DOI: 10.48011/asba.v2i1.1597

Path Planning Collision Avoidance using Reinforcement Learning

Josias G. Batista * **Felipe J. S. Vasconcelos** **
Kaio M. Ramos ** **Darielson A. Souza** ** **José L. N. Silva** *

* *IFCE - Instituto Federal de Educação Ciência e Tecnologia do Ceará - Fortaleza, CE*
(e-mail: josiasbatista@ifce.edu.br, leonardo.silva@ifce.edu.br)
** *UFC - Universidade Federal do Ceará - Fortaleza, CE*
(e-mail: felipe.sousa.vasconcelos@dee.ufc.br, kaiomartins@dee.ufc.br, darielson@dee.ufc.br)

Abstract: Industrial robots have grown over the years making production systems more and more efficient, requiring the need for efficient trajectory generation algorithms that optimize and, if possible, generate collision-free trajectories without interrupting the production process. In this work is presented the use of Reinforcement Learning (RL), based on the Q-Learning algorithm, in the trajectory generation of a robotic manipulator and also a comparison of its use with and without constraints of the manipulator kinematics, in order to generate collision-free trajectories. The results of the simulations are presented with respect to the efficiency of the algorithm and its use in trajectory generation, a comparison of the computational cost for the use of constraints is also presented.

Keywords: Path Planning, Collision Avoidance, Reinforcement Learning, Robotic Manipulator, Trajectory Generation.

1. INTRODUCTION

The industry has found in automation an important ally for the modernization and growth of its activities. The use of robots has increased exponentially in order to perform tasks. In fact, the use of robots with manipulation functions has shown significant growth in the context of industrial production. The aforementioned diffusion of robots in an industrial environment has provided, over the years, that several methods were developed in order to monitor and to control mobile or manipulator robots, giving them the ability to operate in environments that are dangerous to humans (Pinto et al., 2014).

In this sense, optimization becomes a key part, since it is necessary to complete tasks with some desired characteristics, such as minimum time, shorter trajectory, minimal wear of the mechanical parts, among others. Thus, the trajectory performed by the robot is extremely important. In most cases, the cost functions, which are minimized during the optimization process, will have some kind of robot model as a component. This makes them applicable only for a given robot model and sometimes does not apply to be developed and implemented in other robot models (Csiszar, 2016).

The trajectory generation refers to a point in the workspace of the manipulator that can be translated into suitable conditions for a point in the joint space. The problem is to take the manipulator to the specified position, regardless of the initial position and the environment variables. This problem is more generally defined as robot navigation (Sciavicco et al., 1997). Trajectory planning consists of determining a curve in the workspace, connecting the desired, initial and final position of the actuator, avoiding any obstacle. The union of positions in the Cartesian space defines two types of profiles for linear and circular displacements (Khatib, 1986), (Batista et al., 2020b), (Batista et al., 2018).

Based on these concepts, the present work aims to generate the trajectory of a robotic manipulator using Reinforcement Learning (RL), the learning algorithm used is *Q-Learning*. The algorithm is used to generate trajectory in an environment where some obstacles are considered and the trajectories generated must be collision free.

This paper is organized as follows. Section 2 provides some information about the robotic manipulator and presents the model of the forward and inverse kinematics of the same. Section 3 presents the use of RL and the *Q-Learning* algorithm used. The algorithm simulation is presented in Section 4. Finally, the conclusions and future work are mentioned in Section 5.

2. LITERATURE REVIEW

The Q-Learning algorithm is a popular method of RL, and is used for collision prevention where an optimal action-state policy is calculated based on the Markov Decision Process (MDP). The Q-Learning is used to train robots to reach a certain target point, moving through obstacles and avoiding collisions with them. The actions considered are discrete and correspond to moving and rotating in different directions. However, the movement of the robot does not seem natural due to discrete control actions. In addition,

a simplistic reward function that depends only on actions taken and collision events should be used (Shim and Li, 2017).

RL is a popular area of research, widely used in several areas such as manufacturing technology, multi-agent technology or computer vision. Robotics RL is often applied to wheel control mobile robots, manipulating robots, or humanoid robots. The Q-Learning algorithm was used as a trajectory generator to reach a target and implemented in an industrial manipulator robot (Miljković et al., 2013).

In Park et al. (2007), real-time path planning is proposed, combining Probabilistic Roadmap (PRM) and RL to deal with uncertain dynamic environments and similar environments. A series of experiments demonstrate that the proposed hybrid path planning can generate a collision-free path even for dynamic environments in which objects block the pre-planned global path. It is also shown that hybrid path planning can adapt to similar environments previously learned without significant additional learning.

In Gu et al. (2017) it was presented that a recent RL algorithm based on training off-policy of Q functions can switch to complex 3D manipulation tasks and learn deep neural network policies with sufficient efficiency to train in physical robots real. It was shown that training times can be further reduced by parallelizing the algorithm on several robots that group their policy updates asynchronously. The paper presents an experimental evaluation and shows that the method can learn a variety of 3D manipulation skills in simulation and a complex door opening skill in real robots.

In the work from Nair and Supriya (2018), an approach called Modified Temporal Difference Learning for path planning and obstacle avoidance was proposed for static obstacles. The algorithm was developed in MATLAB software and path planning was implemented in a 4×4 grid environment. A GUI for the same is created, which access the user inputs like obstacle number, positions and type. The developed algorithm was compared with the conventional path planning Dijkstra's algorithm in the same environment. It was observed that computational complexity was less in the proposed approach compared to the conventional method.

In Jing et al. (2018) was propose a novel computational framework to automatically generate efficient robotic path online for surface/shape inspection application. Within the computational framework, a MDP formulation was proposed for the coverage planning problem in the industrial surface inspection with a robotic manipulator. A reinforcement learning-based search algorithm was proposed in the computational framework to generate planning policy online with the MDP formulation of the robotic inspection problem for robotic inspection applications. It was observed that the proposed method could automatically generate the inspection path online for different target objects to meet the coverage requirement, with the presence of pose variation of the target object.

In Liu et al. (2019) was proposed a knowledge fusion algorithm for upgrading a shared model deployed on the cloud. Then, effective transfer learning methods in Lifelong Federated Reinforcement Learning (LFRL) was introduced. LFRL was consistent with human cognitive science and fited well in cloud robotic systems. Experiments show that LFRL greatly improves the efficiency of reinforcement learning for robot navigation. The cloud robotic system deployment also showed that LFRL was capable of fusing prior knowledge.

3. ROBOTIC MANIPULATOR

The manipulator used in this work is called *Selective Compliance Assembly Robot Arm* (SCARA). He is a robot that has four degrees of freedom (DOF). Considering the first two joints from the base, it is noted that its workspace is the horizontal type (XY). As the joints revolve around vertical axes in a rotational manner it possible to make an analogy to planar robots of two DOF. Therefore, for this study only two DOF are needed. Figure 1 shows an image of the robot.

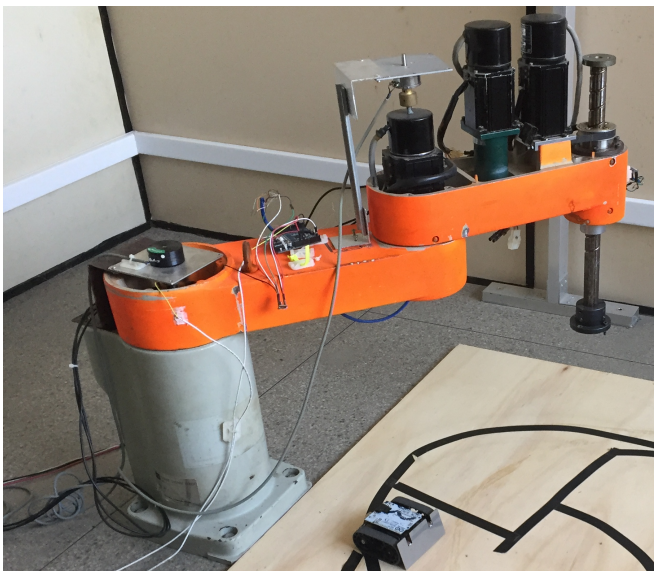

Figure 1. SCARA robotic manipulator.

3.1 Kinematics Model

The robot kinematics is defined as the study of its movement in relation to a reference system. Thus, it is about the analytical description of the robot spatial movement as a function of time, and in particular of the relationships between the position and orientation of its tool with the values that make its articular coordinates. The problem of direct kinematics is to determine the position and orientation of the manipulator actuator, in relation to a fixed reference coordinate system, known the joint values; the problem of inverse kinematics solves the configuration that the robot must adopt for a position and orientation of the known extreme (Romano, 2002).

3.2 Forward kinematics

In order to define the forward kinematics model, the Denavit-Hartenberg (DH) convention and the coordinator system of the manipulator, which are shown in Figure 2, are used. Thus, the needed parameters of angle and size can be selected (Hartenberg and Denavit, 1964).

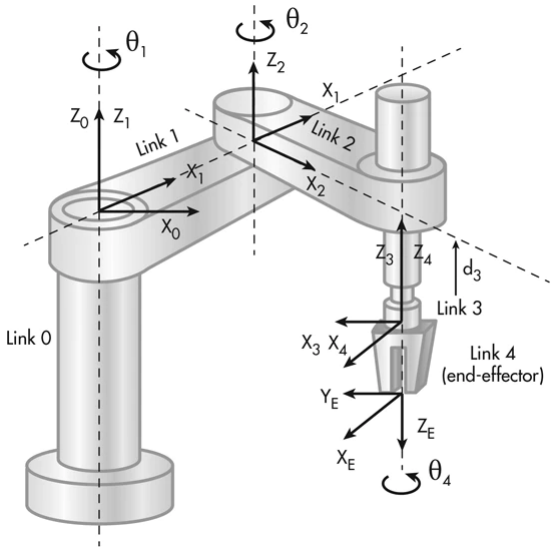

Figure 2. Robot joint coordinate systems SCARA Gonzalez (2017).

From the DH parameters the homogeneous transformation matrices of the coordinate systems are formed. These matrices are used in conjunction with the comparison of the geometric shape of the robot in order to determine the position of its end-effector in the workspace through the coordinates in the joint space as described by the equations

$$P_x = 0.35\cos(\theta_1) + 0.30\cos(\theta_1 + \theta_2), \quad (1)$$

$$P_y = 0.35\sin(\theta_1) + 0.30\sin(\theta_1 + \theta_2). \quad (2)$$

The equations (1) and (2) represent the solution to the problem of forward kinematics for the SCARA manipulator.

3.3 Inverse kinematics

The inverse kinematics is intended to solve a configuration that the robot must adopt in relation to the position and orientation of a known point. A simple way to solve this problem is to use the geometric method where is possible to determine the value of the joint angle, in order to correctly position the manipulator so that the trajectories in the joint space can be generated (Silva, 2016; Batista et al., 2020a).

From the direct kinematics equations (1) and (2) and applying some trigonometric transformations, one may find the inverse kinematics equations given by:

$$\theta_1 = tan^{-1}\left[\frac{P_y(L_1 + L_2\cos(\theta_2)) - P_x L_2\sin(\theta_2)}{P_x(L_1 + L_2\cos(\theta_2)) - P_y L_2\sin(\theta_2}\right], \quad (3)$$

$$\theta_2 = \cos^{-1}\left(\frac{P_x^2 + P_y^2 - L_1^2 - L_2^2}{2L_1L_2}\right), \quad (4)$$

where $L_1 = 0.35$ m and $L_2 = 0.30$ m are the values of the lengths of each joint of the manipulator as shown in Figure 2.

The equations (3) and (4) are the solutions to the SCARA manipulator inverse kinematics problem and are used to generate its trajectory.

4. USING OF RL IN THE GENERATION OF TRAJECTORIES

The RL is an approach that can be used to learn the robots movements, based on feedback received from the environment. The basic idea is inspired by natural learning, the way animals (including humans) learn. This technique involves several different attempts to solve a problem for later verification. When good results occur, the tendency is to repeat the behavior; if the resulting responses are not satisfactory, the direction to be followed is to avoid them (Sutton and Barto, 1998). In RL, the robot tries different actions (in fact, all the actions at its disposal) in all states in which it enters. For each action, there is continuous monitoring, so that all information is stored in some type of representation (usually, some type of table or matrix).

The RL theory is based on Markovian decision-making processes, although its ideas and methods can be extended and applied to more general applications and processes. An environment satisfies Markov property if its state summarizes the past compactly, without losing the ability to predict the future. That is, one can predict what the next state and the next expected reward will be, given the current state and action (Sutton and Barto, 1998). A Markov process is a sequence of states, with the property that any future state value prediction will depend only on the current state and action, and not on the sequence of past states.

A Markovian decision process is defined by a set $\langle S, A, P, R\rangle$, in which one have: the finite set of states S of the system, the finite number of actions A, and a P state transition model, which maps the state-action pairs into a probability distribution over the set of states, and finally, a R reward function, which specifies the reinforcement. The agent receives for choosing a particular action $a \in A$ in the state $s \in S$. The state s and a are the current actions that determine (i) the next state s' according to the probability $P(s'|s,a)$, and (ii) the reward $r(s,a)$ associated. Figure 3 shows the generic structure of the RL.

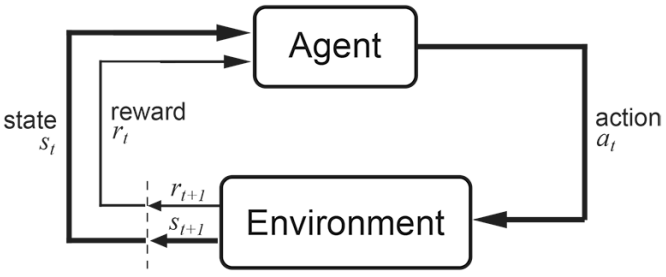

Figure 3. Generic structure of the RL Sutton and Barto (1998).

4.1 Q-Learning

In the RL algorithm used here, the robot and the environment interact in a discrete sequence of steps in time.

The state and the action at a given moment determine the probability distribution for the state s_{t+1} and the reinforcement r_t. The purpose of the robot is usually to choose actions in order to maximize a discounted sum of subsequent reinforcements:

$$r_t = \sum_{k=0}^{T} \gamma^k r_{t+k} \tag{5}$$

The actions are selected by the robot from a function state (control policy) $\pi : S \rightarrow A$. The value of utility of a state, given a policy is the expected reinforcement starting from the following state and following the policy:

$$V^{\pi}(s) = E_{\pi}(R_t | s_t = s) \tag{6}$$

and the optimal policy of actions is that sequence that maximizes the value of the state:

$$V^{*}(s) = \max_{\pi} V^{\pi}(s) \tag{7}$$

There is always at least one optimal policy that produces the maximum utility value in all states $s \in S$. Alongside these two state value functions, there are two share value functions:

$$Q^{\pi}(s, a) = E_{\pi}(R_t | s_t = s, a_t = a) \tag{8}$$

and

$$Q^{*}(s, a) = \max_{\pi} Q^{\pi}(s, a) \tag{9}$$

From Q^*, you can determine an optimal policy by making

$$\pi^{*}(s_t) = \arg\max_{a} Q(s, a) \tag{10}$$

Therefore, Q-Learning is an algorithm that allows one to automatically establish an action policy in a interactive. The algorithm converges to an optimal control procedure, when the Q state-action pair learning hypothesis is represented by a complete table containing the information value of each pair (Sutton and Barto, 1998). Convergence occurs both in deterministic and non-deterministic MDP. The Q-learning algorithm learns an optimal assessment function over the entire state-action pair space $S \times A$ (Ribeiro, 2002).

The function $Q(s, a)$ rewards the future action by choosing the action a in the state s, and is learned through trial and error according to the following equation

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha[r_t + \gamma V_t(s_{t+1}) - Q_{t+1}(s_t, a_t)] \tag{11}$$

where α is the learning fee, r is the reward, or punishment, resulting from taking action in the s state, γ is the discount factor and the term $V_t(s_{t+1}) = \max_a Q(s_{t+1}, a)$ is the utility of the state s resulting from the action, obtained using the function Q that has been learned to date .

The Q function represents the discounted reward expected when taking a a action when visiting the s state, and following an optimal policy since then. The generic *Q-Learning* algorithm is shown in Algorithm 1.

Algorithm 1 Pseudocode for Q-Learning

```
for every s ∈ S, a ∈ A do
    initializes table Q(s, a)
end for
generates an initial state
repeat
    selects an action a and runs
    receives reward r = r(s, a)
    observe the new state s'
    update table Q(s, a) as follows:
    Q(s, a) ← r + γ max_a' Q(s', a')
    s ← s'
until s be terminal
```

Until the stopping criterion is reached which can be a maximum number of episodes, or the shortest state sequence to achieve the task objective.

5. RESULTS

In this section, the results and simulations for the *Q-Learning* algorithm are presented.

5.1 Simulations

To perform the computer simulation, a space is created that corresponds to the manipulator workspace, shown in Figure 4. In this space, it is considered a starting point (SP), where the robot starts its movement to perform the trajectory, and a final point (FP), which refers to the target point, where the robot must arrive to complete the trajectory. The gray squares found in the workspace are considered obstacles, and the robot must deflect them to avoid collision.

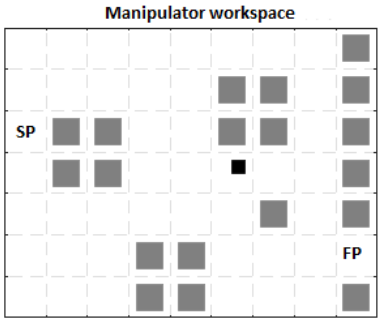

Figure 4. Workspace of the manipulator.

The actions that can be performed are shown in the tables below, and are performed at a constant speed and within a discrete time interval that is also constant. Table 1 shows the actions for the robot kinematic without constrains. Table 2 shows the possible actions for the constrained robot. The change in the direction of rotation of the joint is made so that the robot can avoid the obstacles and make the shortest path.

In Table 2, the third column represents the rotation movement of the manipulator joints, which can rotate 90°,

Table 1. Unconstrained actions.

Action performed	Motion
A1	Forward
A2	Back
A3	To the left
A4	On the right

Table 2. Constrained actions.

Actions performed	Motion	Rotation angle (°)
A1	Forward	0
A2	Back	0
A3	Rotation from the left	-90
A4	Rotation to the left	-90
A5	Right from the right	90
A6	Rotation back to the right	90

considering a reference point 0 (zero) to the left or to the right.

Simulations are performed with and without constraints on the manipulator kinematics for 100 episodes and the following values are used:

- Maximum number of steps per episode = 2000
- Learning rate, $\alpha = 0.1$
- Discount factor, $\gamma = 0.95$
- Probability of random selection, $\epsilon = 0.1$
- Eligibility trace, $\lambda = 0.5$

5.2 Results

The results of the simulations are presented by means of the figures 5 and 6 that show the learning curves during each training of the algorithm in the generation of the manipulator's trajectory.

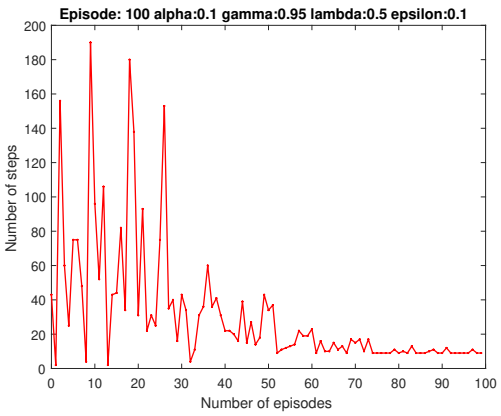

Figure 5. Algorithm without kinematic constraints.

Also extracted are the sequences of states that define the optimal policy, that is the optimum trajectory performed by the robot, which is 13 steps for the algorithm unconstrained and 8 steps for algorithm with constrains. The figures 7 and 8, below show the trajectories generated by the algorithm.

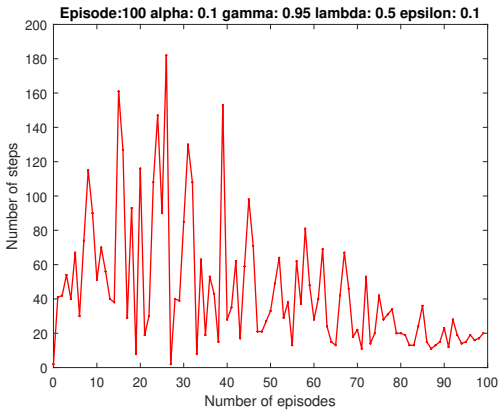

Figure 6. Algorithm with kinematic restrictions.

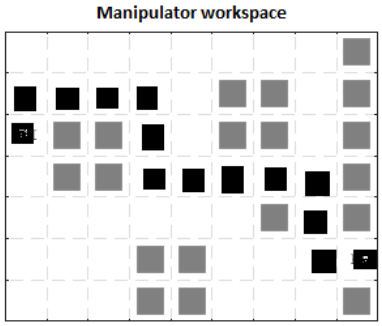

Figure 7. 13-step trajectory (algorithm without restriction).

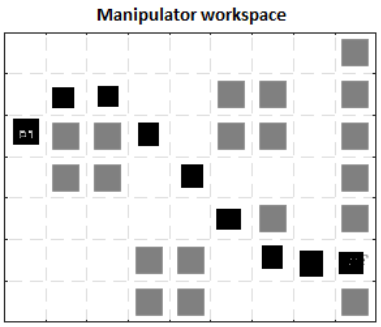

Figure 8. 8-step trajectory (restricted algorithm).

5.3 Discussions

A comparison was made between the Q-learning algorithm with and without constrains. The computational cost was verified and showed in Table 3.

Table 3. Computational cost.

Algorithm	Comp. cost [s]
Without restriction	121.780873
With restriction	169.463518

The RL Q-learnig algorithm also functioned as collision-free trajectory generator of obstacles placed in the manipulator workspace. The simulations showed and compared the two forms of the algorithm used, where the algorithm

with the kinematics constrained presented a better result, as shown in the previous section. The same algorithm also showed a better performance in the generation of the trajectory, that is, the trajectory had a smaller number of steps.

6. CONCLUSION

This work presented simulations using RL in order to generate trajectories of a robot. The algorithm used was *Q-Learning*, where a comparison was made between the algorithm with and without kinematic constrains. It can be extracted, as the main conclusion of this work, that an RL can be used efficiently in the generation of trajectories of manipulators. It can also be concluded that either the algorithm shows satisfactory results and can also be used to generate collision-free trajectories.

The authors are researching other ways to improve the algorithm, such as: considering all kinematic restrictions, such as the length of links; integration the *Q-Learning* algorithm with *Robotics System Toolbox*; comparison *Q-Learning* with other collision-free trajectory generation algorithms, such as artificially obtained fields; and to perform an implementation of the *Q-Learning* algorithm in the robotic manipulator.

ACKNOWLEDGMENTS

The authors thank IFCE, UFC and CAPES for the use of equipment, space and financial support.

REFERENCES

Batista, J., Souza, D., dos Reis, L., Barbosa, A., and Araújo, R. (2020a). Dynamic model and inverse kinematic identification of a 3-dof manipulator using rlspso. *Sensors*, 20(2), 416.

Batista, J., Souza, D., Silva, J., Ramos, K., Costa, J., dos Reis, L., and Braga, A. (2020b). Trajectory planning using artificial potential fields with metaheuristics. *IEEE Latin America Transactions*, 18(05), 914–922.

Batista, J.G., da Silva, J.L., and Thé, G.A. (2018). Can artificial potentials suit for collision avoidance in factory floor?

Csiszar, A. (2016). A combinatorial approach to the automated generation of inverse kinematics equations for robot arms. In *Automation Science and Engineering (CASE), 2016 IEEE International Conference on*, 984–989. IEEE.

Gonzalez, C. (2017). What's the difference between industrial robots? URL `https://www.machinedesign.com/robotics/what-s-difference-between-industrial-robots`.

Gu, S., Holly, E., Lillicrap, T., and Levine, S. (2017). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, 3389–3396. IEEE.

Hartenberg, R.S. and Denavit, J. (1964). *Kinematic synthesis of linkages*. McGraw-Hill.

Jing, W., Goh, C.F., Rajaraman, M., Gao, F., Park, S., Liu, Y., and Shimada, K. (2018). A computational framework for automatic online path generation of robotic inspection tasks via coverage planning and reinforcement learning. *IEEE Access*, 6, 54854–54864.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1), 90–98.

Liu, B., Wang, L., and Liu, M. (2019). Lifelong federated reinforcement learning: a learning architecture for navigation in cloud robotic systems. *IEEE Robotics and Automation Letters*, 4(4), 4555–4562.

Miljković, Z., Mitić, M., Lazarević, M., and Babić, B. (2013). Neural network reinforcement learning for visual control of robot manipulators. *Expert Systems with Applications*, 40(5), 1721–1736.

Nair, D.S. and Supriya, P. (2018). Comparison of temporal difference learning algorithm and dijkstra's algorithm for robotic path planning. In *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, 1619–1624. IEEE.

Park, J.J., Kim, J.H., and Song, J.B. (2007). Path planning for a robot manipulator based on probabilistic roadmap and reinforcement learning. *International Journal of Control, Automation, and Systems*, 5(6), 674–680.

Pinto, M.F., Mendonça, T.R., Olivi, L.R., Costa, E.B., and Marcato, A.L. (2014). Modified approach using variable charges to solve inherent limitations of potential fields method. In *Industry Applications (INDUSCON), 2014 11th IEEE/IAS International Conference on*, 1–6. IEEE.

Ribeiro, C.H.C. (2002). *Estudo de Desempenho de Algoritmos de Aprendizagem sob Condições de Ambigüidade Sensorial*. Ph.D. thesis, Instituto Tecnológico de Aeronáutica.

Romano, V.F. (2002). Robótica industrial. *São Paulo: Edgard Blücher*.

Sciavicco, L., Siciliano, B., and Dawson, D. (1997). Modeling and control of robot manipulators. *IEEE Transactions on Robotics and Automation*, 13(2), 315.

Shim, M.S. and Li, P. (2017). Biologically inspired reinforcement learning for mobile robot collision avoidance. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, 3098–3105. IEEE.

Silva, L.J.N. (2016). *Projeto e Realização de Controle Fuzzy para Manipulador Industrial Tipo SCARA e sua Avaliação de Desempenho a Luz da Norma ISO 9283*. Dissertação submetida a Coordenação do Curso de Pos-Graduação em Engenharia de Teleinformatica, da Universidade Federal do Ceara, como parte dos requisitos exigidos para obtenção do grau de Mestre em Engenharia de Teleinformatica.

Sutton, R.S. and Barto, A.G. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.