IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

A Comparative Analysis of Path Planning Algorithms for Industrial Manipulators in Dynamic Environment

Harshinii Rajesh
Department of Robotics and Automation
B.E Robotics and Automation, PSG College of Technology
Coimbatore
22r213@psgtech.ac.in

Sowndara T
Department of Robotics and Automation
B.E Robotics and Automation, PSG College of Technology
Coimbatore
22r246@psgtech.ac.in

M.P.Anbarasi
Department of Robotics and Automation
B.E Robotics and Automation, PSG College of Technology
Coimbatore
mpa.rae@psgtech.ac.in

Abstract*— Path planning is a critical component in the deployment of industrial robot manipulators, particularly given the dynamic and interactive nature of real-world environments. This paper provides a comprehensive review of various path-planning algorithms, categorizing them into classical, sampling-based, optimization-based, and machine learning-driven approaches. Each algorithm is evaluated on terms of computational efficiency, path optimality, adaptability to dynamic environments, and scalability. Under classical methods, algorithms such as Dijkstra's and A* are analyzed for their precision and deterministic nature, while sampling-based techniques like Rapidly-exploring Random Trees (RRT) and Probabilistic Roadmaps (PRM) are explored for their efficiency in high-dimensional spaces. Optimization-based strategies such as Particle Swarm Optimization (PSO) are evaluated for their ability to balance computational efficiency with path optimality. Emerging machine learning-based approaches, including Reinforcement Learning (RL) and Deep Learning (DL), are highlighted for their adaptability and potential in handling complex, dynamic environments. The paper also discusses key challenges in collision avoidance, real-time adaptability, and computational overhead, offering insights into future research directions to advance path-planning techniques for industrial robotics. This also brings into light the merits and demerits of each approach with the hopes of establishing how appropriate each algorithm is suitable for specific industries.

Keywords*—algorithms, adaptability, collision free, comparison, dynamic environment, efficiency, industry, path planning, robot manipulators.

I. INTRODUCTION

Path planning is not only a foundational aspect of robotics but also a primary function necessary for enabling robots to perform collision-free navigation in complex environments. Industrial robot manipulators, such as those used in assembly lines, welding tasks, and material handling, must operate with high levels of precision and stability. These tasks are often conducted in unpredictable terrains or cluttered environments, where real-time adaptability becomes crucial. Robot navigation has been a popular research area for quite a number of years in robotics. It gives an explanation of how one can find a reasonable way that the robot can use to move itself from one position to another while avoiding certain factors like time, energy used and distance. The main issues arising due to path planning are dynamic obstacles, uncertainty in environment and the fact that path planning should always be performed in real time, which makes it very difficult to decide on what is the best algorithm for that particular job. In industrial settings, such as factories with constantly shifting workstations, conveyors, and human workers, robot manipulators must quickly adapt to changing environments. These challenges necessitate the development and refinement of algorithms that can handle dynamic, uncertain conditions while maintaining high performance. In industrial settings, path planning for robots also requires careful consideration of human-robot interactions. Robots frequently share spaces with human workers. This poses significant challenges in path planning, as robots need to not only avoid stationary and dynamic obstacles but also ensure the safety of nearby workers. Algorithms need to account for the unpredictable movements of humans while maintaining efficiency and avoiding collisions.

A lot of work has been accomplished in this field and numerous sets of the path planning algorithms have been introduced to deal with these problems. This paper aims to review and compare the most widely used algorithms, grouped into four main categories: Classical methods which include algorithms like Dijkstra's and A* focus on deterministic graph-based approaches to find the shortest path, Randomized algorithms include RRT and PRM, Swarm-based algorithms like PSO explores multiple potential paths simultaneously, and Machine Learning-based approaches such as Reinforcement Learning (RL) and Deep Learning (DL) techniques have emerged as promising solutions for dynamic environments[4]. These approaches will be evaluated according to their computation time, the optimality of the paths, their capability for modification, their applicability at large scale industrial environment, as well as their prospects for further development. The high computational costs associated with these methods are a

barrier to their real-time deployment in industrial settings, highlighting the need for continued research into optimizing machine learning approaches for faster path planning. One major area of research is the integration of hybrid approaches that combine the strengths of different algorithmic methods. Another important research direction is the development of more robust real-time algorithms capable of handling a wider range of dynamic obstacles and uncertainties, particularly in environments with human-robot interactions. As industries continue to adopt more advanced robotic systems, the demand for efficient, real-time path planning solutions will only grow. The paper is structured as follows: Section II gives an initial information on robot manipulators. Section III contains the various path planning algorithms. In this context, Section IV presents a comparative review of these algorithms while Section VI recalls the main conclusions of the paper.

II. Robot Manipulators and its Integration with Path Planning

At present, industrial robot manipulators are widely used all over the world in manufacturing industries where accuracy, consistency and speed are the elements that are needed to perform their functions. These robots mostly include segmental units which are hinged together to allow the segments to move in space. It is the nature and number of joints of a particular robot that define it can have a particular number of degrees of freedom that dictate the possibility or the ease with which it can reach certain points within its working envelope. Manipulators include the, articulated robots, SCARA robots and Cartesian and each type offer different motion profiles and operational characteristics. Regardless of the specific nature of the manipulators that are utilized in various applications ranging from automobile manufacturing to production of semiconductor devices, all have the same activity of providing feedback control of the end point of the manipulator together with the tool which is mounted to it and interacts with the environment. Both kinematics and dynamics make it easier to explain how these industrial robots are programmed to move and with the surroundings.

Kinematics on the other hand deals with the aspects of the motion of objects, without giving any credit to the reason behind it. In the robot manipulators, Kinematics targets the joints and links and in so doing determines the position as well as orientation of the end tool. Dynamics is extended to examining the forces and torques which lead to motion in the manipulator. In kinematics analysis, only the dimensions of the robot are considered whereas in dynamics, other factors such as mass of the links, moment of inertia, friction in joints and force acting on the links are taken into consideration.

Path planning is a crucial part of robotic systems, especially in industrial applications, where robot manipulators are employed for tasks such as welding, painting, and assembly lines. In such applications they require the capability to navigate and perform in a complex environment particularly steering clear of obstacles including other robotic structures or equipment to achieve the desired results. In all such scenarios path planning algorithms must consider issues pertaining to joint constraints, real time optimality and adaptability with added aspects related to optimality and collision free smooth operations.

III. Classification of Path Planning Algorithms

A. Classical Approaches

Path planning is a fundamental problem in robotics, where the goal is to find a collision-free path for a robot to move from a start location to a goal location in a complex environment. This problem has been extensively studied in the field of robotics, and various algorithms have been developed to tackle it. Among the classical approaches, Dijkstra's algorithm and its extension, A* algorithm, are two of the most well-known and widely used methods.

Dijkstra Algorithm

The Dijkstra algorithm is a classical method to determine the shortest path between points or nodes in a graph. These algorithms were named for Dr. Edsger W.Dijkstra, a Dutch computer scientist and software engineer. In the field of robotic systems, especially in the use of industrial robots in dynamic environments Dijkstra is used for finding the path that costs least by providing the best way for the robot to move. While it is effective in static environments, the challenge arises when applying Dijkstra in dynamic settings where the obstacles in the environment are subjected to change. The main limitation of Dijkstra in dynamic environments is its need for frequent recalculations to update its path for moving through obstacles in the environment. Each time the environment changes, the algorithm has to redefine the entire path, which can be a time consuming and slower process. However, Dijkstra is valued for its accuracy, as it always gives the shortest path when the environment is fully known. To improve the algorithm's performance in dynamic settings, path smoothing techniques are applied. Path smoothing does away with sharp angles that would in any case be characteristic of Dijkstra's grid centered path determination algorithm. This is particularly important in a condition of change because the smooth, uninterrupted motion of the robots leads to higher efficiency and reduced stress. The basic idea is that the distances of all nodes to get to the source node are stored, and can be altered if those distances are found to be changeable.

The algorithm proceeds as follows: First the starting node is identified as S and a distance of 0 is set to it while all the other nodes are set to '∞'. Choose the unvisited node with the least distance (initially the source), add it to the visited nodes, and take into consideration the new path to all its neighbors if shorter paths are found through the current node[9]. Repeat this process until all the nodes are added to the visited list.

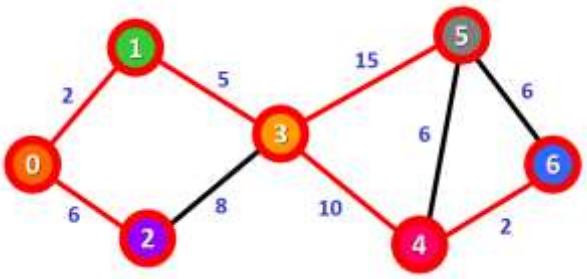

Fig 1: Algorithm for establishing the shortest achievable

Dijkstra's Algorithm cannot be used with negative weight graphs since the edge weights are added in the calculations. If the graph is containing the negative of weights, then the algorithm will not run properly. If a node is S (visited), the present path is the shortest to that node in the whole network graph. Problems may be caused by positive and negative weight which alter the total weight as a node is being processed.

A* Algorithm

The A algorithm* is established based on Dijkstra by adding heuristics to calculate the shortest path more effectively. While Dijkstra focuses on overall edges or nodes have no relation to the target, the A* technique has a better influence on the heuristic function that leads the robot to the path found in fewer numbers of steps. This makes it particularly beneficial for use in environments where the obstacles are constantly changing. A* enables an industrial robot to adapt more quickly as a result of shifting focus on the paths that are closer to the goal when the obstacles of the environment are

moving, instead of reconstructing the entire graph. This makes A * more efficient in the situations that expect higher changes in virtually all the contexts presented. A* can be enhanced with path smoothing techniques similar to Dijkstra, ensuring that the robot's movement remains smooth even in real-time dynamic environments. This smoothing is particularly helpful in eliminating the sharp edge motion so that instead the robot can smoothly go forward and backward with little variance in velocity [3]. In dynamic environments where speed, safety, and flexibility are important the incorporation of A* with smoothing techniques is useful for real-time motion planning in robotics. The environment is described in terms as a graph, where nodes are configurations or positions of the end-effector and edges correspond to possible movements between two nodes. An approach of the robot is to traverse this graph from a start to a goal node while possibly dynamic obstacles exist or shift. A* uses both the actual cost to reach a node and an estimated heuristic of the cost to reach the goal.

The algorithm proceeds as follows: two main lists are used that are the Open list (O), which contains all nodes yet to be evaluated, and the Closed list (C), which includes nodes that have already been fully assessed. Initially, only the start node is placed in the Open list. The algorithm utilizes cost functions to guide the path search, where the total cost function is defined as f(q)=g(q)+h(q). Here, g(n) represents the known cost from the start node to node n, and h(n) is the heuristic estimate from node n to the goal. As long as the Open list is not empty, the algorithm searches for the current node in the Open list with the least total cost. If this node happens to be the goal, the path is considered found. The current node is then removed from the Open list and added to the Closed list. Next, the algorithm generates valid neighbouring configurations for the current node, ensuring these neighbours align with the robot's kinematic constraints and joint limits. Each neighbouring configuration is checked for collision-free movement between the current and neighbour nodes. If a collision is detected, the neighbour node is discarded. For each valid neighbour, the total cost is calculated. If the neighbour is not in the Closed list, and the cost to reach it via the current node is lower than any previously computed cost, the neighbour's cost is updated, and it is added to the Open list if not already present.

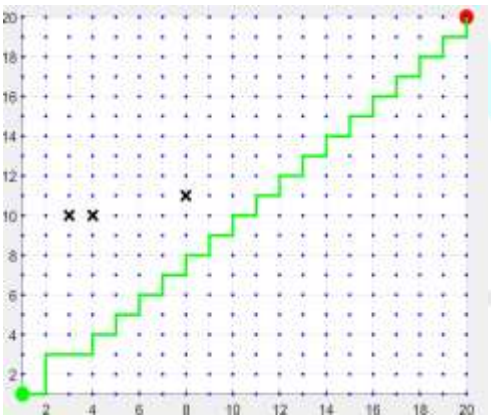

Fig 2: MATLAB Output for A* algorithm in a dynamic environment.

In dynamic environments, where obstacles can move unpredictably, the algorithm continuously monitors the positions and velocities of these obstacles. If a dynamic obstacle is predicted to intersect the planned path, the algorithm replans by re-evaluating the Open list and adjusting the path accordingly. Once the goal is reached, the optimal path is constructed by backtracking from the goal node to the start node using the parent nodes. This approach ensures that the robot navigates effectively in dynamic, cluttered environments while adhering to constraints and avoiding collisions in real-time.

B. Randomized Algorithm

While classical approaches such as Dijkstra's algorithm and A* algorithm have been widely used, they can be limited by their computational complexity and inability to handle high-dimensional spaces. In recent years, randomized algorithms have emerged as a promising approach to path planning, offering a trade-off between computational efficiency and solution quality. Among the randomized algorithms, Rapidly-exploring Random Tree (RRT) and Probabilistic Roadmap Method (PRM) are two of the most popular and widely used methods.

Rapidly-exploring Random Tree (RRT)

From LaValle's work, one of the fundamental paths planning algorithms is the RRT algorithm [2]. This algorithm can be used for robot path planning especially in complex high-dimensional space. It achieves this by gradually creating a tree solution in increments and using random points within the structure of an environment to link them to the nearest node in the tree. This tree helps the robot to determine its optimal path to the goal when it is among the obstacles. Using the concepts of a tree structure, every node is a state, and the branches connecting these nodes are the edges. Since industrial robots are applied for dynamic environments, the result indicates that large space can be efficiently sampled using RRT even for complicated scenes in real-time path planning consisting of obstacle avoidance or finding the shortest path.

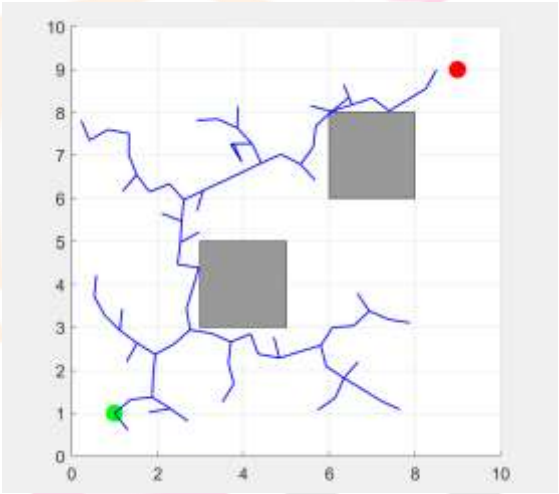

Fig3: MATLAB implementation of the Rapidly-exploring Random Tree (RRT) algorithm for path planning in a 2D environment with obstacles.

However, RRT often provides near-optimal or jerky type of paths, and hence additional path smoothing methods are required to make these paths suitable for real life uses in manufacturing. Its applicability in a dynamic environment since they are able to replan the paths whenever the obstacle is moving makes it even more appropriate for use in industries where changes are common. The main advantages of RRT are probability, easy to implement, perfectly scalable and so on. Several variants of the RRT algorithm such as RRT Biased and RRT* are used to enhance computational efficiency [1].

Probabilistic Roadmap (PRM)

PRM algorithm is a highly beneficial technique for solving path planning issues in both static and dynamic environments. PRM is a multiple query algorithm where it pre defines its roadmap for other path planning work. This makes it very much suitable for what industrial robots used in dynamic environments. Instead of complex maps, these

roadmaps consist of points sampled in the robot's environment, linked with simple paths. A roadmap is a data architecture in the form of an undirected graph which is a set of connected nodes or vertices and edges. The PRM algorithm mainly consists of two parts: construction phase and query phase to select the optimum and non-interference path. For the first steps, the nodes are taken at random, and these points are joined up in what could be described as a graph. In the query phase, the robot identifies the precomputed graph from the start to the goal position. The main strength of PRM can be seen particularly in the context of multi-query problems: the same robot or different robots at different times pass through the same environment [5]. In dynamic environments, the roadmap is updated online to enable the robot to navigate around the obstacles and perform an efficient path replanning.

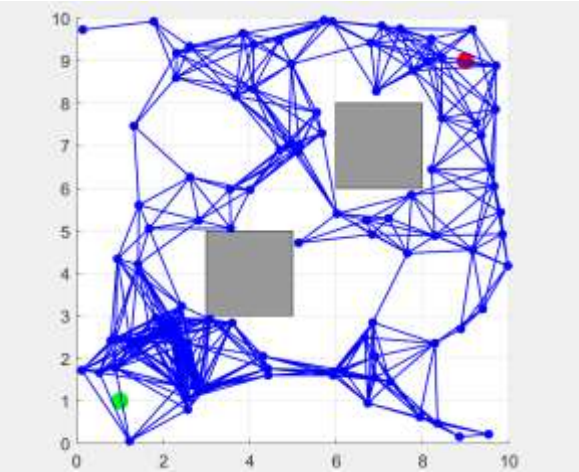

Fig4: MATLAB output of the roadmap

C. Swarm based algorithm

Particle Swarm Optimization (PSO) is an optimization technique originally inspired by the behavior of a social swarm like bird flocking, fish schooling. In dynamic environments, where obstacles may move or change over time, PSO is particularly useful due to its adaptability and robustness. In the case of industrial robots, such as robotic arms or mobile robots, PSO simulates a swarm of particles, each showing a potential path solution. These particles search in the solution space and as such move with respect to their best positions (known as personal best) and the best positions of other particles in the swarm (known as global best). Therefore, PSO leads the swarm toward an optimal path that minimizes a specific cost function, such as path length, energy consumption, or obstacle avoidance. It's an essential requirement for industrial robots to constantly update their plans when navigating through the environment. PSO allows robots to adapt to such environments by updating their paths in real-time, based on the movement of particles in the solution space. As particles evaluate new positions and share information, the swarm converges on an optimal path that avoids obstacles and ensures efficient movement [7]. An example for such a case is an industrial robot needed to move within a workstation in a manufacturing unit to deliver goods to various workstations; the robot must also navigate around other robots, machines or workers on the floor.

The PSO algorithm operates as follows: firstly, initialization of a set of particles, where each particle represents a potential path for the robot. These particles are randomly assigned initial positions and velocities within the search space. The fitness of each particle is then evaluated using a fitness function that typically considers several factors, including the particle's proximity to the target, its distance from obstacles, and the smoothness of the proposed path. Based on these fitness evaluations, each particle updates its position and velocity according to two key influences: its own personal best position and the global best position discovered by the entire swarm. This update process allows particles to explore new paths while converging toward better solutions. The algorithm iterates over several generations, gradually guiding the particles closer to an optimal or near-optimal path. The process continues until a valid solution is found or a predefined stopping criterion, such as a maximum number of iterations, is met. This approach leverages collective intelligence to efficiently search for feasible and optimal paths in complex environments.

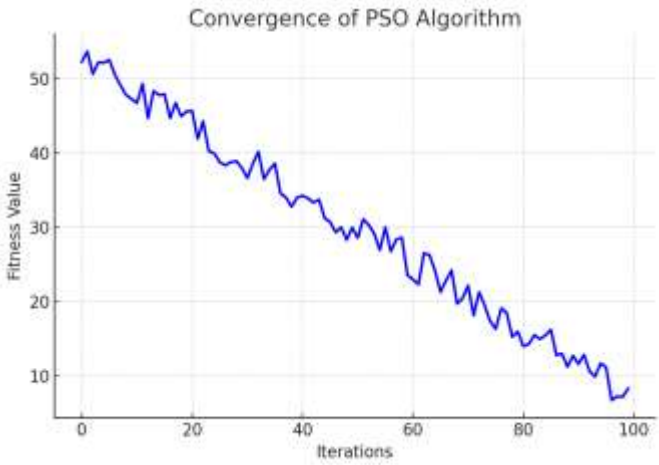

Fig 5: Convergence of PSO Algorithm

The optimization process of the algorithm is visualized in Fig 5, which illustrates the decreasing trend of the fitness value over successive iterations. This graph demonstrates the algorithm's convergence towards the best-known solutions, as the fitness value decreases with each iteration, indicating that the algorithm is approaching the optimal solution. Fig 6 visualizes the movement of particles across iterations, providing insight into the algorithm's search process. Each line in the graph represents a particle's position at each iteration, showing how particles search the space and gradually move to the global best path. The graph illustrates the particles' initial random exploration of the search space, followed by their convergence to the optimal path, demonstrating the algorithm's ability to efficiently search the solution space.

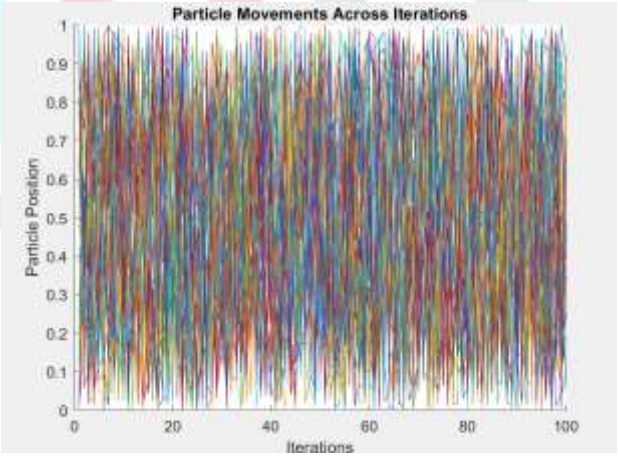

Fig 6: Particle Movement across iterations

In practical industrial applications, PSO has demonstrated its effectiveness in real-time path planning under dynamic conditions. A study involved applying PSO to a six-axis industrial robot [8]. The robot was required to navigate a workspace filled with obstacles, with the aim to minimize the

cycle time. The algorithm proved effective in finding collision-free paths and optimizing performance under dynamic conditions. Another application utilized PSO for controlling a two-link robotic manipulator in a dynamic environment. The robot was able to avoid obstacles and optimize its movement, further demonstrating PSO's robustness in solving complex path planning problems in real-world industrial environments. By incorporating PSO into path planning, manufacturers can achieve more efficient, collision-free, and adaptable robotic operations in dynamic environments, improving productivity and safety

D. Machine Learning based algorithms

Machine learning algorithms have become essential tools for path planning in industrial robots, especially in dynamic environments where obstacles and conditions change frequently. Unlike traditional methods that rely on fixed rules or predefined paths, machine learning approaches allow robots to learn and adapt to their surroundings in real time. These algorithms enable more flexible, efficient, and autonomous decision-making, improving robots' ability to navigate complex spaces. In dynamic environments, machine learning techniques like Reinforcement Learning (RL) and Deep Learning (DL) play a critical role by allowing robots to continuously update their strategies based on environmental feedback, optimizing path planning even in unpredictable settings.

Reinforcement Learning (RL)

Reinforcement Learning (RL) is a unique method of teaching robots how to navigate the correct ways most efficiently by using trial and error or testing values. In industrial robots, RL is aimed at teaching the robot how to move in the environment that may have some barriers at one time, while at another time it may be clear, and which may change regularly. The robot's ''agent'' operates based on experience which is in a way gained from the environment giving rewards if the robot successfully takes an action or penalizing it for physical contact with the environment. The RL algorithm is most useful for dynamic environments because the environment constantly changes. Deep reinforcement learning (DRL) is a combination of deep neural networks with those of RL [6]. In this method, the robot builds its own perceptual model of the environment through the DL and chooses the best action using RL. On one hand, the recognition of the given environment is provided by supporting subservices of DL while RL helps the robot not only recognize the environment but also to learn the most effective ways to pass various conditions of unpredictable configurations on its way.

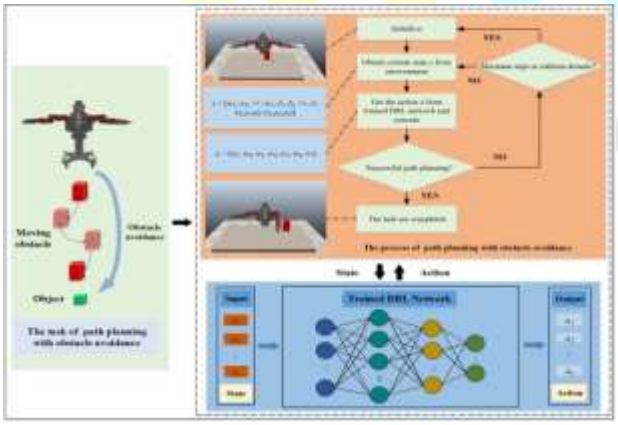

Fig 7: RL based path planning for dynamic environment

DQL - deep Q-learning is one of the most effective RL techniques for developing path planning algorithms for robots. On this approach, the environment of the robot is considered a topological map [11]. The robot is trained to choose the correct path with the help of rewards obtained by simulation of possible paths with successful passing of the goal with least contact with obstacles detected on the way. It has been established to offer high accuracy in path learning, especially in dynamic environments such as industries.

Deep Learning (DL)

Deep Learning (DL) in turn is the use of neural networks to analyze or learn from big data. By means of path planning for robots, DL models can analyze sensory data of the robot in real time, lighter, such as camera data or LiDAR data, to rapidly understand the position of the obstacle, and then plan the safest and most efficient path. CNN and RNN are widely used in dynamic path planning because of the high performance of deep learning. Through the use of DL in dynamic environments, robots are able to increasingly capture and understand details of the environment quicker.

CNNs are perfect for the image data coming from cameras and for making 2D or 3D maps. CNNs are employed for object detection from sensors and for the recognition of empty areas or paths and generation of the best paths, respectively. In this configuration, CNN works on a cell-based representation of the surrounding environment. Each of the items of the grid is a part of the environment occupied by the robot with obstacles being defined as taken and free spaces as vacant. The CNN takes in this data, derives high level features, and estimates the best move that the robot has to make in every state. This is especially applicable where there are fixed and dynamic obstacles in the working environment.

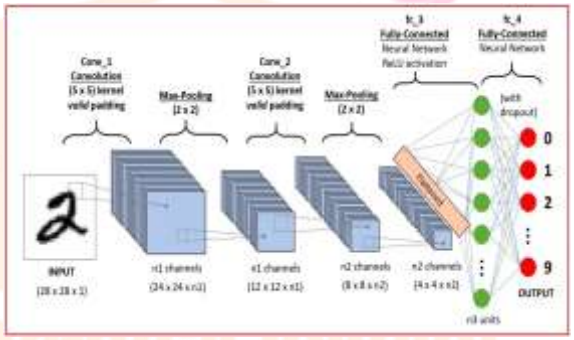

Fig 8: CNN architecture

RNNs especially the Long Short-Term Memory (LSTM) networks are the most suitable when handling time varying data. RNNs are usually used in conjunction with CNNs to improve the decision-making processes taking into account not only the state of the object but the sequence of events as well, which can be particularly useful in a number of robotic tasks which require robots to operate in different environments with various objects – for instance, in the frameworks of warehouse automation. The DL-based approach improves its performance due to real-time decision making which will reduce interference for the robot navigation.

IV. Comparative Analysis

*A. Classical Approaches: Dijkstra and A**

These algorithms are mainly used for pathfinding in static environments. They provide the shortest path, and are easy to implement. But they are less efficient when it comes to dynamic or high-dimensional space and computational cost is expensive. Dijkstra's Algorithm finds all possible paths to determine the shortest path but does not account for changes in the environment. A* is more efficient than Dijkstra's,

although there are cavity conditions when the performance depends on the condition of the heuristic function.

B. *Randomized Algorithms: RRT and PRM*

The RRT and PRM offer a more flexible solution to work in high dimensional or dynamic systems. Real time applications are preferred by these algorithms because the method used to resolve dynamic obstacles is very efficient. The advantage of RRT is particularly suitable for the feasibility of search space exploration in high-dimensional space. Still, the paths may turn uneven on its curve and may be jagged. PRM is effective for multi-query environments where repeated path planning is required [5]. It develops an initial set of directions which can be recycled for a number of processes. If the environment is dynamic then PRM suffers as often as a company needs to update the roadmap.

C. *Swarm-Based Algorithm: Particle Swarm Optimization (PSO)*

PSO is great at adapting to changes in dynamic environments, as it lets particles adjust their paths in real time when new obstacles appear. This flexibility makes PSO useful for situations where things are constantly changing. However, it's often slower than traditional methods and doesn't always find the shortest path since it relies on randomness in its search. While PSO can handle real-time challenges well, it can also be more computationally demanding, and the solutions it finds aren't always as optimal as those from classical algorithms, which are more predictable and efficient.

D. *Machine Learning-Based Algorithms: Reinforcement Learning (RL) and Deep Learning (DL)*

Main strength of machine learning based approach is the versatility and possibility of difficult path planning in dynamic environment, and incremental learning and capability to perform complex operations. However, the process of training is substantially time consuming and requires large quantities of training data. Reinforcement Learning (RL) allows the robot to improve the selection of the path on its own by asking feedback from the environment through trial and error. It can readily operate in a difficult, ever-changing condition. Deep Learning (DL) algorithms can learn complex patterns from the environment, making them highly effective in dynamic situations [10].

Table 1: Comparison of the different path planning methods based on key performance metrics

Algorithm	Computational Efficiency	Adaptability	Scalability	Path Optimality	Best Use Case
Dijkstra	Low	Low	Low	High	Static environments
A*	Medium	Medium	Low	High	Heuristic-driven pathfinding in semi-dynamic environments
RRT	High	High	High	Low	Real-time, dynamic, high-dimensional spaces
PRM	High (multi-query)	Medium	High	High (with RRT*)	Multi-query, semi-static environment
PSO	Medium	High	Medium	Medium	Multi-robot systems in dynamic environments
Reinforcement Learning (RL)	Low (training phase)	High	High	High (with good training)	Learning-based navigation in complex dynamic spaces
Deep Learning (DL)	Low (training phase)	High	High	High	High-dimensional sensor input handling in dynamic environments

V. Conclusion

This paper presents a synthesis of classical, probabilistic and AI based planar path planning techniques for industrial manipulators. While the classical methods such as A* are efficient in solving problems in well-defined spaces, they are not efficient in real environments, especially the large and dynamic ones. The probabilistic algorithms like RRT are developed to handle high dimensional environments well but face some constraints like having to ensure both probabilistic possibility and the optimality of the final path at the same time. Machine learning promotes controllability and flexibility at the cost of control complexity while traditional methods improve computational control complexity in exchange for improved controllability. The nature of the chosen algorithm should, therefore, reflect the nature of the industrial task that needs to be accomplished. Continuing the analysis of the subject matter for future research, an attempt should be made to focus on the development of new, more productive transitional models based on such approaches as classical, probabilistic, and AI ones integrated for improved path planning in optimized dynamic industrial environments with less time and energy consumption.

References

[1] H. Zhang, Y. Wang, J. Zheng, and J. Yu, "Path Planning of Industrial Robot Based on Improved RRT Algorithm in Complex Environments," *IEEE Access*, vol. 6, pp. 53250-53260, Sep. 2018,

[2] J. J. Kuffner Jr. and S. M. LaValle, "RRT-Connect: An Efficient Approach to Single-Query Path Planning," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, CA, USA, 2000, pp. 995-1001.

[3] B. Yan, T. Chen, X. Zhu, Y. Yue, B. Xu, and K. Shi, "Comprehensive Survey and Analysis on Path Planning Algorithms and Heuristic Functions," *Intelligent Computing*, SAI 2020, pp. 581–598, July 2020.

[4] A. Ravankar, A. A. Ravankar, Y. Kobayashi, and T. Emaru, "Path smoothing extension for various robot path planners," *IEEE*, [Online].

[5] M. Huppi, L. Bartolomei, R. Mascaro, and M. Chli, "T-PRM: Temporal Probabilistic Roadmap for Path Planning in Dynamic Environments,"

[6] P. Chen, J. Pei, W. Lu, and M. Li, "A deep reinforcement learning based method for real-time path planning and dynamic obstacle avoidance," vol.497, pp. 64-74,2022.

[7] R. Kumar and K. Kumar, "An optimized approach of intelligent path planning of a robot manipulator using PSO algorithm," AIP Conf. Proc., vol. 2548, no. 030024, 2023.

[8] S. Sahu and B. B. Choudhury, "PSO Based Path Planning of a Six-Axis Industrial Robot," in *Computational Intelligence in Data Mining*, H. Behera, J. Nayak, B. Naik, and D. Pelusi, Eds. Singapore: Springer, 2020, vol. 990, pp. 213–220

[9] R. Saxena, "Dijkstra's shortest path algorithm - A visual introduction," FreeCodeCamp, Sep. 6, 2022.

[10] LeCun, Y., Bengio, Y., & Hinton, G. (2015). *Deep learning*. Nature, 521(7553), 436-444

[11] Y. Long and H. He, "Robot path planning based on deep reinforcement learning,"