

TABLE 2 Simulation Tests Results.

Number of Test	DQN		EKF-DQN s_{k+1}		EKF-DQN s_{k+2}	
1st Test	**Reward Average**	4.18	**Reward Average**	5.51	**Reward Average**	**5.66**
	Success rate	8.66%	**Success rate**	20.00%	**Success rate**	**22.66%**
2nd Test	**Reward Average**	3.89	**Reward Average**	**6.59**	**Reward Average**	5.17
	Success rate	5.33%	**Success rate**	**32.66%**	**Success rate**	9.33%
3rd Test	**Reward Average**	4.35	**Reward Average**	**4.32**	**Reward Average**	6.26
	Success rate	11.33%	**Success rate**	**14.00%**	**Success rate**	22.66%
4th Test	**Reward Average**	3.88	**Reward Average**	**5.89**	**Reward Average**	4.39
	Success rate	4.66%	**Success rate**	**22.66%**	**Success rate**	22.66%
5th Test	**Reward Average**	4.77	**Reward Average**	**5.50**	**Reward Average**	4.92
	Success rate	12.00%	**Success rate**	**24.66%**	**Success rate**	10.00%
Final Average	4.21		**5.56**		5.28	
Success rate Average	8.40%		**22.66%**		15.26%	

The DQN and EKF-DQN algorithms present random variables, that is, the neural network itself presents initializations mainly of synaptic weights that depend on probabilistic functions and can assume different values for different tests. This means that, in a given test, a network can boot into a relatively "good" region, making it easier for the agent to learn. Thus, it was decided to perform five different tests for each algorithm, thus generating greater reliability in the performance comparisons.

Another method used for the analysis of the results was the strategy of generating a single seed for the *framework* of random numbers in *Python*. Thus, the results obtained with the algorithms are fair and provided by an adequate manner, since randomness, especially in exploration, is generated by a single seed.

6.2 | Evaluation via Statistical Test

The *t-Student* test is a statistical method used to compare mean values between two different experiments, in order to analyze the difference between experiments[28]. In this article this technique is used to confirm whether or not the performance of the EKF-DQN algorithm is superior to that of the DQN. To this end, two hypotheses were raised and the values **t** and **p** (parameters) of the *t-Student* test were calculated to evaluate the average of the success rates of the experiments, as shown in Table 2:

- Hypothesis 0 (H_0): The performace of the EKF-DQN is not superior to that of the DQN algorithm. In this hypothesis, their success rates are equal.

- Hypothesis 1 (H_1): The performace of the EKF-DQN is superior to that of the DQN algorithm. In this hypothesis, their success rates are different.

The calculated p-value was 0.0102 and the **t** value was $-3,345$ for the five independent tests performed. This result indicates that with a confidence level of 95% we reject the null hypothesis, that is, it is concluded that the DQN network actually is benefited from the hybridization with the EKF with a significance level of 5%.

6.3 | Trajectory for the Proposed Mission

The cases and situations where the EKF-DQN outperformed the DQN are highlighted in bold in Table 2. It is possible to verify that in five tests, that is, for ten avaliable metrics, the EKF-DQN surpassed the DQN in nine, including four times the *average* and five times the *success rate*. The DQN was slightly higher only in Test Number 2. The results numerically prove the efficiency of the EKF-DQN, as already highlighted visually in Figure 5.

An example of the successful trajectory provided by the EKF-DQN algorithm to the mobile robot is shown in Figure 12 . It is possible to verify that the mobile vehicle left the initial position, passed through the intermediate circles (green and white) and arrives at the final circle (black) where it completes its mission.

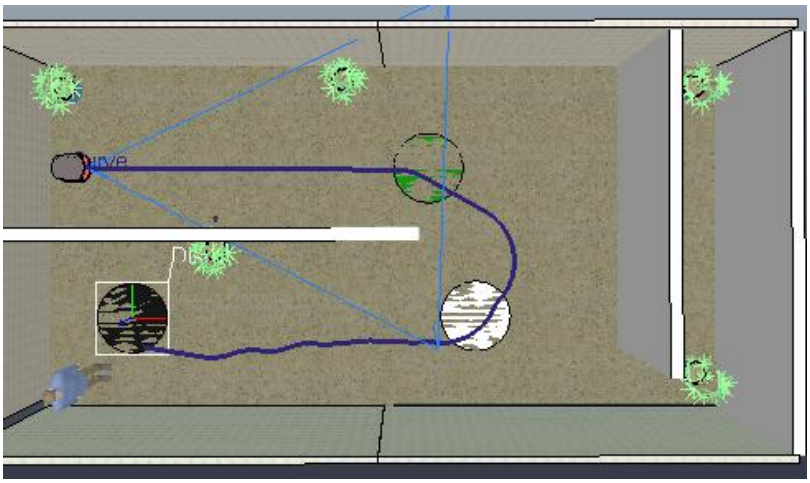

FIGURE 12 Trajectory of the Mobile Robot in the Simulation Environment.

The same trajectory, but with the EKF estimates in the Cartesian plane, is shown in Figure 13 . Even with the nonlinearities (curvilinear trajectory) performed by the agent, it is possible to notice a faithful approximation between the measured state (blue star) and the estimated state performed by the Extended Kalman filter (dashed orange line). It is worth to mention that it is only possible to obtain estimates of non-linear trajectories thanks to the Jacobian matrix of the EKF, which allowed the execution of the EKF-DQN algorithm for the proposed navigation task.

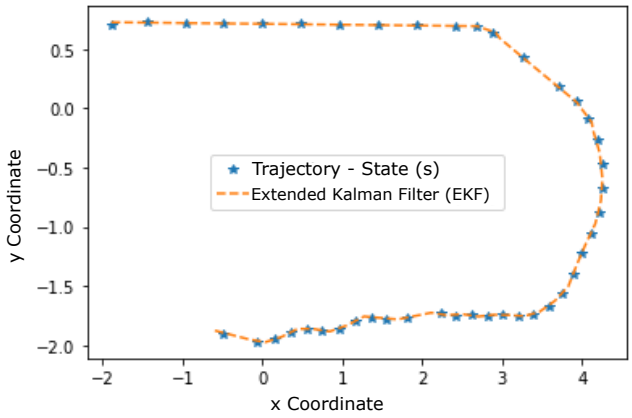

FIGURE 13 Trajectory of the Mobile Robot in the Cartesian Plane.

6.4 | Security Module Test

To test the safety auxiliary module, synthetic people moving on the track were added to the simulation environment. The main role of the auxiliary module is to brake the autonomous vehicle to avoid a collision. In this way, the ResNet50 network that was adjusted for this purpose works by receiving images from the RGB-D camera. If a person is identified next to the mobile vehicle, the power to the engines is cut off. Figure 14 presents the aforementioned simulation scenario.

It is observed that the safety module contributes positively to the reduction of collisions in the EKF-DQN controller. When compared to the system without a safety module, it is possible to evaluate the reduction of approximately 50% of collisions during the completion of the mission.

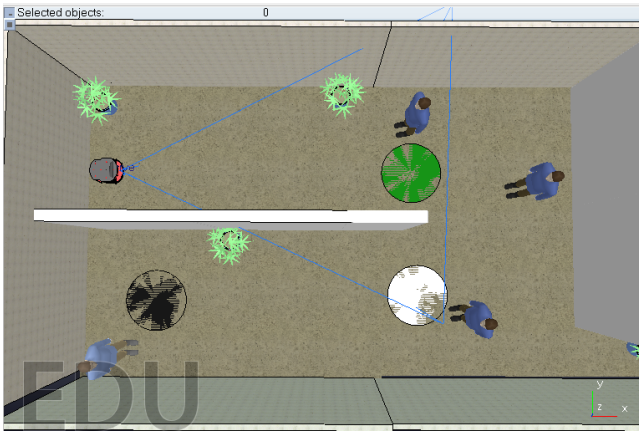

FIGURE 14 Synthetic people moving on the track.

7 | CONCLUSIONS

The results obtained in this work shows that the EKF-DQN algorithm presented relevant improvements in relation to the traditional DQN according to the established metrics. The agent training with the proposed reinforcement learning based algorithm confirmed the hypothesis raised at the beginning of this article if it is possible, through the use of EKF, to predict partial navigation states and to accelerate the convergence of the DQN by inserting predicted values of states in memory replay.

The performance of the EKF-DQN in terms of reward average and success rate was superior than the traditional DQN in all five tests performed for each algorithm. For the ten considered metrics, the EKF-DQN was superior in nine, confirming the efficiency of the proposed method and that the EKF-DQN provides the optimal navigation policy to the robot faster than the traditional DQN and Q-Learning algorithsm.

The extended Kalman filter (EKF) also presented state predictions that were faithful to the real measurements, even with curvilinear trajectories, showing to be an accurate and reliable algorithm for mobile vehicle navigation. Besides, another interesting characteristic of the EKF is its computational cost, its implementation basically does not generate high processing time, which makes it relatively fast and cheap compared to the DQN algorithm.

As a future work, we intend to reproduce the experiment through a hardware implementation. To this end, we are developing a practical test environment, similar to the simulation environment, so that the trained neural network can be applied to a real vehicle. It is also intended to merge the method with distance sensors, such as ultrasound or LIDAR, allowing the generalization of navigation in a real environment.

References

1. Vasilev I, Slater D, Spacagna G, Roelants P, Zocca V. *Python Deep Learning: Exploring deep learning techniques and neural network architectures with PyTorch, Keras, and TensorFlow, 2nd Edition*. Packt Publishing . 2019.

2. Krohn J, Beyleveld G, Bassens A. *Deep Learning Illustrated: A Visual, Interactive Guide to Artificial Intelligence*. The Addison-Wesley data & analytics seriesAddison Wesley . 2019.

3. Geron A., ed.*Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media . 2019.

4. Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. *Nature* 2015; 518(7540): 529–533.

5. Zhu K, Zhang T. Deep reinforcement learning based mobile robot navigation: A review. *Tsinghua Science and Technology* 2021; 26(5): 674-691. doi: 10.26599/TST.2021.9010012

6. Hasselt Hv, Guez A, Silver D. Deep Reinforcement Learning with Double Q-Learning. In: AAAI'16. AAAI Press; 2016: 2094–2100.

7. Wang Z, Schaul T, Hessel M, Hasselt vH, Lanctot M, Freitas dN. Dueling Network Architectures for Deep Reinforcement Learning. 2016.

8. Yue P, Xin J, Zhao H, Liu D, Shan M, Zhang J. Experimental Research on Deep Reinforcement Learning in Autonomous navigation of Mobile Robot. In: ; 2019: 1612-1616

9. Kalman RE, Others . A new approach to linear filtering and prediction problems. *Journal of basic Engineering* 1960; 82(1): 35–45.

10. Chrif L, Kadda ZM. Aircraft Control System Using LQG and LQR Controller with Optimal Estimation-Kalman Filter Design. *Procedia Engineering* 2014; 80: 245-257. 3rd International Symposium on Aircraft Airworthiness (ISAA 2013)doi: https://doi.org/10.1016/j.proeng.2014.09.084

11. Lichota P, Dul F, Karbowski A. System Identification and LQR Controller Design with Incomplete State Observation for Aircraft Trajectory Tracking. *Energies* 2020; 13(20).

12. Ahumada GA, Nettle CJ, Solis MA. Accelerating Q-Learning through Kalman Filter Estimations Applied in a RoboCup SSL Simulation. In: ; 2013: 112-117

13. Gao Y, Huang CM. Evaluation of Socially-Aware Robot Navigation. *Frontiers in Robotics and AI* 2022; 8. doi: 10.3389/frobt.2021.721317

14. Jin J, Nguyen NM, Sakib N, Graves D, Yao H, Jagersand M. Mapless Navigation among Dynamics with Social-safety-awareness: a reinforcement learning approach from 2D laser scans. *2020 IEEE International Conference on Robotics and Automation (ICRA)* 2020. doi: 10.1109/icra40945.2020.9197148

15. Sutton RS, Barto AG. *Reinforcement Learning: An Introduction*. The MIT Press. second ed. 2018.

16. Srichandan A, Dhingra J, Hota K. An Improved Q-learning Approach with Kalman Filter for Self-balancing Robot Using OpenAI. *J Control Autom Electr Syst 32, 1521–1530* 2021; 32(1).

17. Arıkan A, Kayaduman A, Polat S, et al. Control method simulation and application for autonomous vehicles. In: ; 2018: 1-4.

18. Hayes-Roth B. An Architecture for Adaptive Intelligent Systems. *Artif. Intell.* 1995; 72(1-2): 329–365.

19. Arkin RC. *Behavior-Based Robotics*. MIT Press . 1998.

20. Osorio F, Romero R, Prestes E, Wolf D. *Robótica Móvel*. Editora LTC . 2014.

21. Carreras M. *A Proposal of Behavior-Based Control Architeture With Reinforcement Learning for an Autonomous Underwater Robot*. PhD thesis. Spain; 2003.

22. Carreras M, Yuh J, Batlle J, Ridao P. A behavior-based scheme using reinforcement learning for autonomous underwater vehicles. *IEEE Journal of Oceanic Engineering* 2005; 30(2): 416-427. doi: 10.1109/JOE.2004.835805

23. Stefano. N. *Behavioral and Cognitive Robotics: An Adaptive Perspective*. Institute of Cognitive Sciences and Technologies, National Research Council (CNR-ISTC) . 2021.

24. Aguirre L. *Introdução à Identificação de Sistemas – Técnicas Lineares e Não-Lineares Aplicadas a Sistemas Reais*. Editora UFMG . 2015.

25. Dudek G, Jenkin M. *Computational Principles of Mobile Robotics*. USA: Cambridge University Press. 2nd ed. 2010.

26. Luber M, Spinello L, Arras K. People tracking in RGB-D Data with on-line boosted target models. In: ; 2011: 3844-3849

27. Lin LJ. *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis. USA; 1992.

28. Haslwanter T. *An Introduction to Statistics with Python: With Applications in the Life Sciences*. Statistics and ComputingSpringer International Publishing . 2016.

How to cite this article: Carlos D. Bezerra, Flávio H. T. Vieira, Anderson S. Soares (2022), Deep-Q-Network Hybridization with Extended Kalman Filter for Accelerate Learning in Autonomous Navigation with the Auxiliary Security Module, *Computational Intelligence*, *XXX;00:X–X*.