IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 4, 2020, accepted December 27, 2020, date of publication January 28, 2021, date of current version February 8, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3055455*

# Combining a Monte Carlo Tree Search and a Feasibility Database to Plan and Execute Rearranging Tasks

**PEDRO MIGUEL URIGUEN ELJURI, (Graduate Student Member, IEEE), GUSTAVO ALFONSO GARCIA RICARDEZ, (Member, IEEE), NISHANTH KOGANTI, JUN TAKAMATSU, (Member, IEEE), AND TSUKASA OGASAWARA, (Member, IEEE)**
Nara Institute of Science and Technology, Nara 630-0192, Japan

Corresponding author: Pedro Miguel Uriguen Eljuri (pedro.uriguen.pl3@is.naist.jp)

**ABSTRACT** In this paper, we address the problem of solving rearranging tasks using a robot. Rearranging tasks are challenging because they include many problems to solve at the same time, such as determining how to pick the items as well as planning how and where to place them. Solving a rearranging task usually consists of finding a set of pick-and-place instructions with a symbolic planner to perform the task. However, if the symbolic planner does not consider the robot's capability to execute the instructions, it will likely generate many infeasible instructions, which wastes time in multiple trials and failures. Therefore, we propose to combine symbolic and motion planning to confirm a sequence of instructions before its execution by the robot. To achieve this combination, we use a Motion Feasibility Checker (MFC), which selects a set of feasible poses for the robot from a feasibility database. The MFC verifies that the instructions of the symbolic planning are valid and searches for a pick-and-place pair of poses to execute the instructions. We use a Monte Carlo Tree Search (MCTS) as the symbolic planner, and we combine it with the MFC when creating or expanding the states in the tree. After the MCTS finds a set of instructions for the rearranging task, we execute those instructions with the robot. As these instructions were previously validated, the robot is able to execute them. The proposed method was tested in a simulation environment that reproduces the scenario of rearranging products on a shelf of a convenience store. The experiment results show that the proposed method outperforms the conventional method in various initial states of increasing levels of difficulty.

**INDEX TERMS** Rearranging task, task planning, symbolic planning, motion planning, service robot.

## I. INTRODUCTION

The use of robots outside factories has become more common nowadays. We have robots that help us with daily tasks, such as floor sweeping or vacuum cleaning, e.g., Roomba[1] and RURO.[2] However, there is still a tedious and time-consuming task performed by humans daily: rearranging task. This is the second most common task in a daily-life routine [1], so its automation using robots is expected. Examples of rearranging tasks are tidying up a room, organizing items on a desk, and moving items on a shelf.

A rearranging task is challenging because of the technical difficulties of determining how to pick-and-place the items with the robot and where to place them in the environment. This is difficult because the robot often requires to perform at least one re-grasp to be able to place the item in the target pose. On top of that, realistic scenarios also present disturbances, e.g., target items moved by humans or by other phenomena.

In this work, we focus on the task planning of rearranging tasks. Specifically, we focus on the symbolic and motion planning, i.e., how to obtain the sequence of instructions that the robot needs to execute to reach the goal state or final rearrangement. Furthermore, we consider realistic conditions where re-grasps are needed and disturbances are expected.

The most common approach to solve a rearranging task is to first solve the problem logically at the symbolic level by obtaining a set of instructions and then execute them with the robot [2]. This approach is time-consuming because, when the robot fails to execute the instruction (e.g., due to a failure in creating the trajectory during motion planning), it is required to execute the symbolic planner again to obtain a new set of instructions. This pattern of getting new instructions without knowing if the robot will be able to

The associate editor coordinating the review of this manuscript and approving it for publication was Min Wang.

[1]Roomba robot from iRobot, https://www.irobot.com/roomba

[2]RURO robot from Panasonic, https://panasonic.jp/tourist/en/soji/

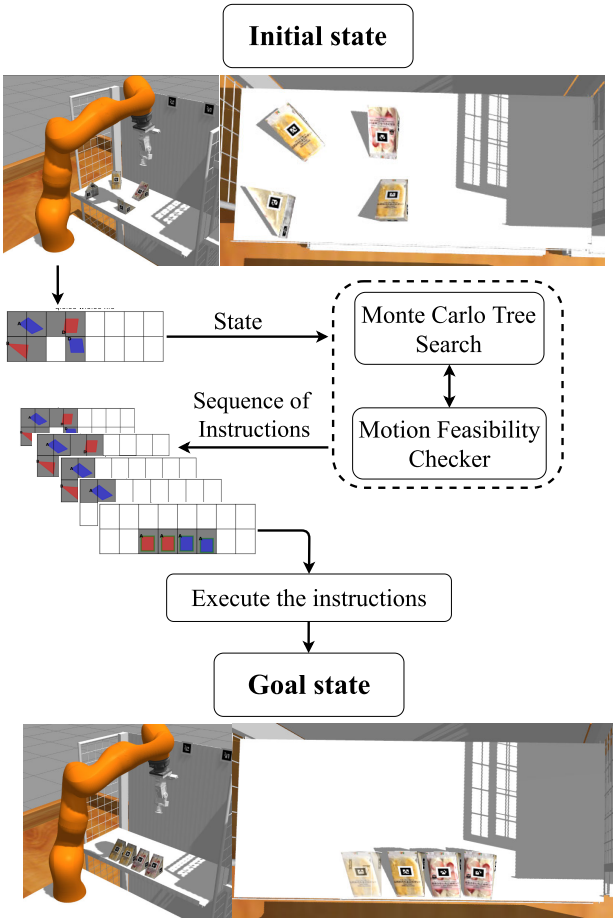



**FIGURE 1.** Overview of the proposed method.

execute them increases the risk of falling into a loop of invalid instructions, thus spending time trying to find a solution that ultimately may not be executable with the robot.

In this paper, we propose to solve a rearranging task by combining symbolic and motion planning at the moment of generating the set of instructions. The proposed method avoids spending time in poses where the motion planning can fail by using a valid set of pre-computed poses obtained using motion planning. This early rejection shortens the overall planning time. The validation is done using the Motion Feasibility Checker (MFC) and the pre-computed poses stored in a *feasibility database*.

We propose to use the Monte Carlo Tree Search (MCTS) [3], [4] as the symbolic planner, as it considers multiple options or sequences to reach the goal. Fig. 1 shows an overview of the proposed method. First, the initial state is given to the proposed method. Second, the proposed method uses the MCTS and MFC to create a tree of states and search for a sequence of instructions. Third, the proposed method founds a sequence of instructions. Finally, the robot executes those instructions and reaches the goal state.

Through simulation experiments, we evaluate the performance of the conventional and proposed methods, as well as their robustness to disturbances (e.g., unexpected item movements). We set a task of rearranging items in a convenience store scenario where items need to be nicely ordered for new customers, and we prepared initial cases with three levels of difficulty. We evaluate the compared methods under two conditions: without disturbances in the environment, and with forced disturbances. The results show that our proposed method can successfully solve the rearranging task, even when re-grasps are necessary and disturbances are present, and outperform the conventional approach.

The rest of the paper is organized as follows. Section II explains related works on solving rearranging tasks. Section III describes our proposed method. Section IV introduces the rearranging task used to test our method and compare it to other methods. Section V presents the results obtained in the experiments. Section VI includes a discussion. Finally, Section VII concludes this paper and provides some directions for future work.

## II. RELATED WORKS

Rearranging tasks are a complex problem that has been tried to be solved before, e.g., [5]–[7]. The goal of a rearranging task is to find a sequence of instructions to move a set of items from an initial state to a target arrangement (goal state). There are multiple approaches to solve a rearranging task, such as hierarchical [7]–[9] and randomized approaches [10]. Some approaches focus on verifying how the actions of the robot would affect the environment using a geometric planner and selecting the first solution that has no collisions with the environment [5], [11].

To combine the symbolic and motion planners to perform an organizing task has been proposed [12]. This approach focuses on using a Probabilistic Roadmap Method [13] to generate new states, validate them and select the state that has the lowest cost among them. Dantam *et al.* [14] proposed to obtain multiple solutions to be saved in a set, regardless of the cost. In the case that the robot can not execute the task, the task planner will attempt to execute the next solution from the set.

In most recent works, it has been proposed to use a machine learning approach to solve rearranging tasks [15]–[17]. These works focus mainly on how to reach the goal state with less movements of the items or on rapidly obtaining a new state to move the items [16], [18]. One of their limitations is that they assume that the motion planning can always execute the pick-and-place motion and that the approach to the items is always from above. In our proposed method, we validate the motions of the robot before executing them. We do these validations using a database of feasible poses of the robot, which avoids failures in motion planning.

## III. PROPOSED METHOD

### A. OVERVIEW

We propose to combine the symbolic and motion planning when generating the solution for a rearranging task.

We assume that a rearranging task can be achieved by repeating the following actions: 1) a robot moves to pick an item, 2) returns to a neutral pose keeping the grasp of an item, 3) moves to release an item in the target location,

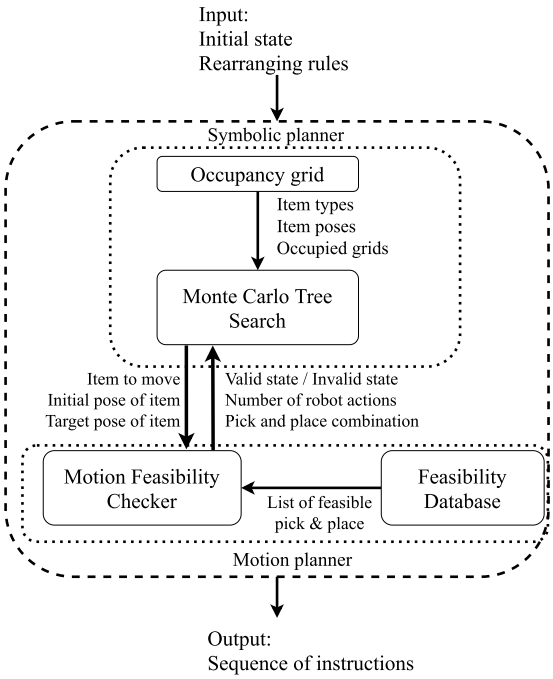



**FIGURE 2.** Components of the proposed method.

and 4) returns to a neutral pose. We consider that the robot temporarily releases the item before a re-grasp. We also consider the possible grasping points of the items as known information, which is determined beforehand based on the geometry of the items. These grasping points are manually defined and the proposed method chooses automatically the grasping point to do the pick-and-place of the item. Thus, the symbolic planning should decide what item to pick, how to grasp it, and where to release it, whereas motion planning should generate the robots movements to achieve the instructions in the symbolic plan. Since the robot always returns to a neutral pose, we can classify the movements into those between a picking pose and a neutral pose, and those between a placing pose and a neutral pose. Fig. 2 shows the components of the proposed method, and the input and output of each component.

The MFC verifies if the trajectory between the neutral pose and the given pose to pick or place exists. The MFC considers the feasibility of the robot and does not consider collisions to other items. To accelerate the judgment, we use a pre-computed feasible motion database.

The MCTS explores various object arrangements, i.e., states, and selects the path with the highest ratio between cumulative reward of the states and number of visits to each state. One of the advantages of the MCTS is that it can obtain a solution at any moment during the tree search.

Since it is useless to explore states that the robot cannot reach, the MCTS collaborates with the MFC for rejecting invalid states with respect to reachability. The MFC evaluates the validity with respect to the kinematics (e.g., picking an item with a certain grasping pose and being able to release it at the target configuration). Thanks to the MFC and the feasibility database, checking the validity of the state and their transition is very quick.

Note that calculating the collision-free movement is a task not for the MCTS and the MFC, but for motion planning. After the MCTS finds the state transitions to reach the rearrangement goal, the method generates the actual robot movements using the MFC's output and motion planning. It is also important to mention that the pick-and-place poses are obtained offline, and during the execution of the task the motion planner is executed online. Finding a trajectory between the neutral configuration and the pick or place poses obtained offline is faster, because all the poses were already tested offline so the probability of failure in finding a trajectory is relatively low.

### B. STATE REPRESENTATION

To solve a rearranging task, we need to create a representation of the environment that can be used by the symbolic planner. We use the properties of the items as a state: the geometry of the item, its pose in the environment, and the type of item.

From that state, we can create an occupancy grid representation of $m$ columns by $n$ rows to keep track of the spaces occupied by the items and the available spaces, which are candidates for place positions. In cases where an item uses more than one grid space, all the spaces used by that item are considered occupied. We also consider that more than one item can share the same grid. The initial state of the environment is considered as the root of the tree for the symbolic planning.

### C. SYMBOLIC PLANNING

At the symbolic level, solving a rearranging task is creating the instructions the robot needs to execute. We chose an MCTS [3], [4] as a symbolic planner because it can obtain a solution at any time, and its randomness allows a balance between exploration and exploitation of the tree. One of the advantages is that an MCTS does not require exploring all the states of the tree to find a solution. This is very useful to select a solution that has a high probability of success.

The MCTS efficiently explores and exploits the states of a tree to solve a complex problem. Examples of these problems can be games such as chess, Go, or tic-tac-toe. The decisions of the MCTS are represented as a tree of states, where each state is linked by state transitions of possible actions of the robot moving an item. The MCTS chooses the solution which has the highest ratio between rewards and visits. The MCTS generates new states by moving the items to the available spaces in the occupancy grid. The MCTS considers moving the item to the target goal pose. The MFC checks if this movement of the item is feasible or not. Fig. 3 shows the four stages of the MCTS: selection, expansion, simulation, and back-propagation [3], [4]. The execution of these four stages constitutes one iteration of the MCTS. In our proposed method, we modified the expansion and simulation stages of the MCTS to use the MFC.

In the expansion stage, the MCTS generates one or more states that have yet to be explored. From the generated states,

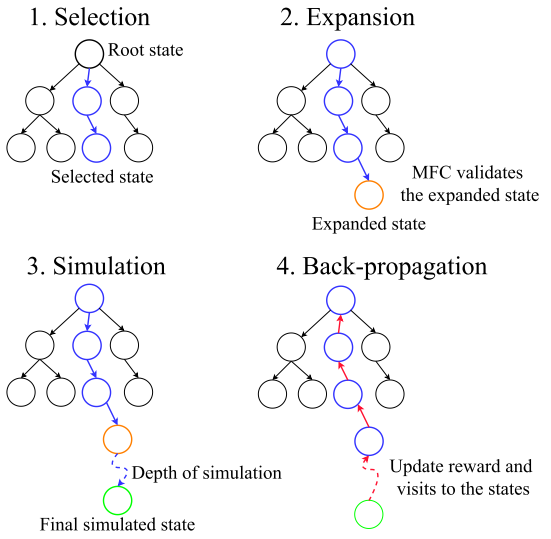



**FIGURE 3.** Stages of the Monte Carlo Tree Search using the MFC.

one state is randomly selected to be used in the simulation stage. During the expansion stage, we prune the tree using the MFC to remove invalid states. We perform this tree pruning to avoid the MCTS spending time in invalid states during its iterations.

In the simulation stage, from the new state created in the expansion stage, the MCTS generates random valid states, moving one item randomly, until one of the following three conditions is met: it reaches a final state that cannot be further explored, it reaches the goal state, or it reaches the maximum depth of the simulation. These generated states are validated with the MFC to ensure that all the states used in the MCTS are feasible. The maximum depth of simulation allows the MCTS to finish the simulation stage early if it does not reach a final state and to continue with the execution of the algorithm. The last simulated state is evaluated using the rearranging rules, which are the guidelines of how and where to place the items in the environment.

The back-propagation stage updates the reward values and number of visits to each state in the path until the expanded state in the tree. We use a reward $r(s_i)$, where $s_i$ is the last simulated state in the simulation stage. The reward is based on the rearranging rules.

Once we complete an iteration of the MCTS, the process is repeated until the tree search is finished. The tree search ends when one of the following stopping conditions is achieved: we obtain the maximum score in the rearranging task or a set time has passed. In case that the search ends because of the maximum score, the path that reaches the maximum score is selected. Otherwise, the sequence selected is based on the states in the tree that have the highest ratio between the accumulative reward and the number of visits.

### D. MOTION FEASIBILITY CHECKER

The MFC validates the received state in regard to the robot motions, searches for a combination of pick and place poses from the feasibility database, and determines the number of pick-place actions that the robot needs to reach the states in the MCTS. The input of the MFC is the current state and the target state of the item. The output of the MFC is the validity of the state, and the sequence of pick-and-place poses of the end-effector to reach the target state, if the state is deemed valid.

The target state of the MCTS has the information of the item to be moved, namely, the initial pose $T_{\text{initial}}$ and target pose $T_{\text{target}}$ of the item. Based on $T_{\text{initial}}$, the MFC decides the grasping point of the item and then obtains a set of pick and place candidates that are close to the grasping point from the feasibility database. These candidates can be slightly deviated from the planned grasping point because of the sampling of the database. Then, the MFC uses these poses to search for a combination of pick and place poses that closely approximates the final pose of the item to the required target pose in the state of the MCTS.

The MFC obtains $T_{\text{predicted}}$ by calculating the transform of the item, if it were picked and placed with a pick and place poses from the databases. This is an approximation of how the robot would place the item in the environment. Then, the MFC compares and calculates the difference between $T_{\text{predicted}}$ and $T_{\text{target}}$ using (1).

$$f(T_{\text{target}}, T_{\text{predicted}}) = w_1 f_d(T_{\text{target}}, T_{\text{predicted}}) + w_2 f_g(T_{\text{target}}, T_{\text{predicted}}), \quad (1)$$

where $f_d$ is the Euclidean distance between the positions and $f_g$ is the difference in the orientation. We use the geodesic unit sphere [19] to calculate the difference between the orientations of the target and predicted poses. We use $w_1$ and $w_2$ to represent the respective weights for the position and orientation. In case that the MFC receives multiple pick-and-place candidates from the feasibility database, the MFC will calculate all the possible combinations between those pick and place poses, then select the combination that places the item the closest to $T_{\text{target}}$. We use a threshold $th$ to compare with the result of (1) between $T_{\text{target}}$ and $T_{\text{predicted}}$. If the value is greater than $th$, we need to do a re-grasp of the item to achieve the task. The intermediate pose of the item for the re-grasp is the place pose of the item $T_{\text{predicted}}$ from the previous search that was the closest to $T_{\text{target}}$. This $T_{\text{predicted}}$ is considered as the new initial pose $T_{\text{initial}}$ for the new search.

The total number of pick-place actions required to reach $T_{\text{target}}$ is also considered when the MCTS needs to evaluate the state. We compare the total number of pick-place actions to a maximum number of actions already defined. In case that the total number of actions is greater than the maximum number of actions, the MFC will determine that the tentative state is invalid and the MCTS will erase that invalid state from the tree. In case that the tentative state is valid, the MFC will return the combination of pick and place poses and the total number of actions of the robot that are necessary to reach the tentative state. Fig. 4 shows a flowchart of how the MFC validates a state.

The process to obtain the pick and place candidates from the database is the following. First, based on the initial pose

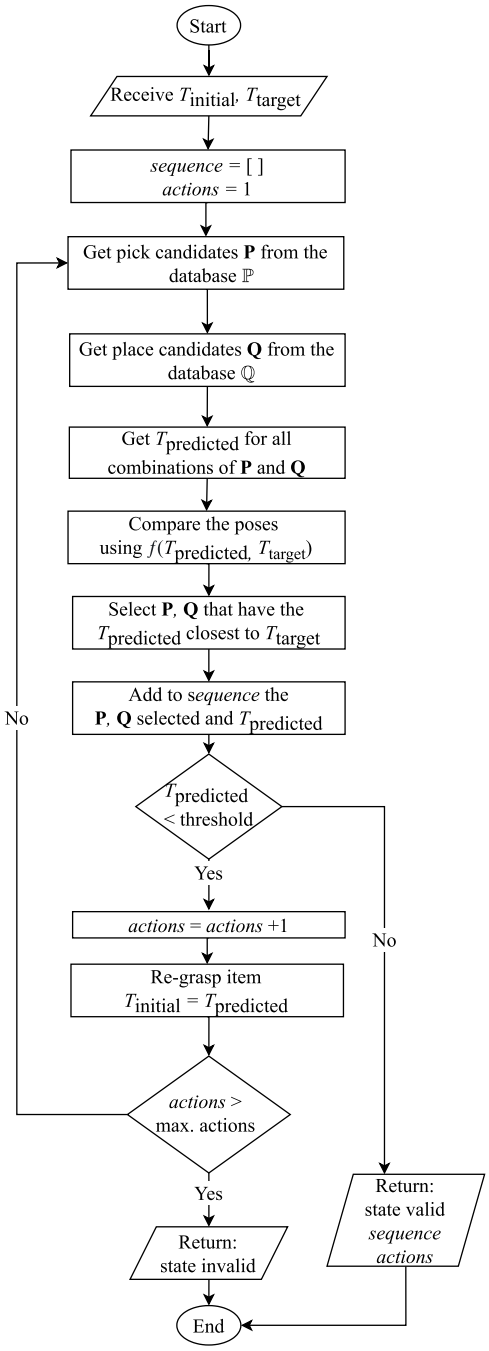



**FIGURE 4.** Validation of a state using the MFC.

of the item $T_{initial}$, we select a grasping point. The grasping points are defined beforehand depending on the geometry of the item (e.g., center of the faces). Second, we obtain the pose of the grasping point with respect to the robot. Finally, we search and select in the database for the poses that are close to the grasping point. The distance that we consider to the poses from the grasping point depends on the sampling of the database. To select the place candidates we use the grasping point obtained for the pick. Then, using the target pose of the item $T_{target}$, we obtain the ideal pose of the grasping point in the target. Finally, we search in the database for the poses close to that point, as we did for the pick.

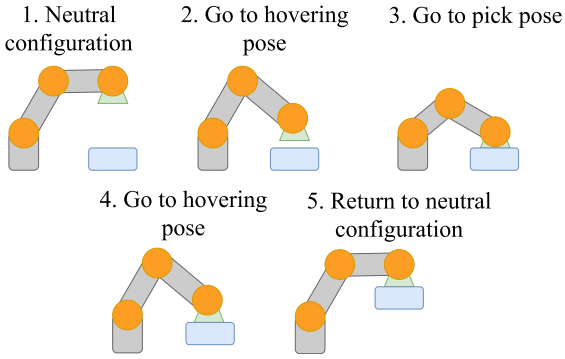



**FIGURE 5.** Illustration of the pick or place maneuvers.

Note that the MFC is an approximation of the motion planner. Thus, the MFC has some limitations compared to the motion planner. These limitations are on the sampling of the feasibility database and the collisions of the robot with the environment that the MFC does not consider when it checks if a state is executable. The limitation of the collisions with the environment can be solved by adding the environment information at the moment of creating the feasibility database.

### *E. FEASIBILITY DATABASE*

The feasibility database contains valid poses of the robot's end-effector. We consider as a feasible pose a pose that is reachable by the robot in different orientations and where we can execute a pick or place maneuver.

We consider as a pick or place maneuver the motion from a neutral configuration (home position) of the robot to a hovering pose over the grasping point, from the grasping point back to the hovering pose, and finally from the hovering pose to the neutral configuration, as shown in Fig. 5. The neutral configuration of the robot is defined as a joint angle configuration; this ensures the trajectories to the pick and place poses start always from the same robot configuration.

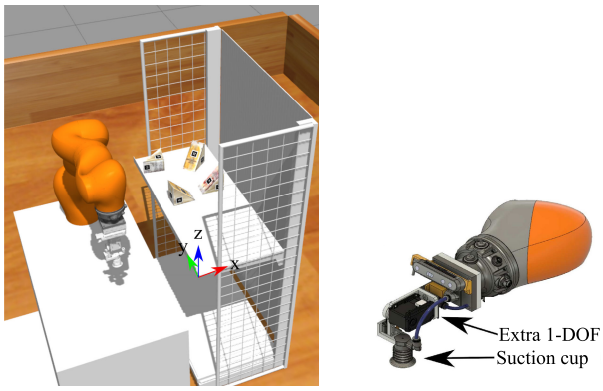
The process of creating the feasibility database is the following. First, we create a set of pick and place pose candidates $\mathbb{P}$ and $\mathbb{Q}$, respectively. Then, each pose is validated with the motion planner moving from the neutral configuration to hovering pose and, finally, to the pick or place poses. We discard the invalid poses from the set.

To validate the pose, we use the motion planner to create a trajectory to the pose of $x$, $y$, $z_{\text{hovering}}$, roll, pitch, yaw where $z_{\text{hovering}}$ is an offset in the $z$-axis of the pose and roll, pitch, yaw are the orientation of the end-effector. Second, in case that the planner can create a trajectory to the hovering pose, we attempt to create the trajectory to reach the pick or place poses. Finally, we return to the neutral configuration. If the motion planner succeeds in creating a trajectory for the pick or place poses, we add that pose to the set of valid poses. A reachable pose is not always a feasible pose. This is because we need to reach the pose from a specific orientation.

After checking all the poses for pick and place, we obtain

$$\mathbb{P} = \left[\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \cdots, \mathbf{P}_n\right],$$

$$\mathbb{Q} = \left[\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{Q}_3, \cdots, \mathbf{Q}_m\right],$$




(a) Simulation environment. (b) End-effector.

**FIGURE 6.** Simulation environment and end-effector used in the experiments.

where $\mathbb{P}$ and $\mathbb{Q}$ are the pick and place poses sets, respectively. Each element $\mathbf{P}$ and $\mathbf{Q}$ is a $x$, $y$, $z$, roll, pitch, yaw pose of the end-effector, $n$ and $m$ are the number of valid poses for the sets of pick and place, respectively.

The reason why we use two different sets of valid poses is because we consider that we can pick an item from multiple directions but, at the moment of placing the item, we have more restrictions in how to place it based on the rearranging rules.

## IV. EXPERIMENTAL SETUP

### A. SIMULATION ENVIRONMENT

To evaluate our proposed method, we executed a sandwich rearranging task in a simulation environment. We used a simulation environment in Gazebo.[3] We used a robot-arm *KUKA LBR iiwa 14 R820*[4] controlled through the open-source package *iiwa_stack* [20] with ROS.[5]

The robot arm has 7 DOF and is mounted on a fixed base in front of a shelf, as shown in Fig. 6. The position of the shelf's bin is at 0.34 m, −0.60 m and −0.15 m in $x$, $y$, $z$, respectively, from the base of the robot arm. To manipulate the objects, we used a custom-made end-effector with an extra DOF and a suction cup, as shown in Fig. 6b.

### B. REARRANGING RULES

The rearranging task is based on the restock and disposal task of the Future Convenience Store Challenge (FCSC) [21], [22], one of the challenges in the World Robot Challenge 2018[6] (WRC) held in the World Robot Summit 2018. The aim of the FCSC is to automate various tasks done in a convenience store.

We consider this sandwich rearranging task challenging because it requires to pick-and-place an item complying with a set of rules. Also, during FCSC 2018, none of the teams that participated were able to complete the sandwich rearranging task. One of the many challenges of this task is to change the orientation of the item. To achieve this, the robot requires to do multiple re-grasps of the item.

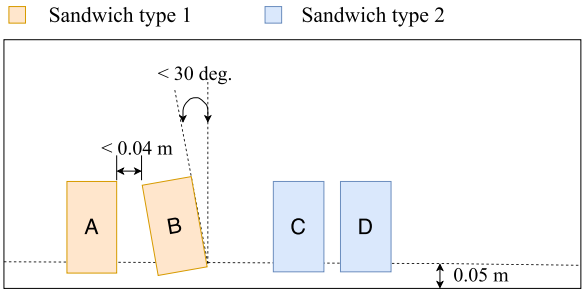



**FIGURE 7.** Illustration of a correct final state, all the items are in the correct position and orientation on the shelf. Item B is not aligned as the others, but its orientation is still valid by the rules. Items of the same type are together. The total score of this rearrangement is 12 points, which is the maximum score.

The sandwich rearranging task consists of rearranging four sandwiches in a shelf. The rules of the rearranging task are the same as in the restock and disposal task of FCSC[7]:

- The bottom surface of the product must be in contact with the shelf.
- The label of the product faces the front.
- The tolerance of the orientation is 30°.
- All products should be placed within 0.05 m from the edge of the shelf.
- Items of the same type must be grouped and placed within 0.04 m from each other.

We evaluate the final rearrangement based on the rearranging rules. The score for each item in correct position and orientation is three points. The maximum score is 12 points. Fig. 7 shows an illustration of a correctly rearranged state.

### C. METHODS FOR COMPARISON

To execute this sandwich rearranging task, we consider the following three methods and their variations:

- **Conventional method** (symbolic planner independent of the motion planner). In this method, if there is a failure in the motion planner while executing the instructions with the robot, the symbolic planner is executed again. The symbolic planner is just a greedy algorithm that searches for the first valid solution for moving an item. This algorithm receives the information of the items, checks if there is an item of the same type in the front of the shelf and selects the closest empty grid to the existing item. If there are not other items of the same type, it selects the grid with more empty grids next to it in the front of the shelf. We consider two variations of this method for the experiments. In the variation A, the symbolic planner only obtains the information of the environment one time. Then, the robot executes the instructions from the symbolic planner. In the variation B, after moving an item, the symbolic planner

[3]Gazebo, http://gazebosim.org/
[4]KUKA LBR iiwa 14 R820, https://www.kuka.com/en-de/products/robot-systems/industrial-robots/lbr-iiwa
[5]Robot Operating System, http://www.ros.org/
[6]World Robot Challenge, https://worldrobotsummit.org/en/index2018.html
[7]FCSC rule book, https://worldrobotsummit.org/en/wrc2018/service/pdf/Rulebook_task1.pdf

obtains again the information of the environment and generates a new set of instructions for moving the items.

- **Conventional method using MFC**. This method is similar to the Conventional method, but the instructions are validated with the MFC before executing them with the robot. If an instruction is not valid, the symbolic planner is executed again. Similar to the Conventional method, we consider the variations A and B for this method.
- **Proposed method**. This method uses an MCTS and the MFC to combine the symbolic and motion planning. The MCTS creates a tree of states and validates the states with the MFC. Then, the MCTS finds the instructions that can be executed with the robot. In the proposed method, we also consider two variations for the experiments. In the variation A, the stop condition is when the MCTS finds a path that obtains the maximum score. In case that there is a disturbance in the environment during the execution, the previously obtained solution is abandoned and the MCTS will search for a new solution from the current state. This variation obtains the instructions for moving all the items before the execution. In the variation B, the stop condition is a set time. When the stop condition is reached, the MCTS will select the state with highest possibility of success (i.e., the ratio between the accumulative reward of the state and its number of visits). After each movement of the item, the MCTS will update the information of the environment. This variation obtains the instructions for moving one item at a time.

During the execution of the instructions, the robot receives the information of the pose of the item, before grasping it. This is done for all the methods to compensate for disturbances in the environment before the pick. In the case of the methods that use the MFC, a new search of the MFC is done and a new sequence of pick and place poses of the robot end-effector is obtained. Because all the methods obtain the current pose of the item before the robot grasps it, we can say that all the methods are robust to pick the item.

### D. EVALUATION

We evaluate the performance of the proposed method using the score (i.e., score using the FCSC 2018 rules), task completion time, symbolic planning time, and motion planning time. The task completion time of the rearranging task considers the time to obtain a symbolic solution and the time to plan and execute the trajectories with the robot.

There are two conditions for the experiments: one, where there are no external disturbances in the environment and two, where there are external disturbances. We consider that in the real world, while the robot is doing the rearranging task, a user can take an item and move it to a different part of the shelf. The second condition is to simulate those actions of the user. We consider the disturbance as a random movement of an item, after the robot executes an instruction. These are the conditions to simulate external disturbances to the items:

- The random movement is in the range of $-0.1$ m to $+0.1$ m in $x$-axis and $y$-axis.
- The item is moved while collision with the other items does not occur.
- The final pose of the item must be inside the area of the bin of the shelf.

In the variation A, we set the stop condition for the MCTS to be when it finds a branch of the tree that has the maximum score. In the variation B, the stop condition is a search time of 1 minute. After that, the state with the highest possibility of success is selected.

In total, we performed 200 trials with each method, 100 per disturbance condition. Each trial is a rearranging task with four items on random poses on top on the shelf.

We consider the difficulty of the rearranging task based on the initial state of the environment. We want to evaluate the performance of the proposed method with different difficulties for the same task. Furthermore, a low-difficulty state is where the robot needs few actions to reach the goal, whereas a high-difficulty state is where the robot needs to do multiple actions to reach the goal state. In Appendix A, we explain how we define the difficulty for the initial state.

### E. CREATION OF THE FEASIBILITY DATABASE

The feasibility database was created by validating possible pick and place poses on the top of the bin of the shelf. The size of the bin is 0.9 m in length by 0.4 m in depth. We sampled poses on the bin at every 0.01 m in $x$-axis and $y$-axis and we sampled in the $z$-axis every 0.01 m starting from the surface of the bin to a height of 0.1 m. We considered nine possible directions to pick an item and six possible directions to place an item, these pick and place orientations are shown in Fig. 8. In each direction for pick, we consider five rotations of the end-effector: $0°$, $\pm 30°$ and $\pm 45°$. In the case of the place, we consider three rotations: $0°$ and $\pm 45°$. The created feasibility database has a total of 620714 valid pick poses and 136679 valid place poses.

### F. OTHER PARAMETERS

Based on the rearranging rules, we set a threshold $th$ to determine if the item complies with the rearranging rules or the robot needs to do a re-grasp. The value of $th$ is 0.3. Any value greater than this means that the item does not complies with the rules and the robot needs to do a re-grasp of the item. We set the values of $w_1$ and $w_2$ in (1) to 0.5 to give the same importance to the orientation and position of the items.

To represent the occupancy of the state we use a grid of two rows by eight columns. Using this representation, the number of rows can be used to determine if an item is on the back, front or middle of the shelf, in case that is using both rows. The width of the columns is approximately the same as the width of the sandwiches. This helps to simplify the number of possible movements of an item in the symbolic planning, because we can only have one correct item per grid. In the MFC, we set the maximum number of actions of the robot to move an item to six. Based on some preliminary experiments,

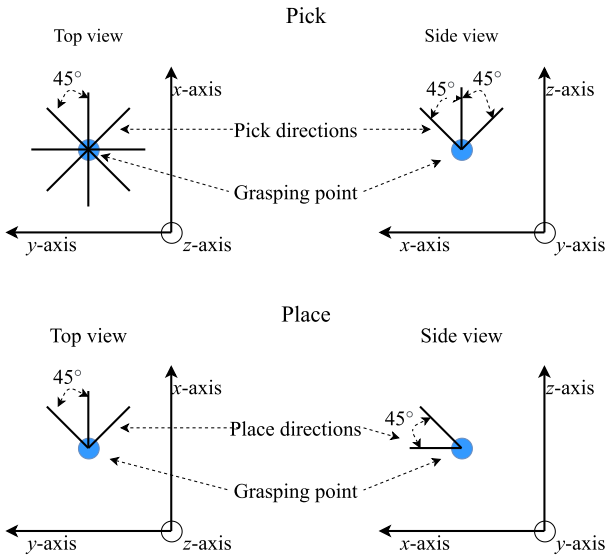



**FIGURE 8.** Pick and place directions of the end-effector used in the creation of the feasibility database.

we know that in the worst case the robot needs six actions to put an item in front of the shelf. In the simulation stage of the MCTS, we set the depth of simulation to five.

## V. RESULTS

We divided the results of the experiments in two subsections. Subsection V-A shows the results of the experiments without external disturbances, whereas V-B shows the results with the disturbances. In addition, we separated the experiments based on the difficulty of their initial states in three levels: low, medium and high.

### A. EXPERIMENTS WITHOUT DISTURBANCES

In the experiments without disturbances, we evaluate the time and performance of the methods. We consider the variation A of the Conventional and Conventional using MFC methods because, without external disturbances in the environment, the poses of the items do not change apart from when the robot does the pick and place motion. Thus, the solution of variations A and B should be the same.

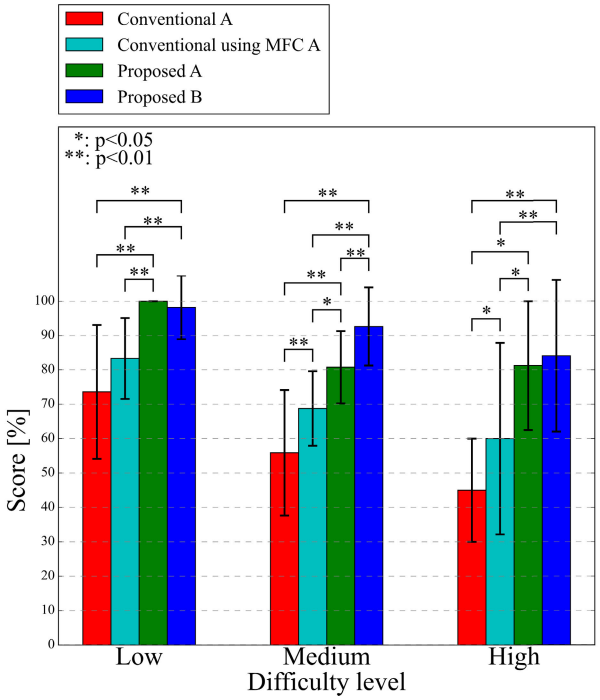



**FIGURE 9.** Scores of the experiments without disturbances divided in three groups by the difficulty level of the initial state.

Fig. 9 shows the score of the methods based on the difficulty of their initial states. We can observe the advantage of using any of the variations of the proposed method compared to the other methods. The proposed methods A and B outperform the others in all the difficulties. Fig. 9 also shows that, when the level of difficulty of the initial state is high, the greedy approach of the Conventional method does not perform well.

Table 1 shows the score and task completion time of the methods. As we can see in this table, the scores of the proposed methods A and B are higher than the other methods. Regarding the task completion time, the proposed methods A and B take more to complete the task. This is because the proposed method builds a tree with multiple states and searches between those states for a solution. The proposed

**TABLE 1.** Results of the experiments without disturbances.

| Score [%] | | | | | | |
|---|---|---|---|---|---|---|
| | Low difficulty | | Medium difficulty | | High difficulty | |
| Method | Mean | Standard deviation | Mean | Standard deviation | Mean | Standard deviation |
| Conventional [A] | 73.61 | 19.49 | 55.88 | 18.25 | 45.00 | 15.00 |
| Conventional using MFC [A] | 83.33 | 11.79 | 68.75 | 10.83 | 60.00 | 27.84 |
| Proposed [A] | **100.00** | 0.00 | 80.77 | 10.53 | 81.25 | 18.75 |
| Proposed [B] | 98.21 | 9.28 | **92.65** | 11.39 | **84.09** | 22.00 |

| Task completion time [s] | | | | | | |
|---|---|---|---|---|---|---|
| | Low difficulty | | Medium difficulty | | High difficulty | |
| Method | Mean | Standard deviation | Mean | Standard deviation | Mean | Standard deviation |
| Conventional [A] | 235.40 | 52.13 | 371.28 | 110.28 | 566.95 | 246.63 |
| Conventional using MFC [A] | **103.56** | 22.06 | **125.14** | 56.75 | **294.74** | 105.26 |
| Proposed [A] | 1245.58 | 441.81 | 1633.27 | 522.19 | 2427.64 | 1185.15 |
| Proposed [B] | 998.98 | 630.19 | 1018.60 | 821.08 | 1177.57 | 632.55 |

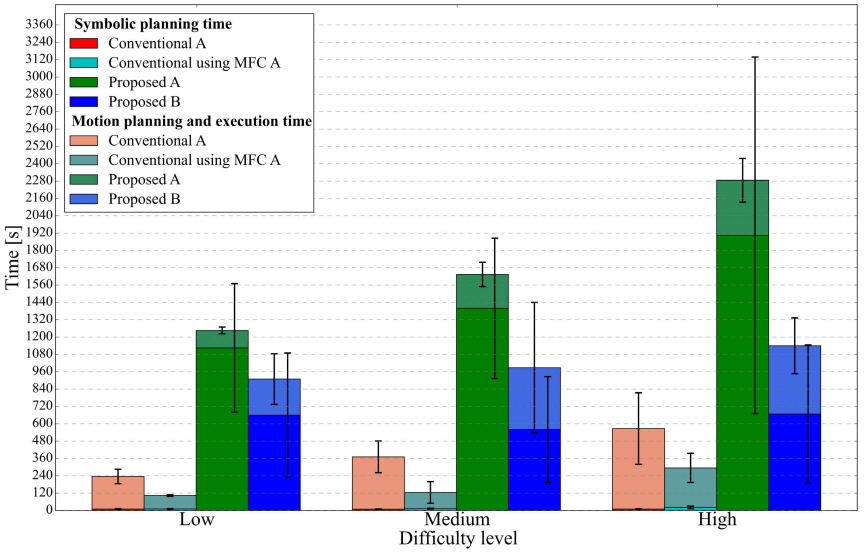



**FIGURE 10.** Task completion time of the experiments without disturbances divided in three groups by the difficulty level of the initial state.

method A uses more time to find a solution compared to the variation B, which is directly related to the stop condition. In the variation B, we stop the search after one minute and select a state with the highest possibility of success. In the variation A, there is a deeper exploration of the tree, i.e., until a state with the maximum score is reached.

The advantage of the proposed methods A and B are that they confirms that the generated instructions are valid. In this way, the proposed methods ensure that the solution can be executed with the robot. This is a trade-off between the time to find a solution and the obtained score.

Fig. 10 shows the time that each method used in searching for a solution and performing the actions with the robot. The proposed methods A and B use most of their time searching for the solution but, in the execution, their planning an execution time is similar or shorter than the Conventional method.

We can observe that between the Conventional and Conventional using MFC, the latter is multiple times faster than the former. This proves that it is better to validate the instructions before executing them with the robot. Moreover, using the MFC reduces the failures in the motion planner compared to the Conventional method. Using the MFC to confirm the trajectories before executing them is a better approach than just attempting to execute the motion with the motion planner. Overall, the proposed method proved to be efficient in finding a solution for the rearranging task.

### B. EXPERIMENTS WITH DISTURBANCES

We compare our proposed methods A and B to the Conventional using MFC A and B. The results obtained in the environment without disturbances showed that the Conventional method obtains a lower score compared to the others methods. In the experiments with disturbances, the Conventional method is not used, because its results will not be useful as a benchmark to compare with the others.

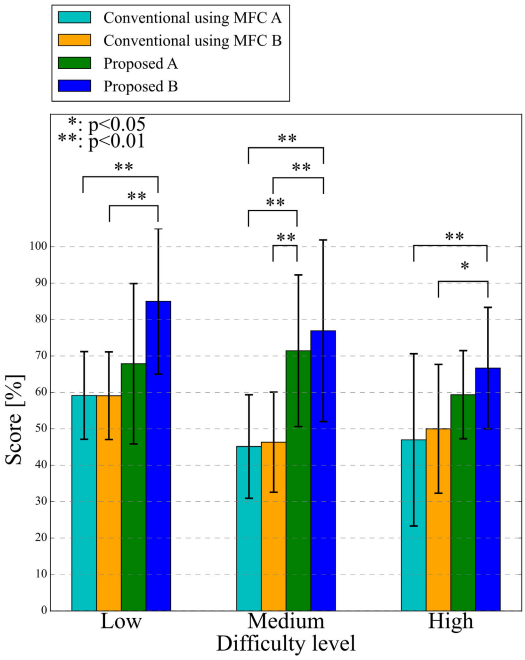



**FIGURE 11.** Scores of the experiments with disturbances divided in three groups by the difficulty level of the initial state.

Fig. 11 shows the scores obtained by the methods. The Conventional using MFC A and B have a lower score compared to the proposed methods. The Conventional using MFC only obtains a solution for moving one item at a time, whereas the proposed method explores the tree considering the whole sequence to move the items. This shows the robustness of our proposed method when there are disturbances in the environment.

Table 2 and Fig. 12 show the score and task completion time of the methods. Similar to the results previously obtained, the proposed methods take more time to complete the task, but obtain a higher score than the others.

**TABLE 2.** Results of the experiments with disturbances.

| Score [%] | | | | | | |
|---|---|---|---|---|---|---|
| | Low difficulty | | Medium difficulty | | High difficulty | |
| Method | Mean | Standard deviation | Mean | Standard deviation | Mean | Standard deviation |
| Conventional using MFC [A] | 59.17 | 12.05 | 45.14 | 14.40 | 46.97 | 23.64 |
| Conventional using MFC [B] | 58.33 | 11.79 | 46.32 | 13.74 | 50.00 | 17.68 |
| Proposed [A] | 67.86 | 22.02 | 71.43 | 20.82 | 56.25 | 10.83 |
| Proposed [B] | **85.00** | 20.00 | **76.92** | 24.93 | **62.67** | 16.67 |

| Task completion time [s] | | | | | | |
|---|---|---|---|---|---|---|
| | Low difficulty | | Medium difficulty | | High difficulty | |
| Method | Mean | Standard deviation | Mean | Standard deviation | Mean | Standard deviation |
| Conventional using MFC [A] | **104.69** | 12.77 | **123.68** | 78.04 | **321.31** | 125.95 |
| Conventional using MFC [B] | 141.41 | 80.40 | 282.51 | 302.66 | 438.45 | 364.19 |
| Proposed [A] | 12496.78 | 8763.34 | 12163.11 | 8453.08 | 17381.56 | 9812.62 |
| Proposed [B] | 2707.58 | 2568.91 | 5262.22 | 4181.67 | 8003.34 | 3717.08 |

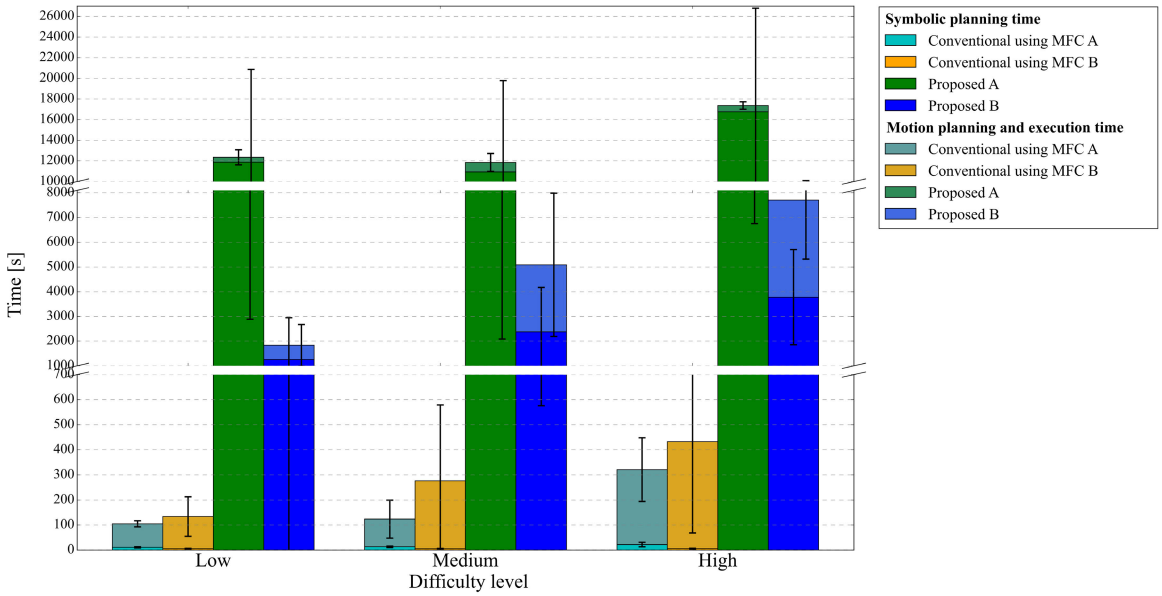



**FIGURE 12.** Task completion time of the experiments with disturbances divided in three groups by the difficulty level of the initial state.

We can conclude that, in all the cases, the variations of the proposed method obtain a higher score than the other methods, regardless of the difficulty of the initial state and the presence of disturbances in the environment.

Fig. 13 shows the robot re-grasping an item multiple times to move it to the front of the shelf. The MFC is used to obtain the poses to pick-and-place the item. Fig. 14 shows the robot performing the rearranging task, the instructions are obtained using the proposed method B. This YouTube playlist[8] contains the videos of the experiments.

## VI. DISCUSSION

In Section V, we showed that the proposed method variations A and B obtain a higher score than the other methods. We also verified that the proposed method can solve the task even when disturbances in the environment occur and obtain a score higher than the other methods.

[8]Playlist of the robot rearranging an environment, https://www.youtube.com/playlist?list=PLMnssJ3KtZsmVXuuibJqJdjU-NhpiGL1a

Recent works on rearranging tasks [15]–[17] assume that the motion planning will always be successful when moving the items. Theoretically speaking, these approaches are equivalent to having the motion and symbolic planning separated as in the Conventional method A and B in our experiments. On the other hand, the approach of Dantam *et al.* [14] is similar to what we do by validating the instructions beforehand but they do not consider the disturbances in the environment. In the case of a disturbance, their approach requires to search for a new solution, which is similar to the proposed method A that turned out to be time-consuming.

It is also important to mention that the recent works in rearranging tasks [14]–[17] only consider grasping the items from the top and they do not consider the re-grasping of the items, which is usually necessary in realistic scenarios. In contrast, our proposed method was evaluated in a realistic rearranging task scenario where, to reach the goal state, the items need to be re-grasped more than once.

We consider that, in such scenarios, our proposed method B is a better solution because it it leverages the advantages of the

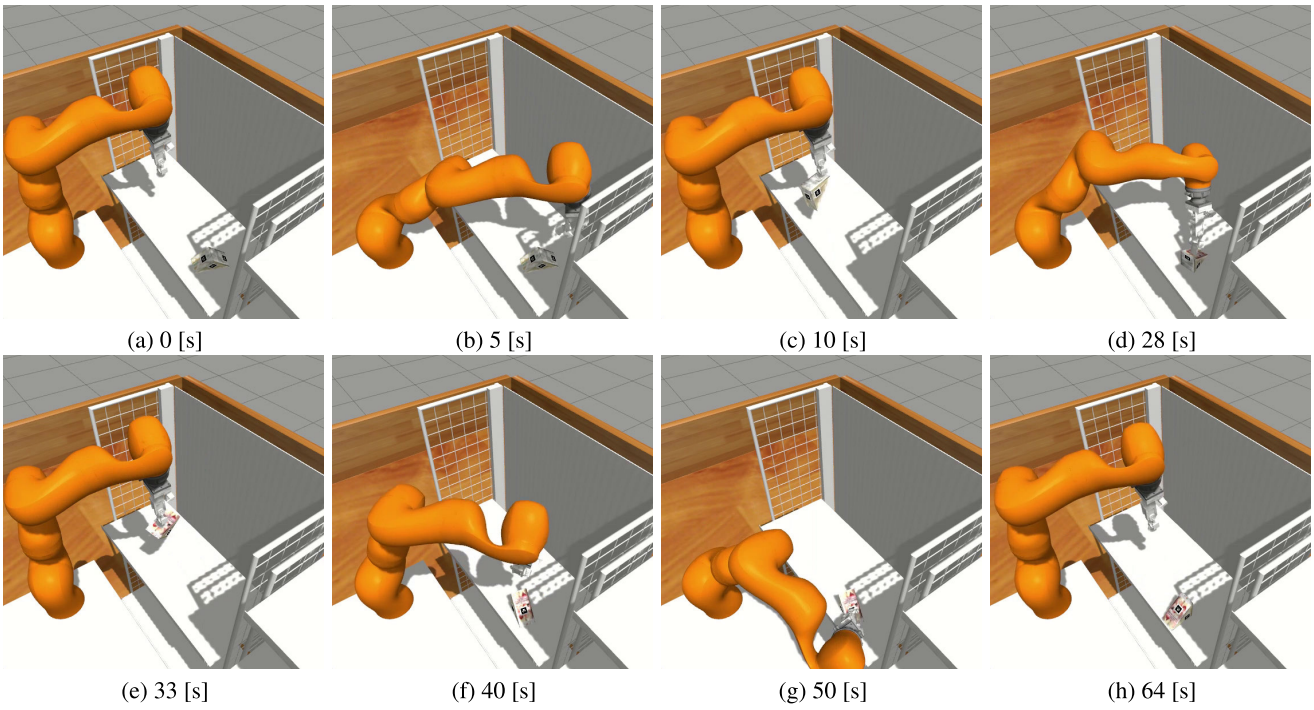



**FIGURE 13.** Robot rearranging an item by performing multiple re-grasps.

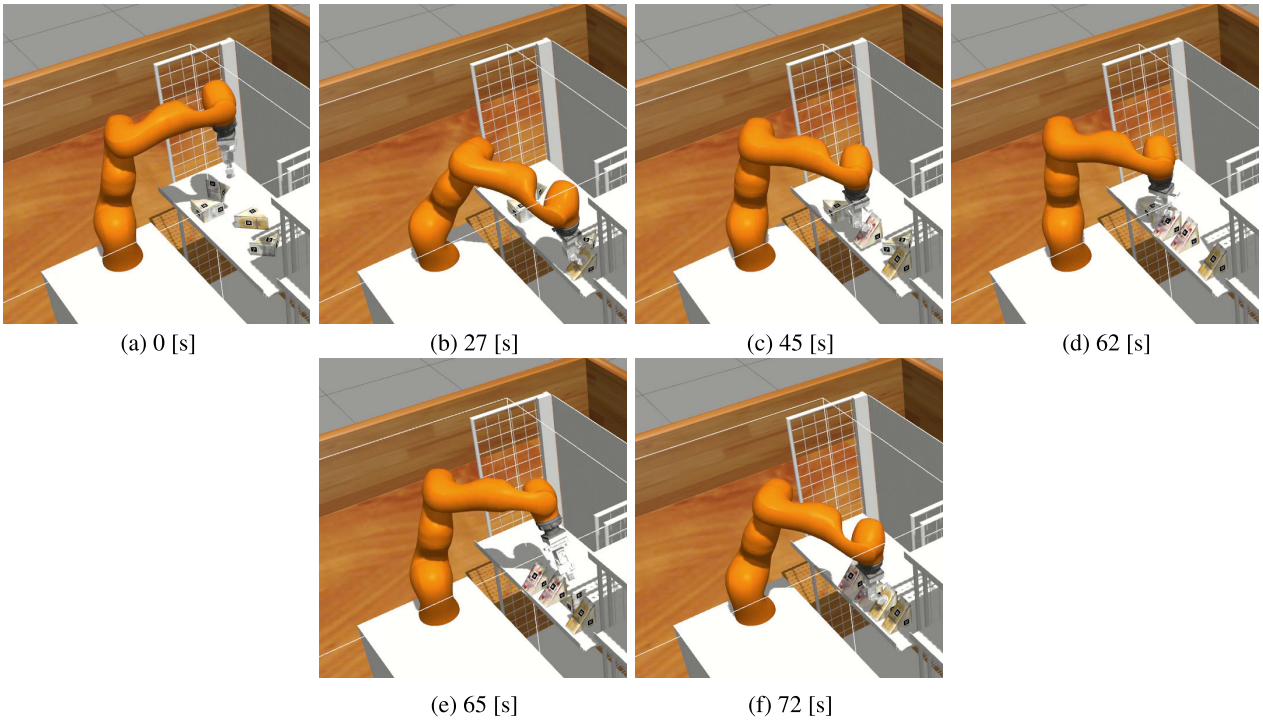



**FIGURE 14.** Robot rearranging an environment. The sequence of actions is obtained from the MCTS, and the re-grasping poses come from the MFC.

MCTS such as finding a solution at any moment during the tree search. Together with the MFC that early rejects invalid instructions. This allows the proposed method B to find a solution with a high rate of success, and the MFC ensures that the robot can perform the instructions. The MFC also obtains the number of actions required to perform the re-grasping of the items and the pick-and-place poses for the end-effector.

It is also important to mention that the proposed methods A and B did not obtain the maximum score in the experiments without disturbances because of the following reasons.

The MCTS in the proposed method finds a solution that can be executed, but during the execution of the instructions there are collisions between the items that were not considered in the solution. The proposed method only considers the collisions of the robot with the environment, but not of the items.

## VII. CONCLUSION

In this paper, we proposed a novel approach to solve a rearranging task by combining the symbolic and motion planning. Our proposed method finds a solution by combining an MCTS and the MFC. The MFC validates the solution using a pre-computed feasible motion database, to determine if there is a pick-and-place sequence that satisfies each instruction in the solution. The obtained results show that the proposed method outperforms the other methods and it is robust against disturbances in the environment. In particular, the variation B of the proposed method obtains a higher score than the other methods in all the scenarios.

We consider as future work to reduce the time that the MCTS uses in the tree search. This can be achieved by modifying the MFC to return the first valid combination of pick and place candidates that places the item complying with the rearranging rules, instead of doing all the possible pick and place combinations and then selecting the closest to the target. Another future improvement to our method is to use a machine learning approach to accelerate the tree search, training a model with different patterns of rearranging an environment. We also consider evaluating the proposed method rearranging different environments changing the type of items and rules.

## APPENDIX A. DIFFICULTY OF A STATE

We formulate the difficulty of a state based on the maximum distance that we could move the items in the shelf, the number of actions to change the side of the item that is facing down and the difference between the orientations of the initial and target pose of the item. Equation (2) shows how we calculate the difficulty for one item.

$$f(\mathbf{p}, n, \mathbf{q}) = \frac{w_1 f_1(\mathbf{p})}{p_{\max}} + \frac{w_2 f_2(n)}{n_{\max}} + \frac{w_3 f_3(\mathbf{q})}{2\pi}, \tag{2}$$

where $\mathbf{p}$, $n$, $\mathbf{q}$ are the initial position of the item, side of the item that is facing down, orientation of the item, respectively; $f_1$, $p_{\max}$, $f_2$, $n_{\max}$ and $f_3$ are the distance to move an item to a corner of the shelf, the maximum distance that an item can move inside the shelf, the number of actions required to change the side that is facing down to 0 and the difference between the two orientations, respectively and $w_1$, $w_2$ and $w_3$ are the weights to balance the importance of the metrics in the equation.

We define the difficulty of a state $D$ as the sum of the difficulties of rearranging each item as shown in (3).

$$D = \sum_{i}^{m} f(\mathbf{p}_i, n_i, \mathbf{q}_i), \tag{3}$$


**FIGURE 15.** Sides of the item.

where $m$ is the set of items to rearrange and $i$ is an item in the set. A difficulty of 0 is an easy initial state that does not require many actions of the robot, whereas a difficulty of 1 is a difficult initial state, where the robot needs to perform multiple maneuvers and re-grasps of the items. We divide the initial states into three groups based on their difficulty: low (0.0 to 0.33), medium (0.34 to 0.66) and high (0.67 to 1.0).

Equations (4), (5) and (6) show how to calculate the distance, number of actions of the robot and the difference between two quaternions, respectively.

$$f_1(\mathbf{p}) = \max(f_d(\mathbf{p}, \mathbf{p}_r), f_d(\mathbf{p}, \mathbf{p}_l)), \tag{4}$$

where $\mathbf{p}_r$ and $\mathbf{p}_l$ are the positions of the right and left corner of the front of the shelf, respectively, and $f_d$ is the euclidean distance between the two positions. We use the distance to the right and left corners of the shelf because we consider the worst case for moving the item. This would be moving the item from one side of the shelf to the corner in the opposite side. Fig. 15 shows the assigned numbers for each side of the item. Our target configuration of the item is when the side 0 is facing down in contact with the shelf.

Based on preliminary experiments, we found that when the side facing down is 1 or 2, the robot needs three actions to move the item in the worst case. If the side that is facing down is 3 or 4, the robot needs six actions to move the item in the worst case. Based on this information, we define (5) to determine the number of actions required by the robot to move the item so the side that is facing down is 0.

$$f_2(n) = \begin{cases} 0, & \text{if } n = 0, \\ 3, & \text{if } n = 1 \text{ or } n = 2, \\ 6, & \text{otherwise}, \end{cases} \tag{5}$$

where $n$ is the current side of the item.

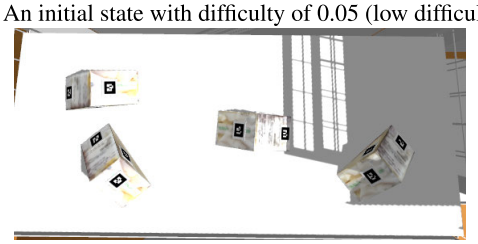
To determine the difference between the quaternions of the initial pose and target pose rotation, we use as metric the geodesic unit sphere [19]:

$$f_3(\mathbf{q}) = 2 \arccos(\mathbf{q} \cdot \mathbf{q}_{\text{target}}), \tag{6}$$

where $\mathbf{q}$ and $\mathbf{q}_{\text{target}}$ are the rotations of the initial and target poses, respectively, in this case $\mathbf{q}_{\text{target}}$ is the rotation of the item when it is placed in front of the shelf. $\mathbf{q}_{\text{target}}$ is the same as an identity quaternion.

For the value $p_{\max}$ we consider the maximum distance that we can move the item in the shelf, that is the diagonal of the bin of the shelf (0.98 m). In the case of the maximum number

(a) An initial state with difficulty of 0.05 (low difficulty)



(b) An initial state with difficulty of 0.87 (high difficulty)

**FIGURE 16.** **Initial states with different levels of difficulty.**

of actions $n_{\max}$, based on preliminary experiments we know that in the worst case is when the sides 3 or 4 are facing down. So the value of $n_{\max}$ is six.

Fig. 16 shows the difficulty of two initial cases. Fig. 16a is a low difficulty state, where the items are in the correct configuration in front of the shelf, but the items of the same type are not grouped together. Fig. 16b is a high difficulty state where the items are scattered in the shelf. This state the robot needs to perform multiple re-grasps to solve the rearranging task.

## REFERENCES


[1] M. Cakmak and L. Takayama, "Towards a comprehensive chore list for domestic robots," in *Proc. 8th ACM/IEEE Int. Conf. Hum.-Robot Interact. (HRI)*, Mar. 2013, pp. 93–94.

[2] T. Takahama, K. Nagatani, and Y. Tanaka, "Motion planning for dual-arm mobile manipulator-realization of tidying a room motion," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, vol. 5, Apr. 2004, pp. 4338–4343.

[3] R. Coulom, "Efficient selectivity and backup operators in Monte Carlo tree search," in *Proc. Int. Conf. Comput. Games*. Turin, Italy: Springer, 2006, pp. 72–83.

[4] G. M. J.-B. Chaslot, M. H. M. Winands, H. J. Van Den Herik, J. W. H. M. Uiterwijk, and B. Bouzy, "Progressive strategies for Monte Carlo tree search," *New Math. Natural Comput.*, vol. 4, no. 3, pp. 343–357, Nov. 2008.

[5] S. Cambon, F. Gravot, and R. Alami, "A robot task planner that merges symbolic and geometric reasoning," in *Proc. 16th Eur. Conf. Artif. Intell.*, 2004, pp. 895–899.

[6] G. Havur, G. Ozbilgin, E. Erdem, and V. Patoglu, "Hybrid reasoning for geometric rearrangement of multiple movable objects on cluttered surfaces," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, Jun. 2014, pp. 445–452.

[7] A. Krontiris and K. E. Bekris, "Dealing with difficult instances of object rearrangement," in *Robotics: Science and Systems*. Rome, Italy: Sapienza Univ. of Rome, 2015.

[8] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical planning in the now," in *Proc. 24th AAAI Conf. Artif. Intell. Workshops*. Ann Arbor, MI, USA: Univ. of Michigan, May 2010, pp. 1470–1477.

[9] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "Incremental task and motion planning: A constraint-based approach," in *Robotics: Science and Systems*, vol. 12. 2016, pp. 52–63.

[10] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *Int. J. Robot. Res.*, vol. 23, nos. 7–8, pp. 729–746, Aug. 2004.

[11] C. Dornhege, M. Gissler, M. Teschner, and B. Nebel, "Integrating symbolic and geometric planning for mobile manipulation," in *Proc. IEEE Int. Workshop Saf., Secur. Rescue Robot. (SSRR)*, Nov. 2009, pp. 1–6.

[12] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "FFRob: Leveraging symbolic planning for efficient task and motion planning," *Int. J. Robot. Res.*, vol. 37, no. 1, pp. 104–136, Jan. 2018.

[13] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[14] N. T. Dantam, S. Chaudhuri, and L. E. Kavraki, "The task motion kit," *IEEE Robot. Autom. Mag.*, vol. 25, no. 3, pp. 61–70, Sep. 2018.

[15] C. Paxton, Y. Barnoy, K. Katyal, R. Arora, and G. D. Hager, "Visual robot task planning," in *Proc. Int. Conf. Robot. Automat. (ICRA)*, May 2019, pp. 8832–8838.

[16] Y. Labbe, S. Zagoruyko, I. Kalevatykh, I. Laptev, J. Carpentier, M. Aubry, and J. Sivic, "Monte Carlo tree search for efficient visually guided rearrangement planning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3715–3722, Apr. 2020.

[17] H. Song, J. A. Haustein, W. Yuan, K. Hang, M. Y. Wang, D. Kragic, and J. A. Stork, "Multi-object rearrangement with Monte Carlo tree search: A case study on planar nonprehensile sorting," in *Proc. IEEE Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 9433–9440.

[18] J. Lee, Y. Cho, C. Nam, J. Park, and C. Kim, "Efficient obstacle rearrangement for object manipulation tasks in cluttered environments," in *Proc. Int. Conf. Robot. Automat. (ICRA)*, May 2019, pp. 183–189.

[19] D. J. Huggins, "Comparing distance metrics for rotation using the $k$-nearest neighbors algorithm for entropy estimation," *J. Comput. Chem.*, vol. 35, no. 5, pp. 377–385, Feb. 2014.

[20] C. Hennersperger, B. Fuerst, S. Virga, O. Zettinig, B. Frisch, T. Neff, and N. Navab, "Towards MRI-based autonomous robotic US acquisitions: A first feasibility study," *IEEE Trans. Med. Imag.*, vol. 36, no. 2, pp. 538–548, Feb. 2017.

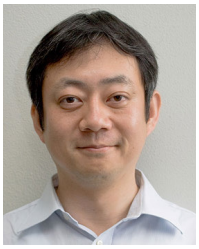
[21] K. Wada, "New robot technology challenge for convenience store," in *Proc. IEEE/SICE Int. Symp. Syst. Integr. (SII)*, Dec. 2017, pp. 1086–1091.

[22] G. A. G. Ricardez, S. Okada, N. Koganti, A. Yasuda, P. M. U. Eljuri, T. Sano, P.-C. Yang, L. El Hafi, M. Yamamoto, J. Takamatsu, and T. Ogasawara, "Restock and straightening system for retail automation using compliant and mobile manipulation," *Adv. Robot.*, vol. 34, nos. 3–4, pp. 235–249, Feb. 2020.


**PEDRO MIGUEL URIGUEN ELJURI** (Graduate Student Member, IEEE) received the M.E. degree from the Nara Institute of Science and Technology, Japan, in 2018, where he is currently pursuing the Ph.D. degree with the Robotics Laboratory, Division of Information Science. His research interests include robot control, service robots, task planning, and robotics competitions.

**GUSTAVO ALFONSO GARCIA RICARDEZ** (Member, IEEE) received the M.E. and Ph.D. degrees from the Nara Institute of Science and Technology, Japan, in 2013 and 2016, respectively. He is currently an Assistant Professor with the Robotics Laboratory, Division of Information Science, Nara Institute of Science and Technology, and a Research Advisor for robotics competitions at the Robotics Hub of Panasonic Corporation. His research interests include human-safe, efficient robot control, human–robot interaction, manipulation, and task planning, and he leads numerous research projects and teams in international robotics competitions.

**NISHANTH KOGANTI** received the B.Tech. degree in electrical engineering from the Indian Institutes of Technology Jodhpur, India, in 2012, and the M.E. and Ph.D. degrees from the Nara Institute of Science and Technology, Japan in 2014, and 2017, respectively. He has previously worked as a Postdoctoral Researcher at the University of Tokyo, Japan, and as an Assistant Professor with the Nara Institute of Science and Technology, Japan. He is currently working as a Senior Data Scientist with GEP Worldwide Inc., India. His research interests include reinforcement learning, robotics, and natural language processing.



**JUN TAKAMATSU** (Member, IEEE) received the Ph.D. degree in computer science from the University of Tokyo, Japan, in 2004. From 2004 to 2008, he was with the Institute of Industrial Science, University of Tokyo. In 2007, he was with Microsoft Research Asia as a Visiting Researcher. In 2008, he joined the Nara Institute of Science and Technology, Japan, as an Associate Professor. His research interests are in robotics, including learning-from-observation, task and motion planning, feasible motion analysis, 3D shape modeling and analysis, and physics-based vision.

**TSUKASA OGASAWARA** (Member, IEEE) received the Ph.D. degree from the University of Tokyo, Japan, in 1983. From 1983 to 1998, he was with the Electrotechnical Laboratory, Ministry of International Trade and Industry, Japan. From 1993 to 1994, he was with the Institute for Real-Time Computer Systems and Robotics, University of Karlsruhe, Germany, as a Humboldt Research Fellow. He joined the Nara Institute of Science and Technology, Nara, Japan, in 1998, where he is currently a Professor with the Division of Information Science. He is also a Vice President of the Nara Institute of Science and Technology and the Dean of the Graduate School of Science and Technology. His research interests include human–robot interaction, dexterous manipulation, human modeling, and bio-inspired robotics.