TiAGo: A Sorting Robot for Hagelslag

Yuezhe Zhang Siyuan Wu Danning Zhao

***Abstract*—The promising future of retail requires the robots to use a variety of skills to perceive the environment, navigate around it and manipulate objects. Equipped with symbolic knowledge using Planning Domain Definition Language (PDDL), our robot TIAGo, can sort among empty and full hagelsag and place them to shelves or drop them to the basket in ROS simulation. Our simulation has shown that both grippers can perform the actions of pick, place and discard independently and the system can also cope with the situation when new products are added to the environment without any changes to the framework other than the information to the knowledge base.**

***Index Terms*—PDDL, Knowledge Base, Robot Planning**

I. Introduction

During the pandemic of COVID-19, people in grocery store were asked to keep ”social distance” to protect both customers and employees from the virus. During this hard period, the unprecedented increase for online ordering and absence of employees have forced grocery stores to rely on innovation and technology to keep up with heavy demand. The retail automation, which is using robot to replace human employees in grocery stores and warehouses, has proven its ability to make shops and warehouses more efficient. Robots are deployed to improve productivity to help deal with this sky-high demand by performing repetitive tasks. At the moment, robots are able to clean floors, direct customers, sort merchandise [1]. However, no robot can do more than one of these things. Robots do not have the ability to plan and accomplish a sequence of multiple task independently which may seem quite simple to human employees. For example, one common task for human employees in grocery stores is to transport merchandise from warehouse, place them on the empty shelf, and discard damaged goods.

Our goal is to enable a typical service robot, TIAGo, to release human workers from heavy workload in grocery store. TIAGo has a base to move in indoor environments, a mobile manipulator to pick and place objects, and necessary sensors for environment perception. However, its potential to work in grocery stores is limited by the lacking ability to understand scenarios and reason from known information. We would like to implement TIAGo's behaviour by Knowledge Representation and Reasoning (KRR) approaches with reasoning using symbolic knowledge.

To be specific, our task is to create a KRR system to support the operation of the robot in following scenarios: due to the outer packaging of the products could have been damaged during transit, our robot should figure out which products have been damaged and to separate them from the good ones. It should separate damaged products according to its camera and discard the empty or damaged ones to the basket. For other products, it should place them on the shelves just like human employees. The specific tasks include:

- Identify and separate the empty products from the other products on the table.
- Grasp products using both grippers.
- Move to the basket and discard the empty products.
- Move to the shelf and place the full products with both grippers.

To develop our KRR solutions, we utilized Planning Domain Definition Language (PDDL) [2], [3] and ROSPlan [4]. PDDL is a problem-specification language for formulating and solving planning problems using predicates and actions, which consists of two main parts, a domain description and a problem description. The ROSPlan framework provides a collection of tools for planning in a ROS system. It has a variety of nodes which encapsulate reasoning, problem generation, and execution. It possesses a simple interface, and links to ROS libraries, including MoveIt [5] and move_base [6].

II. World Model and Knowledge Base

A. World model

To simplify our task to a typical occasion, we built a simulation environment in Gazebo as shown in Fig. 1, where we place a table, a basket and a shelf in retail stores. The TIAGo robot is at the center of the retail store. It has an extendable torso and two manipulator arms to grab products. We place 4 objects on the table, which are illustrated in Fig. 2. The four products are hagelslags with red and blue packaging. We use

All authors are master students at TU Delft, The Netherlands. E-mail addresses: { Y.ZHANG-130, S.Wu-14, }@student.tudelft.nl.

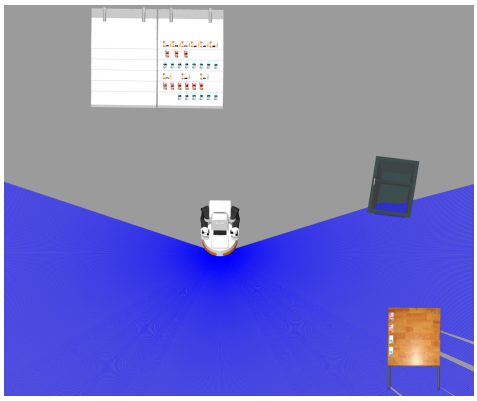

Fig. 1: A simulation world we built to simplify the task.

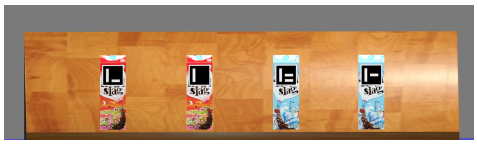

Fig. 2: One typical arrangement of four products on the table. The left two products are red, indicating packaging is broken. The rest products are blue which are well-preserved.

red hagelslag model to represent empty products with broken outer packaging, which should be discarded by the TIAGo robot. Meantime, the blue hagelslag model represents full products which should be placed carefully on the shelf. Note the arrangement of these products on the table is random and TIAGo robot should scan aruco on the products to locate these products. Fig. 2 only gives one typical arrangement of four products on the table.

We build the environment of our own since it is closely related to the tasks we want to accomplish. We want to sort the products between empty ones and full ones, it is thus much vivider and more genuine to use the red and blue hagelslag items instead of using multi-cubes. And we also choose the shelf containing many products instead of the original empty shelf. It is worth mentioning that since our new shelf does not contain the information of the aruco markers, we have to place the items on the shelf based on hard-coded positions.

B. Task in simulation

Our simplified task in the typical simulation environment can be described as follows. The initial position is at the center of the room. The basket and shelf are initially empty. There are 4 products on the table, two of them are red and the other are blue. All of these products can be picked and placed by both grippers. TIAGo should be able to move to all three waypoints: table, basket and shelf. Our task in this simulation is to throw all red hagelslags into the basket and place all blue hagelslags on the shelf.

C. Knowledge base

There are many KRR methods we can choose for solving this task. Two major options are PDDL and Prolog with Situation Calculus. The reasons we choose PDDL as our knowledge base are threefold. Firstly, it is much convenient in expressing different planning problems within the same framework, since the same domain description can be used with many different problem descriptions to solve different planning problems in the same domain. Secondly, the parameterisation of actions uses the variables that represent objects from a specific problem instance, which makes it more intuitive and understandable. Thirdly, pre- and post-conditions of actions are expressed as logical propositions constructed from predicates, objects and logical connectives, which are easy to implement.

Our knowledge base contains following knowledge:

- Our robot has four actions: move, pick, place and discard.
- It can move when it is free and locate at the starting point.
- Our robot can pick a product when it is at the waypoint where the product locates. And the product can be picked up and be held by the corresponding free gripper.
- Our robot can discard an empty product by grippers when the robot is at the desired location – basket. And then the robot is free to perform next action.
- Our robot can place a full product by grippers when the robot is at the desired location – shelf. And then the robot is free.

III. Reasoning and Planning

The design details of PDDL domain file can be seen in Table I. The domain file defines the comprehensive aspects of the problem and it is applicable to different specific problem situations. The file normally contains object types, predicates and actions to define the model. In our case, we use `is_full(obj)` and `is_empty(obj)` to determine which item is full or empty. The predicate `can_pick(g, obj)` is used to determine which gripper can pick which item. The predicate `can_place(obj, wp)` is used to determine where the object can be placed. The predicate `can_discard(wp)` is used to determine where we can drop the items. The predicate `is_classified` describes the state the item is sorted well, which means

TABLE I: Details of PDDL domain file

Name	Contents
Types	waypoint-wp, object-obj, robot-v, gripper-g
Predicates	visited(wp), gripper_free (g), robot-free(v), robot-at(v, wp), object-at(obj, wp), can_pick(g, obj), can_place(obj, wp), can_discard(wp), is_holding(g, obj), is_full(obj), is_empty(obj), is_classified(obj)
Durative actions	move, pick, place, discard

the empty item is dropped or the full item is placed on the shelf.

We attach the code of the action discard at Listing 1 as an example to show how the durative action works. The pre-conditions of the discard action are that the robot is at the location for dropping items, and it is holding an empty object with one gripper, which can be discarded at this location. The effects of the action are that the robot is still at this location but the gripper is free and not holding anything anymore, and the item is dropped at the place and hence well-classified. The time duration for this action is 2s.

```
(:durative-action discard
    :parameters (?v - robot
                 ?wp - waypoint
                 ?obj -object
                 ?g - gripper)
    :duration (= ?duration 2)
    :condition (and
        (at start (robot-at ?v ?wp))
        (at start (robot-free ?v))
        (at start (is_holding ?g ?obj))
        (at start (is_empty ?obj))
        (at start (can_discard ?wp))
    )
    :effect (and
        (at end (robot-at ?v ?wp))
        (at start (not (robot-free ?v)))
        (at end (robot-free ?v))
        (at end (free ?g))
        (at end (not (is_holding ?g ?obj)))
        (at end (object-at ?obj ?wp))
        (at end (is_classified ?obj))
    )
)
```

Listing 1: Action discard in PDDL.

The design details of PDDL problem file is shown in Table II. The `Objects` define the items we will use in the world model, including tiago robot, a few location points, two grippers and 4 items of hagelslag. The `Ground truth` describes some rules of the items. We define two red products as empty and two blue products as full. The empty products should be discarded at the basket and the full products can be placed at two shelf locations. We also define which gripper can pick which item. The `Initial states` describe the situations when the program starts. In our case, the robot is still at the start point and two gippers are free. All four items are at the table. The `Goal` describes what we want to achieve in this task. Here the goal is to make all the items classified. The preconditions of classified has been explained above.

TABLE II: Details of PDDL problem file

Name	Contents
Objects	tiago - robot wp0 wp_table_1 wp_table_2 wp_shelf_1 wp_shelf_2 wp_basket - waypoint leftgrip rightgrip - gripper AH_hagelslag_aruco_16 AH_hagelslag_aruco_17 AH_hagelslag_aruco_0 AH_hagelslag_aruco_1 - object
Ground truth	(is_full AH_hagelslag_aruco_16) (is_full AH_hagelslag_aruco_17) (is_empty AH_hagelslag_aruco_1) (is_empty AH_hagelslag_aruco_0) (can_place AH_hagelslag_aruco_16 wp_shelf_1) (can_place AH_hagelslag_aruco_17 wp_shelf_2) (can_discard wp_basket) (can_pick leftgrip AH_hagelslag_aruco_17) (can_pick leftgrip AH_hagelslag_aruco_1) (can_pick rightgrip AH_hagelslag_aruco_16) (can_pick rightgrip AH_hagelslag_aruco_0)
Initial states	(robot-at tiago wp0) (robot-free tiago) (free leftgrip) (free rightgrip) (object-at AH_hagelslag_aruco_16 wp_table_1) (object-at AH_hagelslag_aruco_17 wp_table_1) (object-at AH_hagelslag_aruco_0 wp_table_1) (object-at AH_hagelslag_aruco_1 wp_table_1)
Goal	(is_classified AH_hagelslag_aruco_1) (is_classified AH_hagelslag_aruco_0) (is_classified AH_hagelslag_aruco_16) (is_classified AH_hagelslag_aruco_17)

The solution of the PDDL planning can be shown in Fig.3. To accomplish the goals, the robot will move to the table and pick two empty items using left gripper and right gripper sequentially. And then it will move to the shelf and place them. After that the robot will return to the table and pick two full items using two grippers and move to the basket to drop them. It is worth mentioning that the solution here is the outcome of the PDDL online editor, which uses an optimal solver and can provide an optimal solution. However, since the ROSPlan utilizes the default PDDL solver, which can not provide an optimal solution but a satisfying solution. Therefore the result of the action sequence in the simulation is different from what we show here.

IV. Results

A. Results of the primary task

Here is a video demonstration with 8x speed showing the execution of our task in the simulation.

The task was accomplished successfully, but it can be seen from the video that TIAGo's action sequence is not optimal. Tiago first moves to basket and discards the objects after picking them, instead of moving to shelf and placing them.

B. Results of the extended task: adding new products

The extended task is the previous a pick-place-discard task with the addition of new empty products. In the problem file, we add a new empty hagelslag `AH_hagelslag_aruco_2`. We also add the following initial states and goals to the problem setting.

```
(object-at AH_hagelslag_aruco_2 wp_table_1)
(is_empty AH_hagelslag_aruco_2)
(can_pick leftgrip AH_hagelslag_aruco_2)
(can_pick rightgrip AH_hagelslag_aruco_2)
```

```
(:goal (and
    (is_classified AH_hagelslag_aruco_16)
    (is_classified AH_hagelslag_aruco_17)
    (is_classified AH_hagelslag_aruco_0)
    (is_classified AH_hagelslag_aruco_1)
    (is_classified AH_hagelslag_aruco_2)
)
```

Thanks to the advantage of PDDL approach, we do not need to modify any domain file or ROS code. We have shown that the extended task can be solved without any change or additions other than the information to the knowledge base.

The planning result of extended task can be visualized in the following read session shown in Fig. 4:

V. Discussion

In the simulation, we have done some modifications and implementations to achieve our task:

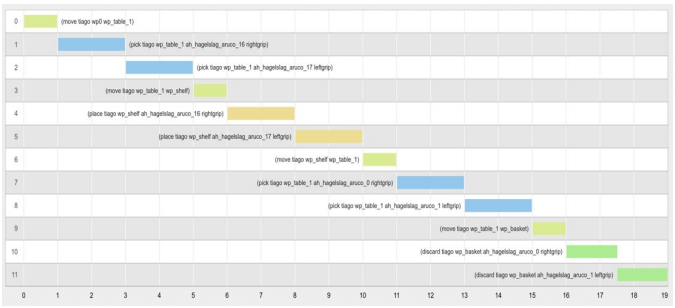

Fig. 3: Reasoning solution including pick(blue), place(orange), discard(green), and move(light green).

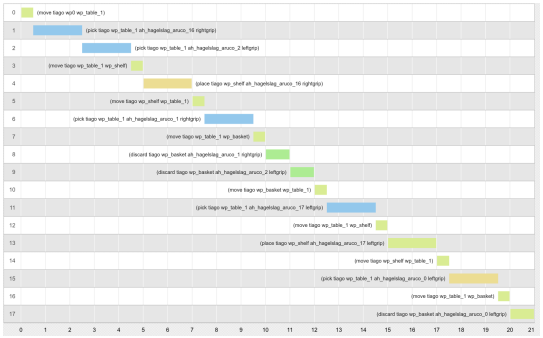

Fig. 4: Timeline view of extended task with new products adding to the previous problem

- We created the environment of our own to suit the needs of our task.
- Both grippers can pick, place and discard products.
- We refined the grasp location, since grasping a hagelslag is different from grasping a cube, and we need to shift the grasp location to the new mass center to avoid products falling off the hands.
- We implemented pre- and post- grasp pose to prevent bringing down other products and enable accurate picking as well.
- We implemented pre- and post place pose to enable accurate placing and thus reduced the risk of failure of RRTConnect.
- We refined the quaternion of the placing pose to prevent making the product upside down.

Even though it seems that we have achieved our task quite well, there are still some issues we have encountered along the way which can be addressed and improved in our future study:

- Our system still lacks robustness:
  - Sometimes the gripper server fails to receive command and get stuck.
  - Sometimes the robot is unable to find a path in narrow passage, especially in shelf compartment.
- Our robot cannot get feedback information from actions and reconfigure itself. After the pick fails, TIAGo will keep the current configuration and cannot return to the default posture to make re-pick. Developing a self-adaptive system that can switch the system configuration at run-time, such as MROS, could be a way to address this issue.
- Placing location is hard-coded, it would be better to perceive the partially-observable environment by integrating multimodal sensors.

REFERENCES

[1] "5 robots now in grocery stores provide a prevew of retail automation," Apr. 2020. [Online]. Available: https://www.roboticsbusinessreview.com/retail-hospitality/5-robots-grocery-stores-now/

[2] M. Fox and D. Long, "PDDL2.1: An extension to pddl for expressing temporal planning domains," *Journal of Artificial Intelligence Research*, vol. 20, pp. 61–124, 2003.

[3] D. McDermott, M. Ghallab, A. E. Howe, C. A. Knoblock, A. Ram, M. M. Veloso, D. S. Weld, and D. E. Wilkins, "Pddl-the planning domain definition language," 1998.

[4] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, N. Hurtos, and M. Carreras, "Rosplan: Planning in the robot operating system," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 25, 2015, pp. 333–341.

[5] "MoveIt Motion Planning Framework." [Online]. Available: https://moveit.ros.org/

[6] "Move_base - ROS Wiki." [Online]. Available: http://wiki.ros.org/move_base