Doctoral Dissertation

Rearranging Tasks by a Robot Using Motion Feasibility and a Monte Carlo Tree Search

Pedro Miguel URIGUEN ELJURI
Program of Information Science and Engineering
Graduate School of Science and Technology
Nara Institute of Science and Technology

Supervisor: Professor Tsukasa Ogasawara
(Division of Information Science)

Submitted on March 17, 2021

A Doctoral Dissertation
submitted to the Graduate School of Science and Technology,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Pedro Miguel URIGUEN ELJURI

Thesis Committee:

Professor Tsukasa Ogasawara	(Supervisor)
Professor Kenji Sugimoto	(Co-supervisor)
Associate Professor Jun Takamatsu	(Co-supervisor)
Assistant Professor Gustavo Alfonso GARCIA RICARDEZ	(Co-supervisor)
Assistant Professor Sung-Gwi Cho	(Co-supervisor)

Rearranging Tasks by a Robot Using Motion Feasibility and a Monte Carlo Tree Search*

Pedro Miguel URIGUEN ELJURI

Abstract

In this dissertation, I focus on how to solve the problem of rearranging tasks with a robot. Developing a method to obtain the sequence of robot actions to manipulate items in an environment to realize a rearranging task while explicitly considering the motion capabilities of the robot. The resulting solution has a high success rate because it only uses actions that can be performed by the robot.

First, I propose to combine symbolic planning with motion planning to generate a sequence of instructions and confirm them before the execution with the robot. I propose to use a Motion Feasibility Checker (MFC) to verify if an instruction can be executed with the robot. This is achieved by estimating the final pose of the item if the robot executed the instruction and the item was picked and placed with a set of pick and place poses. The MFC uses a set of feasible poses stored in a feasibility database, which is created in advance to know the poses the robot end-effector can reach in the environment. A feasible pose is a pose of the end-effector where the robot can perform a pick or place maneuver.

Second, I propose to use the MFC with a Monte Carlo Tree Search (MCTS) to search for a solution to the rearranging task in a tree with possible states of the environment. Each state in the tree is the result of an action of the robot. The combination of symbolic planning with motion planning is achieved when the MFC verifies the instructions of the MCTS. The MCTS prunes the tree using the output of the MFC to keep only valid states. As the tree only has valid states, the path of states selected by the MCTS should also be valid and the robot should be able to execute them.

*Doctoral Dissertation, Graduate School of Science and Technology, Nara Institute of Science and Technology, March 17, 2021.

Finally, I evaluate our proposed method by doing a shelf rearranging task in a convenience store setup. We compare our proposed method to a conventional approach (i.e., symbolic and motion planners are independent) and to a conventional approach that uses the MFC. We evaluate the results based on the task completion time and a score assigned to the rearrangement task based on some rearranging rules. The proposed method uses more time to find the solution, but the obtained score is higher than the scores of the other methods.

Keywords:

Rearranging task, task planning, symbolic planning, motion planning, manipulation, service robot

Contents

1 Introduction **2**

1.1 Background 2

1.2 Contributions 3

1.3 Dissertation Layout 5

2 Related Works **6**

3 Proposed Method **7**

3.1 Overview 7

3.2 State Representation 9

3.3 Symbolic Planning 9

3.4 Motion Feasibility Checker 11

3.5 Feasibility Database 13

4 Experimental Setup **17**

4.1 Simulation Environment 17

4.2 Rearranging Rules 17

4.3 Methods for Comparison 19

4.4 Evaluation 20

4.5 Feasibility Database 22

4.6 Other Parameters 22

5 Results **25**

5.1 Score and time results 25

5.2 Experiments with disturbances 26

5.3 Experiments using different types of items 36

6 Discussion **39**

7 Conclusions **41**

Acknowledgements **45**

References 46

Appendix 50

A Difficulty of a state 50

List of Figures

1 Overview of the proposed method. 4
2 Components of the proposed method. 8
3 Stages of the Monte Carlo Tree Search using the MFC. 10
4 Validation of a state using the MFC. 15
5 Illustration of the pick or place maneuvers. 16
6 Simulation environment and end-effector used in the experiments. . . 18
7 Illustration of a correct final state. 19
8 Pick and place directions of the end-effector used in the creation of the feasibility database. 23
9 Grid representation of the environment 24
10 Scores divided in three groups by the difficulty level of the initial state, without disturbances in the environment. 28
11 Task completion time divided in three groups by the difficulty level of the initial state, without disturbances in the environment. 29
12 Scores divided in three groups by the difficulty level of the initial state, with disturbances in the environment. 31
13 Task completion time divided in three groups by the difficulty level of the initial state, with disturbances in the environment 32
14 Screenshots of the robot re-grasping an item. 34
15 Screenshots of the robot rearranging an environment. 35
16 Screenshots of the robot rearranging items of different size. 38
17 Sides of the item. 51
18 Initial states with different levels of difficulty. 52

List of Tables

1 Results of the experiments without disturbances 30
2 Results of the experiments with disturbances. 33
3 Results of the experiments using different items. 37

Chapter 1

Introduction

1.1 Background

In recent years, the use of robots in our daily-life has been becoming more common [1, 2]; nowadays some robots are specifically designed to help in a house environment doing tasks such as floor sweeping or vacuum cleaning, *e.g.,* Roomba[1] and RURO[2].

Nevertheless, the use of robots in this field is still under development [3, 4], where many repetitive and time-consuming tasks are yet being done by the human and could be relegated to a robot. One of these tedious and time-consuming tasks that humans do daily is a rearranging task.

A rearranging task can be something small as moving items on a shelf or organizing items on a desk to something big such as tidying up a room. In a daily-life routine all these rearranging tasks are the second most common tasks [5] so their automation using robots is expected.

A rearranging task is challenging because it has many technical difficulties such as the manipulation of the item specifically determining how to pick and place the items with the robot and where to place them in the environment. Deciding how to pick and place the item 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.

In this work, I focus on the task planning of rearranging tasks. Specifically, I 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, I 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 [6]. This approach is time-consuming because, when the robot fails to

[1] Roomba robot from iRobot, https://www.irobot.com/roomba
[2] RURO robot from Panasonic, https://panasonic.jp/tourist/en/soji/

execute an instruction (*e.g.,* due to a failure in creating the trajectory during motion planning), the robot needs to obtain a new set of instructions. This pattern of getting new instructions without knowing if the robot will be able to execute them increases the risk of falling into a loop of invalid instructions, thus spending time trying to find a solution that ultimately may be executable with the robot.

1.2 Contributions

The contributions of this research are twofold. First, a Motion Feasibility Checker (MFC) to validate the instruction to be executed with the robot using pre-computed poses obtained from a database. Second, a method to solve rearranging tasks by combining symbolic and motion planning using a Monte Carlo Tree Search (MCTS) [7, 8] and the MFC.

The following is a brief description of each contribution:

1. To avoid spending time in poses where the motion planning can fail by using a valid set of pre-computed poses obtained using a motion planner. The MFC uses the pre-computed poses to estimate the intermediate states of the item during its manipulation with the robot and determines is its possible to execute or not the instruction.

2. A method to solve a rearranging task by using a MCTS as the symbolic planner and validating the states of the tree using the MFC. The MCTS will only do the tree search between the valid states determined by the MFC. The advantage of the MCTS is that it can find a solution at any moment.

Fig. 1 shows an overview of the proposed method to solve a rearranging task. First, the initial state is given to the proposed method. Second, the proposed method uses an 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.

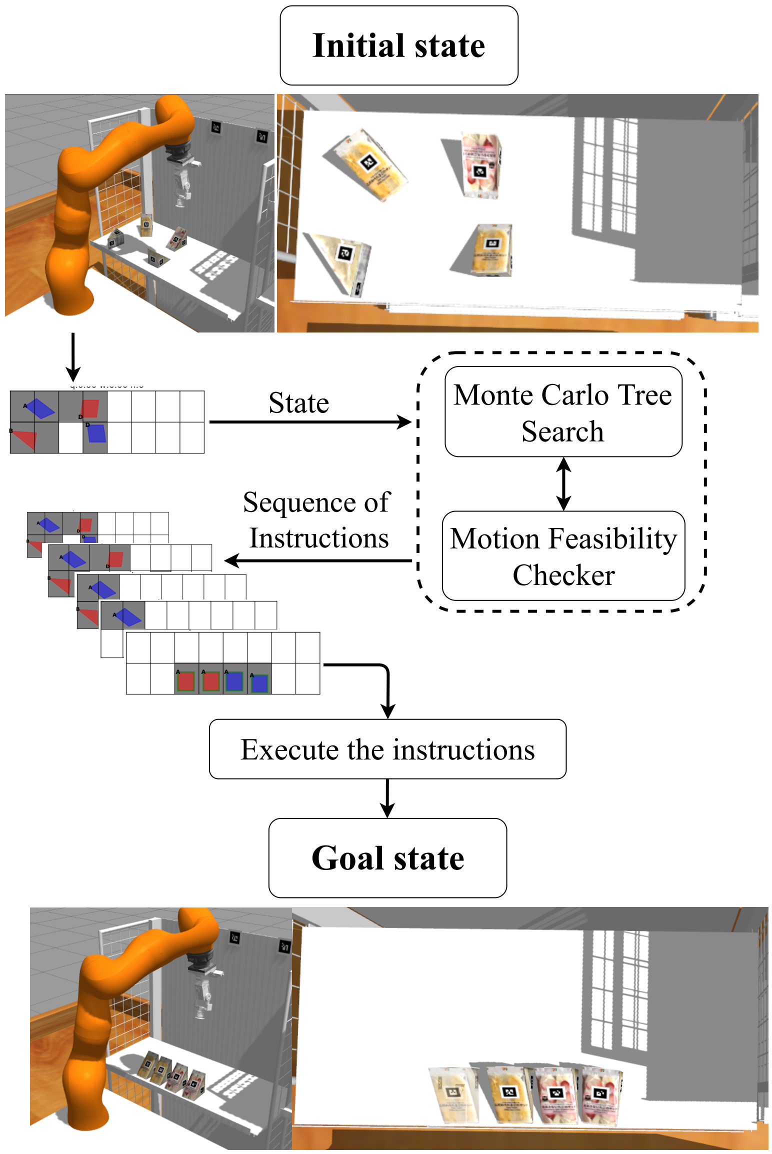


Figure 1: Overview of the proposed method.

1.3 Dissertation Layout

The rest of this dissertation is organized as follows. Chapter 2 explains related works on solving rearranging tasks. Chapter 3 describes our proposed method. Chapter 4 introduces the rearranging task used to test our method and compare it to other methods. Chapter 5 presents the results obtained in the experiments. Chapter 6 includes a discussion. Finally, Chapter 7 concludes this dissertation and provides some directions for future work.

Chapter 2

Related Works

Rearranging tasks are a complex problem that has been tried to be solved before *e.g.,* [9–14]. 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 [13,15,16] where they focus on rearranging the items using some guidelines of the sequence of items that need to be moved (*e.g.,* in a desk moving the books first, then the other items). There is also randomized approaches [17–19], where there is no specific sequence in how to move the items to reach the goal state. 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 [10,20].

To combine the symbolic and motion planner to perform an organizing task has been proposed [21]. However, that approach focuses on using a Probabilistic Roadmap Method [22] to generate new states, validate them and select the state that has the lowest cost among them. Dantam *et al.* [23,24] proposed to obtain multiple solutions, to be saved in a set, regardless of the cost. In such a 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 [13,19,25–27]. 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 [26,28]. One of their limitations is that they assume that the motion planning can always execute the pick and place motion and that the pick and place approach to the items is always from above.

In our proposed method, we try a different approach to solve a rearranging task by pre-validating the motion of the robots before executing them. We do these validations using a database of feasible poses of the robot, which avoids failures in the motion planning.

Chapter 3

Proposed Method

3.1 Overview

This dissertation proposes to combine the symbolic and motion planning when generating the solution for a rearranging task.

Assuming that a rearranging task can be achieved by repeating the following actions to manipulate an item: 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, 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 automatically chooses 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, the movements can be classified 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, the input and output of each component.

The Motion Feasibility Checker (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 Monte Carlo Tree Search (MCTS) explores various object arrangements, *i.e.* states, and selects the path with the highest ratio between cumulative reward of the nodes and number of visits to each node. One of the advantages of the MCTS is that it can obtain a solution at any moment.

Since it is useless to explore states that the robot cannot achieve, the MCTS collaborates with the MFC for rejecting invalid states that can not be executed with the robot

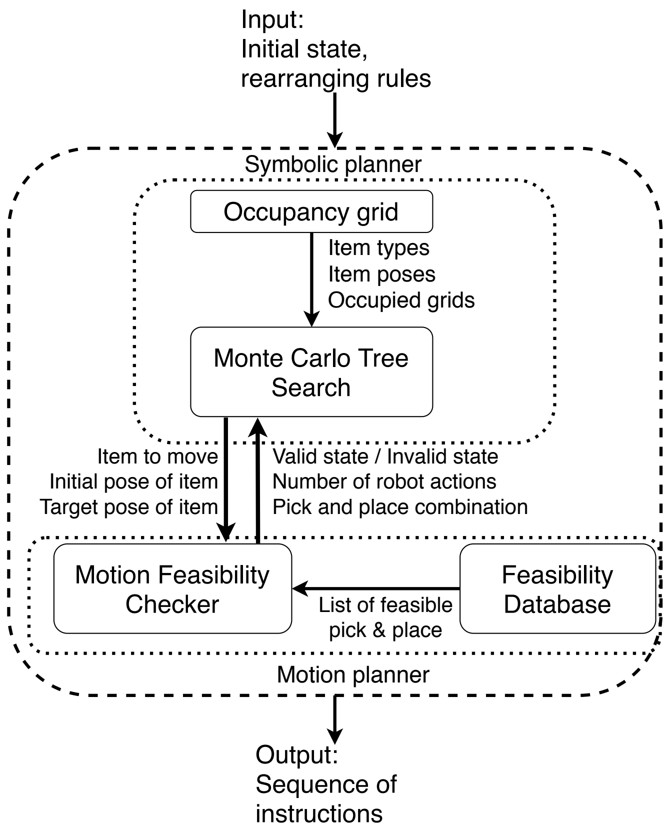


Figure 2: Components of the proposed method.

based on its feasibility. 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.

Calculating the collision-free movement is not a task 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 poses are obtained offline, which allows the motion planner, when executed online, to find a trajectory between the neutral configuration and the pick or place pose faster. This is because all the poses were already tested offline so the probability of failure in finding a trajectory is relatively low.

3.2 State Representation

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

From that state, we can obtain 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 be sharing the same grid. The initial state of the environment is considered as the root of the tree for the symbolic planning.

3.3 Symbolic Planning

At the symbolic level, solving a rearranging task is creating the instructions the robot needs to execute. We chose an MCTS [7, 8] as a symbolic planner because it searches for a solution in a tree with multiple possible states, and can obtain a solution at any time and its randomness allows a balance between exploration and exploitation of the tree [19, 28, 29]. One of the advantages is that an MCTS does not require exploring all the nodes of the tree to find a solution. This is very useful to decide for a solution that has a high probability of success.

The MCTS efficiently explores and exploits the nodes 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. Each state in the tree is linked by an action of the robot. The MCTS always chooses the solution which has the highest ratio between rewards and visits. The MCTS generates new states, 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 [7, 8, 30]. 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

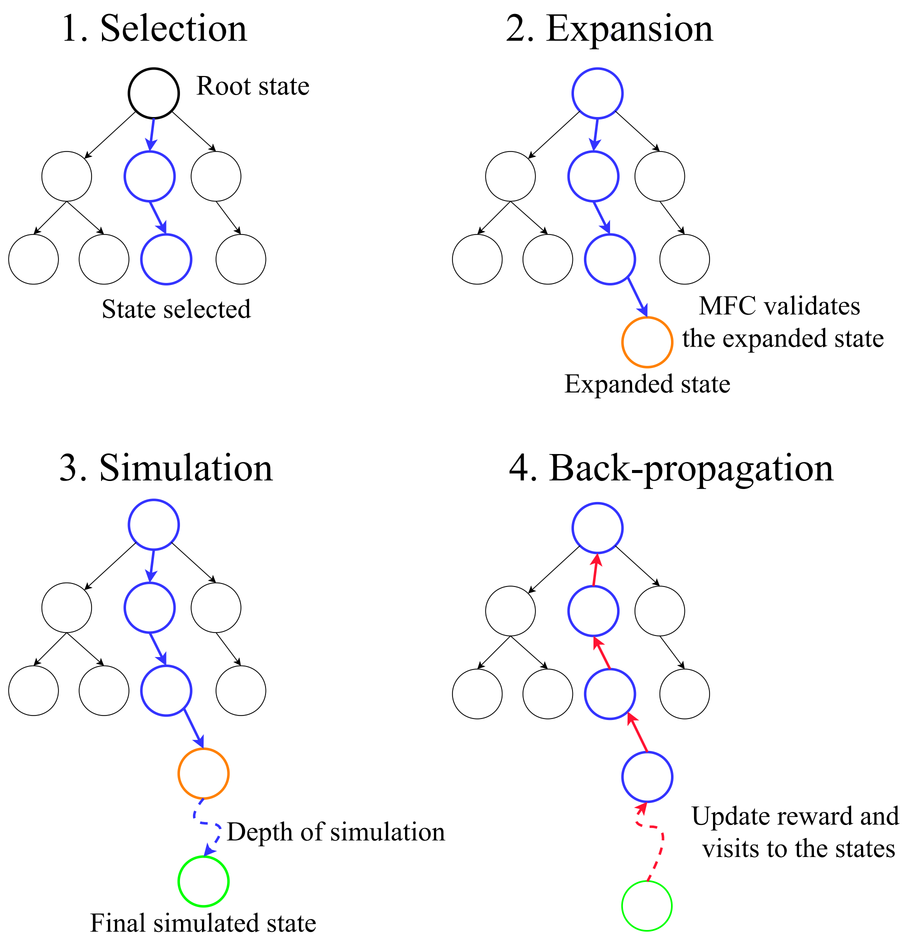


Figure 3: Stages of the Monte Carlo Tree Search using the MFC.

be explored. From the generated states, 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 the 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. The rearranging rules are a set of instructions that we need to follow while executing the rearranging task. These rules 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 states 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.

3.4 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 MCF 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_{initial} and target pose T_{target} of the item. Based on T_{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 would the robot places the item in the environment. Then, the MFC compares and calculates the difference between $T_{\text{predicted}}$ and T_{target} using (1).

$$
\begin{aligned}
f(T_{\text{target}}, T_{\text{predicted}}) = {} & w_1 f_d(T_{\text{target}}, T_{\text{predicted}}) \\
& + w_2 f_g(T_{\text{target}}, T_{\text{predicted}}),
\end{aligned}
\tag{1}
$$

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 [31] to calculate the difference between the two quaternions 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_{target}. We use a threshold th_1 to compare with the result of (1) between T_{target} and $T_{\text{predicted}}$. If the value is greater than th_1, 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_{target}. This $T_{\text{predicted}}$ is considered as the new initial pose T_{initial} for the new search.

The total number of pick-place actions required to reach T_{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 of the item T_{initial}, we select a grasping point. The grasping points are defined before-hand 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 within a range to the grasping point. The distance range 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 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.

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.

3.5 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 pose. 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_{hovering}, roll, pitch, yaw, where z_{hovering} is an offset in the height 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 pose. Finally, we return to the neutral configuration. If the motion planner succeeds in creating a trajectory for the pick or place pose, we add that pose to the set of valid poses. A reachable pose is not always a feasible pose, the reason of this is because we need to reach the pose from an specific orientation.

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

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

where $\mathbb{P}$ and $\mathbb{Q}$ are the pick and place poses sets, respectively. Each element $\mathbf{P}$ and $\mathbf{Q}$ is an x, y, z, roll, pitch, yaw pose of the end-effector, n and m are the numbers 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 for placing the item. We have to comply with the rearranging rules, so we have to add more restrictions in how the robot place the item in the environment.

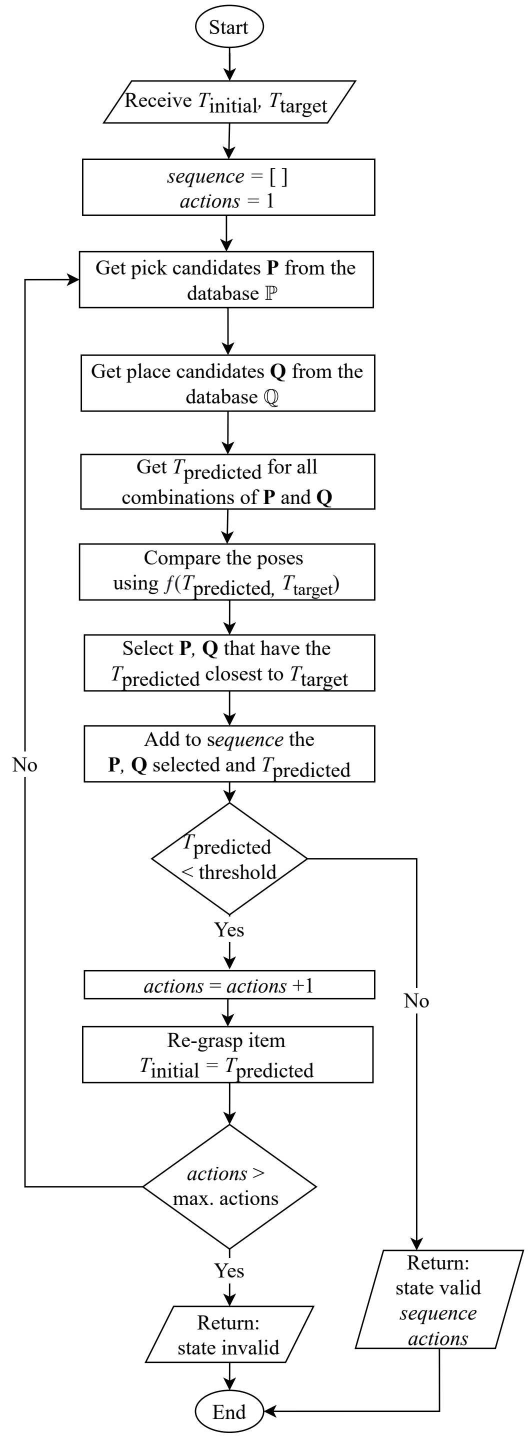


Figure 4: Validation of a state using the MFC.

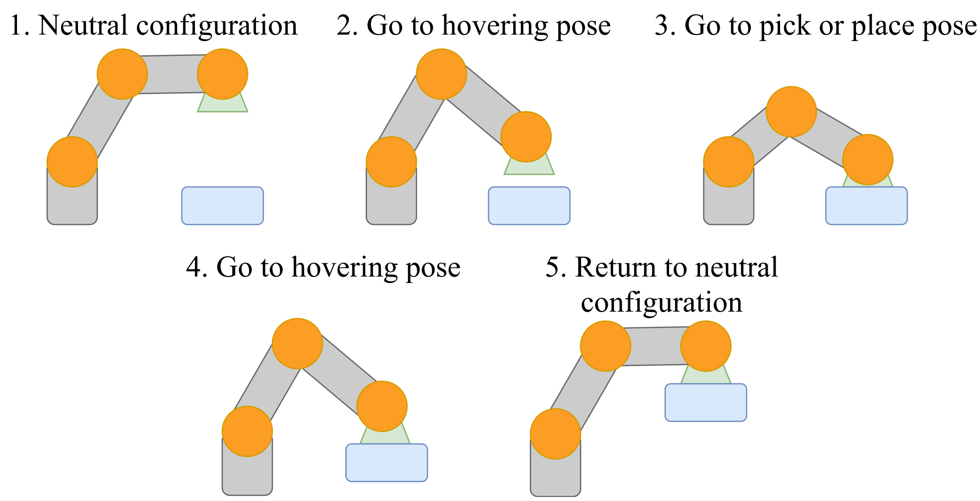


Figure 5: Illustration of the pick or place maneuvers.

Chapter 4

Experimental Setup

4.1 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* [32] in ROS[5].

The robot arm has 7 DOF and is mounted on a fixed base in front of a shelf, as shown in Fig. 6a. 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, considering the right front corner as the origin of the bin. To manipulate the objects, we used a custom-made end-effector with an extra DOF and a suction cup, as shown in Fig. 6b.

4.2 Rearranging Rules

The rearranging task is based on the restock task of the Future Convenience Store Challenge (FCSC) [33, 34], 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 items complying with a set of rules. Also, during WRC 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.

The sandwich rearranging task consists of rearranging four sandwiches in a shelf.

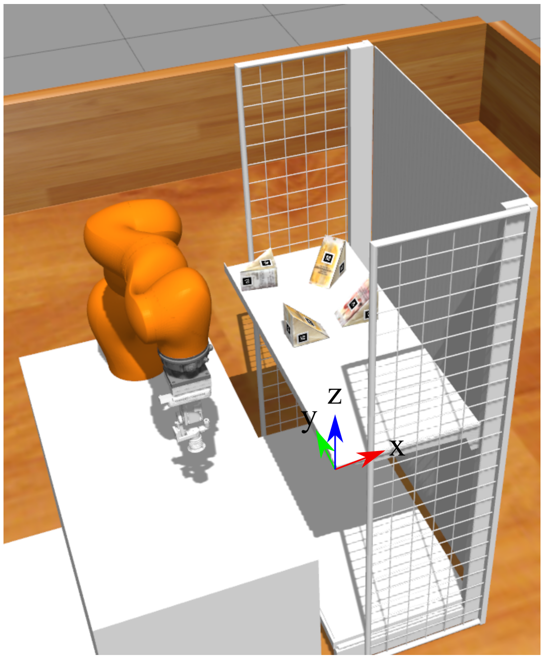
---

[3] Gazebo, http://gazebosim.org/

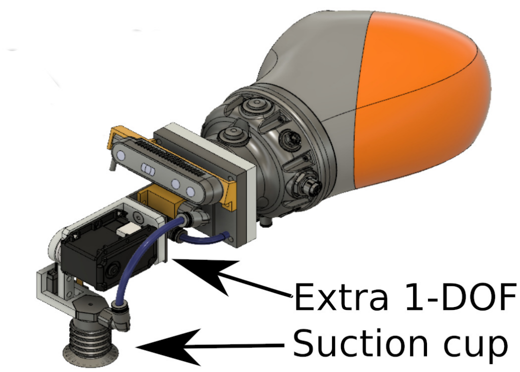
[4] KUKA LBR iiwa 14 R820, `https://www.kuka.com/en-de/products/robot-systems/industrial-robots/lbr-iiwa`

[5] Robot Operating System, http://www.ros.org/

[6] World Robot Challenge, `https://worldrobotsummit.org/en/index2018.html`



(a) Simulation environment.



(b) End-effector.

Figure 6: Simulation environment and end-effector used in the experiments.

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.

[7]FCSC rule book, `https://worldrobotsummit.org/en/wrc2018/service/pdf/Rulebook_task1.pdf`

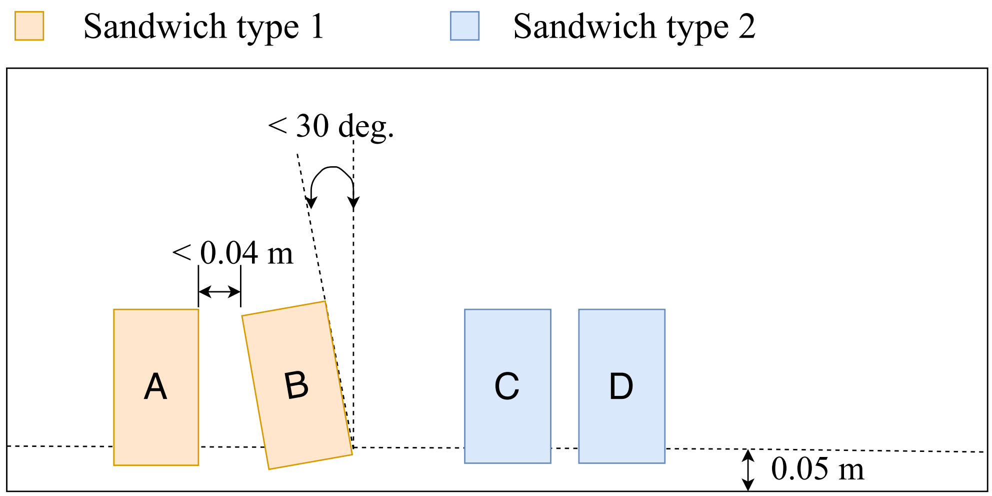


Figure 7: Illustration of a correct final state, all the items are in the correct position and orientation on the shelf. Item B is not aligned as the others, but its orientation is still valid by the rules. Items of the same type are together. The total score of this rearrangement is 12 points, which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

4.3 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 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 state 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. 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, before grasping the item, the robot receives the information of the pose of the item. This is done for all the methods to compensate for disturbances in the environment during 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 for picking the item.

4.4 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, time to plan and execute the trajectories with the robot.

We consider two conditions for the experiments, one where there are not 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 considerations 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 happen.
- The final pose of the item must be inside the area of the bin of the shelf.

We consider the difficulty of the rearranging task based on the initial state of the environment. We want to evaluate how the proposed method behaves 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.

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, time to plan and execute the trajectories with the robot.

There are two conditions for the experiments, one where the environment does not have any external disturbances 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 its to simulate those actions of the user. We consider the disturbance as a random movement of an item, before or 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 happen.
- The final pose of the item must be inside the area of bin of the shelf.

We set the stop conditions for the MCTS in the variation A to be only when it found a branch of the tree that has the maximum score. In the variation B, the stop condition is when the search time is 1 minute 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.

4.5 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. To plan the trajectories from the neutral configuration to the pick and place poses, we used an RRT planner [35].

We considered nine possible directions to pick an item and six possible directions to place an item, these pick and place orientations can be observed in Fig. 8. In each direction for pick, we consider five rotations of the end-effector: 0°, ±30° and ±45°. In the case of the place rotations we consider three rotations: 0° and ±45°.

The created feasibility database has a total of 620714 valid pick poses and 136679 valid place poses.

4.6 Other Parameters

Based on the rearranging rules, we set a threshold to determine if the item complies with the rearranging rules or the robot needs to do a re-grasp. The value of the threshold th_1 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 case that we have items of different sizes,

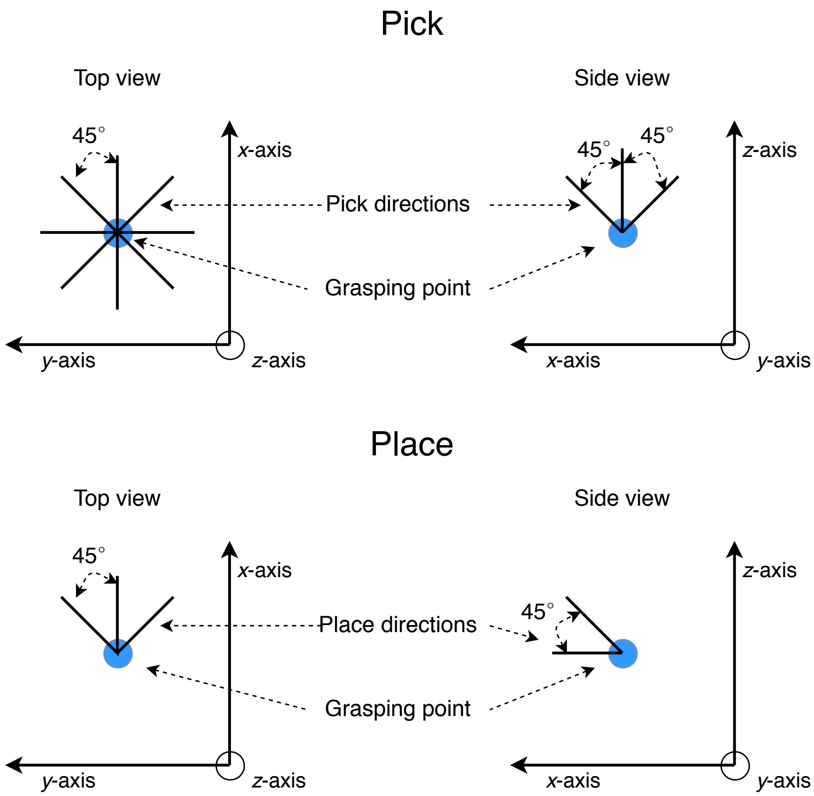
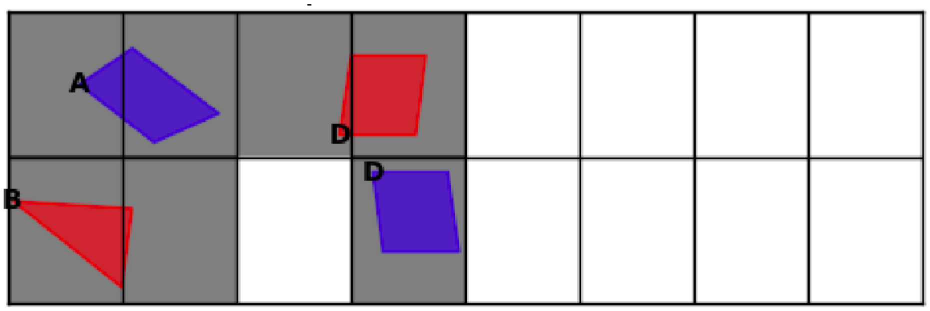


Figure 8: Pick and place directions of the end-effector used in the creation of the feasibility database.

the grid size is determined by the smallest item that we have to manipulate. Fig.9 shows an example of an initial state and its grid representation.

In the MFC, we set the maximum number of actions of the robot to move an item to six, based on some preliminary experiments, we know that in the worst case the robot needs 6 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.

(a) Environment to rearrange.



(b) Grid representation.

Figure 9: Grid representation of the environment. The occupied grids are marked in gray, items of the same type have the same color. The letter next to each item is the face of the item that is facing up.

Chapter 5

Results

We divided the results of the experiments in three subsections. Subsection 5.1 shows the results of the experiments without external disturbances, whereas 5.2 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. Subsection 5.3 shows the results of experiments rearranging different types of items.

5.1 Score and time results

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.

Fig. 10 shows the score of the methods based on the difficulty of their initial states and the statistical significance using ANOVA. We can observe the advantage of using any of the variations of the Proposed method compared to the other methods. The Proposed method outperforms the others in all the difficulties. Fig. 10 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 method are higher than the other methods. Regarding the task completion time, the Proposed method 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 method A uses more time to find a solution compared to the variation B, which is directly related to its stop condition. In the variation B, we stop the search after one minute and select a state with a high 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 method is that it confirms that the instructions generated are valid. In this way, the proposed method ensures that the solution can be executed with the robot. This is a trade-off between the time to find a solution and the score obtained.

Fig. 11 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 than the planning and execution time of the Conventional method.

We can observe that between the Conventional and Conventional using MFC methods, 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.

5.2 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.

Fig. 12 shows the scores obtained by the methods. The Conventional using MFC A and B have a lower score compared to the Proposed method. 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. 13 show the score and task completion time of the methods. Similar to the results previously obtained, the Proposed method takes more time to complete the task, but obtains a higher score than the others.

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. 14 shows how the robot re-grasps an item multiple times to move it to the front of the shelf. Whereas Fig. 15 shows he robot performing the rearranging task, the instructions are obtained using the Proposed method B, and the MFC is used to verify the instruction and obtain the poses to move the item. This YouTube playlist[8] contains the videos of the experiments.

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

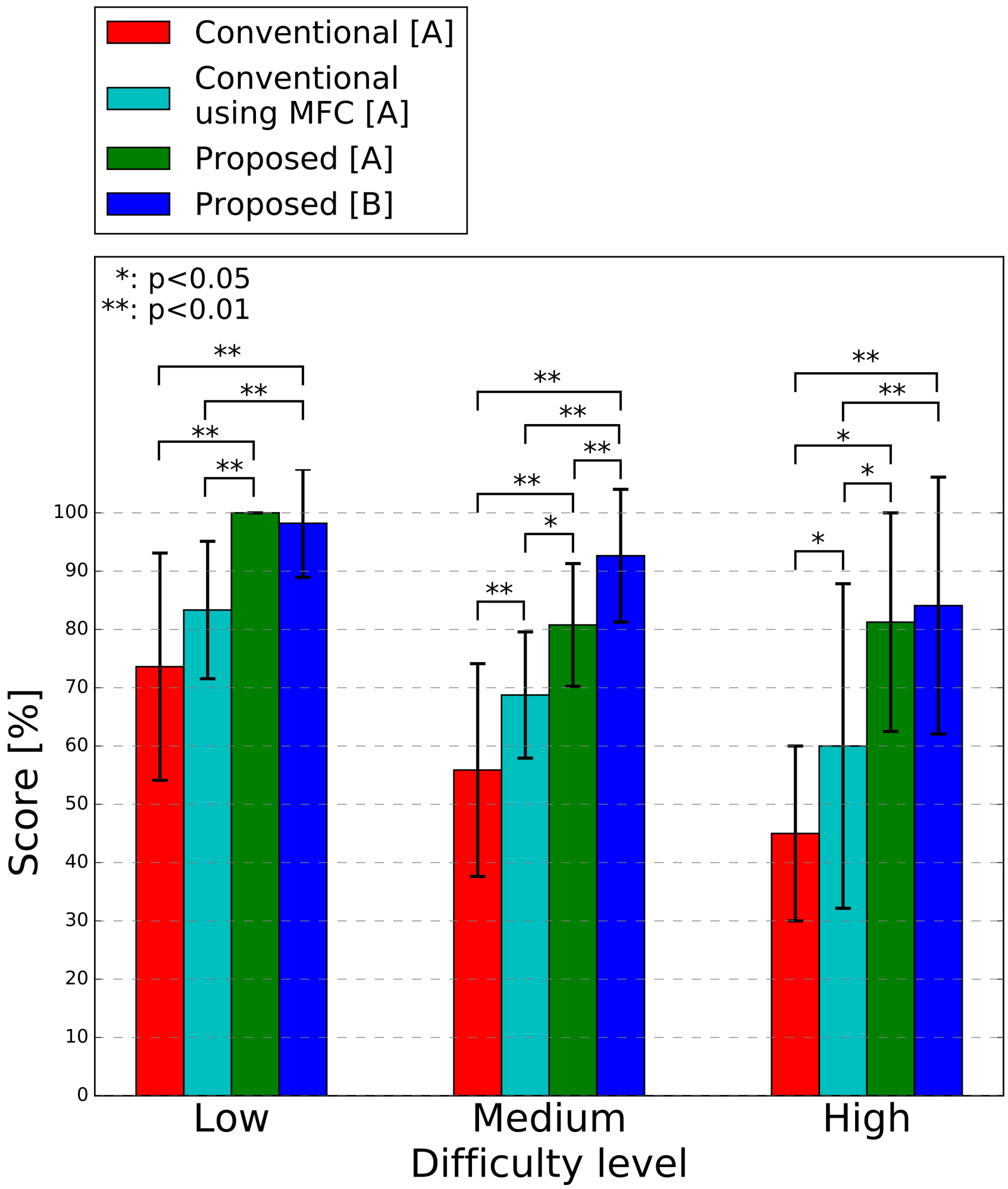


Figure 10: Scores divided in three groups by the difficulty level of the initial state, without disturbances in the environment.

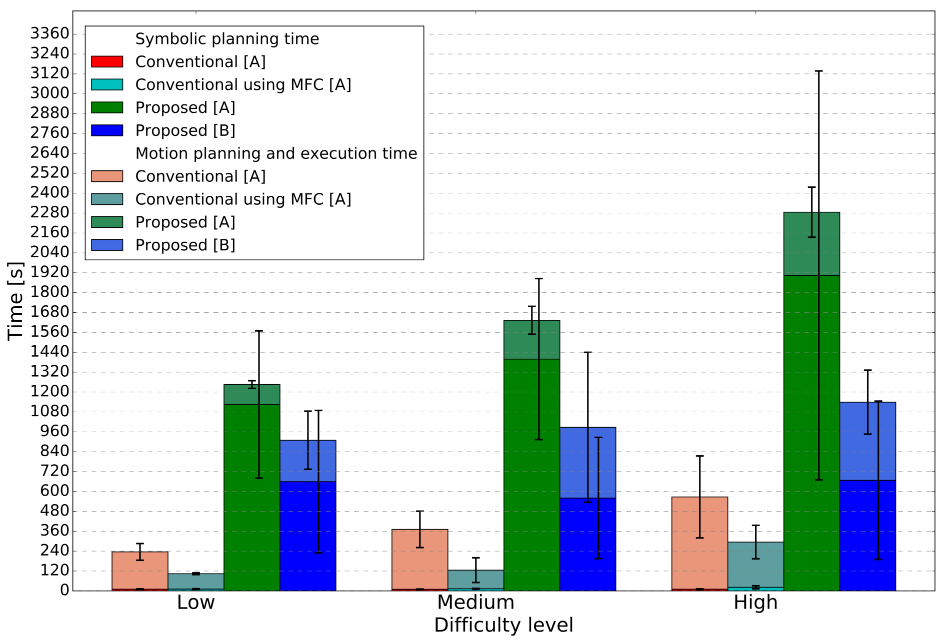


Figure 11: Task completion time divided in three groups by the difficulty level of the initial state, without disturbances in the environment.

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**	0.00	80.77	10.53	81.25	18.75
Proposed [B]	98.21	9.28	**92.65**	11.39	**84.09**	22.0

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

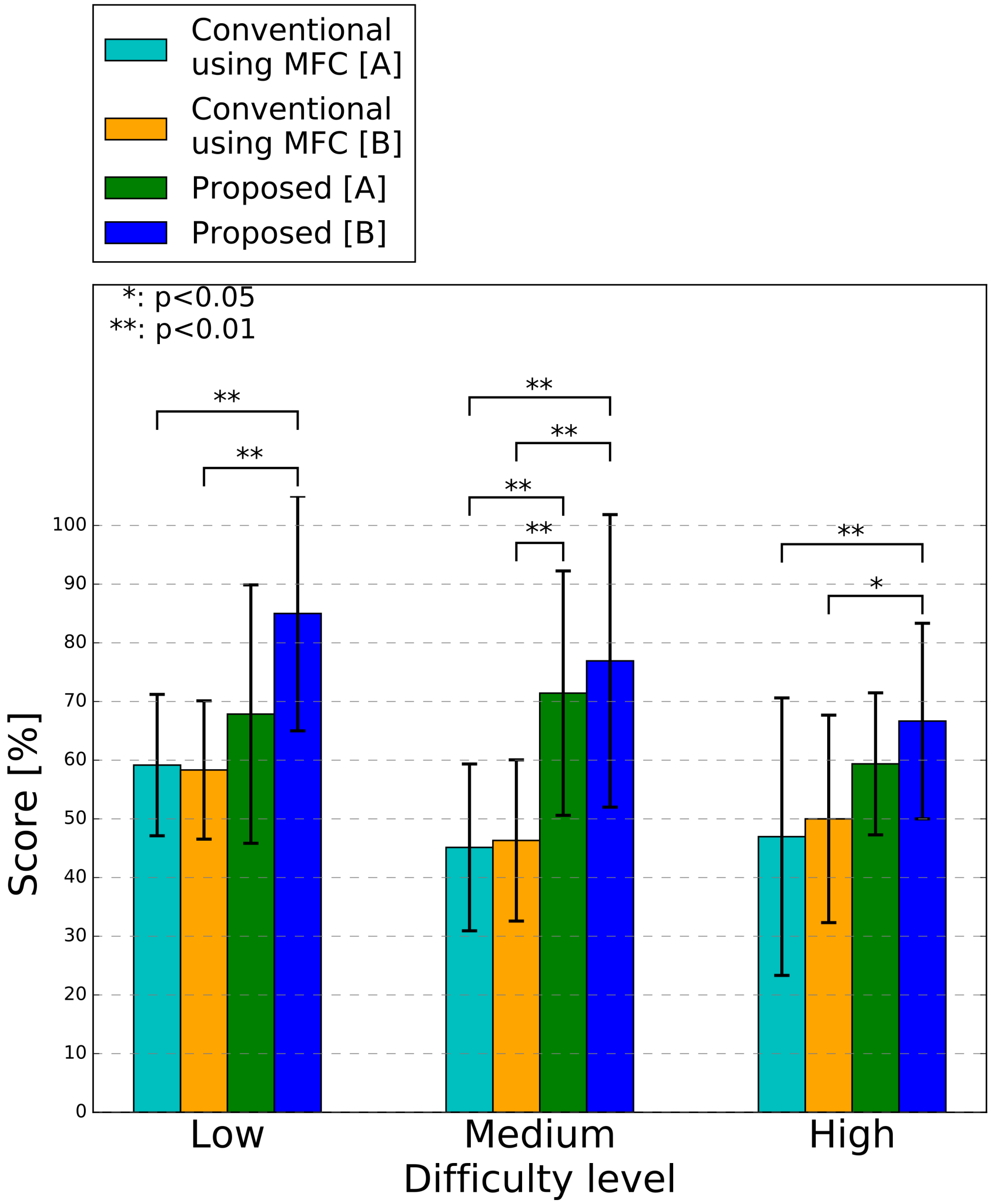


Figure 12: Scores divided in three groups by the difficulty level of the initial state, with disturbances in the environment.

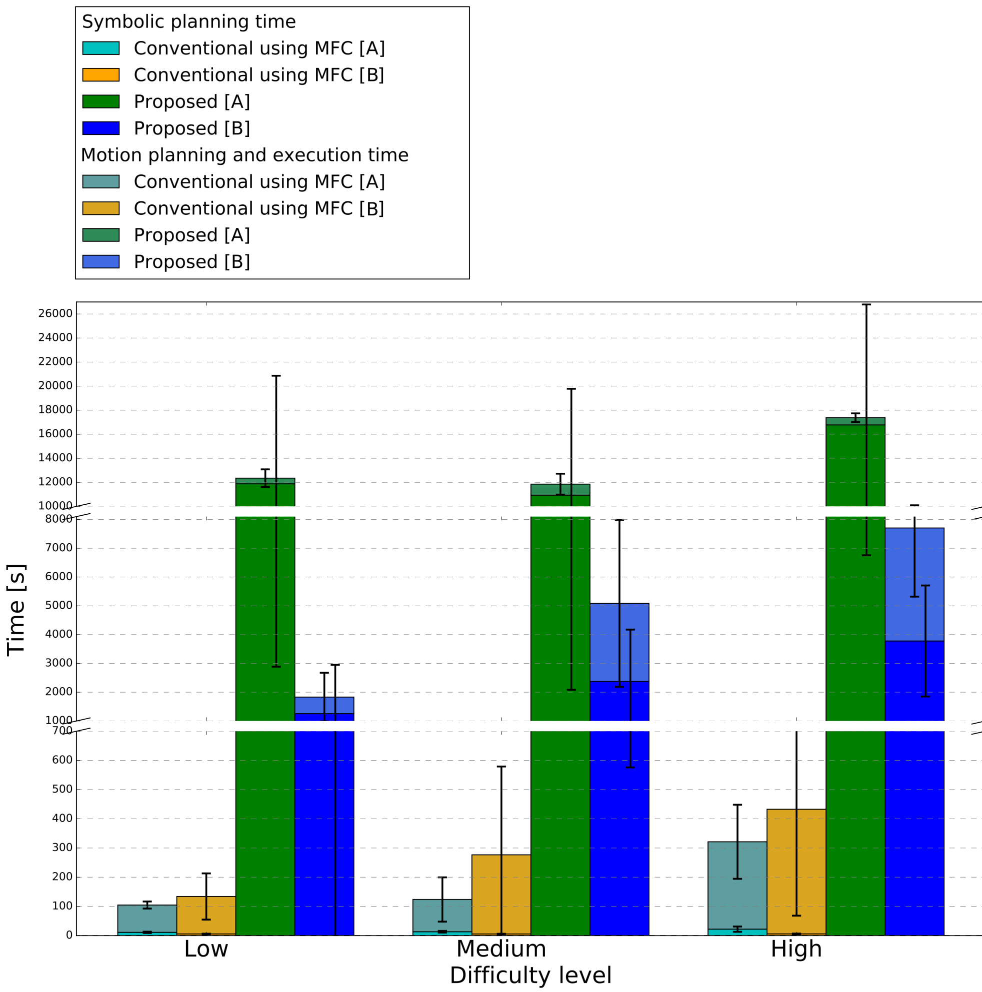
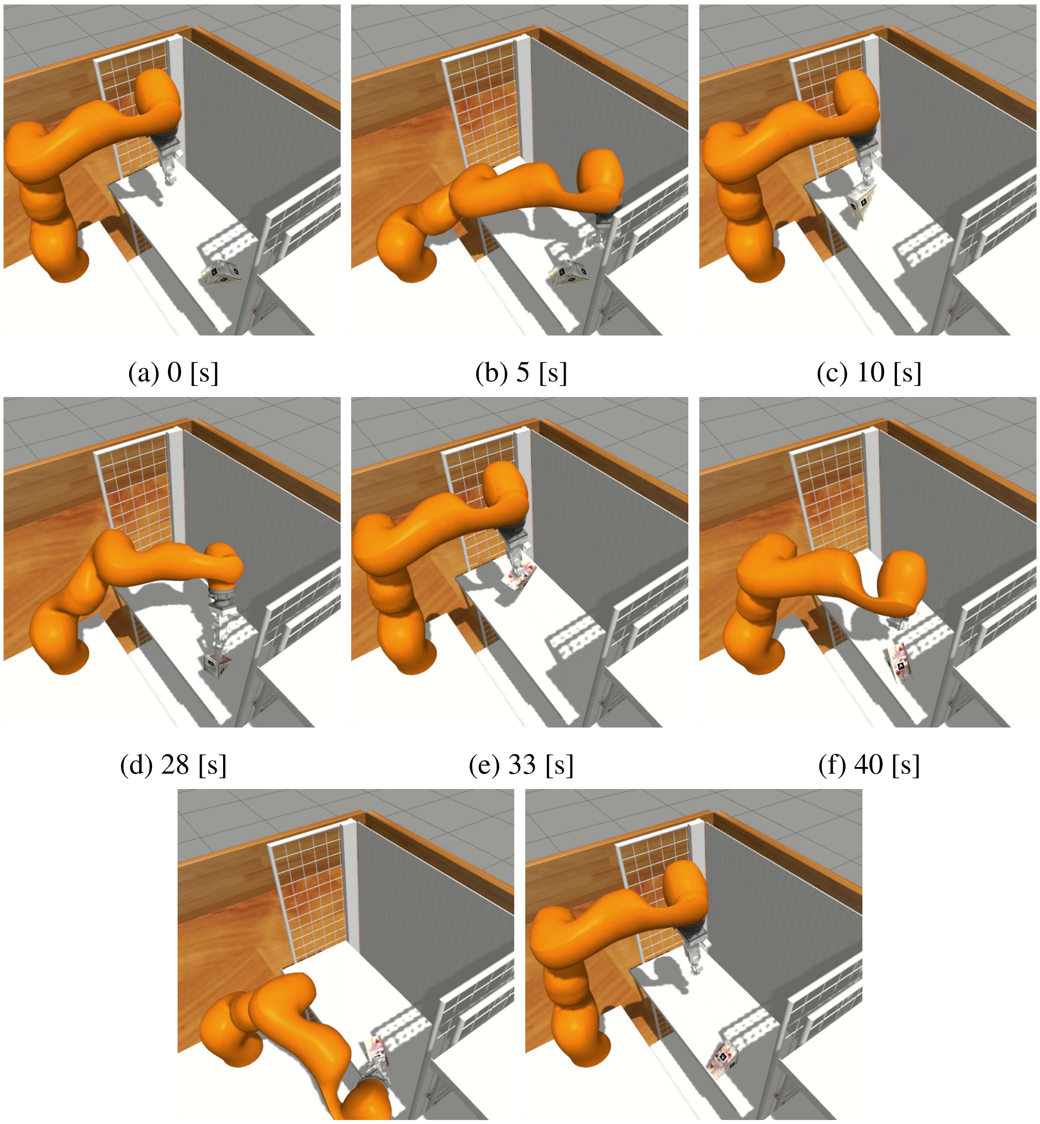


Figure 13: Task completion time divided in three groups by the difficulty level of the initial state, with disturbances in the environment

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

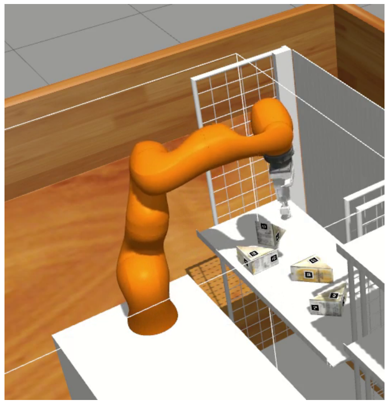


(a) 0 [s] (b) 5 [s] (c) 10 [s]

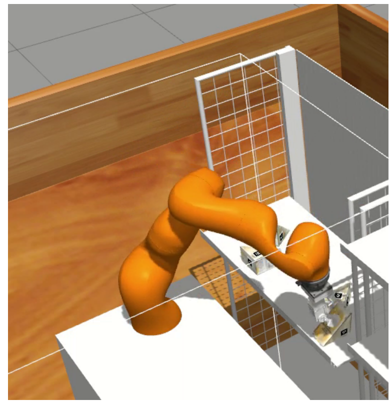
(d) 28 [s] (e) 33 [s] (f) 40 [s]

(g) 50 [s] (h) 64 [s]

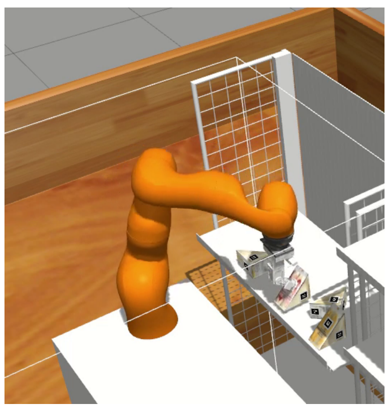
Figure 14: Screenshots of the robot rearranging an item performing multiple re-grasps of it.



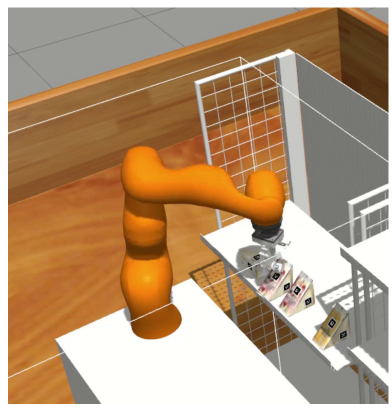
(a) 0 [s]



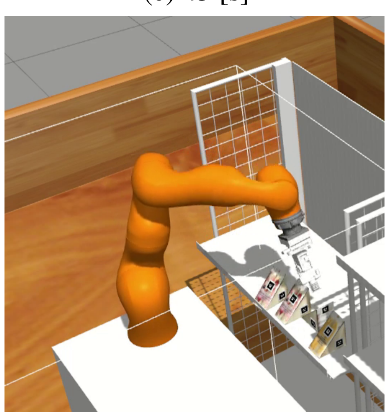
(b) 27 [s]



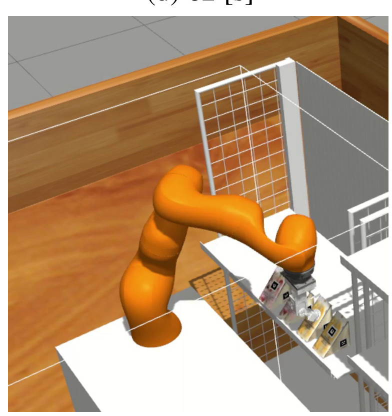
(c) 45 [s]



(d) 62 [s]



(e) 65 [s]



(f) 72 [s]

Figure 15: Screenshots of the robot rearranging an environment. The sequence of actions were obtained from the MCTS, and the re-grasp poses come from the MFC.

5.3 Experiments using different types of items

We evaluated the proposed method A and B doing rearranging experiment using different types of items. For these experiments, we used three rice balls in a clear packaging and two sandwiches. The rearranging rules are the same as in chapter 4. For these new experiments we do not consider disturbances in the environment, because our main focus is to analyze how the propose method deals with multiple items of different sizes. We executed 10 trials in 10 different initial states for each variation of the proposed method.

The information of the items for these experiments is the following:

- Rice ball size: $0.9 \times 0.077 \times 0.045$ m
- Sandwich size: $0.07 \times 0.11 \times 0.11$ m

Table 3 shows the scores obtained by the methods, as we can see the proposed method A and B obtain a score higher than 75%. It is important to mention that for the new item (rice balls), we had to consider more orientations of the end-effector. We considered these orientations 0°, ±30° and ±45°. In these experiments the orientations considered to pick and to place the items is the same, so there is only one feasibility database.

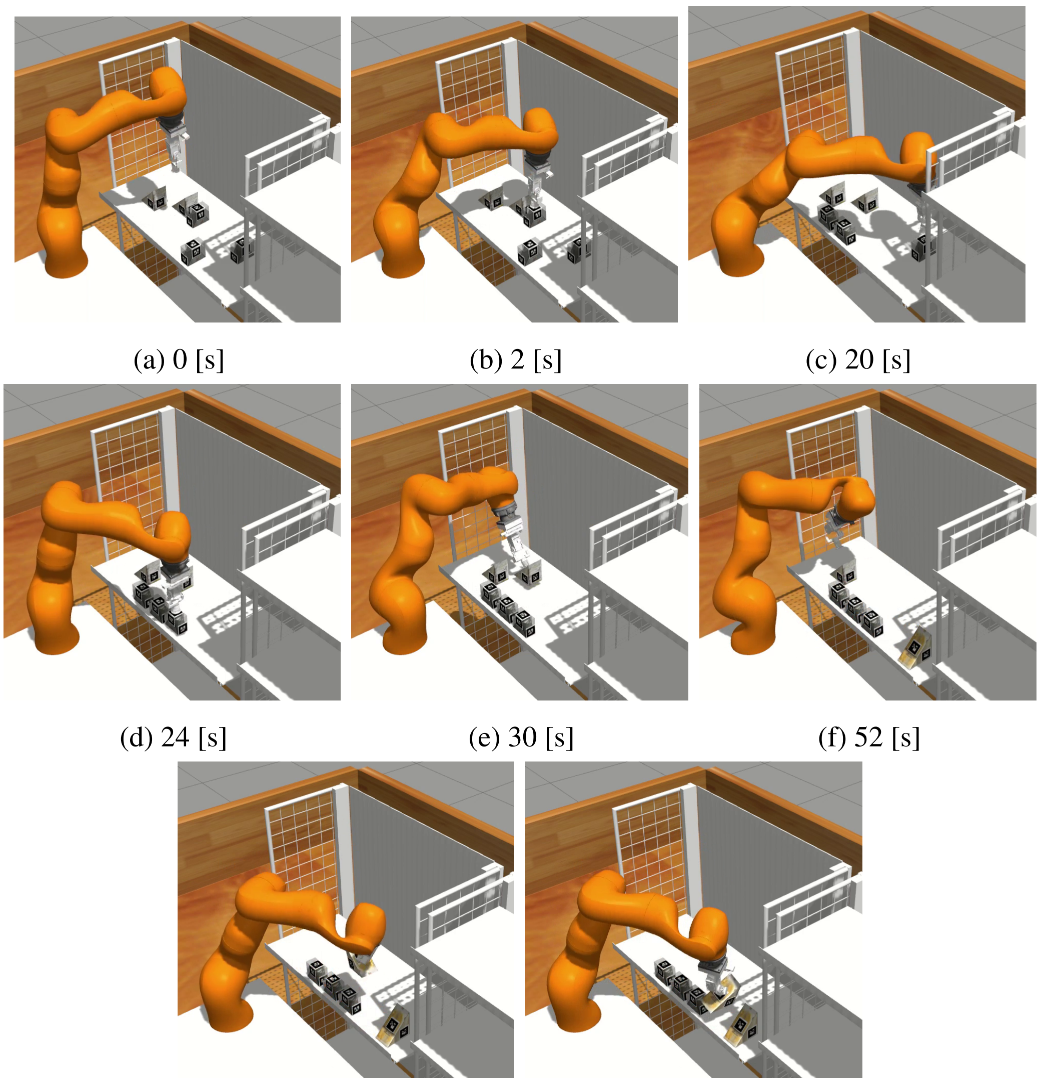
The table also shows that the time for rearranging five items is multiple times higher than rearranging four items. This shows that our proposed method will require more time to find a solution every time we add new items to the environment. This increase of time is a issue that can be improved later in a future work.

Fig. 16 shows an execution of one of the experiments rearranging items of different size.

Table 3: Results of the experiments using different items.

Rearranging five items (2 sandwiches, 3 rice balls)				
	Score [%]		Time [s]	
Method	Mean	Standard deviation	Mean	Standard deviation
Proposed A	74.54	23.34	6020.81	2702.95
Proposed B	86.06	16.68	2929.87	977.90

Rearranging four items (4 sandwiches)				
	Score [%]		Time [s]	
Method	Mean	Standard deviation	Mean	Standard deviation
Proposed A	95.34	11.98	1744.23	993.25
Proposed B	97.61	9.92	1075.35	653.51



(a) 0 [s] (b) 2 [s] (c) 20 [s]

(d) 24 [s] (e) 30 [s] (f) 52 [s]

(g) 60 [s] (h) 70 [s]

Figure 16: Screenshots of the robot rearranging an environment with items of different size. The sequence of actions were obtained from the MCTS, and the re-grasp poses come from the MFC.

Chapter 6

Discussion

In Chapter 5, 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.

Recent works on rearranging tasks [25–27] 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.* [23] 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 solution 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 [23,25–27] 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 leverages the advantages of the MCTS combined with the MFC. 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. The advantage of the proposed method compared to only using the MFC, is that the proposed method considers the whole sequence for rearranging the items, where as the conventional methods focus on finding the first valid solution for all the items.

It is also important to mention that the proposed method 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 does not consider the collision of between items or between the item and the shelf.

The current implementation of the proposed method is affected by the number of items it needs to rearrange. If the number of items is increased, the size of the tree will increase and it will require more time to expand the states and search for a solution, likewise if we change the size of the grid. With a smaller grid, the proposed method will be faster because the the possible movements of the items in the grid decreases directly affecting the size of the tree. This was proved by the experiments rearranging five items, where the proposed method A required thrice the time as before to find the solution.

Finally, to be able to use the proposed method in new environments or with different types of items this is some of the information that we need to have in consideration. The feasibility database is specific to the robot, in case that the robot changes. We will need to create a new database, or in case that the environment changes (*e.g.*, the height of the shelf, or the items). Also, the geometry of the items to rearrange is important, because based on that we manually defined the grasping points and the sampling of the orientations of the end-effector to pick and place the items.

To avoid these possible issues, we can create a more general feasibility database considering not only the workspace of the robot such as the shelf, but the whole reachibility of the robot using multiple end-effector orientations even if they are not required for picking and placing the items to rearrange. There is a disadvantage in using this approach, that it is too time consuming. The advantage is that the database would only be built once and then could be used in different environments. Also, having a big database will also increase the time it takes for the MFC to find the possible candidates and test the possible combinations, we could prune the database to reduce the size of the database depending the environment, we just keep the poses that are within our target workspace. This would only add an extra step in the MFC before getting the pick and place candidates.

Chapter 7

Conclusions

In this dissertation, 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 proposed MFC validates the instructions using a pre-computed feasible motion database to determine if there is a pick-place sequence to manipulate the item that satisfies the instruction received. This early validation rejects invalid instructions early so we do not waste time trying to execute those instructions. The MCTS uses the MFC to validate the states in the tree, in this way the motion and symbolic planning are combined. The solution found by the MCTS can be executed directly by the robot with a low probability of failure in the motion planning.

We evaluated our proposed method doing a shelf rearranging task with different types and shapes of items, the results obtained show that the proposed method outperforms the other methods and it is robust against disturbances in the environment. We proposed two variations of the proposed method A and B, B being our preferred approach to solve the rearranging tasks. The variation B of the proposed method is faster than A and in general obtains a higher score than the other methods in all the scenarios evaluated.

We consider as future work, to reduce the time that the MCTS uses in the tree search. We think that this could be achieved by first, modifying the MFC so it does not verify all possible combinations of pick and place candidates, instead we could use a threshold and if the comparison between target and predicted poses is less than the threshold. We use directly that pick and place combination. Second, we could use a machine learning approach to accelerate the tree search, where we will train a model using different patterns of how to rearrange the environment. Finally, we could fine-tune the parameters of the MCTS such as the exploration constant or depth of simulation. Other improvements to the propose method can be in the metric used to compare the target and predicted poses in the MFC. Currently, we are using the geodesic unit sphere, but this metric calculates the angle between two quaternions. There are some ambiguities while comparing two different poses. We consider that changing this metric can also help in reducing the number of re-grasps that the item requires to reach the target pose.

Publication list

Refereed International Journal

1. **Pedro Miguel Uriguen Eljuri**, Gustavo Alfonso Garcia Ricardez, Nishanth Koganti, Jun Takamatsu and Tsukasa Ogasawara, "Combining a Monte Carlo Tree Search and a Feasibility Database to Plan and Execute Rearranging Tasks," *IEEE Access*, vol. 9, pp. 21721-21734, 2021.

Refereed International Conference Papers

1. **Pedro Miguel Uriguen Eljuri**, Gustavo Alfonso Garcia Ricardez, Nishanth Koganti, Jun Takamatsu and Tsukasa Ogasawara, "Rearranging Tasks in Daily-life Environments Using a Monte Carlo Search and a Feasibility Database," *in Proceedings of IEEE/SICE International Symposium on System Integration*, pp. 330-335, 2021.

2. **Pedro Miguel Uriguen Eljuri**, Gustavo Alfonso Garcia Ricardez, Nishanth Koganti, Jun Takamatsu and Tsukasa Ogasawara, "Combining Symbolic and Motion Planners for Rearranging Tasks in Daily-life Environments," *in Proceedings of 2020 Fourth IEEE International Conference on Robotic Computing*, pp. 71-74, 2020.

Other Refereed International Journals

1. Gustavo Alfonso Garcia Ricardez, Seigo Okada, Nishanth Koganti, Akiya Yasuda, **Pedro Miguel Uriguen Eljuri**, Tetsuya Sano, Pin-Chu Yang, Lotfi El Hafi, Masaki Yamamoto, Jun Takamatsu and Tsukasa Ogasawara, "Restock and Straightening System for Retail Automation using Compliant and Mobile Manipulation," *in RSJ Advanced Robotics*, vol. 34, no. 3-4, pp. 235-249, 2020.

2. Gustavo Alfonso Garcia Ricardez, Nishanth Koganti, Pin-Chu Yang, Seigo Okada, **Pedro Miguel Uriguen Eljuri**, Akiya Yasuda, Tetsuya Sano, Lotfi El Hafi, Masaki Yamamoto, Jun Takamatsu and Tsukasa Ogasawara, "Adaptive Motion Generation using Imitation Learning and Highly-Compliant End Effector for Autonomous

Cleaning," *in RSJ Advanced Robotics,* vol. 34, no. 3-4, pp. 189-201, 2020.

Acknowledgements

This thesis would not be completed without the people who helped me and supported me during this challenge. I would like to take this opportunity to show my appreciation to them.

To Professor Tsukasa Ogasawara, I would like to express my sincere gratitude for accepting me as a Master student in his laboratory and later as a Doctor Student. None of this would have been possible without his kindness, wisdom and continue support that helped me to achieve my dreams.

To Associate Professor Jun Takamatsu, I am very grateful to him for his kind advice, patience, and guidance that helped me to realize new possibilities and challenges.

To Assistant Professor Gustavo Garcia, I am very grateful for his explanations, help, and comments during the different stages of my research. Thank you very much for your patience, especially while checking my papers and presentations.

To Assistant Professor Sung-Gwi Cho, I am also thankful for his valuable suggestions and questions that give me a new perspective.

I am also grateful to my thesis committee: Professor Tsukasa Ogasawara, Professor Kenji Sugimoto, Associate Professor Jun Takamatsu, Assistant Professor Gustavo Garcia, Assistant Professor Sung-Gwi Cho for questions and comments that helped me to improve.

I would like to deeply thank the Ministry of Education, Culture, Sports, Science and Technology of Japan for having granted me with the MEXT Scholarship to help me achieve my dream of studying in Japan

I would like to thank especially to my family and friends that even being far away. They always find a way of not making me feel the distance, supporting me with their affection and understanding.

To my teammates in the World Robot Summit with whom I share many happy and stressful moments during the robot competitions.

To my colleagues and all members in the Robotics laboratory, who have encouraged since my first day in the laboratory.

To all my friends for the wonderful moments that we shared. Especially thanks to Abi Uy, Anirban Chakraborty, Erica Gonzales, and Diana Romero whom shared with me since my arrival in Japan.

References

[1] T.-M. Wang, Y. Tao, and H. Liu, "Current researches and future development trend of intelligent robot: A review," *International Journal of Automation and Computing*, vol. 15, no. 5, pp. 525–546, 2018.

[2] D. Belanche, L. V. Casaló, C. Flavián, and J. Schepers, "Service robot implementation: a theoretical framework and research agenda," *The Service Industries Journal*, vol. 40, no. 3-4, pp. 203–225, 2020.

[3] B. Gates, "A robot in every home," *Scientific American*, vol. 296, no. 1, pp. 58–65, 2007.

[4] J.-D. Dessimoz and P.-F. Gauthey, "RH5-Y–toward a cooperating robot for home applications," in *Robocup-at-Home League, Proceedings Robocup07 Symposium and World Competition, Georgia Tech, Atlanta, USA*, vol. 30. Citeseer, 2010.

[5] M. Cakmak and L. Takayama, "Towards a comprehensive chore list for domestic robots," in *Proc. of the 8th ACM/IEEE Int. Conf. on Human-Robot Interaction (HRI)*, 2013, pp. 93–94.

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

[7] R. Coulom, "Efficient selectivity and backup operators in Monte-Carlo Tree Search," in *Proc. of the Int. Conf. on Computers and Games*. Springer, 2006, pp. 72–83.

[8] G. M. J.-B. Chaslot, M. H. M. Winands, H. J. V. D. Herik, J. W. H. M. Uiterwijk, and B. Bouzy, "Progressive strategies for Monte-Carlo Tree Search," *New Mathematics and Natural Computation*, vol. 4, no. 3, pp. 343–357, 2008.

[9] K. L. Myers, "Towards a framework for continuous planning and execution," in *Proceedings of the AAAI Fall Symposium on Distributed Continual Planning*, vol. 12. Citeseer, 1998, p. 13.

[10] S. Cambon, F. Gravot, and R. Alami, "A robot task planner that merges symbolic and geometric reasoning," in *Proc. of the 16th European Conference on Artificial Intelligence*, 2004, pp. 895–899.

[11] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas, "Symbolic planning and control of robot motion [grand challenges of robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 61–70, 2007.

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

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

[14] C. Phiquepal and M. Toussaint, "Combined task and motion planning under partial observability: An optimization-based approach," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9000–9006.

[15] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical planning in the now," in *Workshops at the Twenty-Fourth AAAI Conf. on Artificial Intelligence*, 2010.

[16] 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.

[17] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 729–746, 2004.

[18] T. Welschehold, N. Abdo, C. Dornhege, and W. Burgard, "Combined task and action learning from human demonstrations for mobile manipulation applications," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 20019, pp. 4317–4324.

[19] J. E. King, V. Ranganeni, and S. S. Srinivasa, "Unobservable monte carlo planning for nonprehensile rearrangement tasks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4681–4688.

[20] C. Dornhege, M. Gissler, M. Teschner, and B. Nebel, "Integrating symbolic and geometric planning for mobile manipulation," in *Proc. of the IEEE Int. Workshop on Safety, Security & Rescue Robotics (SSRR)*, 2009, pp. 1–6.

[21] C. R. Garrett, T. Lozano-Perez, and L. P. Kaelbling, "FFRob: Leveraging symbolic planning for efficient task and motion planning," *The International Journal of Robotics Research*, vol. 37, no. 1, pp. 104–136, 2018.

[22] L. E. Kavraki, P. Svestka, J. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[23] N. T. Dantam, S. Chaudhuri, and L. E. Kavraki, "The task motion kit," *IEEE Robotics and Automation Magazine*, vol. 25, pp. 61–70, 2018.

[24] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "An incremental constraint-based framework for task and motion planning," *The International Journal of Robotics Research*, vol. 37, no. 10, pp. 1134–1151, 2018.

[25] C. Paxton, Y. Barnoy, K. Katyal, R. Arora, and G. D. Hager, "Visual robot task planning," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2019, pp. 8832–8838.

[26] Y. Labbé, S. Zagoruyko, I. Kalevatykh, I. Laptev, J. Carpentier, M. Aubry, and J. Sivic, "Monte-carlo tree search for efficient visually guided rearrangement planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3715–3722, 2020.

[27] 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. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, 2020, pp. 9433–9440.

[28] J. Lee, Y. Cho, C. Nam, and C. Kim, "Efficient obstacle rearrangement for object manipulation tasks in cluttered environments," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol. 5, 2019, pp. 183–189.

[29] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*. Springer, 2006, pp. 282–293.

[30] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.

[31] D. Huggins, "Comparing distance metrics for rotation using the k-nearest neighbors algorithm for entropy estimation," *Journal of Computational Chemistry*, vol. 35, no. 5, pp. 377–385, 2014.

[32] 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 Transactions on Medical Imaging*, vol. 36, no. 2, pp. 538–548, 2017.

[33] K. Wada, "New robot technology challenge for convenience store," in *2017 IEEE/SICE Int. Symposium on System Integration (SII 2017)*, 2017, pp. 1086–1091.

[34] G. A. Garcia Ricardez, S. Okada, N. Koganti, A. Yasuda, P. M. Uriguen 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," *Advanced Robotics*, vol. 34, no. 3-4, pp. 235–249, 2020.

[35] L. Zhang and D. Manocha, "An efficient retraction-based RRT planner," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2008, pp. 3743–3750.

Appendix

Chapter 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 rotations 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, quaternion representation of the rotation 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 quaternion representation of two rotations respectively; 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).

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

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 an difficult initial state, where the robot needs to perform multiple maneuvers and re-grasping of the items. We divide the initial states into three groups based on their difficulty: low (0.0 - 0.33), medium (0.34 - 0.66) and high (0.67 - 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.

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

Figure 17: Sides of the item.

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. 17 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.

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

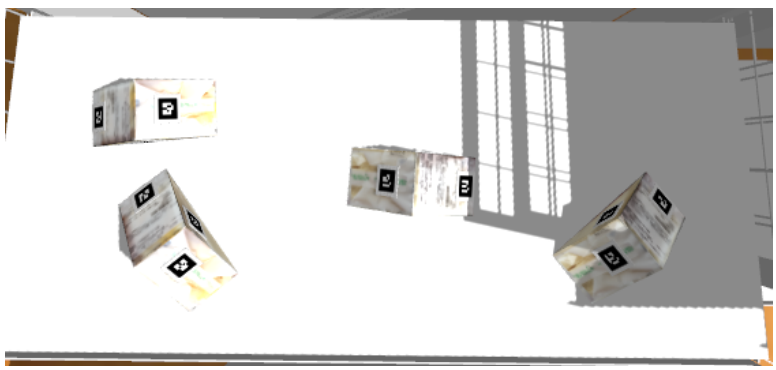
where n is the current side of the item.

To determine the difference between the quaternions of the initial pose and target pose rotation, we use as metric the geodesic unit sphere [31]:

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

where $\mathbf{q}$ and $\mathbf{q}_{\text{target}}$ are the quaternion representation of the rotation of the initial and

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



(b) A difficult initial state with difficulty of 0.87 (high difficulty)

Figure 18: Initial states with different levels of difficulty.

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

Fig. 18 shows the difficulty of two initial cases. Fig. 18a 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. 18b is a high difficulty state where the items are scattered in the shelf. This state the robot needs to perform multiple re-graspings to solve the rearranging task.