Comparison of RRT and TRRT Object Arrangement Path Planning Robot for Retail Warehouse Using CoppeliaSim

L. H. Kwee*, N. M. M. Sobran
Faculty of Electrical Engineering, Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya, 76100 Durian Tunggal, Melaka, Malaysia
*corresponding author's email: B011910146@student.utem.edu.my

Abstract – *Shelf stocking in retail warehouses requires high accuracy, multitasking, and precise timing. Automating the process has become a popular solution due to difficulty in facing continuous demand in replenishing items at shelves. The traditional path planning, Rapid-exploring Random Tree (RRT) approach is inefficient and produce suboptimal path generation when it comes to complex workspaces such as retail warehouse. Furthermore, the simple linear interpolation method used for robotic arm manipulation can result in jerky, unrealistic motion, and collisions with obstacles. To address these issues, the Transition-based Rapidly-exploring Random Tree Star (TRRT) is proposed to achieve the shortest distance in shelving items at warehouse. The study also implements inverse kinematics algorithms for smooth arm manipulation. In this study, the TRRT is constructed in CoppeliaSim software using the OMPL Library as well as Inverse Kinematics (IK) algorithms. The results show that the TRRT approach generates shorter paths than the RRT approach but takes longer calculation time. Moreover, the TRRT approach demonstrates a good repetition result compared to the RRT approach. Furthermore, the accuracy percentage between the joints angle obtained from CoppeliaSim and the IK calculation in MATLAB reaches 99.27%. In conclusion, the TRRT approach and inverse kinematics algorithms improve the efficiency and smoothness of the object arrangement path planning robot, making it a viable solution for automating the shelf-stocking process in retail warehouses.*

Keywords: *CoppeliaSim, Inverse Kinematics, Object Arrangement Path Planning Robot, Retail Warehouse Robot, Transition-based Rapidly-exploring Random Tree (TRRT)*

Article History
Received 19 June 2023
Received in revised form 1 September 2023
Accepted 15 September 2023

I. Introduction

Shelf stocking in retail warehouses and stores is a challenging task for robotic applications. The advancement of manufacturing and software technologies has led to a significant increase in the use of robotics in retail in recent years [1]-[2]. However, despite recent advancements in visual processing and robot manipulation technologies, humans are still required for shelf-related jobs such as shelf picking and replenishment due to their adaptability and dependability [1], [3]. Therefore, there is growing interest in developing robotic systems capable of shelf replenishment manipulation, as demonstrated by initiatives such as the Amazon Robotic Challenge (ARC) and Amazon Picking Challenge (APC), which promote autonomous robotic manipulations in the chaotic surroundings of warehouses [4].

Retail replenishment requires multitasking, accuracy, and impeccable timing, which necessitates a lot of human resources. Efficient shelf replenishment methods are crucial for ensuring that products are correctly arranged on shelves, as poor arrangement can lead to decreased sales. In the retail warehouse or store, shelf replenishment is an essential role that requires a lot of manual effort by store employees, especially for large companies like Amazon that have thousands of objects to replenish on the shelves every day [3], [5]. Shelf replenishment is undoubtedly the most time-consuming activity, with 50% of the time spent locating the proper spot on the shelf [6]. As a result, it is expensive and inefficient, requiring a lot of human resources and time. Furthermore, the task is repetitive and tedious, leading to errors and shelf storage without getting fully utilized. There is also a shortage of labor in the warehouse due to its continuous expansion.

To make the shelf replenishment process more efficient, this study proposes the design of object arrangement path planning robot for retail warehouses that can run smoothly and take the shortest manipulation arrangement path. For example, the TX SCARA Robot was currently used in Family Mart in Japan, but it can only restock a limited number of beverage bottles and cans in a cylindrical shape from the refrigerated store to the refrigerator rack. Additionally, it does not have a mobile base to carry objects, making it unsuitable for the long distances required in a retail warehouse environment [7].

Currently, the RRT algorithm is extensively utilized in the realm of robot path planning [8]. However, the traditional RRT algorithm tends to suffer from reduced calculation efficiency in complex workspaces, and the generation of suboptimal paths that may take a long time to reach due to its reliance on randomness and heuristic techniques. The long and convoluted paths generated are computationally intensive, particularly in environments such as large warehouses or stores with a high number of items to be arranged. This significantly increases the overall time and resources required for the object replenishment process, making it less efficient and less cost-effective [9].

Moreover, some of the robot's joint angles are calculated using a simple linear interpolation method, which can lead to jerky, unrealistic motion, compromising the precision and smoothness of the robotic arm's movement. It can also cause the robot to collide with obstacles in its environment, potentially damaging its structure and affecting its functionality. Linear interpolation may be limited in accuracy as it does not account for nonlinearities in the robot's joint dynamics, such as joint limits and singularities, which can further degrade the robotic arm's motion [10]-[11].

The inefficiencies of the traditional RRT algorithm in complex workspaces and the limitations of linear interpolation for calculating robot joint angles have led researchers to explore alternative methods for robot path planning. In the context of object arrangement path planning in retail warehouse environments, two crucial aspects are the object arrangement strategy and motion planning approach. Therefore, in recent years, researchers have proposed different methods to address these aspects and improve the efficiency and accuracy of robot path planning in such environments.

Abdo et al. [12] proposed a collaborative filtering-based method for arranging objects on shelves and in boxes by predicting pairwise preferences between objects and maximizing user preferences. However, this method relies heavily on user preferences, which may not always be practical in a busy retail warehouse environment. Furthermore, the method may not be suitable for complex and cluttered environments, where there are many objects and limited space for arranging them. Other than that, Jiang et al. [13] proposed a supervised learning approach that considers stacking of objects, stability, object-area relationship, and common placing constraints. Although this approach addresses various criteria for object arrangement, it requires a large amount of training data and may not be suitable for real-time processing. Furthermore, it may not be able to handle unexpected situations, such as sudden changes in the environment or the addition of new objects. Furthermore, Kang et al. [14] proposed an approach that utilizes hierarchical, spatial, and pairwise relationships to arrange cluttered objects. This approach improves object placement using extracted connections and ergonomic criteria, such as visibility and accessibility. However, the approach requires the prior identification of object connections and may not be effective for environments with constantly changing objects and layouts.

For motion planning, researchers have proposed different methods to improve the performance of traditional RRT. RRT-Connect, RRT*, TRRT, and Hybrid RRT-PRM are all variants method of the Rapidly-exploring Random Tree (RRT) algorithm, which is a widely used approach for robot path planning in complex environments. While the traditional RRT algorithm suffers from reduced calculation efficiency and may generate suboptimal paths, these variants aim to overcome these limitations and improve the performance of the algorithm [15]-[18].

RRT-Connect was introduced by Kuffner [9], [19] as an extension of the original RRT algorithm, with the main improvement being the addition of a second tree that grows from the goal configuration towards the start configuration. By connecting these two trees, RRT-Connect can generate a feasible path more efficiently and with higher success rates than the traditional RRT algorithm.

RRT* is another extension of RRT that was introduced by Karaman and Frazzoli [20]-[23]. This algorithm improves the path quality by using a re-wiring step that iteratively re-connects nodes to improve the path quality. By continuously improving the path quality, RRT* generates higher quality paths compared to traditional RRT.

TRRT, or Transition-based RRT, is a further improvement over RRT* that aims to reduce the computational cost by using local optimization techniques [24],[25]. TRRT achieves this by focusing on the transitions between nodes rather than the nodes themselves and using a local planner to optimize these transitions. This allows for more efficient exploration of the search space and results in faster convergence towards optimal solutions. One of the benefits of this approach is that it can improve the quality of the path by reducing its length. By optimizing the transitions between nodes, TRRT can generate smoother and shorter paths compared to traditional RRT. This can be particularly useful in retail warehouse environments where efficient use of space is essential and shorter paths can lead to faster and more efficient object handling.

Hybrid Rapidly-exploring Random Tree (RRT)-Probabilistic Roadmap (PRM) technique is a novel approach to motion planning in robotics. The Hybrid RRT-PRM technique combines two sampling-based algorithms, RRT and PRM, to generate shorter paths in a given configuration space [26]. RRT explores the configuration space by constructing a tree of random samples, while PRM plans for multiple paths by building a roadmap of the free space. By leveraging the strengths of both algorithms, the Hybrid RRT-PRM technique can generate shorter paths more efficiently compared to RRT and PRM.

In this study, inspired from Kang et al. [14] approach that focused on spatial aspect in robotic manipulation and Transition-based RRT was used to develop the object arrangement path planning robot in a retail warehouse environment. Then, in order to achieve accurate and smooth object arrangement manipulation, inverse kinematics was used as well for the UR5 Cobot movement. With this, the main contribution of this study is to provide the performance analysis of the proposed TRRT object arrangement path planning in the environment of warehouse structure using CoppeliaSim compared to RRT approach.

The overall organization of the study is as follows. Section 2 covers the theoretical aspect of RRT and TRRT approach. Section 3 explains the methodology of the study, Section 4 describes experimental result and discussion. Lastly, section 5 concludes the study and the future directions.

II. Path Planning Approaches

A. Rapidly-exploring Random Tree (RRT)

The Rapidly-exploring Random Tree (RRT) algorithm [15]-[18] is widely-used method for exploring high-dimensional free space while searching for low-cost paths. However, due to its inherent random nature, the paths generated by the traditional RRT algorithm can be suboptimal and lack systematicity. To address this limitation, an enhanced variant called Transition-based RRT (T-RRT) is introduced, which incorporates additional mechanisms to guide the search towards low-cost regions and produce paths that closely adhere to minimal work paths.

The RRT algorithm initiates with an initial state and proceeds iteratively by adding new states to a tree structure until either a goal state is reached, or a maximum number of iterations is reached. During each iteration, the algorithm randomly samples a new state within the search space. This sampled state is then connected to the existing tree by finding the nearest node in the tree. This connection creates a new edge in the tree, which represents a potential path from the nearest node to the newly sampled state.

In order to ensure the feasibility of the path, the algorithm performs collision checks to verify if the newly created path collides with any obstacles in the environment. Additionally, the algorithm may incorporate other constraints, such as velocity or acceleration bounds, to further refine the feasibility of the path. If the path satisfies all constraints and is collision-free, the newly sampled state is added to the tree, expanding the exploration of the search space, and the process continues.

The traditional RRT algorithm is effective at exploring arbitrary search spaces and finding feasible paths through complex obstacles, provided that a feasible path exists within the given configuration space. However, due to its random nature, the paths generated by the RRT algorithm can lack systematicity and be suboptimal in terms of cost or other metrics. The illustration results of RRT algorithm shown in Fig. 1(a).

To address these limitations, the T-RRT algorithm is introduced. T-RRT builds upon the foundation of the RRT algorithm but incorporates additional mechanisms to bias the search towards low-cost regions and generate paths that closely align with minimal work paths. By doing so, T-RRT aims to improve the efficiency and optimality of the path planning process.

B. Transition-based Rapidly-exploring Random Tree (TRRT)

The Transition-based Rapidly-exploring Random Tree (TRRT) algorithm [24]-[25] illustrate in Fig. 1(b) is a planning algorithm that combines the strengths of two methods to efficiently find low-cost paths in complex configuration spaces. The algorithm consists of two stages: exploration and optimization.

In the exploration stage, the algorithm samples a random configuration q_{rand} in the configuration space C and extends the tree T from the nearest node q_{near} towards q_{rand} with a fixed step size. The algorithm ensures that the step size is small enough to approximate well the cost variation between q_{near} and q_{new}, where q_{new} is the new node that results from extending the tree towards q_{rand}. Additionally, the algorithm checks for collision detection, rejecting nodes that collide with obstacles in the space.

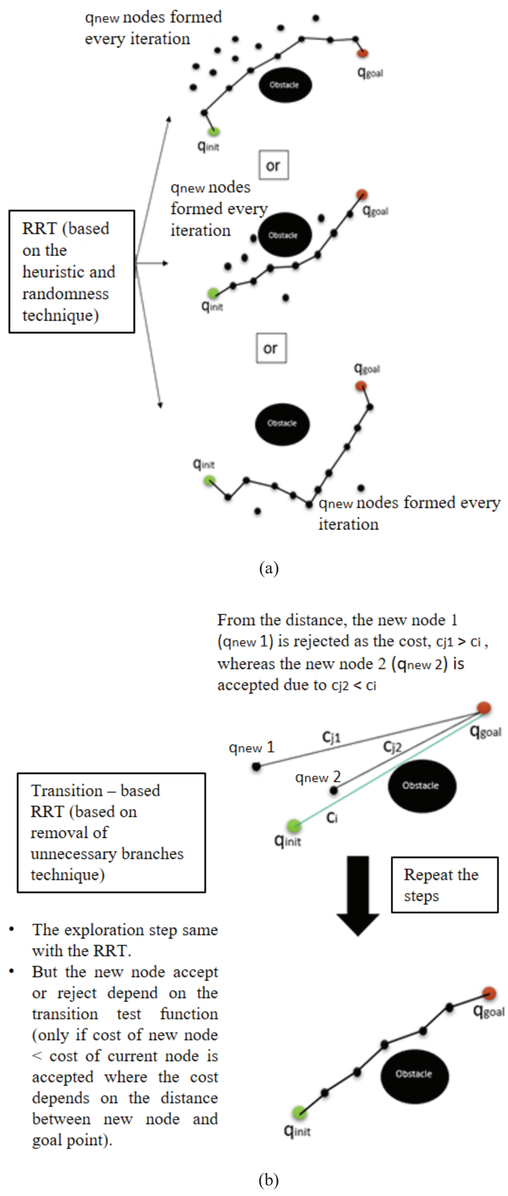
qnew nodes formed every iteration
Obstacle
qgoal
qinit
or
qnew nodes formed every iteration
RRT (based on the heuristic and randomness technique)
Obstacle
qgoal
qinit
or
qgoal
Obstacle
qinit
qnew nodes formed every iteration
(a)
From the distance, the new node 1 (qnew 1) is rejected as the cost, cj1 > ci , whereas the new node 2 (qnew 2) is accepted due to cj2 < ci
qnew 1
Cj1
Cj2
qgoal
qnew 2
Obstacle
Ci
qinit
Transition – based RRT (based on removal of unnecessary branches technique)
Repeat the steps
The exploration step same with the RRT.
But the new node accept or reject depend on the transition test function (only if cost of new node < cost of current node is accepted where the cost depends on the distance between new node and goal point).
Obstacle
qgoal
qinit
(b)

Fig. 1. (a) RRT algorithm and (b) TRRT algorithm

In the optimization stage, the algorithm filters out irrelevant configurations regarding the search of low-cost paths before inserting q_{new} into the tree. The algorithm uses the TransitionTest function, which is based on the Metropolis criterion commonly used in stochastic optimization methods. The function takes the current temperature T and the increase rate T_{rate} as inputs and returns true if the transition from the current node q_{near} to the new node q_{new} is accepted, and false otherwise. The temperature T parameter, despite its seemingly unrelated name, is derived from the Metropolis criterion and acts as a control parameter influencing the acceptance or rejection of transitions. It represents a trade-off between exploration and exploitation within the optimization process. By adjusting the temperature, the algorithm can introduce randomness that aids in escaping local optima and potentially discovering more optimal solutions. The function computes the cost difference between the current node ci and the new node cj, and if the difference is positive, the transition is accepted. Otherwise, the function uses a probability function that depends on the temperature T to accept or reject the transition.

The main loop of the algorithm starts by initializing the tree T with the initial configuration q_{init}. The loop iterates until a stopping criterion is met, such as reaching the goal configuration. At each iteration, the algorithm samples a random configuration q_{rand} and finds the nearest neighbour q_{near} in the tree T. The algorithm extends the tree towards q_{rand}, resulting in a new node q_{new}. The algorithm then checks if the transition from q_{near} to q_{new} is accepted by using the TransitionTest function. If the transition is accepted, the algorithm adds the new node and edge to the tree T.

Overall, the TRRT algorithm efficiently explores the configuration space and finds low-cost paths by combining the exploration bias of RRT-like algorithms with the efficiency of stochastic optimization methods.

III. Methodology

The developed Object Arrangement Robot mainly consists of the block diagram architecture as shown in Fig. 2. To facilitate the implementation and performance analysis of the Transition-based Rapidly-exploring Random Tree (TRRT) and Inverse Kinematics (IK), a dedicated object arrangement path planning robot is developed. The robot development is carried out using the CoppeliaSim software, utilizing the Lua programming language. The designed Object Arrangement Path Planning Robot for retail warehouse environments is depicted in Fig. 3, while Table I provides an overview of the components employed [29]-[33]. Then, the overall process of the object arrangement robot is shown in Fig. 4 in detail.

The UR5 robot inverse kinematics of joint angle calculation formula is used in this study to determine the joint angles required for a robotic arm manipulator to achieve a desired end-effector position. This formula, derived from [27] and [28], serves as a mathematical representation to solve for the joint angles. In this study, these formulae are implemented in MATLAB to calculate the joint angles manually and compare them with the results obtained using the proposed algorithms in CoppeliaSim. This comparison enables an assessment of the accuracy and effectiveness of the algorithms in generating appropriate joint angles for the robotic arm manipulator. By validating the algorithms against the manual calculations, we can evaluate their performance and ascertain their suitability for real-world applications. The formula and the subsequent comparison contribute to assessing the algorithms' accuracy and provide insights into their practical implementation in robotic arm control and motion planning. The joint angle formula for UR5 robot is shown below.

$$\theta_1 = atan2\left({}^{0}P_{5y}\,, {}^{0}P_{5x}\right) \pm \mathrm{acos}\left(\frac{d_4}{\sqrt{{}^{0}P_{5x}{}^{2} + {}^{0}P_{5y}{}^{2}}}\right) + \frac{\pi}{2} \tag{1}$$

$$\theta_2 = \emptyset_1 - \emptyset_2 = atan2(-{}^{1}P_{4z}\,, -{}^{1}P_{4x}) - \mathrm{ASIN}\left(\frac{-a_3\,\mathrm{SIN}\,\theta_3}{|{}^{1}P_{4xz}|}\right) \tag{2}$$

$$\theta_3 = \pm\,\mathrm{ACOS}\left(\frac{|{}^{1}P_{4xz}|^2 - a_2{}^2 - a_3{}^2}{2a_2a_3}\right) \tag{3}$$

$$\theta_4 = atan2({}^{3}\hat{X}_{4y}, {}^{3}\hat{X}_{4x}) \tag{4}$$

$$\theta_5 = \pm\,acos\left(\frac{{}^{0}P_{6x}\,sin\,\theta_1 - {}^{0}P_{6y}\,cos\,\theta_1 - d_4}{d_6}\right) \tag{5}$$

$$\theta_6 = atan2\left(\frac{-{}^{6}\hat{X}_{0y}\cdot sin\,\theta_1 + {}^{6}\hat{Y}_{0y}\cdot cos\,\theta_1}{sin\,\theta_5}, \frac{-{}^{6}\hat{X}_{0x}\cdot sin\,\theta_1 + {}^{6}\hat{Y}_{0x}\cdot cos\,\theta_1}{sin\,\theta_5}\right) \tag{6}$$

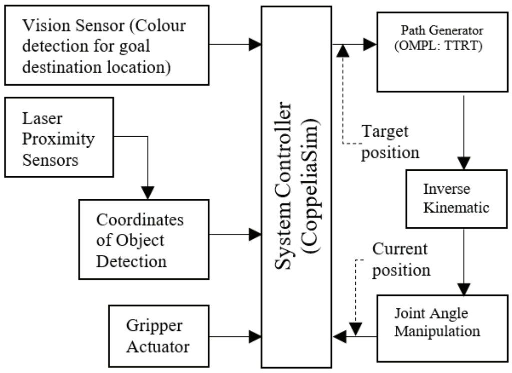
Vision Sensor (Colour detection for goal destination location)
Laser Proximity Sensors
Coordinates of Object Detection
Gripper Actuator
System Controller (CoppeliaSim)
Path Generator (OMPL: TTRT)
Target position
Inverse Kinematic
Current position
Joint Angle Manipulation

Fig. 2. System prototype architecture of Object Arrangement Robot

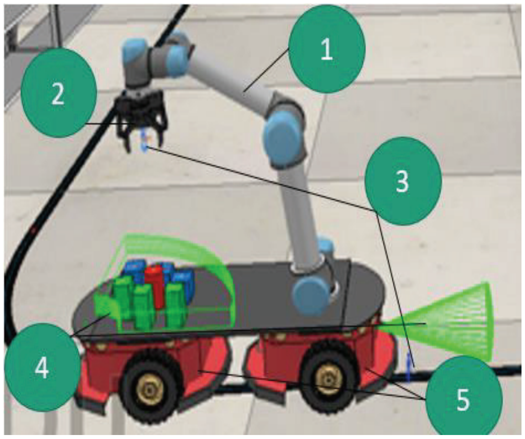

Fig. 3. Object Arrangement Path Planning Robot

TABLE I
COMPONENTS OF OBJECT ARRANGEMENT ROBOT

No.	COMPONENT
1	UR5 Cobot (Collaborative Robot)
2	Robotiq_85 gripper
3	Vision Sensor
4	Proximity Sensor
5	Pioneer PD3X

IV. Experimental Results and Discussion

To evaluate the performance of the basic traditional RRT algorithm and its T-RRT variant, the implementation of object arrangement path planning robot is designed in CoppeliaSim software. The evaluation consists of three experiments to assess various aspects, including path length, algorithm calculation time, repetition performance, manipulation accuracy, and the smoothness of the inverse kinematics implementation in the robotic arm.

A. *Evaluation of Path Length & Calculation Time*

In this section, the T-RRT algorithm is compared to the basic traditional RRT algorithm in terms of path length and calculation time. The recorded path lengths and calculation times are presented in Table II and Table III, respectively. Subsequently, the results are further compared in Table IV, showcasing the differences in path length and calculation time between the T-RRT algorithm and the basic traditional RRT algorithm.

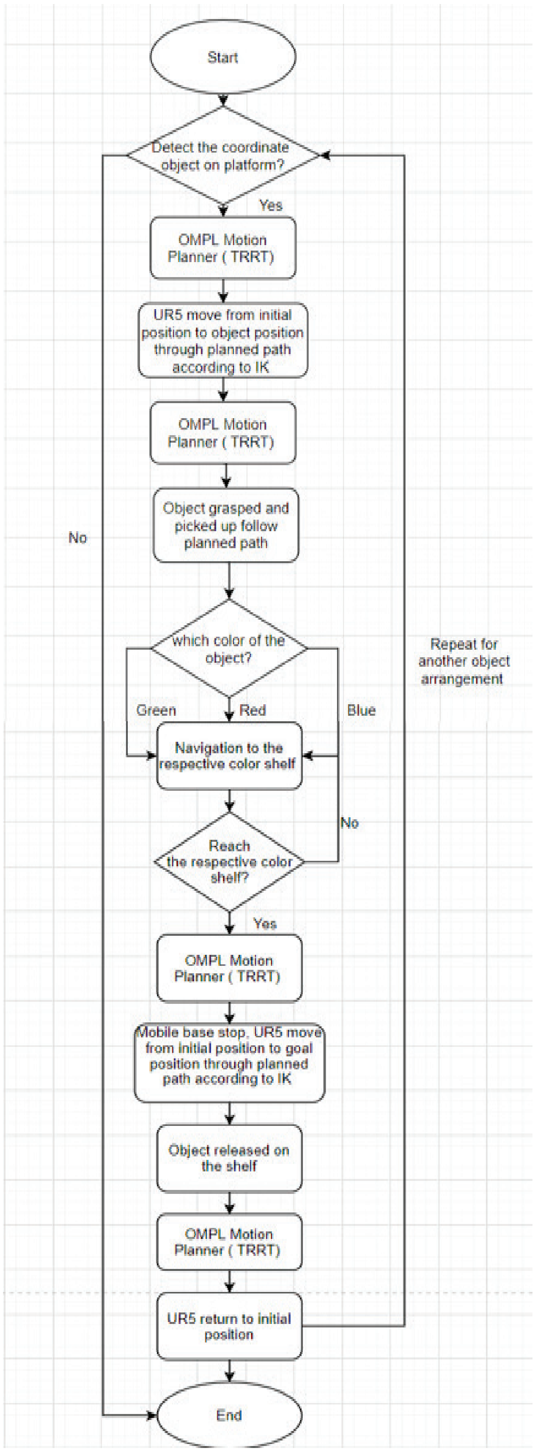
Start
Detect the coordinate object on platform?
Yes
No
OMPL Motion Planner (TRRT)
UR5 move from initial position to object position through planned path according to IK
OMPL Motion Planner (TRRT)
Object grasped and picked up follow planned path
which color of the object?
Green
Red
Blue
Repeat for another object arrangement
Navigation to the respective color shelf
No
Reach the respective color shelf?
Yes
OMPL Motion Planner (TRRT)
Mobile base stop, UR5 move from initial position to goal position through planned path according to IK
Object released on the shelf
OMPL Motion Planner (TRRT)
UR5 return to initial position
End

Fig. 4. Flowchart of Object Arrangement Robot

TABLE II
SIMULATION RESULT OF EXPERIMENT 1 OF RRT

Path	Path Length (m)	Calculation Time (s)
1	1.261	19.44
2	1.263	12.63
3	1.372	16.19
4	1.388	14.19
5	1.413	10.81
6	1.409	11.94
7	1.350	9.81
8	1.349	7.50
9	1.621	8.44
10	1.620	11.19
11	1.378	10.44
12	1.379	5.94
13	1.481	9.56
14	1.630	19.50
15	1.485	18.88
16	1.480	12.19
17	1.781	22.63
18	1.769	27.00
19	2.746	14.31
20	2.741	17.94
21	1.356	6.06
22	1.353	11.63
23	4.211	20.75
24	4.212	20.63
25	1.589	9.56
26	1.579	11.50
27	2.551	14.06
28	2.550	21.63
∑	51.317	396.35

TABLE III
SIMULATION RESULT OF EXPERIMENT 1 OF TRRT

Path	Path Length (m)	Calculation Time (s)
1	1.260	19.38
2	1.259	11.19
3	1.375	17.56
4	1.393	16.25
5	1.401	21.38
6	1.400	20.31
7	1.346	20.31
8	1.346	17.81
9	1.616	22.94
10	1.617	22.19
11	1.387	10.06
12	1.389	17.50
13	1.467	18.50
14	1.483	29.94
15	1.562	22.19
16	1.479	19.63
17	1.782	16.94
18	1.796	19.41

19	2.772	24.13
20	2.745	24.50
21	1.357	16.50
22	1.364	17.00
23	2.641	19.75
24	2.639	25.69
25	1.593	22.63
26	1.579	23.19
27	2.554	25.75
28	2.553	26.81
∑	48.155	569.44

TABLE IV
SUMMATION RESULT FOR RRT AND TRRT OF EXPERIMENT 1

	Path Length (m)	Time(s)
RRT	51.317	396.35
TRRT	48.155	569.44
Difference	3.162	173.09

The results show that when planning paths for object manipulation in a retail warehouse using the RRT and TRRT algorithms, the TRRT algorithm produces a slightly shorter path (difference of 3.162 m) than the RRT algorithm. This finding indicates the TRRT algorithm's ability to find a more direct path from the start to the goal configuration. However, the TRRT algorithm takes longer computation time (173.09 s) due to its time-consuming reduction step involved in pruning branches.

In contrast, the RRT algorithm is faster in computation time because it does not require a reduction step to find a collision-free path. Instead, the RRT algorithm explores the configuration space more randomly, which can result in longer paths. While the step reduction in the TRRT algorithm can lead to a shorter paths, it increases the computation time.

Overall, this experiment shows that the TRRT algorithm is more accurate and efficient in generating shorter paths for object manipulation in this particular environment, albeit with slightly longer computation time than the RRT algorithm. The results provide valuable insights for path planning in retail warehouse environments, where path length and time efficiency are crucial for optimizing operations.

B. *Evaluation of Repetition Comparative Performance*

Repetition tests are inevitable when it comes to robotics performance analysis. In the second experiment, the focus will be on evaluating the repetition performance of the RRT and TRRT algorithms. As an example, the performance of Path 1 planned by both algorithms, simulating its repetition five times, recording and analyzing the path length and calculation time. The comparative repetition performance between the RRT and TRRT algorithms in terms of path length and calculation time is presented in Table V and Table VI.

TABLE V
SIMULATION RESULT OF EXPERIMENT 2 IN TERMS OF PATH LENGTH REPETITION

	1	2	3	4	5	Average
RRT	1.261	1.260	5.03	1.261	4.796	2.722
TRRT	1.260	1.260	1.260	1.260	1.260	1.260

TABLE VI
SIMULATION RESULT OF EXPERIMENT 2 IN TERMS OF CALCULATION TIME REPETITION

	1	2	3	4	5	Average
RRT	13.063	7.563	20.125	13.938	24.5	15.838
TRRT	17.875	21.625	14.625	12.313	15.813	16.450

The results demonstrate that, on average, the TRRT algorithm consistently generates shorter paths (average length of 1.260 meters) compared to the RRT algorithm (average length of 2.722 meters). This finding highlights the superiority of the TRRT algorithm in generating shorter paths. The RRT algorithm's poor performance is attributed to its reliance on randomness and heuristics, resulting in paths of varying lengths. Conversely, the deterministic nature of the TRRT algorithm, combined with its efficient exploration of the configuration space through the "removal of unnecessary branches" technique, consistently generates feasible paths of the same length. However, the TRRT algorithm took slightly longer to compute the paths, with an average calculation time of 16.450 seconds compared to 15.838 seconds for the RRT algorithm.

Overall, this study suggests that the TRRT algorithm is more accurate and efficient in generating shorter paths than the RRT algorithm for path planning of objects in this particular environment. Although the TRRT algorithm may require slightly more computation time, its ability to consistently generate shorter paths is a valuable asset for optimizing path planning in this context.

C. *Evaluation of Smoothness & Accuracy of IK Implementation Evaluation*

In this section, the evaluation performance of the inverse kinematics-based path trajectory in terms of accuracy and smoothness of the robotic arm manipulation will be focused. The analysis involves studying the position of the UR5 Cobot's joint angles, which change according to the position of the end effector through inverse kinematics in CoppeliaSim graphs. The manipulation of joint angles is visualized in Fig. 5, showcasing the changes in each joint's angle over time as the end effector moves to different positions. Simultaneously, Fig. 6 depicts the corresponding movement of the end effector over time. Both figures provided a representative view of the manipulation process, demonstrating the smooth and realistic motion achieved by manipulating the joint angles using inverse kinematics. To assess the accuracy of the system, the joint angle manipulation for the initial point of the end effector from the CoppeliaSim simulation is compared with the Inverse Kinematics Calculation in MATLAB. The results are presented in Table VII. The comparison aims to validate the accuracy of the UR5 Cobot manipulation based on the inverse kinematics method.

TABLE VII
COMPARISON OF JOINTS ANGLE OF UR5 ROBOT AT INITIAL POSITION IN UNIT RADIAN

Angle at 1st point (rad)	Joint					
	J1	J2	J3	J4	J5	J6
Coppelia-Sim	0	0.1544	0.9365	0.5055	-1.5732	-0.2143
MATLAB	0	0.1500	0.9401	0.5100	-1.5708	-0.2142
Error	0	0.0044	0.0036	0.0045	0.0024	0.0001
Accuracy	1	0.9707	0.9962	0.9912	0.9985	0.9995
Average	0.9927 or 99.27%					

Where $accuracy = \frac{|Manual\ Calulation - Simulation\ Calculation|}{|Manual\ Calculation|}$

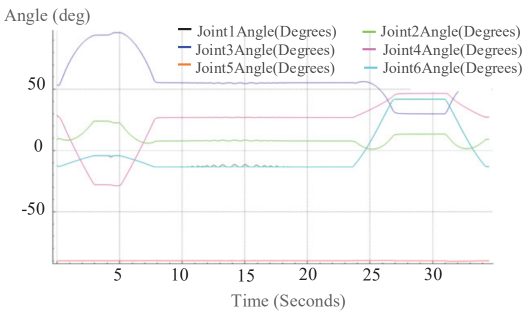

Fig. 5. Angle Joint 1 to 6 of the UR5 Cobot in the first 30 seconds

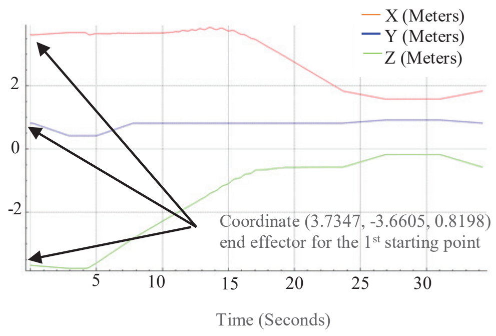

Fig. 6. Coordinates of the end effector position of UR5 Cobot in the first 30 second

Fig. 5 illustrates the manipulation of the UR5 Cobot's joints angles to reach the end-effector positions depicted in Fig. 6, with an update occurring every second. The graph in Fig. 5 shows the changes in the angles of each joint over time, as the end-effector moves to reach different positions. On the other hand, Fig. 6 illustrates the corresponding movement of the end-effector over time, as it reaches different positions. Both figures only display the first 30 seconds of the manipulation process, as the entire process, which lasted approximately 2 minutes, would be too extensive to show in the report. The first 30 seconds, which correspond to the pick and place task of the first object, have been selected to provide a representative view of the manipulation change. It is important to note that the joint angle manipulation in Fig. 5 is done using inverse kinematics, which is a more complex and accurate method. On the other hand, inverse kinematics considers the dynamics of the robot's joints, such as joint limits and singularities, to calculate the optimal joint angles that results in smooth, realistic motion of the end-effector. The smooth and realistic motion can be observed from the graph as it illustrates the smooth motion without any jerky and unrealistic motion. Overall, the graphs demonstrate that the UR5 reaches the destination coordinates by manipulating the joints angles in a smooth and accurate manner.

Furthermore, Table VII compares the joint angles of the UR5 Cobot's first starting point in CoppeliaSim with the inverse kinematics calculation in MATLAB. The results, presented in Table VII, show a small error, indicating high accuracy in the manipulation of the UR5 Cobot based on the inverse kinematics method. This comparison further confirmed the effectiveness of the inverse kinematics approach in producing smooth and realistic motion for the UR5 Cobot. Thus, the combination of graphical visualization and accuracy assessment provide comprehensive insights into the performance of the inverse kinematics-based path trajectory, highlighting its accuracy and ability to produce smooth and realistic motion for the UR5 Cobot.

V. Conclusion

In this study, the TRRT approach is implemented to obtain the shortest path for robotic arm manipulation in warehouse environment. Furthermore, the IK approach is studied and analyzed for smoothness of the arm's movements during manipulation. The results show that the TRRT method generates shorter paths with good repetition performance compared to the traditional RRT approach. Moreover, the IK implementation improves the smoothness and accuracy of the arm's movements and reaches the accuracy of 99.27%. In overall, combining TRRT and IK can result in an efficient and accurate object arrangement path planning robot, leading to increased productivity and performance in retail warehouse environments.

For future research, the TRRT algorithm can be further optimized to reduce computation time and improve path planning performance. There is potential for the TRRT algorithm to be enhanced to handle dynamic obstacles in the environment. At present, the TRRT algorithm assumes a static environment, which is not always the case in real-world applications. It is therefore essential to develop a dynamic TRRT algorithm that can adapt to these situations. Moreover, the TRRT algorithm can be expanded to support multiple robots collaborating on a task. This could be achieved by modifying the transition test function to consider the movements of other robots in the environment. Additionally, the performance of the TRRT algorithm can be tested in more complex environments with tighter spaces and more obstacles. Finally, the TRRT algorithm can be integrated with other planning algorithms, such as a task planner, to generate high-level plans for the robot before using the TRRT algorithm for motion planning. It is anticipated that with these enhancements, the TRRT algorithm will experience significant time improvements while also offering more robust and efficient path planning capabilities.

Acknowledgements

The authors thank to Faculty of Electrical Engineering (FKE) and UTeM for providing the workspace to complete this project.

Conflict of Interest

The authors declare no conflict of interest in the publication process of the research article.

Author Contributions

L.H. Kwee planned the direction of study, experimentation, data analysis and writing the original draft paper. N.M.M. Sobran supervised the study, draft review and editing.

References

[1] T. Motoda, D. Petit, T. Nishi, K. Nagata, W. Wan, and K. Harada, “Shelf Replenishment Based on Object Arrangement Detection and Collapse Prediction for Bimanual Manipulation,” *Robotics*, vol. 11, no. 5, Art. no. 5, Oct. 2022, doi: 10.3390/robotics11050104.

[2] M. Fujita et al., “What are the important technologies for bin picking? Technology analysis of robots in competitions based on a set of performance metrics,” Advanced Robotics, vol. 34, no. 7–8, pp. 560–574, Apr. 2020, doi: 10.1080/01691864.2019.1698463.

[3] S. Jotawar, M. Soni, and S. Kumar, “Motion Planning for an Automated Pick and Place Robot in a Retail Warehouse,” in Proceedings of the Advances in Robotics on - AIR ’17, New Delhi, India: ACM Press, 2017, pp. 1–6. doi: 10.1145/3132446.3134904.

[4] C. Eppner et al., “Four aspects of building robotic systems: lessons from the Amazon Picking Challenge 2015,” Auton Robot, vol. 42, no. 7, pp. 1459–1475, Oct. 2018, doi: 10.1007/s10514-018-9761-2.

[5] “Shelf Stocking Recruitment,” iPlace Recruitment. https://www.iplacerecruitment.com.au/client/recruitment-services /technical-operations/shelf-stocking/ (accessed Nov. 21, 2022).

[6] M. Costanzo, G. De Maria, G. Lettera, and C. Natale, “Can Robots Refill a Supermarket Shelf?: Motion Planning and Grasp Control,” IEEE Robotics & Automation Magazine, vol. 28, no. 2, pp. 61–73, Jun. 2021, doi: 10.1109/MRA.2021.3064754.

[7] “Telexistence to Install AI Restocking Robots in 300 Convenience Stores Across Japan; Validation of technology by Japan’s top-tier convenience store chain sets stage for expansion of ‘robot-as-a-service’ solution in the U.S. – TELEXISTENCE inc.” https://tx-inc.com/en/blog/2022/08/10/11712/ (accessed Dec. 17, 2022).

[8] “Path planning and assembly mode-changes of 6-DOF Stewart-Gough-type parallel manipulators - ScienceDirect.” https://www.sciencedirect.com/science/article/pii/S0094114X1630 1835 (accessed Mar. 22, 2023).

[9] J. J. Kuffner and S. M. LaValle, “RRT-connect: An efficient approach to single-query path planning,” in Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), Apr. 2000, pp. 995–1001 vol.2. doi: 10.1109/ROBOT.2000.844730.

[10] “trajectory.pdf.” Accessed: Apr. 23, 2023. [Online]. Available: http://www.cs.columbia.edu/~allen/F15/NOTES/trajectory.pdf

[11] “Lecture: Keyframe Animation and Linear Interpolation.” http://titan.csit.rmit.edu.au/~e20068/teaching/i3dg&a/2017/animation.html (accessed Apr. 26, 2023).

[12] N. Abdo, C. Stachniss, L. Spinello, and W. Burgard, “Robot, organize my shelves! Tidying up objects by predicting user preferences,” Proceedings - IEEE International Conference on Robotics and Automation, vol. 2015, pp. 1557–1564, Jun. 2015, doi: 10.1109/ICRA.2015.7139396.

[13] Y. Jiang, M. Lim, C. Zheng, and A. Saxena, “Learning to Place New Objects in a Scene.” arXiv, Feb. 08, 2012. doi: 10.48550/arXiv.1202.1694.

[14] M. Kang, Y. Kwon, and S. Yoon, Automated task planning using object arrangement optimization. 2018, p. 341. doi: 10.1109/URAI.2018.8442210.

[15] “Lav98c.pdf.” Accessed: Mar. 22, 2023. [Online]. Available: http://msl.cs.illinois.edu/~lavalle/papers/Lav98c.pdf

[16] Y. Liu, H. Zhao, X. Liu, and Y. Xu, “An Improved RRT Based Obstacle Avoidance Path Planning Algorithm for Industrial Robot,” Information and Control, vol. 50, no. 2, pp. 235–246 and 256, 2021, doi: 10.13976/j.cnki.xk.2021.0259.

[17] Y. Wang, J. Tian, Z. Liu, H. Liu, J. Liu, and L. Xie, “Path planning of manipulator based on improved RRT algorithm,” J. Phys.: Conf. Ser., vol. 2216, no. 1, p. 012012, Mar. 2022, doi: 10.1088/1742-6596/2216/1/012012.

[18] R.R.W, “Robotik - Ep.2: Intro to Motion Planning and Rapidly-exploring Random Tree (RRT),” rey’s blog - democratizing robotics., May 16, 2020. https://rrwiyatn.github.io/blog/robotik/ 2020/05/16/rrt.html (accessed Nov. 06, 2022).

[19] Y. Chen, Y. Fu, B. Zhang, W. Fu, and C. Shen, “Path planning of the fruit tree pruning manipulator based on improved RRT-Connect algorithm,” International Journal of Agricultural and Biological Engineering, vol. 15, no. 2, pp. 177–188, 2022, doi: 10.25165/j.ijabe.20221502.6249.

[20] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” The International Journal of Robotics Research, vol. 30, no. 7, pp. 846–894, Jun. 2011, doi: 10.1177/0278364911406761.

[21] C. Dath and L. Lundin, “Comparing different sampling-based motion planners in multiple configuration spaces,” p. 55, 2018.

[22] F. Islam, J. Nasir, U. Malik, Y. Ayaz, and O. Hasan, “RRT∗-Smart: Rapid convergence implementation of RRT∗ towards optimal solution,” in 2012 IEEE International Conference on Mechatronics

and Automation, Aug. 2012, pp. 1651–1656. doi: 10.1109/ICMA.2012.6284384.
[23] M. Shahabi, H. Ghariblu, and M. Beschi, "Comparison of different sample-based motion planning methods in redundant robotic manipulators," Robotica, vol. 40, no. 9, pp. 3104–3119, Sep. 2022, doi: 10.1017/S026357472200008X.
[24] L. Jaillet, J. Cortes, and T. Simeon, "Transition-based RRT for path planning in continuous cost spaces," in 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice: IEEE, Sep. 2008, pp. 2145–2150. doi: 10.1109/IROS.2008.4650993.
[25] D. Devaurs, T. Siméon, and J. Cortés, "A multi-tree extension of the transition-based RRT: Application to ordering-and-pathfinding problems in continuous cost spaces," in 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Sep. 2014, pp. 2991–2996. doi: 10.1109/IROS.2014.6942975.
[26] J. Jermyn, "A Comparison of the Effectiveness of the RRT, PRM, and Novel Hybrid RRT-PRM Path Planners," IJRASET, vol. 9, no. 12, pp. 600–611, Dec. 2021, doi: 10.22214/ijraset.2021.39297.
[27] "ur5_kinematics.pdf." Accessed: Dec. 18, 2022. [Online]. Available: http://rasmusan.blog.aau.dk/files/ur5_kinematics.pdf
[28] R. Keating and U. N. J. Cowan, "M.E. 530.646 UR5 Inverse Kinematics".
[29] J. Villalobos, I. Y. Sanchez, and F. Martell, "Singularity Analysis and Complete Methods to Compute the Inverse Kinematics for a 6-DOF UR/TM-Type Robot," Robotics, vol. 11, no. 6, Art. no. 6, Dec. 2022, doi: 10.3390/robotics11060137.
[30] "V-REP Robot Simulator," AZoRobotics.com. https://www.azorobotics.com/software-details.aspx? SoftwareID= 27 (accessed Dec. 20, 2022).
[31] "Pioneer3DX-P3DX-RevA.pdf." Accessed: Dec. 17, 2022. [Online]. Available: https://www.generationrobots.com/media/ Pioneer3DX-P3DX-RevA.pdf
[32] "UR5," Clearpath Robotics. https://store.clearpathrobotics.com/ products/universal-robots-ur5 (accessed Dec. 17, 2022).
[33] "Specifications." https://assets.robotiq.com/website-assets /support_documents/document/online/2F-85_2F-140_TM_InstructionManual_HTML5_20190503.zip/2F-85_2F-140_TMInstruction_Manual_HTML5/Content/6.%20Specifications.htm (accessed Dec. 17, 2022).