ISSN: 2182-2069 / E-ISSN: 2182-2077

Dynamic Path Planning Algorithm for Mobile Robots: Leveraging Reinforcement Learning for Efficient Navigation

Sivayazi Kappagantula[1], and Dr. Giriraj Mannayee[2*]

[1]Department of Mechatronics, Manipal Institute of Technology, Manipal Academy of Higher Education, (MAHE), Manipal, Karnataka, India; Department of Design and Automation, School of Mechanical Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India.
sivayazi.k@manipal.edu, sivayazi.kappagantula2018@vitstudent.ac.in,
https://orcid.org/0000-0001-6497-5390

[2*]Department of Design and Automation, School of Mechanical Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India. m.giriraj@vit.ac.in,
https://orcid.org/0000-0002-8726-2972

Received: January 07, 2024; Revised: February 27, 2024; Accepted: March 24, 2024; Published: May 30, 2024

Abstract

Traversing unfamiliar terrain presents a considerable challenge, particularly concerning the task of locating a viable pathway, regardless of its actual existence. This paper presents a novel navigation algorithm leveraging reinforcement learning, specifically the Markov Decision Process, to address the challenges of navigating dynamic environments. In contrast to traditional methods, this approach offers adaptability and efficiency in scenarios ranging from mobile robot navigation to complex industrial settings. The algorithm integrates an enhanced A* algorithm, showcasing its versatility in handling various tasks, from pathfinding to obstacle avoidance. To evaluate its effectiveness, the algorithm undergoes rigorous testing across multiple scenarios, comparing its performance with and without reinforcement learning. Through extensive experimentation, the algorithm demonstrates superior performance in terms of efficiency and adaptability, particularly in scenarios.

The results presented highlight the algorithm's learning progress and effectiveness in finding the shortest path. Notably, the algorithm's performance surpasses that of conventional approaches, underscoring its potential for real-world applications in mobile robot navigation and beyond. In conclusion, the proposed algorithm represents a significant advancement in navigation techniques, offering a robust solution for addressing the challenges posed by dynamic environments. Its integration of reinforcement learning enhances adaptability and efficiency, making it a promising tool for various industries and applications.

Keywords: Robotics, Path Planning, Algorithm, Reinforcement Learning.

1 Introduction

Robots are being deployed today in a wide range of industrial and civilian applications. Robots were already being used in high budget applications like defense, space explorations, mega-industries, etc. However, we can progressively see the use of robots in day-today life. A good example is from dispatching parcels from warehouse to delivery to your doorstep, robots perform these tasks already.

Journal of Internet Services and Information Security (JISIS), volume: 14, number: 2 (May), pp. 226-236.
DOI: *10.58346/JISIS.2024.I2.014*

*Corresponding author: Department of Design and Automation, School of Mechanical Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India.

Robots are being used for disinfecting office spaces during pandemics and deliver food and medicines to patients. Another good example is use of drones for photo shoots in weddings and mass gatherings. The applications seen today is just a just a beginning.

The significant rise of use of robotics can be attributed to developments in processing power of computers, development in sensory technologies, well established control system engineering [need citation], etc. Robots built by Boston Dynamics has achieved movement comparable to animal instincts (Feng et al., 2014; Spoorthi et al., 2021; Balamurugan & Nagarajan, 2017).

Autonomous navigation is a key concept in robotic systems. This determines how well the robot can navigate around its environment by avoiding obstacles while taking care of the physical constraints of the robot. For a mobile robot this involves travelling from the start point to its goal point. In the case of a robotic arm, this will involve adjusting the joints of the robot to attain a particular orientation. Often mobile robots have features like robotic arm and some kind actuator to carry out tasks it is intended to build.

Thus, navigation system would constitute of two tasks:

- Two find the shortest distance between the start point and end point keeping kinematic constraints into consideration. This is termed path planning.
- Maneuver the robot through the path considering its dynamics. This is termed as motion planning (Yang et al., 2016).

Many start-ups have emerged all over world providing a wide range of solutions to problems using robotics (Lee et al., 2022). These companies are focusing on scalable technologies which can be deployed in a wide range of environments. Thus, the key to building promising solutions is that they should be scalable.

Solving robot path planning problems is one of the key steps in building an autonomous robotic system. This involves the robot being able to perceive the environment using sensory inputs like IR sensors, depth sensors, etc. installed in the robot or some external system like Global Positioning System (GPS), or combining inputs using sensor fusion techniques, and compute desired path. All this must be done without any human intervention.

Most used path planning algorithms are node-based algorithms like Dijkstra's algorithm, A* algorithm, or modified version of these algorithms (El Halawany et al., 2013). These algorithms rely on finding the shortest path based in some heuristic factor in an environment divided into grids or nodes. Sampling based algorithms include methods like Probabilistic Roadmap Method (PRM), Rapidly exploring Random Tree (RRT). These algorithms are used to perceive an environment and convert it into forms which the robot can used to solve for finding path. Often node-based algorithms are used in combination with these algorithms. Potential field algorithm is a good example for mathematical model-based algorithm. Most of the advanced path planning algorithms are inspired from these basic methods.

Today, the complexity of the problem's robotics must solve is increasing exponentially. The environment the robot must work in also is going to be more unstructured. Conventional path planning algorithms rely on perfect knowledge of the environment. However, we need robots to work in spaces like a shop floor, warehouse, office, hospitals, roads, where the dynamics can never be modelled completely. The dimensionality of the environment also increases manifold, making node based or sample-based algorithms painstakingly slow. In these situations, the algorithms will not perform as desired and most likely fail (Yu et al., 2020; Mathur et al., 2024).

Machine learning techniques are being used to solve complex problems in robotics. Camgozlu & Kutlu (2023) introduce CNN for classification problem with maximum efficiency. Use of machine learning in control systems is a very good example where the performance of the robot can be improved by a significant factor (Perrusquía et al., 2019). Further, controllers can tune itself by learning the environment using these techniques. The ability of machine learning to solve such complex problems makes it a perfect solution for path planning and a lot of work is being carried out in this field.

Machine learning techniques like supervised learning, unsupervised learning and deep learning require a huge database of the expected environment for the model to train on (Juma et al., 2023). Projects in the early stages of development will not have this kind of data to train on. Further, during the training phase, some supervision by human would be necessary. This will increase the cost of implementation. Reinforcement learning is a subset of machine learning where a model can be trained using a reward and penalty method.

Reinforcement learning is a technique where a model is trained by allowing it to perform some actions in an environment. The environment in turn would respond with some reward or penalty. Based on this the model can be trained as to which action to be performed in a particular state, just as a human or an animal would learn to perform a task.

The true potential of reinforcement learning was realized when DeepMind Technologies developed an algorithm based on reinforcement learning where a computer learned to play Atari games all by itself as a human would learn (Mnih et al., 2013; Mishra & Kumar, 2023; Watrianthos, 2020).

Training a robot based on this method does not require a large data set. Robot can perform some action in real-time or in simulation and determine whether the action is desired based on the rewards the algorithm awards. Moreover, this learning can happen in simulation, thus reducing the cost of developing a prototype.

Robotic solutions developed by a certain company or organization are deployed in environment which are often different from each other. Thus, the path planning algorithm must be able to adapt to the given environment without much effort. This is crucial for easy deployment of robots and its ability to scale up or scale down as per requirement. Ability of the system to adapt to the dynamic requirement in an industry or warehouse is a necessary parameter for success of a product.

Thus, the path planning algorithm should be able to perform the following tasks:

- Learn the given environment based on heuristic data provided like distance, obstacles, kinematic constraints of the robot, etc.
- After learning is complete, the algorithm should be able to compute optimum path from its current orientation to desired (goal) orientation, without the need the learn the environment again.
- Algorithm must do this by considering power requirement, battery availability, payload, etc.
- Scale up or scale down environment area as per working conditions.

There will not be any previous data of the environment for training, thus the algorithm should be based on reinforcement learning, i.e., learn directly from the environment.

2 Reinforcement Learning

1) Basic Concepts

Reinforcement learning is a way to train a model without writing codes that maps actions with a particular situation, or better called a **state** *s*. Instead, the model is provided with all sets of **actions** *a* it can perform. Actions can be a robot moving front, back, left, and right or some actuator moving. The entity performing the action is called an **agent.**

When an agent performs an action, the **environment** responds with a **reward** *r*. This reward is a score which tells the agent how well it has performed. The aim of the agent is to obtain maximum reward with each action it takes. Thus, an agent records the rewards for each action state pairs and plans a strategy to obtain maximum rewards. Some other key elements of reinforcement learning are described in the following sections.

A policy π is a formula which determines what action *a* the agent will select at a particular state *s*. Policy can be based on some mathematical calculation, lookup tables, or based on a probability distribution. An agent can have multiple policies. Agent will also have to optimize policy, i.e., determine which policy π would fetch maximum reward when action *a* is selected for a state *s*.

While rewards determine what is the immediate benefit the agent can get, a value function for a state represented as *v(s)*, is the total reward the agent can expect in the long run. Value function is estimated by repeatedly performing action *a* in a state *s* till *v(s)* converges to a satisfactory value.

2) Markov Decision Process

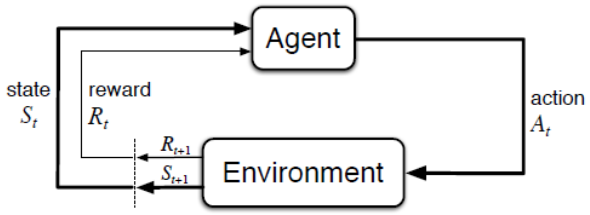

Figure 1: Architecture of Markov Decision Process [7]

MDP is a method for framing problems where an agent interacts with an environment and learns it. This is a continuous process where the agent selects an action A_t time *t* and its state changes from S_t to S_{t+1}. The environment in turn responds with a reward R_t *based* on which the agent learns the environment and can select action which can give maximum rewards.

Thus, using the MDP model Shown in Figure 1, agent can determine sequence of state action pairs which can maximize rewards. Most process can be described as a finite MDP, i.e., finite number of state action pairs and rewards (Sutton & Barto 2018). R_t and S_t *can* be describes using discrete probability distribution as they are dependent only on the preceding state action pair. This can be described as *p(s`, r | s, a)*. This function describes the dynamics of the MDP.

3) Bellman's Equation

Most reinforcement algorithms involve estimation of value functions for its states. Value function of a state is a number which tells the agent how good it is to be in that state. In other words, it is a measure of estimated future rewards in that state (Sutton & Barto 2018). Rewards the agent can expect in future

depends on what action it chooses. This is defined by the policy π the agent uses. Thus, value functions w.r.t the policy also which is denoted as $v_\pi(s)$. This is defined as state-value function for policy π.

Similarly, $q_\pi(s,a)$ is the action-value function for policy π defined as expected return from s and taking action a following policy π.

Relation between the value function, MDP dynamics and rewards is given by Bellman's equation.

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \quad (1)$$

Bellman's equation is used in most reinforcement learning algorithm to find value functions.

$$v_\pi(s) = \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \quad (2)$$

3 Path Planning Algorithm based on Reinforcement Learning

The problem of path planning is described as an MDP. This model is used for developing the algorithm based on reinforcement learning. The following steps are followed for this,

- Here, each node is described as a state s_1, s_2,.... and so on.
- Agent can move to any of the nodes connected to the current node. Each of these will be actions for that state.
- Determine rewards for each action corresponding to the state.
- Select policy for action selection.
- Define dynamics of the environment. The algorithm is modelled as a model-based algorithm.

Required data are fed externally to the algorithm to keep the code universal, that is it can work in any environment. Thus, an algorithm is developed as described in *algorithm 1*.

Pseudo Code for the algorithm Proposed

Algorithm 1 Path planning algorithm based on Reinforcement Learning

Input: Node data, Path distances in.csv files, start and goal nodes

Output: Path from start to goal node

1: Create adjacency matrix A for nodes from input

2: Determine a policy π for action selection

3: Calculate $p(s`, r \mid s, a)$ for each action state pair

4: Determine optimum value function $v_\pi(s)$ for each node under policy π using the Bellman's equation

$v_\pi(s) = \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]$

5: Step 4 repeated till $v_\pi(s)$ converges to a satisfactory accuracy ($\varepsilon = 10^{-5}$)

6: Path is found by moving from current node to the neighboring node having highest $v_\pi(s)$

7: Each new node is appended to a list which gives the path generated

4) Policy

A single policy has been used for action selection in this phase. This has been done for the simple reason to avoid complexity at this phase. The policy is Agent can choose to move to any of the nodes connected to the current node with equal probability.

Thus, if a node has n neighboring nodes, then the dynamics of MDP is given by:

$$p(s`, r \mid s, a) = \frac{1}{n} \quad (3)$$

5) Reward Calculation

Reward had to be allocated dynamically for each action state pairs to keep the program universal. The most obvious way is to link the reward to heuristics of each node, as it was done in Algorithm 1. Thus, equation 3 is used.

6) Value Function Estimation

Bellman equation is used for finding the value functions.

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \quad (4)$$

We are using a single policy only. Thus equation 4.5 can be written as,

$$v_\pi(s) = \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \quad (5)$$

4 Results and Discussions

Developed algorithm, has been initially tested with simple maps to test whether path can be found between the start node and the end goal. For this a few test cases are considered which were visually distinct from each other. Initial test cases had fever nodes. This was progressively increased to test the scaling up capability of the algorithm. Further, the duration for execution of the program was also noted to assess practicality of the algorithm.

1) Test to Determine Ability to Determine Shortest Path

Initially a simple environment is used for testing whether the algorithm can find the smallest path. This map is particularly used to test path planning algorithm in its initial stage to find out whether the shortest path is found, since the shortest path between any two nodes can be visually determined, Test environment 1 shown in Figure 2.

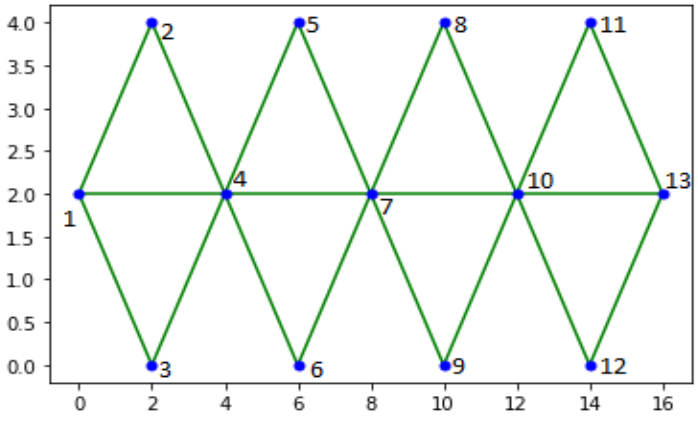

Figure 2: Test environment 1

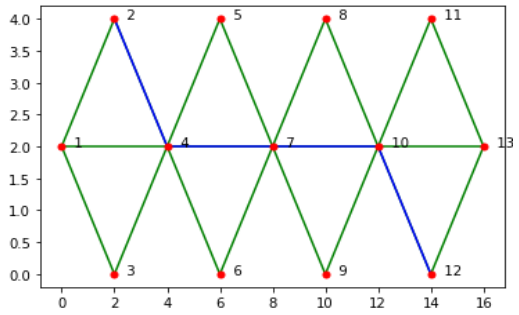

Figure 3: Shortest path found successfully

Here, the start node was given as 2 and the goal node as 12. Green lines are available paths, and the blue lines are the found path. It is observed from Figure 3 that the algorithm can find the shortest path from any two nodes. This exercise was repeated with various start and goal nodes and every time the algorithm found the shortest path successfully.

2) Test to Determine Usage in Non-uniform Map

A more overlapping, non-uniform map is used for testing in case 2. This is to make sure the algorithm can perform in real world scenarios where randomness is common. Figure 4 is used for further testing.

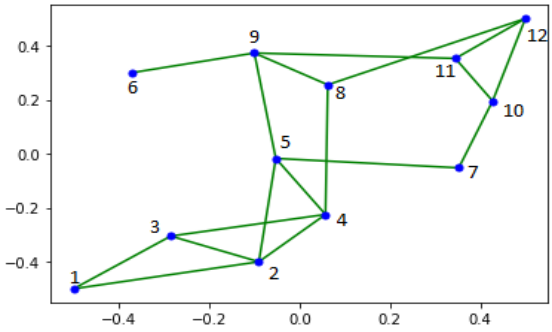

Figure 4: Test environment 2

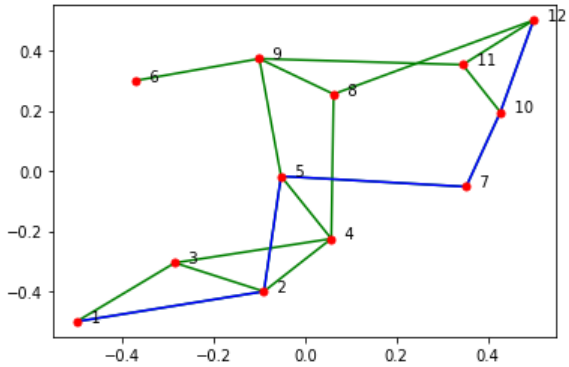

Figure 5: Path is found

In this example, start node is 1 and goal node 12. This was repeated with various combinations and the algorithm was able to find the path successfully. Thus, this test case shows that the algorithm can work in diverse conditions. Table 1 tabulates the value function for each node in the map shown in Figure 5.

Table 1: Node value functions when goal is set to node 12

Node Number	Value function
1	-4.14213562
2	-0.76150547
3	-1.24388723
4	1.50628909
5	2.42991083
6	1.07307444
7	4.28149932
8	4.98570793
9	3.86602698
10	6.86501595
11	7.86034465
12	10

3) Test to Determine Scalability

Next test case is carried out to determine the scalability of the algorithm. A grid like map with 80 nodes is chosen to emulate warehouse like environment. Some nodes are not accessible to emulate obstacles, Testing on larger map shown in Figure 6.

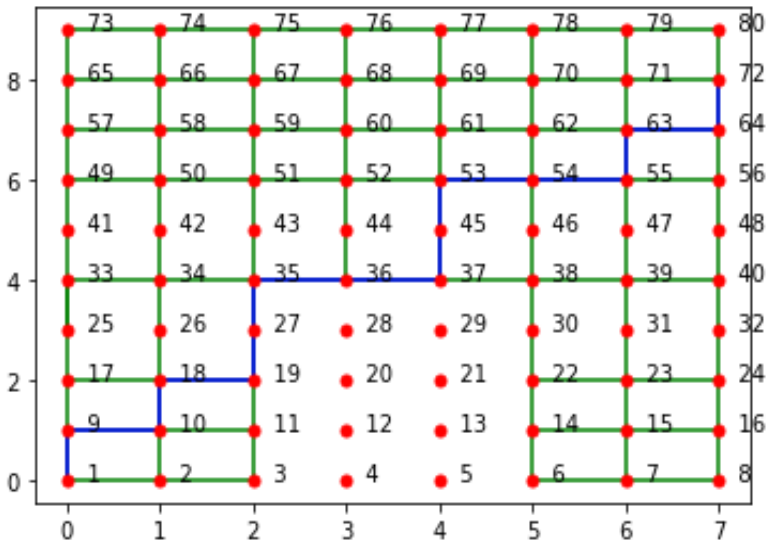

Figure 6: Testing on larger map

The algorithm was able to calculate path between any two nodes easily. The algorithm can be scaled up without any modifications to the existing framework. Thus, achieving another objective.

Time required for execution is calculated withing the program. The below screenshot of the terminal shows the execution time shown in Figure 7.

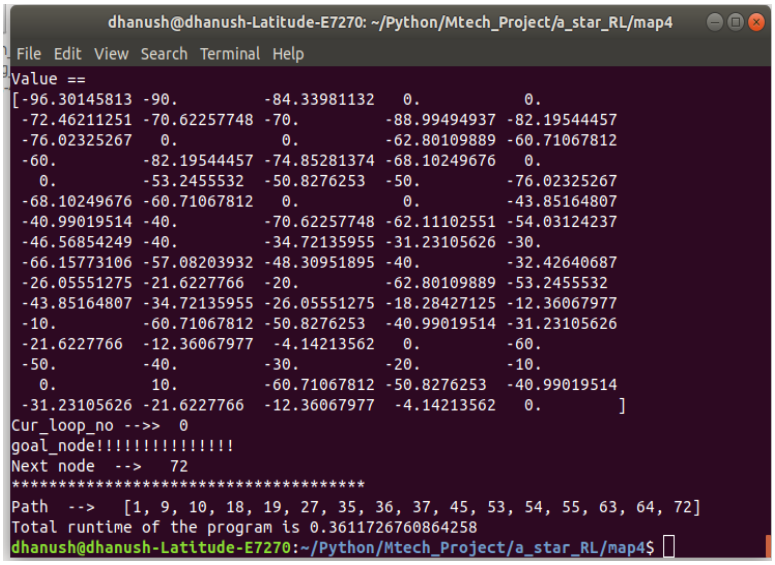

Figure 7: Execution time

Total Runtime of the Program in 0.36 s

The program execution required 0.36 s. This duration is not significant, and the path can be calculated without compromising the practical aspects of using the robots in real world scenarios. Note that this time is including the programming required to visualize the results and plotting. Execution time can be further optimized by removing these statements.

The work carried out has been able develop an algorithm based on reinforcement learning. This algorithm is found to be complete in the environments tested so far, that is, it can find a path between two nodes if it exists. The algorithm also been able to take inputs dynamically and compute path.

The algorithm has been able to demonstrate its capability to perform in various maps successfully. Thus, it can be used in non-uniform and robust environments as per real-world scenarios.

The algorithm is scalable, and the execution time is within acceptable limits for practical usage in industries, warehouses, and domestic applications. This justifies use of reinforcement learning as compared to other machine learning methods as the time required for learning or training is significant. This would limit repeatability of a system.

Use of reinforcement learning has eliminated the requirement of training data, which otherwise is quite a cumbersome exercise in machine learning. This can help in reducing the time required for developing and deploying robots as per requirements.

The algorithm has the capability of to take the dynamic aspects of the situation like robot power usage, battery capacity, terrain limitations, production demand, etc. and calculate the best possible path. This eliminates the limitations of the traditional path planning and control system techniques and thus can be used in developing highly efficient systems enabling robustness and complexity in handling.

5 Conclusion

The algorithm must be tested in actual robots in simulations and the real world. For simulations ROS platform can be used and robots like TurtleBot can be used. This will further test the robustness of the algorithm. A plugin of the algorithm can create and integrate with move_base package of ROS. An

actual robot can be built with capability to map its environment and sense the required data like velocity, obstacles etc., and can be used with this algorithm.

Today, robot fleets are used for execution of task whose volume is huge. This algorithm can be extended for multi-robot path planning. This can be done by dynamically calculating the value functions and use of reinforcement learning for local path planning as well. This would add the functionality of avoiding dynamic obstacle with respect to individual robots. Further this would help in swarm robotics for accomplishing complex tasks with better coordination (Giovanni et al., 2021).

Currently nodes are pre-determined or generated using algorithm like PRM or RRT. The node formation may not be optimum for the task in hand. reinforcement learning can be used for node generation in a mapped environment so as the execute any task optimally. Further, the nodes can be generated such that the path for the robot would be smooth, and minimum power would be required at the robots end to accomplish a task. This can be with respect to the reinforcement learning concepts used, further development is possible. The model of the problem can be altered by adding more possible actions. The algorithm makes uses of basic concepts of reinforcement learning with a single policy. Multiple policies can be evaluated to arrive at a better algorithm.

References

[1] Balamurugan, R., & Nagarajan, N.R. (2017). Automatic Robotic ARM using Hand Gestures. *International Journal of Communication and Computer Technologies (IJCCTS), 5*(2), 43-45.

[2] Camgozlu, Y., & Kutlu, Y. (2023). Leaf Image Classification Based on Pre-trained Convolutional Neural Network Models. *Natural and Engineering Sciences, 8*(3), 214-232.

[3] El Halawany, B.M., Abdel-Kader, H.M., Tag Eldeen, A., Elsayed, A.E., & Nossair, Z.B. (2013). Modified a* algorithm for safer mobile robot navigation. *In 5th International Conference on Modelling, Identification and Control (ICMIC)*, 74-78.

[4] Feng, S., Whitman, E., Xinjilefu, X., & Atkeson, C.G. (2014). Optimization based full body control for the atlas robot. *In IEEE-RAS International Conference on Humanoid Robots*, 120-127.

[5] Giovanni, L., Angelica, M., Fabio, M., Andrea, S., Antonio, L.M., Endika, G.U., & Vıctor, M.V. (2021). Cybsersecurity Issues in Robotics. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JOWUA), 12*(3), 1-28.

[6] Juma, J., Mdodo, R.M., & Gichoya, D. (2023). Multiplier Design using Machine Learning Alogorithms for Energy Efficiency. *Journal of VLSI Circuits and Systems, 5*(1), 28-34.

[7] Lee, H., Enriquez, J.L., & Lee, G. (2022). Robotics 4.0: Challenges and Opportunities in the 4th Industrial Revolution. *Journal of Internet Services and Information Security (JISIS), 12*(4), 39-55.

[8] Mathur, G., Nathani, N., Chauhan, A.S., Kushwah, S.V., & Quttainah, M.A. (2024). Students' Satisfaction and Learning: Assessment of Teaching-Learning Process in Knowledge Organization. *Indian Journal of Information Sources and Services (IJISS), 14*(1), 1–8.

[9] Mishra, D., & Kumar, R. (2023). Institutional Repository: A Green Access for Research Information. *Indian Journal of Information Sources and Services (JISIS), 13*(1), 55–58.

[10] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). *Playing atari with deep reinforcement learning*. https://doi.org/10.48550/arXiv.1312.5602

[11] Perrusquía, A., Yu, W., & Soria, A. (2019). Position/force control of robot manipulators using reinforcement learning. *Industrial Robot: the international journal of robotics research and application, 46*(2), 267-280.

[12] Spoorthi, A., Sunil, T., & Kurian, M. (2021). Implementation of Lora based Autonomous Agriculture Robot. *International Journal of Communication and Computer Technologies (IJCCTS)*, *9*(1), 34-39.
[13] Sutton, R.S., & Barto, A.G. (2018). *Reinforcement learning: An Introduction*. A Bradford Book, Cambridge, MA, USA.
[14] Watrianthos, R. (2020). A Compact Hybrid Ring Patch Antenna for Fixed Communication Applications. *National Journal of Antennas and Propagation (NJAP)*, *2*(1), 13-18.
[15] Yang, L., Qi, J., Song, D., Xiao, J., Han, J., & Xia, Y. (2016). Survey of robot 3D path planning algorithms. *Journal of Control Science and Engineering*, 2016. https://doi.org/10.1155/2016/7426913
[16] Yu, J., Su, Y., & Liao, Y. (2020). The path planning of mobile robot by neural networks and hierarchical reinforcement learning. *Frontiers in Neurorobotics*, *14*, 63. https://doi.org/10.3389/fnbot.2020.00063.

Authors Biography

Sivayazi Kappagantula, Assistant Professor, Department of Mechatronics Engineering, Manipal Institute of Technology, Manipal Manipal academy of higher education. Mr. Sivayazi Kappagantula has received B.E degree in Mechanical Engineering from Andhra University, Vishakhapatnam in the year 2011, M. Tech degree in Mechatronics from Vellore Institute of technology (VIT), Vellore in the year 2013 and pursuing PhD in Autonomous Robot Path planning and Reinforcement Learning from VIT, Vellore. In Aug 2013, he joined Accenture Pvt ltd, Pune as an Associate Software Engineer worked on programming of Cloud Computing Technology. Post to that he, joined as an Assistant Professor and Programme Coordinator for M. Tech, Robotics in Defence Institute of Advanced Technology (Deemed University), Pune, India, An Autonomous Organization, Department of Defence Research & Development, Ministry of Defence, Govt. of India. Later, he joined Senior Executive - Robotics Motion, Tata Advanced Systems Limited, Bangalore as a swarm robotics algorithms designer and developer for Aerospace applications. Currently he is working as an Assistant Professor, Manipal Institute of Technology, Manipal teaching various Robotics courses. He authored and co-authored more than 15 technical papers in International Journals/International Conferences. His research interests include Robotics, Autonomous Systems, Mobile Robotics, Reinforcement Learning. He is currently working on Robot Motion Control, Path Planning and Reinforcement Learning.

Dr. Giriraj Mannayee, Professor, Department of Design and Automation, School of Mechanical Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India. Dr. Giriraj Mannayee pursued his PhD from Anna University, Chennai in Cyber Physical System. He is currently working in Multidisciplinary Engineering like Robotics, Cyber Physical System, MES, Cloud Manufacturing. He published many national and international publications across the globe.