Proceedings of the 2023 International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/41/20230732

Logistics warehousing system shelving station route planning based on X-ARM

Pengao Feng

Bartlett, University College London, United Kingdom, WC1E 6BT

ucbvpf@ucl.ac.uk

Abstract. The need for automated control in warehouse and logistics systems has increased recently, particularly in the area of cargo storage. The use of robot arms in manufacturing is an important area for development. Multi-arm actions, including switching between multiple arms, expand the number of potential operations that numerous operators can do, but they also bring more computing hurdles. Multiple robotic arm systems must operate with careful path planning. In this study, Based on previous research papers, we propose the utilization of *dRRT** and *mmdRRT** algorithms for efficient path planning. The *dRRT** algorithm improves exploration and convergence to an ideal path by combining the advantages of the rapidly-exploring random tree (*RRT*) with the optimal *RRT** algorithm. To address complicated and dynamic situations, the *mmdRRT** algorithm, on the other hand, uses a multi-modal distribution model. We want to improve the reliability and effectiveness of path planning for various robotic arm systems by combining these two techniques.

Keywords: Path-Planning, Logistics and Warehousing System, Coordinating Multi-Arm Operations, *dRRT** & *mmdRRT**.

1. Introduction

Warehouse cargo transportation is a necessary but time-consuming process. Effective route design is vital to guaranteeing smooth operations and increasing productivity, hence it merits in-depth research [1]. In logistics warehousing, there has been an increase in demand for effective, automated control systems. The amount of goods handled and housed in warehouses has greatly expanded as a result of the quick expansion of e-commerce and online shopping. Any goods cannot be delivered more quickly or accurately using manual transportation or dispersed storage facilities [2]. A popular trend is the automatic coordination of many robotic arms for transporting warehouse freight. The positioning of products on shelves must be automated if logistics storage systems are to become more effective and productive overall. Warehouses can lower operational expenses, eliminate human error, and enhance inventory control by introducing automated control methods. Additionally, coordinating numerous manipulators might result in a larger feature set and faster execution [3]. Each robotic arm's beginning and ending points can be limited, and numerous robotic arms can be effectively connected to increase efficiency.

Robotics research was dominated by industrial robots until the 1990s [4] after they were first incorporated into industrial production lines. The automobile and aviation industries, for example, use industrial robots extensively. Industrial robots can be reduced to simple robotic arms, which not only

DOI: 10.54254/2755-2721/41/20230732

reduces the amount of workspace needed but also increases operational flexibility without compromising the level of freedom of the robotic arms. When it comes to repetitive jobs like picking and placing, transportation, assembly, and other similar ones, industrial robots are quite advantageous [4]. However, time, energy, and overall system performance are frequently issues with traditional methods of controlling shelving stations. An improved route planning strategy is therefore required to streamline operations and reduce inefficiencies. We synthesize many research reports on path-planning methods for multiple robotic arms and derive two useful methods, *mmdRRT** and *dRRT**. In this work, they can be introduced independently and contrasted.

The demand for autonomous warehouse robots has substantially expanded in the warehouse management system because of their critical function in the supply chain. As a result, various requests have been made for various components of it.

The construction of the robot arm for the logistics warehouse system must be flexible, intelligent, and adaptive in order for the robot arm to carry out complicated tasks [5]. However, as robotic arms are already highly DOF robots, establishing a bigger configuration space (***C***-space) is necessary to coordinate many robotic arms at the mission planning level [3].

Simulating the grasping and shelf positioning procedure effectively is one of the study's main challenges. The two robotic arms are arranged linearly to simulate their cooperative working procedure. The best plan for this circumstance is determined by contrasting the *dRRT** and *mmdRRT** techniques. This necessitates the use of a reliable modelling and simulation tool that can faithfully depict the dynamics and kinematics of the robotic arm. A good foundation for modelling and simulating such complicated systems is offered by Matlab, a widely used numerical computation and simulation program. This study models the robot arm and analyzes its motion trajectory using the Matlab robot toolkit. This study intends to address this challenge by enhancing the automation of the item placement process in logistics storage systems.

Overall, the *X-Arm* model-based automated control mechanisms that are being developed for logistics storage systems are a result of this research. The course of the robot arm is improved by combining the earlier research techniques. Future research to enhance the automation of the logistics storage system can use the study's findings as a guide.

2. Algorithm Principle

This section primarily explains the fundamentals of the robot arm's route planning. This session's first part examines the *link* functions of two distinct sampling techniques and introduces the *link* function of the robot toolbox that is necessary in Matlab. The second portion of this section compares and summarizes the two path planning techniques that the robotic arm uses: *dRRT** and *mmdRRT**.

2.1. Core function- Link function interpretation

Table 1. Differences between two D-H parameter tables.

	Standard D-H parameters	Improved D-H parameters
Selection of fixed coordinate system	The connecting rod's fixed coordinate system is the rear joint coordinate system.	The connecting rod's fixed coordinate system is the preceding joint coordinate system.
Determination of the X-axis direction	The X axis is determined by the cross product of the current Z axis and the previous Z axis.	The z-axis of the subsequent coordinate is multiplied by the current z-axis to produce the X axis.
Rules of transformation between coordinate systems	Between adjacent joint coordinate systems, the parameter modifications are made in the following order: θ, d, a, α	The following is the order of parameter shifts among neighbouring joint coordinate systems: α, a, θ, d

DOI: 10.54254/2755-2721/41/20230732

For the existing 3D model of the robot arm (*X-Arm6*), Matlab is imported to carry out trajectory motion based on forward and inverse kinematics calculation. Download the robot toolbox in Matlab, enter "*ver"* on the command line to view it.

The Link function is based on *D-H* parameters to establish the link, including the main information of the joint, the input sequence of *D-H* parameters when establishing the link is joint Angle *θ;* joint distance *d;* link length *a*; link Angle *α*; joint type (*0* rotation, *1* movement). In this project, the *X-Arm6* joints are rotated.

In addition, the parameters of the joint variables are "*qlim*" specifies joint limits; "*jointtype*" specifies the jointtype, the default is a revolute joint, *L(x).jointtype= 'P'* means that the *x* link is joined by a prismatic joint; *offset* is the offset of the initial value of the joint. The format of the Link function call is *L(1) = Link([theta1, D1, A1, alpha1, offset1], 'standard') %Standard D-H parameters.*

*2.2. Path planing using dRRT**

In this work, the problem is first defined. Then, two *X-arm6* linear arrangements (m_1, m_2) are specified. The end effector is then arranged to be fitted on the *X-arm6* vacuum suction cups in accordance with the first hypothesis. This experiment focuses on the design path rather than establishing information about obstacles. An empty tree structure is produced by initializing the 3 structure. The root node of the tree represents the initial attitude, while m_1 is at the first starting position and m_2 is there. The next step is to enlarge the sample point, choose a point at random from the sampling space, and locate the closest node in the tree. The next step in the route search is to compute the path from the closest node to the sampling point using inverse kinematics solutions, and then connect that path to the tree to create a new node. Check whether the new node is close to the goal position and meets the requirements to finish path planning when the previous phases are finished. The process of expanding the sampling points, searching for paths, connecting paths, and checking targets is then repeated until a path meeting the conditions is discovered or the predetermined number of iterations has been reached. A specified number of iterations' paths is planned after being repeated numerous times. You can apply a smoothing method or optimization algorithm to shorten the path and enhance the path. The goal is to eliminate needless robotic arm operating time and length, which can save expenses and enable quick reaction. The robot arm's planned path from its starting attitude to its target attitude is known as the ultimate output path.

The dynamic nature of the *dRRT** path planning approach is a benefit. The actual cargo handling operation frequently deviates from the planned route due to alterations in the working environment. The sampling point expansion technique used in the *dRRT** approach is randomly chosen and then dynamically altered during path optimization. As previously noted, *dRRT** can still determine the shortest path in accordance with the changing environment even if the environment changes.

*2.3. Path planing using mmdRRT**

Similar to the *dRRT** approach, *mmdRRT** requires the robot arm's beginning attitude, target attitude, and any information on potential obstacles to be defined upfront. Still, there is no need to define barrier information in this study. Create an empty tree structure with the beginning posture as the tree's root node next to initialize the tree structure. The least average difference sampling approach is used to choose a point in the sample space, after which the node closest to the sampling point is chosen from the tree as part of the node selection process. The path from the chosen node to the sample point is then calculated utilizing the nodes extended using the appropriate motion planning techniques, such as inverse kinematics or trajectory optimization. The path from the old node to the new node is determined when the new node is generated. The new node is then subjected to the target check to see if it is close to the goal attitude. The path planning is finished if the conditions are satisfied. Until a path is discovered that satisfies the requirement or reaches the designated number of iterations, the sample point expansion, node selection, node expansion, and target check processes are again performed. To shorten or enhance the quality of a path, smoothing techniques or optimization algorithms can also be used. Output the robot arm's final path, or its intended route from the starting attitude to the goal attitude.

DOI: 10.54254/2755-2721/41/20230732

Since *mmdRRT** uses the minimum mean difference during sampling to influence the choice of sample space, it is more likely to expand along the minimum path. The *mmdRRT** technique is better able to choose the optimum path quickly for a constant working area by removing all the effects of random factors.

2.4. Compared and discussed dRRT and mmdRRT**

*dRRT** (Dynamic Rapidly-exploring Random Trees) and *mmdRRT** (Minimum Mean Discrepancy Rapidly-exploring Random Trees) is an improved version of two RRT* (Rapidly Exploring Random Trees)-based path planning algorithms [6].

Firstly for dynamic part, *dRRT** is a dynamic algorithm that updates the tree structure in real-time during the planning process to adapt to changes in the environment [7]. It can handle the movement and removal of obstacles in the environment and the addition of new obstacles. While *mmdRRT** is a static algorithm that performs path planning only for static environments. Once the tree structure is established, it does not take into account dynamic changes in the environment [8].

Secondly for shortest path guarantee, *dRRT** ensures that the global shortest path is found, even if there are changes in the environment. It keeps optimizing the tree structure to find shorter paths [9]. While *mmdRRT** can only ensure that a sub-optimal path is found because it will not be optimized after the tree structure is established.

Third part is sampling strategy, *dRRT** uses the random sampling policy to select one point at a time in the completely random sampling space for expansion [10]. This can result in a less-than-ideal growth direction for the tree. While *mmdRRT** Improves the tree growth direction by optimizing the sampling strategy. It uses the Minimum Mean Discrepancy to guide the selection of sample space and is more likely to expand the tree along the shortcut direction.

Next is real-time performance, *dRRT** is superior to *mmdRRT** in terms of real-time performance because it requires only local optimization [11]. However, in high-dimensional environments, the performance of *dRRT** may be limited. While *mmdRRT** will do more optimization when building the tree structure to get a better path. This can lead to longer computation times, especially in high-dimensional environments.

Last but not least is applicability of both methods, based on the methods of previous papers, the design of a suitable Minimum Mean Discrepancy (MMD) sampling strategy can help guide the sampling point selection of *mmdRRT** method, so as to improve the effect of path planning. The current study suggests a multi-modal dRRT* method that focuses on resolving the picking and positioning issues associated with manipulators with a large degree of freedom [3].

*mmdRRT** (multi-machine dual-*RRT**) is a method that extends the *RRT** algorithm, which is utilized to address the joint planning issue of numerous robot arms. By simultaneously creating several RRT* trees, each of which represents the path of a robotic arm, it looks for joint paths. In order to identify a feasible joint path, the *mmdRRT** method takes into account collision detection between robotic arms while maintaining connections between trees during the search process.

*dRRT** (*distributed RRT**) is a method based on distributed path planning, which is used to solve the cooperative motion problem of multiple robot arms. The *dRRT** algorithm breaks down the task into sub-problems, applies the *RRT** algorithm to each robot arm separately to search the path, and coordinates the movement between the robot arms through information exchange to achieve collaborative movement. The *dRRT** algorithm is suitable for scenes with a large number of robotic arms or a large scale.

These methods can help resolve the issue of route planning multiple robotic arms and the specific choice of which method needs to be determined according to the specific situation, including the number of robotic arms, constraints of the workspace, etc.

3. Case Study

The issues that the paper must address are presented in this part. To make things easier, The problem setting for both robotic arms will be described, with an overview of the issue taken from the conference

DOI: 10.54254/2755-2721/41/20230732

article “Anytime Multi-arm Task and Motion Planning for Pick-and-place of Individual Objects via Handoffs” [3] based on *X-arm6*. functioning in a shared workspace are m_1 and m_2. The results can be directly applied to more robotic arm sets by examining the data.

Every robot arm in the workspace has its own dimensional *C*-space (C_{m1}, C_{m2}). The composite configuration space of both robot arms is $C = C_{m1} * C_{m2} \subset R^{d1+d2}$. Then, a composite configuration is $Q = (q^1, q^2)$, where $q^i \in C^{mi}$. $C^{obs} < C$ is the obstacle subset, where a manipulator either collides with static barriers or with itself, or where manipulators collide with one another. $C^{free} = C \backslash C^{obs}$ is the valid subset.

The *X-Arm6* parameters are introduced into a multi-arm control system. By studying the three hypotheses proposed in the original text, *mmdRRT** and *dRRT** methods are used to verify them respectively. In accordance with presumption one, every manipulator has the end-effector that can immobilize an object when it is in a relative position to the end-effector (g). The object can be placed steadily by the manipulator; the end-effector can greatly assist single or multiple-arm systems by enabling the robot arm to more securely grasp objects. In the case of the *X-arm6* parameter, it is particularly clear that the appropriate end-effector may be a device with a vacuum suction cup. Hypothesis two is called object support, when the object is set to a stable position, the object maintains stable contact with the stationary surface, this includes the initial pose p_{init}, or the item is placed in a location where it can be carried by one or more manipulators. The problem as a whole has a state space that is the sum of all its component *C*-spaces. The third hypothesis is “singly manipulable,” meaning that only one manipulator is required to pick up and manipulate the object. Different objects or motion steady states are employed as beginning reference points for the problem to be solved in this study based on the three aforementioned premises. Two different route planning methods, *mmdRRT** and *dRRT**, are used to solve the multi-arm motion trajectory problem.

It is necessary to introduce the mode graph *M* [3] in order to address the *PHP* (pick-and-place) problem. The challenge of multi-robot planning and integrated task and motion planning (*TAMP*) can be resolved with the aid of the mode graph put out by Rahul Shome and Kostas E. Bekris, as seen in the figure 1. below.

The initial (q^1_{init}, q^2_{init}, p_{init}) and final states (q^1_{goal}, q^2_{goal}, p_{goal}) are equivalent to two nodes on *M* indicated as v^{init}_M *and* v^{goal}_M. A pick-and-place issue solution relates to traversing the graph’s edges:

$$\Pi \rightarrow (v^{init}_M, v^i_M \ldots v^j_M, v^{goal}_M) \tag{1}$$

It’s also vital to remember that because the modes do not confine all the arms to configurations, the mapping from a sequence of modes to is not entirely established. To completely characterize the task space route, the solution consists of a series of grounded configurations Q^i and object poses p^i. Let the pair on *M* shown below serve as the definition of a grounded search node:

$$V = < Q, v_M > \tag{2}$$

It only takes a series of Vi to map to a task space solution:

$$(V^{init}, V^i \ldots V^j, V^{goal}) \rightarrow \prod \tag{3}$$

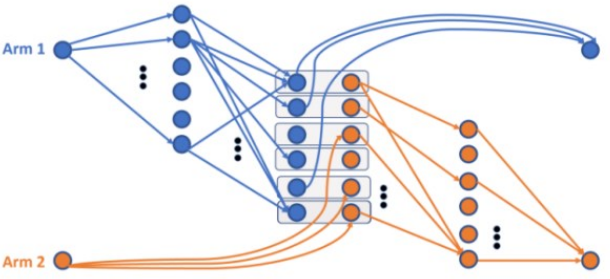


Figure 1. Mode graph (*M*) for pick-and-place issue [3].

DOI: 10.54254/2755-2721/41/20230732

The search tree produced while using *mmdRRT** to plan a route is defined as T in Rahul Shome and Kostas E. Bekris' approach. The composite arm configuration Q and the mode v_M are tracked by the ground search node on the search tree T. The pattern sequence along which these tree nodes are placed, or the traversal of the tree, completely describes the posture of the object created by operations like picking, passing, and inserting. The initial state and pattern that V_{init} outlined serves as the algorithm's starting point for its search. The ground search node in T must be added by the algorithm as soon as it locates the configuration in M that meets the pattern requirement. Then, a number of composite arm configurations that show motions of the arm and item without colliding were found. Each arm's arrangement is determined by the tensor roadmap. Tables 2 and 3 below list the parameters that were utilized and the algorithms created.

Table 2. Algorithmic Notations.

T	The search tree built by *mmdRRT**
V^{init}	Initial configuration Q^{init} and mode v_M^{init}
Π^{best}	Discovered solution with best cost
V^{near}	Node on search tree selected for expansion
V^{new}	New node that is a candidate for adding to T
V^{last}	A better heuristic node from the last expansion

To organize the above steps into a formula, all you need to use is:

Table 3. Algorithm: mmdRRT* (M, v_M^{init}, v_M^{goal}, $\hat{G}$) [3].

1	$\Pi^{best} \leftarrow \phi$; $V^{init} \leftarrow < v_M^{init}.Q, v_M^{init} >$
2	$T.init(V^{init})$; $V^{last} \leftarrow V^{init}$;
3	**while** time.elapsed() < *time_limit* **do**
4	$V^{last} \leftarrow$ Expand_mmdRRT* (T, V^{last},M, $\hat{G}$)
5	$\Pi \leftarrow$ Connect_to_Target ($\hat{G}$,T, v_M^{goal})
6	**if** $\Pi \neq \phi \cap$ cost(Π) <cost(Π^{best}) **then**
7	$\Pi^{best} \leftarrow$ Trace_Path (T, v_M^{goal})
8	**return** Π^{best}

The *X-arm* model is used as a simulation in this work using the aforementioned approach. The six joints of the model, which is designated as *X-arm6* in Table 4, all have various operating ranges. Figures 2 and 3 illustrate the *X-arm6* model's general working range and mounting dimensions, respectively. The plan adopted in this study, which is depicted in Figure 5, simulates the transport cargo by mounting the vacuum head end effector on the *X-arm* in accordance with the first hypothesis given in the preceding section. Table 5 displays the exact characteristics of the vacuum head's end effector.

DOI: 10.54254/2755-2721/41/20230732

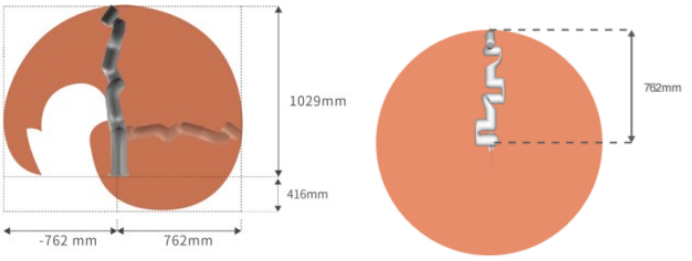


Figure 2. Determine the working space of the robot arm. (*X-Arm6*).

Table 4. Parameter of different *X-Arm.*

	Robot Arm	*X-Arm5*	*X-Arm6*	*X-Arm7*
Maximum Speed		180°/s	180°/s	180°/s
Working Range	Axis 1	±360°	±360°	±360°
	Axis 2	-118°~120°	-118°~120°	-118°~120°
	Axis 3	-225°~11°	-225°~11°	±360°
	Axis 4	-97°~180°	±360°	-11°~225°
	Axis 5	±360°	-97°~180°	±360°
	Axis 6		±360°	-97°~180°
	Axis 7			±360°

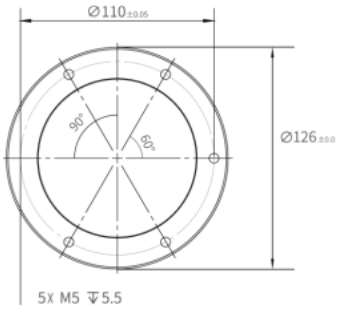


Figure 3. Robot arm installation on X-Arm6. (mm).

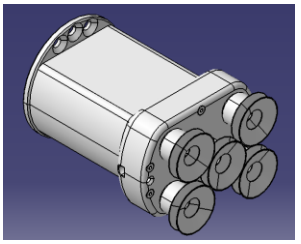


Figure 4. Vacuum Gripper.

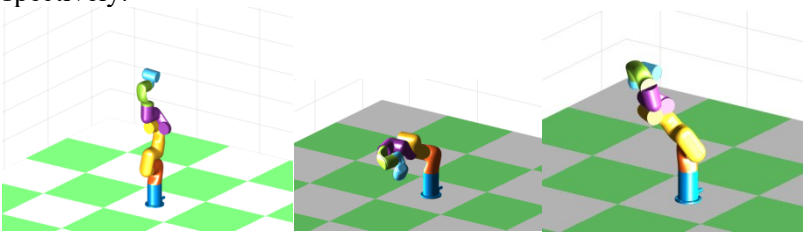
Figure 5. Vacuum Pump Exhaust Suction Cup.

DOI: 10.54254/2755-2721/41/20230732

Table 5. Parameter of vacuum gripper.

Rated supply voltage	24V DC
Maximum supply voltage	28V DC
Static current	30mA
Peak current	400mA
Vacuum degree	78%
Vacuum flow rate (L/min)	>5.6L/min
Mass (g)	610g
Size (L*W*H)	122.5*91.6*75mm
Payload (kg)	⩽5kg
Noise level (within 30 cm)	<60dB
Communication mode	Digital IO
Status light	Power; Working Status
Feedback signal	Air pressure (low or normal)

The route of m_1 in the *X-arm6* is plotted out using Matlab simulation data and *mmdRRT** path planning regulations. The end point of the running track of the m_1 robotic arm is the starting point of the m_2 robotic arm since two robotic arms were chosen as the model of the multi-arm operating system in this research. Figures 6(a), 6(b) and 6(c) illustrate the position state of m_1 at the start point, pick point, and finish point, respectively.



(a) (b) (c)

Figure 6. Using Link Function to Build *X-Arm6* m_1. (a) Initial position; (b) Pick-up cargo location; (c)Moving to *m2* location. The handoff increases each arm's range of motion.

4. Result

A benefits and drawbacks of using *dRRT** or *mmdRRT** in the simulated path planning of multiple robotic arms are discussed in this section, along with the suggested approaches. The three position forms of the *X-arm6* model m_1 were drawn using Matlab and provided the data for this study. The accuracy of the results is shown to be dependent on the following three factors when multi-arm systems are simulated in various situations [12].

The first is whether there are static or moving barriers in the surroundings. Since there is no barrier in the setting chosen for this study, it can already be said that it is a static environment. The route prepared using the *mmdRRT** approach is much shorter and smoother under these circumstances than the route planned using the *dRRT** method. *dRRT** needs a larger buffer time to simulate the path depending on how long it takes to process the path. Therefore, it can be said that under static conditions, the *mmdRRT** approach is preferable to the *dRRT** method.

Second, in the working environment of a linear arrangement of just two robot arms, the path planning speed of the *dRRT** approach is faster than *mmdRRT**. There is only one point of contact between m_1 and m_2 as a result of their shared workspace, and since this point serves as the start of the cargo transit

DOI: 10.54254/2755-2721/41/20230732

chain, there is no need to define it. The best path planning technique in terms of real-time performance is still *dRRT**.

Third, depending on the specific application scenario and requirements, the choice of which method to use depends on the trade-off between dynamism, shortest path assurance, and real-time performance.

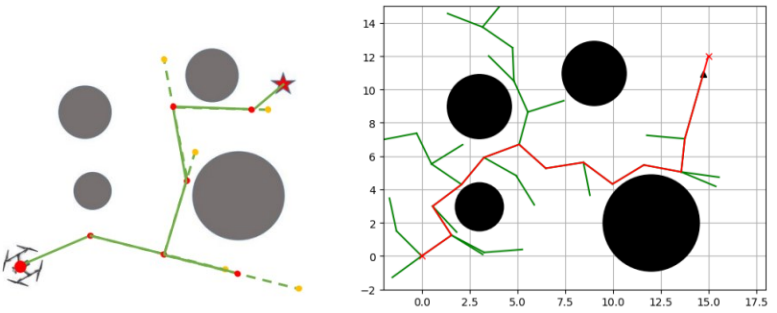


Figure 6. Use *dRRT** sampling to plan the path, randomly sample the dynamic obstacles for path planning, and select the optimal solution by optimizing the path (left). The result of running the code is shown on the right.

As can be seen from the results in the figure, it is necessary to carry out a large number of operations for the path planning of the sampling method of *dRRT** and *mmdRRT**, and finally obtain the optimal solution by optimizing the path. In the real logistics storage management system, the path planning of the robot arm is often based on the barrier-free shortest path.

Based on the path planning of *X-arm6*, there are still many other experimental methods for motion planning of multiple robotic arms besides the basic motion planning algorithms (*dRRT** and *mmdRRT**). For example, environmental simulators. Using a simulation environment can greatly simplify the experiment process and provide repeatability. Open source simulators such as Gazebo, V-REP, or Unity can be used to build virtual environments with multiple robotic arms.

Another example is the experimental parameter setting. By establishing the experiment's parameters, such as the robot arm's beginning posture, the target's posture, the location and nature of the barrier, etc. To mimic various scenarios and settings, the parameters can be changed in accordance with the needs of the situation.

5. Conclusion

The comparison and discussion of *dRRT** and *mmdRRT** in path planning of multi-arm systems are presented based on prior work and the modeling analysis of *X-arm6*. Attempt to resolve the selection and placement issues using manipulators with a high degree of freedom. According to the findings, various path planning techniques should be chosen based on the environment, path planning requirements, response time, and dynamic and static requirements. For a certain working environment model, theoretical demonstration shows that *dRRT** path planning time cost is superior to *mmdRRT**, but accuracy is lower than *mmdRRT**. The comparison established in this work can be investigated further, including the choice of end-effector and the implementation of a multi-arm production line's semi-automation.

References

[1] Khan A, Rehman Sair A, Ekram A, Malik S, Raheel Afzal M, Bin Junaid A and Eizad A 2015 An automated object retrieval system for warehouse *C. ICCAS* pp.95-100

[2] Zai Ur Rahman M, Kumar Kanchi M, A Aleem Pasha M and Kumar Yadla B 2021 Design and development of autonomous warehouse management robot with intelligent software framework C. *ICMNWC* pp. 1-7

[3] Rahul S, Kostas B 2019 Anytime multi-arm task and motion planning for pick-and-place of individual objects via handoffs *C.MRS* pp. 37-43

DOI: 10.54254/2755-2721/41/20230732

[4] Abdelaal M 2019 A study of robot control programing for an industrial robotic arm C. *ACCS&ICNPE&PEIT* pp. 23-28
[5] Wang Y, Liang Y, Chen D, Liu Y and Wang M 2020 A goods sorting robot system for e-commerce logistics warehouse based on robotic arm technology C. *RCAR* pp. 310-314
[6] Reddy M and S.R. N 2014 Integration of robotic arm with vision system C. *CICRC* pp. 1-5
[7] Pei-Chi H and Aloysius K. M 2018 A case study of cyber-physical system design: autonomous pick-and-place robot C. *RTCSA* pp. 22-31
[8] Jaeyeon L, Sangseung K, Kyekyung K, Jaehong K and Joong B K 2013 A development of easily trainable vision system for the multi-purpose dual arm robots C. *URAI* pp. 609-614
[9] Jyothsnaa S, Gandhe A, Deshpande A and Bodas S 2010 Automated inventory management and security surveillance system using image processing techniques C. *IEEE* pp. 2318-2321
[10] Gokul H, Kanna S.V., Akshay K H. and Vignesh R 2020 Design of imitative control modalities for a 3 degree of freedom robotic arm C. *ICCCSP* pp. 1-6
[11] Khairidine B, Jean-Francois B, Francois G and Marc G 2018 Dual arm robot manipulator for grasping boxes of different dimensions in a logistics warehouse C. *ICIT* pp. 147-152
[12] Nazib S and Abu S S 2021 Implementation of Pick & Place Robotic Arm for Warehouse Products Management C. *ICSIMA* pp. 156-161