UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

Dottorato di Ricerca in Tecnologie dell'Informazione

XXXVII Ciclo

Novel design and motion planning strategies for palletization tasks using collaborative robots in automated warehouses

Coordinatore:

Chiar.mo Prof. Marco Locatelli

Tutor:

Chiar.mo Prof. Jacopo Aleotti

Chiar.mo Prof. Riccardo Monica

Dottorando: *Alessio Saccuti*

Anni Accademici 2021/2022 - 2023/2024

In Robotics you have truly done your duty
if you are idle during the experiments.

(In Robotica hai fatto veramente il tuo dovere
se durante gli esperimenti ti giri i pollici.)

Dario Lodi Rizzini

Table of Contents

Abstract **1**

Introduction **3**

1 A Comparative Analysis of Collaborative Robots for Autonomous Mobile Depalletizing Tasks **9**

1.1 Related work 10
1.2 Experimental layout 11
1.2.1 Cobots and end-effector 11
1.2.2 Mounting configuration of the cobot on the mobile base . . 12
1.2.3 Environment model 14
1.3 Experimental protocol 16
1.3.1 Mobile base placement with respect to the pallet 16
1.3.2 Cobot manipulation planning pipeline 16
1.3.3 Motion Planning tools 20
1.4 Experimental evaluation 21
1.4.1 Depalletization of an entire pallet 22
1.4.2 Comparison of mobile base configuration policies 28
1.5 Discussion 30

2 PROTAMP-RRT: A Probabilistic Integrated Task and Motion Planner based on RRT **31**

2.1 Related work 33

2.2 Method . . . 35
2.2.1 Unified state space . . . 37
2.2.2 Probabilistic model . . . 38
2.2.3 Task planner . . . 41
2.2.4 Integrated task and motion planner . . . 43
2.2.5 Probabilistic completeness . . . 45
2.3 Experiments . . . 48
2.3.1 Experimental setup . . . 48
2.3.2 Manipulation Tasks . . . 48
2.3.3 Hyperparameter evaluation . . . 52
2.3.4 Experiments in finite symbolic space . . . 56
2.3.5 Experiments in unlimited symbolic space . . . 59
2.3.6 Scalability . . . 60
2.4 Discussion . . . 62

3 A Prototype Platform for Palletization and Depalletization Tasks using a Collaborative Robot 63
3.1 Mobile robot (Wheeled cart) . . . 65
3.2 Collaborative Manipulator . . . 66
3.2.1 Universal Robot UR10e, characteristics . . . 68
3.2.2 Universal Robot UR10e, supplementary hardware . . . 69
3.3 Vacuum-Based Gripping Systems . . . 70
3.3.1 Comparison of commercial vacuum grippers . . . 71
3.3.2 Design and components of the Schmalz vacuum gripper . . . 73
3.3.3 Gripper Electrical Connection Scheme . . . 75
3.4 Vision System . . . 76

4 Contact-based in-Hand Package Pose Estimation Using a Collaborative Robot 79
4.1 Related Work . . . 82
4.2 Method . . . 84
4.2.1 Induced collisions . . . 85

4.2.2 Displacement error estimation 86
4.3 Experiments 88
4.3.1 Calibration of the contact detection algorithm 91
4.3.2 Accuracy evaluation 93
4.3.3 Palletization tasks and execution time 96
4.4 Discussion 98

5 Conclusions **99**

Bibliography **103**

Abstract

This dissertation introduces novel design and motion planning strategies for collaborative robots in automated warehouses. Traditional warehouses are characterized by human-centred industrial processes where tasks are performed manually by human operators. Automated warehouses were invented to improve efficiency when advanced autonomous robot systems became available. The objective of this thesis is to investigate novel approaches for manipulation tasks in automated warehouses, such as palletization and depalletization of single packages using collaborative robots. Collaborative robots have been the focus of attention due to their flexibility and their ability to work in shared workspace with human operators. The contributions of this dissertation range from developing a prototype platform to implementing novel algorithms. Hardware devices available on the market, such as collaborative robots and vacuum grippers, are compared. In particular, collaborative robots are compared in simulated depalletization tasks, while vacuum grippers are evaluated through manual experiments conducted by a human operator. Based on the comparison results, a prototype platform for depalletization tasks was developed, featuring a collaborative manipulator mounted on a mobile platform. In addition to the prototype platform, this thesis presents two novel strategies for object manipulation: a task and motion planning algorithm that uses a probabilistic model to estimate the feasibility of pick and place actions and a procedure for estimating the displacement error of grasped boxes by inducing collisions between the grasped box and a fixed fixture in the environment.

Introduction

The transportation and stocking of goods have been significantly affected by the evolution of global commerce in recent times. In the past years traditional, non-automated warehouses, represented the standard solution for material handling.

In traditional warehouses goods are organized in pallets and palletization (and consequently depalletization) tasks are typically handled manually or adopting fork-lifts, still operated by humans.

Figure 1: Example of a traditional warehouse (left) and an automated warehouse (right).

Recently, automated warehouses are replacing traditional warehouses to overcome their limitations imposed by the transformation of global commerce (example in Figure 1). In automated warehouses palletization and depalletization tasks are not exclusively performed by human operators. Instead, robotic systems as industrial manipulators and mobile robots are adopted. In the state-of-the-art of automated ware-

houses pallets are relocated by automated guided vehicles (AGV) or laser guided vehicles (LGV). Palletization and depalletization tasks are performed in fixed manipulation stations by industrial robot manipulators with a fixed base. Pallets, loaded on AGVs and LGVs, are positioned inside manipulation areas and relocated again when the manipulation task is terminated by the manipulators.

Figure 2: Example of AGVs and LGVs fleet produced by E80 Group S.p.A. (left) and fixed manipulation stations (right).

An example of industrial company that works in the field of automated warehouses is E80 Group S.p.A., specialized in warehouse automation solutions. The core business of E80 Group S.p.A. is the production of mobile robots (AGVs and LGVs, examples in Figure 2). The company contributed to this thesis by supplying hardware devices, by offering expertise, and by proposing use cases.

A challenging task in automated warehouses is the palletization and depalletization of single objects. This operation is required in tasks such as case picking and forming mixed pallets, i.e. pallets composed of goods that can differ in shape and weight. In case picking tasks a single product unit is extracted from a storage unit, e.g. a pallet. Similarly, the formation of a mixed pallet requires individual manipulation of objects, unlike in the case of homogeneous pallets where robots usually manipulates entire layers. Single objects are typically handled by human operators or by using mobile robots together with fixed manipulation stations. The main drawback

of using fixed manipulation stations is that entire pallets are relocated from storage racks to the manipulation stations. The problem of single-item palletization and de-palletization in automated warehouses is still open and most results are prototypes. A possible alternative solution to the joint use of AGV/LGVs and fixed manipulation stations are mobile manipulators.

A mobile manipulator is a mobile robot equipped with a manipulator mounted on the top. Thanks to the mobile base a mobile manipulator can move in a warehouse similarly to a human worker and, thanks to the manipulator, it can manipulate objects. Several mobile manipulators have been documented in the literature (examples in Figure 3), however the majority of these systems are still in the prototype stage due to the various challenges they face. When developing a mobile manipulator, factors

Figure 3: Example of mobile manipulators. Kuka KMR iiwa (left) and Boston Dynamics Stretch (right).

such as the integration of multiple automation systems, the adoption of advanced algorithms for motion planning, and effective power management must be carefully considered.

Collaborative manipulators, or cobots, have gained popularity for their precision, safety, energy efficiency, reliability and ease of use. Cobots are specifically designed to operate alongside humans in shared workspaces and, unlike industrial manipula-

tors, they can be used in unstructured and non-industrial environments. They are not designed for a specific task; instead, they represent a flexible automation system that can be employed in several contexts. Examples of scenarios other than warehouses in which cobots can improve efficiency include healthcare, automotive, and agriculture. Due to their efficiency in power management and safety features, collaborative robots represent a promising solution to build mobile manipulators.

In this thesis novel design and motion planning strategies for collaborative robots in automated warehouses are presented. In particular, several topics related to automated warehouses are addressed. Hardware components such as collaborative robots and vacuum grippers are compared with the aim of designing a prototype platform for palletization and depalletization tasks. Since palletization is a non-trivial task, and the order in which packages are manipulated may influence the success of the task, an integrated task and motion planning algorithm is proposed. Moreover, to minimize placement error in palletization tasks, a strategy for estimating the displacement error of packages attached to the gripper is presented.

Contributions and thesis structure

The objective of this thesis is to propose advances in automated warehouses by presenting a collaborative prototype mobile manipulation platform and novel algorithms.

The prototypal platform design is presented, with hardware components selected and configured to solve palletization and depalletization tasks in warehouses. Hardware components have been evaluated through experimental tests and the results have been used to select the best components for the prototype. Several commercial collaborative manipulators have been compared in simulated depalletization tasks [1], while vacuum grippers have been tested through manual experiments in which a human operator had to grasp different types of objects. Although mobile manipulators are available on the market, the novel prototype has been designed for possible integration with E80 Group S.p.A. mobile robots.

Two novel problems were addressed related to object manipulation in warehouses. The first is an integrated task and motion planning algorithm that uses a probabilis-

tic model to estimate the feasibility of pick and place actions [2]. The second is a strategy to minimize the placement error of cardboard boxes by inducing collisions between the grasped object and the environment [3].

Specifically, the contributions of this thesis are presented as follows.

In Chapter 1 several collaborative manipulators are compared in simulated autonomous mobile depalletization tasks. The goal of this comparison is to evaluate the performance of mobile manipulators based on collaborative robots. Simulations are conducted in realistic scenarios in which pallets are positioned inside storage racks. The cobots are evaluated in different mounting configurations with respect to the mobile base. Three pallet sizes have been defined: a short, a medium and an high pallet. Moreover, three lateral placement policy for the mobile base with respect to the pallets have been established. Each simulated environment, characterized by a specific cobot, mount configuration, pallet size and mobile base lateral placement policy, is tested with the mobile base positioned at various distances from the pallet and the floor.

In Chapter 2, PROTAMP-RRT, a PRObabilistic integrated Task and Motion Planner based on RRT, is presented. The main contribution of this Chapter is a probabilistic feasibility model for task and motion planning, which evaluates the feasibility of symbolic actions. In PROTAMP-RRT the probabilistic model is actively used by a unified Rapidly-exploring Random Tree (RRT) approach that considers both robot configurations and symbolic information. The solution to a task and motion planning problem consists of a path comprising both abstract actions to be performed, e.g. “pick the box”, and the robot configurations that must be achieved to execute each action. The unified RRT motion planner iteratively samples robot configurations guided by the task planner, which identifies the most promising sequence of actions to perform. An experimental evaluation against previous works is performed in six simulated manipulation tasks.

The collaborative prototype platform for palletization and depalletization tasks is presented in Chapter 3. The prototype is composed of a wheeled cart, a collaborative manipulator, a vacuum gripper, and a vision system. Chapter 3 provides a detailed discussion of the characteristics and features of components available on the mar-

ket. In particular, several commercial vacuum grippers are compared through manual experiments, highlighting their advantages and disadvantages. The selected gripper, which includes an external vacuum generator and a modular gripper system, is described in detail, including its components and electrical connection with the robot. Additionally, this chapter presents a custom solution for attaching vision systems directly to the gripper. In the course of this thesis the prototype is used to evaluate a contact-based in-hand package pose estimation strategy which is discussed in Chapter 4.

In Chapter 4 a contact-based in-hand package pose estimation strategy is proposed. The objective of the proposed strategy is to minimize the placement error of cardboard boxes in manipulation tasks estimating the displacement error, i.e. a relative rigid transformation between the actual and the expected pose of a grasped object. The robot induces collisions between the grasped object and a fixture whose position is known with respect to the manipulator. For each induced collision a $2D$ collision point is saved. Given one or more collision points a non-linear least squares optimization problem is solved to estimate the displacement error. The approach is evaluated with as few as one contact, to minimize cycle time, and a minimal footprint fixture that has been manufactured as a small cylinder.

Finally, in Chapter 5, conclusions and future works are proposed.

Chapter 1

A Comparative Analysis of Collaborative Robots for Autonomous Mobile Depalletizing Tasks

The level of automation in industrial environments, like warehouses, is constantly increasing. A typical manipulation task in automated warehouses is depalletization, i.e. the unloading of pallets that contain overlapping boxes organized in multiple layers. A standard solution for depalletization is to adopt fixed base industrial robots in combination with Automated guided vehicles (AGVs) that perform transportation as well as pick-up and drop-off of entire pallets from storage racks. However, these solutions are inefficient if a single object needs to be picked up from a pallet, as the entire pallet must be moved from its storage location to a robotic cell station. Moreover, unloading stations often require protective fences and, therefore, they do not allow human interaction. Palletization and depalletization tasks of mixed pallets is another example where packages must be manipulated individually.

In order to increase productivity and safety collaborative robots (cobots) are emerging as a new technology in production and manufacturing industries. In particu-

lar, mobile manipulators that consist of a mobile base and a collaborative robot [4, 5], may be a good option in a wide range of industrial applications.

The work presented in this Chapter aims to perform a comparative analysis of several collaborative robots in depalletization tasks, under different mounting configurations for the cobot with respect to the mobile base, and under different distances of the mobile base to the pallet. The purpose of the cobot comparison was to select a commercial manipulator for the prototype presented in Chapter 3. In particular, a realistic scenario was developed with tight spaces due to the presence of storage racks. Different pallet sizes were also considered. The experimental evaluation was carried out in a simulated environment by exploiting a geometric path planner. Some parameters, such as the shape and the size of the mobile base, were kept fixed during the experiments to limit the number of simulated scenarios to be evaluated.

1.1 Related work

Mobile manipulators are being increasingly adopted in industrial environments due to their flexibility and the fact that there is a growing demand of mobile manipulation solutions for collaborative tasks. However, comparative analyses are not available in the literature. In [6] a method was presented to compare fixed-base collaborative robots in terms robot parameters. An overview about the development of collaborative mobile robots in industrial scenarios was presented in [7], where a general system architecture and potential applications were also described. Vitolo et al. [8] discussed the existing challenges about integration and safety issues for mobile manipulators motivated by the lack of guidelines when combining cobots and mobile robots from different manufacturers. Example prototypes of collaborative robots used for mobile manipulation tasks were reported in [9, 10]. In particular, a mobile manipulator was proposed in [9] to fully automate the task of order-picking in industrial shop-floor, whereas in [10] an autonomous depalletizing mobile robot system was developed capable of pulling cardboard boxes from a pallet onto the mobile base for transportation.

Several studies have been carried out in the context of reachability analysis and

motion planning for mobile manipulators. Thakar et al. [11] introduced a motion planning method for picking-up objects with the robot arm while the mobile base moves to increase the efficiency of operations. In [12] an approach was investigated for a mobile manipulator to determine the minimum sequence of base positions to pick objects from multiple trays. In [13] the first mobile manipulation system using deep reinforcement learning in combination with visual perception was presented. In [14] an open source software library was proposed for robot reachability analysis and computation of optimal base placement. Di Marco et al. [15] investigated the problem of base pose estimation in distributed robot systems for cooperative tasks using shared reachability maps.

1.2 Experimental layout

In order to compare different collaborative manipulators a simulated 3D environment of an industrial warehouse was set up. Moreover, five collaborative robot arms available on the market were considered, as well as three configurations of the mobile base and three types of storage racks.

1.2.1 Cobots and end-effector

Given the heterogeneity of depalletization tasks, both in terms of pallet size and shape of the storage racks, the choice of the most appropriate cobot and its configuration on a mobile base is not trivial. Also, several cobots are available on the market that differ in terms of size, payload and number of degrees of freedom. In the proposed comparative analysis 6-axis collaborative manipulators were considered, with a payload of at least 10 Kg. A 3D representation of the cobots that were considered is shown in Figure 1.1, and the list of their parameters is reported in Table 1.1.

Experiments have been carried out using a simulated vacuum gripper of $0.1 \times 0.1 \times 0.11\,m$ size in all cases, assuming an end-effector able to perform grasping by contact with the object, disregarding dynamic effects.

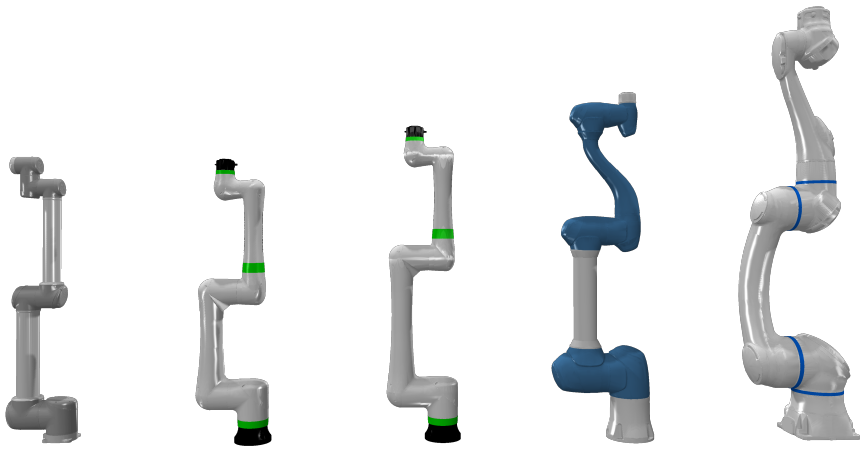


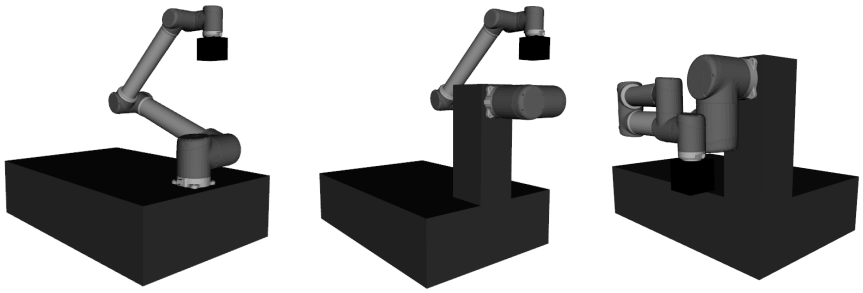
Figure 1.1: 3D models of the collaborative manipulators, from the left: Universal robots UR10, Fanuc CRX-10iA, Fanuc CRX-10iA/L, Doosan H2515, Yaskawa HC20DT.

Table 1.1: Robot parameters

Robot	**Payload (kg)**	**Reach (mm)**	**Weight (kg)**
Universal Robots UR10	10	1300	28.9
Fanuc CRX-10iA	10	1249	40
Fanuc CRX-10iA/L	10	1418	40
Doosan H2515	25	1500	72
Yaskawa HC20DT	20	1700	140

1.2.2 Mounting configuration of the cobot on the mobile base

Three different mobile bases have been simulated as shown in Figure 1.2. All the 3D models consist of a parallelepiped of $1.2 \times 0.8 \times 0.3$ m shape. The first mobile base, Figure 1.2 (a), consists of the base only, and it was used with horizontally mounted cobots (*HC*). The second mobile base, Figure 1.2 (b), includes a centered vertical



(a) (b) (c)

Figure 1.2: Mounting configurations of the cobot on the mobile base: HC (a), VC (b), VRC (c).

extension, and the cobot was mounted vertically on the front side of the extension (*VC*). The third mobile base, Figure 1.2 (c), includes a vertical extension shifted to the right, and the cobot was mounted vertically on the right side of the extension (*VRC*). The horizontal cobot base position was set by adding a longitudinal (Δ_{lon}) and lateral (Δ_{lat}) offset with respect to the mobile base center. In the HC configuration the cobots were mounted along the longitudinal axis with a value Δ_{lon} so that the cobot base is at 5 cm to the mobile base frontal edge. In the VC configuration the cobots were mounted along the longitudinal axis with $\Delta_{lon} = 0.6$ m, while in the VRC configuration the cobots were mounted at $\Delta_{lat} = 0.4$ m and $\Delta_{lon} = 0.5$ m. All the cobots have been evaluated in the three configurations, with the exception of the H2515 and the HC20DT cobots, as the former does not support a vertical mount, and the latter incurs in joint range limitations. Therefore, a total of eleven mobile robots were simulated. Moreover, the height h of the top surface of the mobile base (Figure 1.3) in the simulations with respect to the floor was variable in the range $[0, 0.8]$ m, with discretized steps of 0.1 m. In case HC, parameter h corresponds to the actual height of the cobot base with respect to the floor. Instead, in cases VC and VRC, the actual height of the cobot base is given by h plus the height of the vertical extension (55 cm), minus half of the cobot base width.

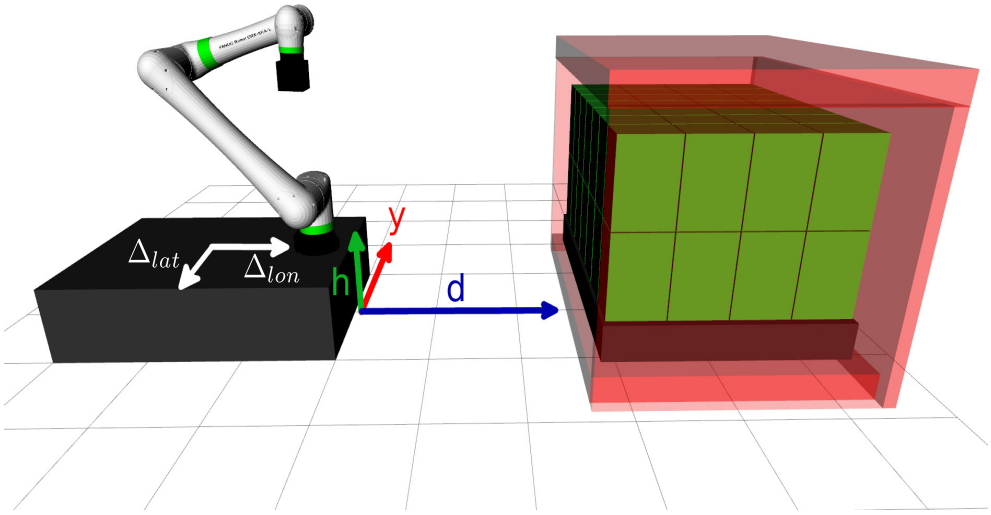


Figure 1.3: Parameters h, y and d used to define the mobile robot configuration.

1.2.3 Environment model

The simulated environment consists of a rack occupied by a fully loaded pallet. All simulated pallets have the same horizontal section, according to the *EPAL-3* standard ($1.2 \times 1 \times 0.14$ m), and they contain boxes of the same size ($0.24 \times 0.16 \times 0.32$ m). Pallets are organized in layers of boxes with a 4 by 7 configuration, with a small gap between adjacent boxes. An example pallet is shown in Figure 1.4. Three types of storage racks (Figure 1.5) have been considered which differ in the number of pallet layers: a *low pallet* with 2 layers, a *medium pallet* with 4 layers, and a *high pallet* with 6 layers. Each storage rack leaves a small amount of free space on both sides of the pallet δ=0.12 m, as well as a small amount of free space at the top of the pallet (γ=0.1 m). Therefore, the depalletization tasks must be carried out in a tight space.

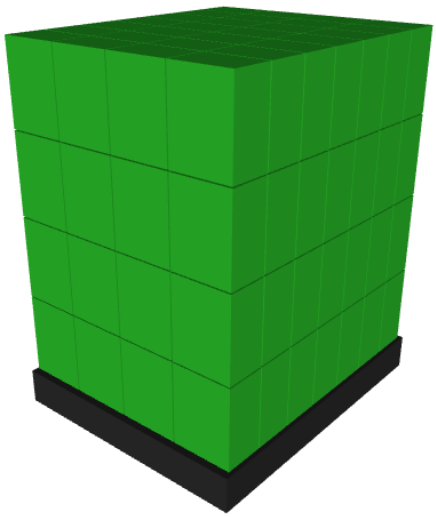


Figure 1.4: Example of an industrial pallet with four layers of boxes.

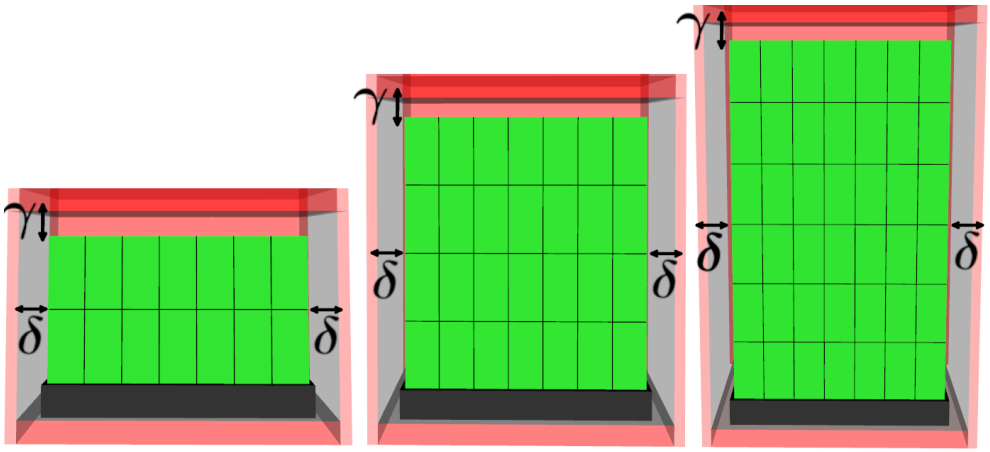


Figure 1.5: Simulated storage racks: *low pallet*, *medium pallet* and *high pallet* (from left to right).

1.3 Experimental protocol

Given a cobot type, a mounting configuration of the cobot on the mobile base (as described in Section 1.2.2), and a pallet type (as defined in Section 1.2.3), a depalletization task is defined as the process of unloading all the packages from the pallet, one after the other. The following Sections describe the motion planning procedure with more details.

1.3.1 Mobile base placement with respect to the pallet

In this Chapter the problem of planning the motion of the mobile base was not considered, that is, in each pick and place task of a package the mobile base was relocated/teleported in a fixed pose and motion planning was limited to generating motions of the cobot. Multiple poses of the mobile base with respect to the pallet have been tested, all with the same orientation parallel to the pallet. In particular, the position of the mobile base was generated by changing the lateral offset (y), and the distance to the pallet (d) (Figure 1.3). The distance to the pallet d was tested in the range $[0.1, 1.1]$ m with discretized steps of 0.1 m. Three policies were considered to evaluate the lateral offset y, as shown in Figure 1.6. In the case of the two symmetrical mobile base configurations HC and VC it was considered a policy where the mobile base is fixed and centered on the pallet (*CP-fixed*) for all the pick and place operations, and another policy where the mobile base is relocated at each pick and place task so that the initial cobot end-effector configuration is centered on the target package (*CP-centered*). In the case of the VRC mobile base configuration, which is asymmetrical, a fixed lateral shift of 0.6 m was applied to the mobile base on the left side with respect to the center of the pallet (*CP-shifted* policy).

1.3.2 Cobot manipulation planning pipeline

The flowchart of the depalletization task is reported in Figure 1.7. Each depalletization task is an iterative process. At each iteration a target package is selected to be unloaded from the pallet, and then a path is planned for the cobot to pick up the pack-

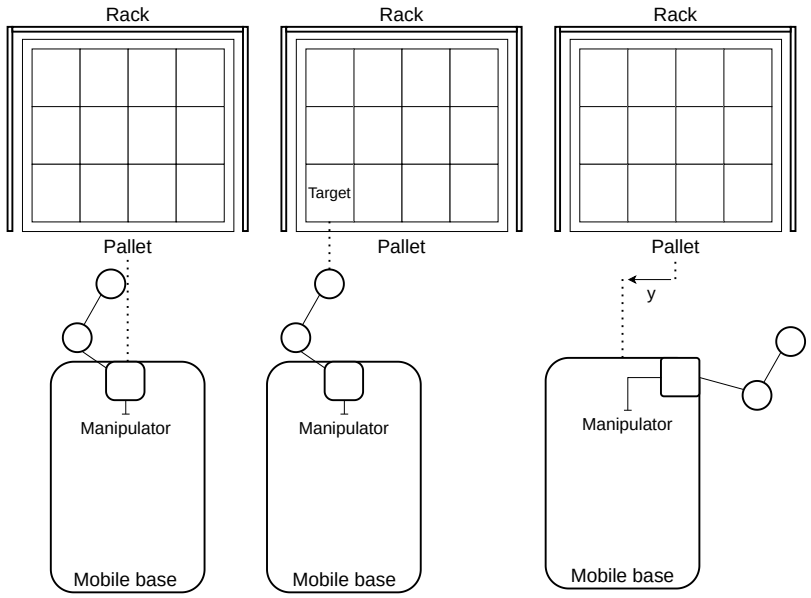


Figure 1.6: Configuration policies for the mobile base lateral placement with respect to the pallet, from the left: *CP-fixed*, *CP-centered*, *CP-shifted*.

age and place it in a deposit configuration on the right side of the mobile base. Boxes can be picked up from different sides, but priority was given to picking up them from the top side. In particular, at each iteration the following steps are performed:

1. The first step selects the box to be picked up, i.e. the closest box to the robot (in the initial configuration) from the set of available boxes. If no boxes are available for picking, the simulation terminates.

2. The second step relocates the mobile base in the environment, according to the mobile base placement policy and the selected values of h and d.

3. Then, the planner attempts to find a motion plan to the grasping configuration where the end-effector is placed against the top surface of the box, and then a motion plan to unload the box and place it on the deposit configuration. If successful, the procedure proceeds to step 5.

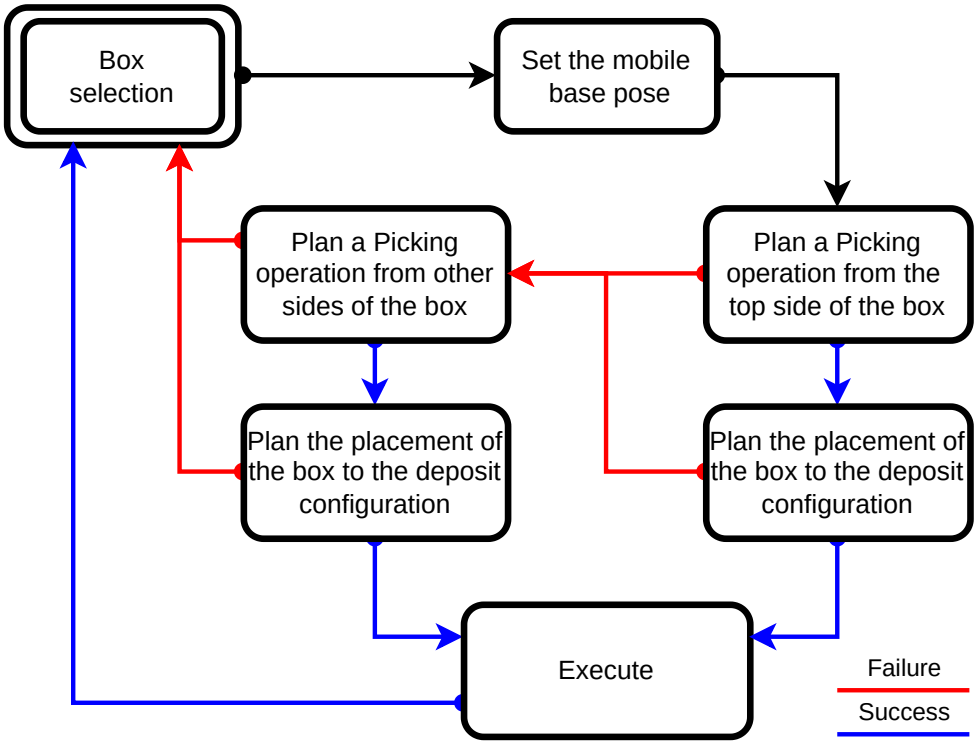


Figure 1.7: Flowchart of the simulated depalletizzation task.

4. Otherwise, the planner attempts to find a motion plan to other potential grasping configurations, where the end-effector is placed against other sides of the box (front, left or right side), and then a motion plan to unload the box and place it on the deposit configuration. If planning fails the box is removed from the set of available boxes, and a different box is selected in step 1.

5. The planned motion plan are executed, and the box is removed from the simulation. Since after the removal of a package from the 3D environment other packages may become reachable, the set of available boxes for grasping is reset to contain all the remaining boxes in the simulation. Then, the simulation proceeds to the next iteration (step 1).

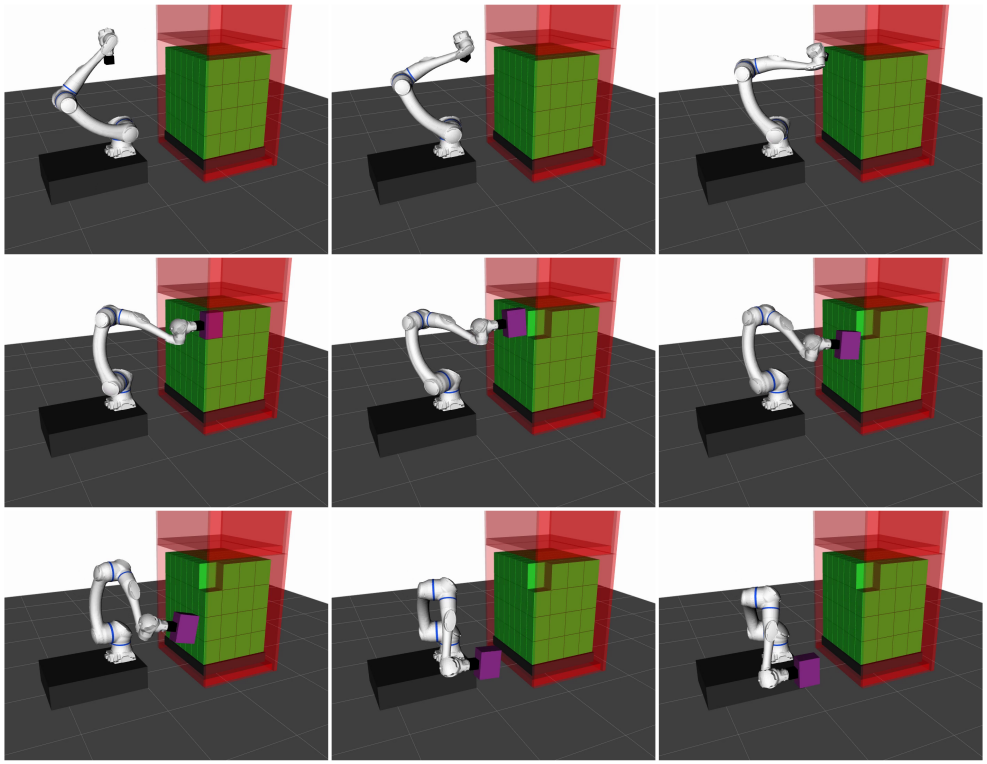


Figure 1.8: Example of a successful package unloading operation (HC robot configuration, HC20DT cobot, *CP-fixed* policy, medium pallet type).

When planning the path to pick up a package the end-effector orientation is unconstrained. Instead, when placing an object on the mobile base, the motion of the cobot is planned by applying a constraint on the orientation of the grasped package so that it remains horizontal, to simulate a safe motion that does not damage the content of the package. Figures 1.8, 1.9 and the video available at `http://rimlab.ce.unipr.it/%7ermonica/saccuti/IRC%5F2022.mp4` show examples of the unloading task.

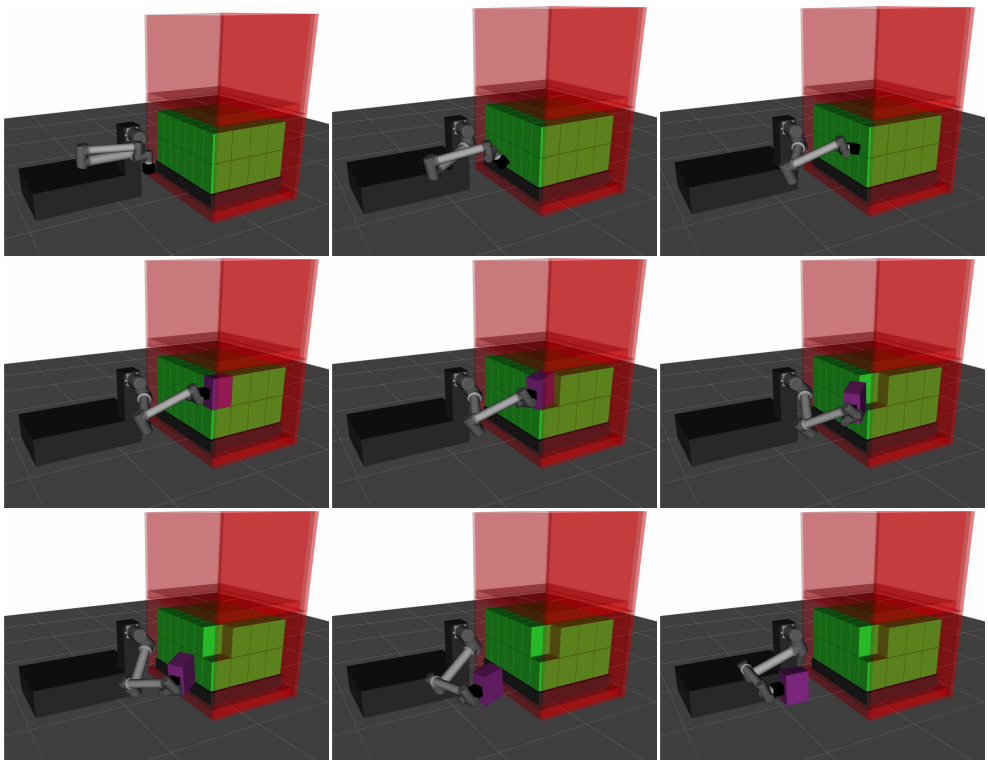


Figure 1.9: Example of a successful package unloading operation (VRC robot configuration, UR10 cobot, low pallet type).

1.3.3 Motion Planning tools

The simulated experimental environment was implemented on top of ROS (Robot Operating System) using the MoveIt! planning framework. In particular, the OMPL planning library [16] was adopted together with the RRT-connect algorithm [17]. The following features of MoveIt! have been used:

- **Inverse kinematics**. MoveIt! offers integrated numerical inverse kinematic

plugins, such as KDL, that can be extended with custom modules. In particular, the trac-ik inverse kinematic plugin [18] was adopted.

- **Collision detection** MoveIt! performs collision detection based on the Flexible Collision Library [19] by default. As traditional collision checking algorithms do, MoveIt! performs a broad and narrow collision checking phase. In the broad phase it uses axis aligned bounding boxes, while in the narrow phase it uses collision shapes pre-defined by the user. The collision shapes need to be defined in a robot description file, for this comparative analysis the convex hull shapes were used to describe the geometry of the robot links.

- **Path constraint**. MoveIt! allows setting constraints such as end-effector orientation. A planner can extract valid constrained states in the Cartesian space or in the joint space. The first approach may lead to trajectory discontinuities while the second can be too slow. To overcome these problems a solution proposed by [20] was applied that pre-computes valid constrained states in a database.

1.4 Experimental evaluation

The experimental evaluation has been carried out on an Intel i9 10900 processor @ 2.80 GHz (32 GB of RAM, Ubuntu 20.04 LTS). In particular, each task was executed given a pallet size (*low*, *medium*, or *high*), a cobot (UR10, CRX-10iA, CRX-10iA/L, H2515, or HC20DT), a cobot mounting configuration (HC, VC, or VRC), a mobile base lateral placement policy (*CP-fixed*, *CP-centered*, or *CP-shifted*), a height of the mobile base top surface ($h \in [0, 0.8]$), and a distance to the pallet ($d \in [0.1, 1.1]$). A total of 57 3D simulation environments have been tested by combining the 5 cobots, 3 mobile bases, 3 lateral placement policies and 3 pallet types. All simulation environments were tested for 99 possible pairs of values (h, d), according to the resolution of parameters h and d.

1.4.1 Depalletization of an entire pallet

In this section the robots are compared to determine which combination of cobot, mounting configuration and lateral placement policy, is the most appropriate for a given pallet size. In particular, for each 3D environment (i.e., pallet size, cobot, mounting configuration and lateral placement policy) and for each value of h and d, a motion planning experiment was executed for the task of unloading an entire pallet. Three scores have been defined to compare the performance of the robots for each 3D environment:

- **Success rate**: number of pairs (h,d) when the robot was able to unload all the packages from the pallet.
- **% top**: percentage of packages that were successfully unloaded by grasping them from the top side.
- **Max** h: maximum value of parameter h when a successful depalletization occurred.

The results are shown in Tables 1.2, 1.3 and 1.4. Each row shows the three scores, given the **Cobot**, the lateral placement policy (**P. policy**), and the pallet size (**P. size**). Table 1.2 refers to the horizontal mounting configuration (HC), Table 1.3 refers to the vertical mount configuration (VC), and Table 1.4 refers to the vertical mounting configuration on the side of the mobile base (VRC). The H2515 and the HC20DT cobots are not present in the latter two tables, as they were not tested in the vertical mount configuration.

In general, the *Success rate* is not high, i.e., only a subset of the 99 pairs (h,d) results in a successful unloading of an entire pallet. This confirms that the robot configuration must be carefully designed and, therefore, a motion planning simulation of the task is advisable.

In Table 1.2 it can be noticed that the HC20DT cobot managed to depalletize medium and high pallets in many cases, due to its large size. However, the HC20DT cobot managed to depalletize a low pallet in one case only, as low pallets reduce the number of collision-free configurations.

Table 1.2: Robot comparison, HC configuration.

Cobot	P. policy	P. size	Succ.	% top	Max h
UR10	*CP-fixed*	low	1	37	0.2 m
		medium	6	53	0.8 m
		high	0	0	-
	CP-centered	low	1	37	0.2 m
		medium	9	56	0.8 m
		high	0	0	-
CRX-10iA	*CP-fixed*	low	1	17	0.2 m
		medium	4	49	0.8 m
		high	0	0	-
	CP-centered	low	7	10	0.3 m
		medium	9	48	0.8 m
		high	0	0	-
CRX-10iA/L	*CP-fixed*	low	1	33	0 m
		medium	10	55	0.7 m
		high	0	0	-
	CP-centered	low	6	25	0.3 m
		medium	15	54	0.7 m
		high	1	68	0.8 m
H2515	*CP-fixed*	low	0	0	-
		medium	14	34	0.6 m
		high	3	45	0.8 m
	CP-centered	low	2	20	0.1 m
		medium	12	47	0.6 m
		high	1	59	0.8 m
HC20DT	*CP-fixed*	low	1	0	0 m
		medium	43	30	0.7 m
		high	28	49	0.8 m
	CP-centered	low	1	1	0 m
		medium	40	34	0.7 m
		high	24	54	0.8 m

Table 1.3: Robot comparison, VC configuration.

Cobot	P. policy	P. size	Succ.	% top	Max *h*
UR10	*CP-fixed*	low	2	40	0.1 m
		medium	9	60	0.6 m
		high	1	70	0.7 m
	CP-centered	low	0	0	-
		medium	7	63	0.2 m
		high	2	75	0.7 m
CRX-10iA	*CP-fixed*	low	0	0	-
		medium	2	31	0.4 m
		high	0	0	-
	CP-centered	low	1	8	0 m
		medium	1	25	0.1 m
		high	0	0	-
CRX-10iA/L	*CP-fixed*	low	0	0	-
		medium	2	35	0.4 m
		high	0	0	-
	CP-centered	low	1	10	0 m
		medium	1	26	0.1 m
		high	0	0	-

Table 1.4: Robot comparison, VRC configuration.

Cobot	P. policy	P. size	Succ.	% top	Max *h*
UR10	*CP-shifted*	low	7	21	0.5 m
		medium	4	51	0.5 m
		high	0	0	-
CRX-10iA	*CP-shifted*	low	2	36	0.5 m
		medium	4	35	0.5 m
		high	0	0	-
CRX-10iA/L	*CP-shifted*	low	0	0	-
		medium	8	52	0.5 m
		high	1	56	0.7 m

Table 1.5: Average planning time and standard deviation (s) for box extraction planning

Cobot	**M. type**	**P. policy**	**Pallet size**		
			Low	Medium	High
UR10	HC	*CP-fixed*	3.1±0.0	2.1±0.3	-
		CP-centered	3.1±0.0	2.1±0.5	-
	VC	*CP-fixed*	5.8±0.5	3.4±0.7	3.6±0.0
		CP-centered	-	3.5±0.8	3.5±0.3
	VRC	*CP-shifted*	5.8±1.1	4.1±0.4	-
CRX--10iA	HC	*CP-fixed*	2.5±0.0	2.9±1.1	-
		CP-centered	1.5±0.4	1.2±0.3	-
	VC	*CP-fixed*	-	4.3±0.7	-
		CP-centered	2.4±0.0	3.5±0.0	-
	VRC	*CP-shifted*	5.1±0.1	3.3±0.4	-
CRX--10iA\L	HC	*CP-fixed*	1.2±0.0	1.9±0.7	-
		CP-centered	1.7±0.7	1.2±0.3	1.4±0.0
	VC	*CP-fixed*	-	4.7±1.1	-
		CP-centered	2.0±0.0	3.9±0.0	-
	VRC	*CP-shifted*	-	3.1±0.5	2.8±0.0
H2515	HC	*CP-fixed*	-	1.9±0.7	1.7±0.2
		CP-centered	4.4±0.5	1.9±0.7	1.1±0.0
HC20DT	HC	*CP-fixed*	5.3±0.0	3.1±1.2	2.4±0.4
		CP-centered	5.5±0.0	2.9±1.1	1.8±0.4

Moreover, the maximum value of parameter h was 0 m, i.e., the HC20DT cobot should be mounted on a mobile base at floor level (**Max** $h = 0$), which would be almost unfeasible. Similarly, the percentage of boxes extracted by grasping them from the top side (**% top**) is lower for large cobots, such as the H2515 and the HC20DT, than for the smaller robots, as large cobots can not easily insert the end-effector above the pallet.

When using small cobots such as UR10, CRX-10iA, and CRX-10iA/L, the *CP-centered* policy achieved better results than the *CP-fixed* policy. In particular, when

using the *CP-centered* policy the CRX-10iA/L cobot was able to depalletize all pallet types from at least one pose of the mobile base, with an acceptable mobile base height.

The VC mounting configuration (Table 1.3) facilitates smallest robots (UR10 and CRX-10iA) to depalletize high pallets, compared to a more standard HC mount. In particular, using the *CP-fixed* policy and the VC mounting configuration the UR10 cobot was able to successfully depalletize all three pallet types from at least one pose of the mobile base. Conversely, the VRC mounting configuration (Table 1.4) facilitates the depalletization capabilities of the smallest robots in the case of short pallets.

The average planning time and the standard deviation are reported in Table 1.5, for each simulated environment (**Cobot**, the mount type **M. type**, the lateral placement policy **P. policy**, and the pallet size). The analysis of the computational time required for motion planning shows that the planner has more difficulties in finding a solution for the extraction of a grasped package in the VC and VRC mounting configurations. In particular, the UR10, CRX-10iA, and CRX-10iA/L cobots required more planning time in the VC and VRC mounting configurations than in the HC mounting configuration. Indeed, the UR10 cobot in the HC mounting configuration took about 2.1 seconds to plan the extraction of a package from a medium pallet, for both for *CP-fixed*, and *CP-centered* policies. In the VC mounting configuration the required planning time increased to 3.4 seconds (*CP-fixed* policy) and to 3.5 seconds (*CP-centered* policy), while for the VRC mounting configuration the UR10 cobot took about 4.1 seconds.

As in some applications a partial depalletization of a pallet may also be acceptable, in Figures 1.10, 1.11 and 1.12 the Success rate when the robots managed to unload at least 75% of the packages in a pallet was reported, as well as the Success rate in the case of a complete depalletization (100% label). It can be seen that the number of successes is higher when a partial depalletization is allowed, and that most robots managed to depalletize 75% of the packages from at least one configuration, with the exception of the UR10 and the CRX-10iA cobots with HC mount type. The results confirm that larger cobots are more suitable to handle larger pallets, while

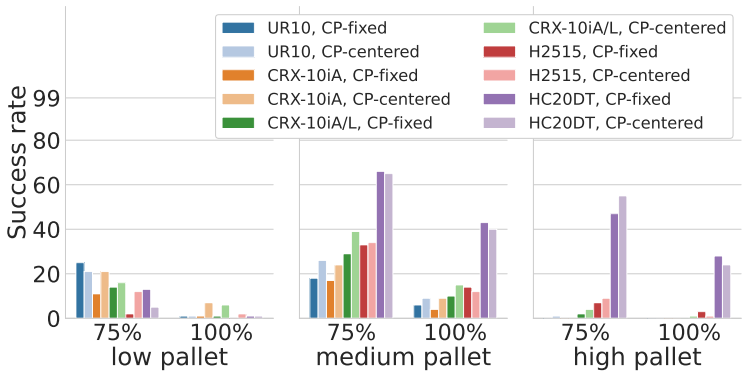


Figure 1.10: Comparison of partial (75%) and full (100%) depalletization tasks, HC configuration.

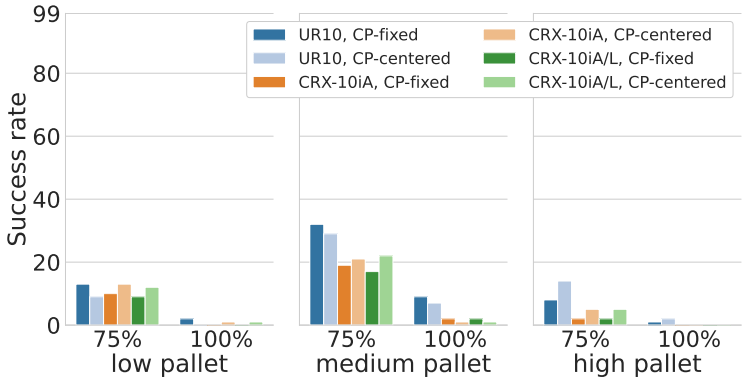


Figure 1.11: Comparison of partial (75%) and full (100%) depalletization tasks, VC configuration.

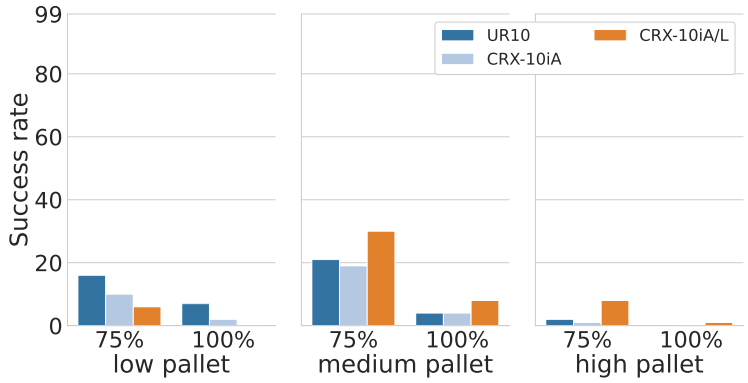


Figure 1.12: Comparison of partial (75%) and full (100%) depalletization tasks, VRC configuration.

smaller cobots are better suited for smaller pallets.

1.4.2 Comparison of mobile base configuration policies

In this section, the effect of the mobile base configuration policy (*CP-fixed* or *CP-centered*) on the robot depalletization capabilities is further analyzed. In particular, the number of boxes that were successfully extracted in each simulation are reported as 2D heat maps for the medium (Fig. 1.13) and the low (Fig. 1.14) pallet size, using the UR10, CRX-10iA, CRX-10iA/L, H2515, HC20DT cobots in the HC configuration, with both the CP-fixed policy and the CP-centered policy. Each cell of the heat map is colored according to the number of extracted boxes, given a mobile base top surface height (h), and the distance of the mobile base to the pallet (d). A more yellow hue means that more packages have been extracted from the pallet in a certain configuration. Thus, the yellow region represents the pairs of values (h,d) where the robot should be placed to depalletize the whole pallet.

When using the medium pallets (Fig. 1.13), the yellow region of the small cobots is higher and more to the right than for the other cobots, as smaller cobots must be placed higher and closer to the pallet to manipulate the packages.

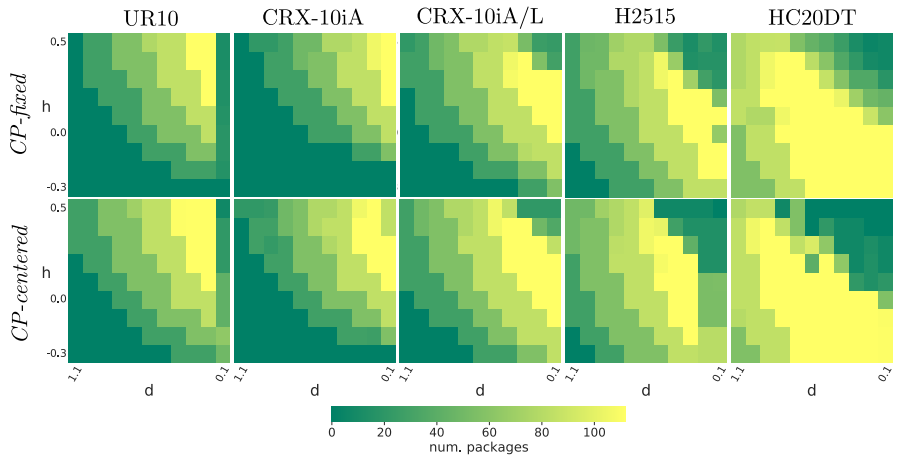


Figure 1.13: UR10, CRX-10iA, CRX-10iA/L, H2515, HC20DT, mobile base configuration policy comparison, medium pallet size.

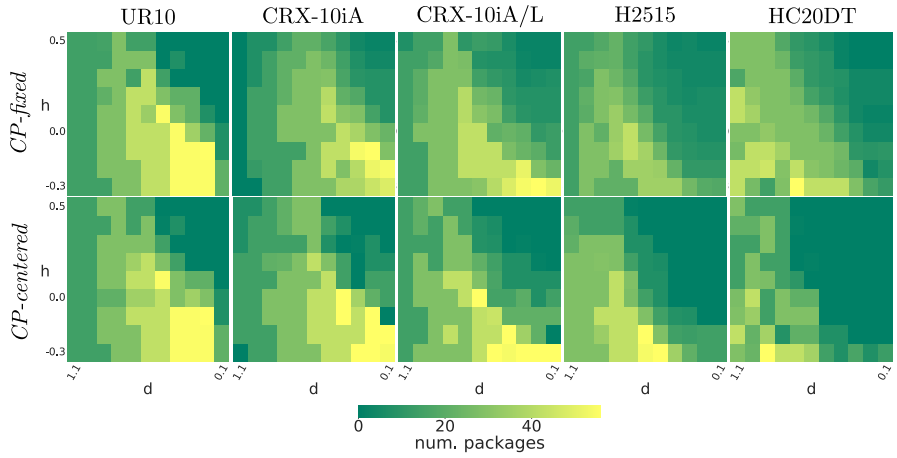


Figure 1.14: UR10, CRX-10iA, CRX-10iA/L, H2515, HC20DT, mobile base configuration policy comparison, low pallet size.

Moreover, the larger the yellow region, the easier the set up of the mobile robot will be. The size of the yellow region increases with the size of the cobot, and reaches its maximum for the HC20DT cobot. Also, the *CP-centered* policy has produced bet-

ter results than the *CP-fixed* for the UR10, CRX-10iA, and the CRX-10iA/L cobots, as the yellow region is larger. Conversely, no such improvement is evident for the larger robots H2515 and HC20DT.

When using the low pallets (Fig. 1.14) small cobots must be placed at low height and closer to the pallet. Instead, large cobots had more difficulties in planning the extraction of packages as the size of the yellow region decreases with the size of the cobot, and it is non-existent for the H2515 and HC20DT cobots.

1.5 Discussion

In this chapter of the thesis a comparison of several collaborative manipulators on a mobile base was performed for depalletization tasks in a simulated environment. Results indicate that the robot configuration and the pallet type strongly affect the performance of the robot. Therefore, a motion planning simulation of the task is advisable to design the robot platform. In particular, the experimental evaluation suggests that centering the mobile base to the target package to be picked up can improve the success rate of the depalletization task. In the case of horizontally mounted cobots, the CRX-10iA/L obtained the best results, while the UR10 cobot achieved the best results in the vertical mount configurations.

Chapter 2

PROTAMP-RRT: A Probabilistic Integrated Task and Motion Planner based on RRT

Planning sequential robot manipulation tasks depends on both logical relations and geometric constraints. Task And Motion Planning (TAMP) approaches combine a task planner, which searches for a valid sequence of high-level symbolic actions (task plan), and a motion planner, which searches for a collision-free path to reach the goal [21]. The symbolic planner uses logical reasoning to determine a sequence of symbolic actions to reach the goal that are exploited to guide the motion planner. On the other hand, the motion planner may discover that some symbolic actions are not feasible due to robot kinematics constraints or collisions with obstacles and, therefore, the exploration of the configuration space can, in turn, affect the symbolic planner. As part of this thesis, a Probabilistic Integrated Task and Motion Planner (PROTAMP-RRT) is proposed, based on a unified Rapidly-exploring Random Tree which operates on both the geometric space and the symbolic space. The unified RRT algorithm, like a standard RRT motion planner [22], samples robot configurations, and it connects them in a tree data structure. However, as the distance between two configurations can not be easily defined if they refer to different high-level symbolic

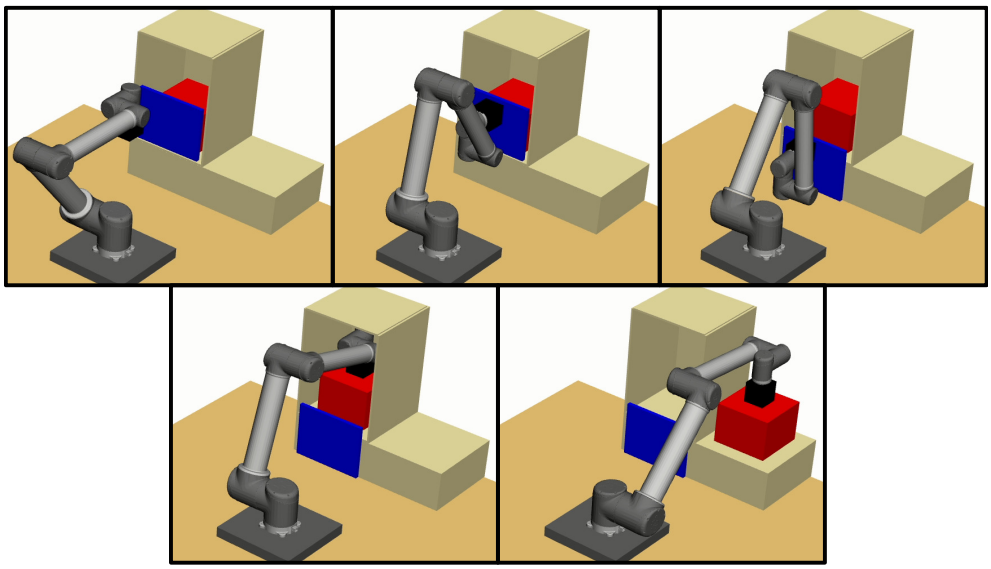


Figure 2.1: A complex manipulation task where the red box must be extracted from a cupboard. Before the extraction, the blue panel must be relocated.

states, a probabilistic model is proposed to enhance the RRT, which is exploited by the task planner to generate a sequence of symbolic actions to reach the goal of the task. Each symbolic action defines a sub-goal configuration of the robot. The RRT expansion is guided towards the sub-goal configuration of the first action in the task plan for which a path has not yet been found. The main contribution is the elaboration of the probabilistic model that evaluates the feasibility of the task plan, by estimating the probability that new collision-free configurations will be sampled. The probabilistic model is updated in the expansion phase of the RRT, and it is used to generate a new task plan if the feasibility of the previous one is unlikely. The probabilistic model takes into account probabilities conditioned on the pose of movable objects, so that the task planner can prevent unfavorable object configurations. PROTAMP-RRT de-

fines symbolic actions using significant object poses, which are geometric poses of movable objects that are also represented as symbols. In order to limit computational load, PROTAMP-RRT starts with few significant object poses, but it can also add more of them if necessary. To do so, PROTAMP-RRT relies on sample generators which are able to extract new significant object poses, and the corresponding robot configurations to grasp and release objects in such poses. For example, if an object must be picked-up from a cupboard and an obstacle obstructs the extraction (Fig. 2.1) new significant object poses are generated to relocate the obstacle first. The evaluation of PROTAMP-RRT was carried out on simulated pick-and-place tasks. Results indicate that PROTAMP-RRT outperforms two state-of-the art TAMP approaches: TM-RRT [23] and Planet [24]. In summary, the contributions presented in this Chapter are: 1) a probabilistic feasibility model for TAMP, 2) a unified RRT approach that takes advantage of the probabilistic model and 3) the experimental evaluation against previous works.

A task and motion planning strategy like PROTAMP-RRT can be adopted for solving complex palletization tasks where standard motion planning techniques may result inefficient. For example when mixed pallets must be formed, i.e. pallets that contain multiple types of packages with different sizes, an approach like PROTAMP-RRT could be beneficial to find the sequence of packages to be manipulated.

2.1 Related work

In most TAMP approaches, the task planner generates symbolic plans that are validated using a motion planner. Backtracking occurs when the motion planner fails, and new symbolic plans are generated. In the Task-Motion Kit [25] new symbolic plans are iteratively deepened if the motion planner fails. In [26], sampled symbolic actions are validated by a RRT-Connect motion planner, until the task planner can build a complete symbolic plan. PDDLStream [27] by Garrett et al. introduces *streams*, i.e. external “black-box” algorithms which add new symbols to the symbolic space. A motion planner is a stream which generates trajectories for abstract actions. These approaches suffer from the fact that sampling-based motion planners, like RRT, may

never terminate if the motion planning problem is not feasible. Therefore, these methods either use a simpler motion planner, or determine failures using a timeout, which depends on the complexity of the task.

The approach in this Chapter of sampling new significant object poses is most similar to PDDLStream [27], where the pose generator is also a stream. Refinement functions in Task-Motion Kit [25] are a related concept, as they convert an abstract action into a geometric goal. Also semantic attachments [28] can produce new values for numeric fluents when actions are evaluated. The difference in the proposed approach is that new significant object poses are sampled independently of existing actions.

As TAMP is computationally intensive, several methods have been proposed to guide task planning without using feedback from motion planning like PROTAMP-RRT. Heuristics have been proposed to estimate the action feasibility in the task planner, using a database [29], a SVM classifier [30] and reinforcement learning [31]. In order to help the task planner with geometric information, scene graphs [32] and neural networks [33] have also been proposed. Xu et al. [34] use deep learning to predict future long-term affordances in the task planner. A few works [35] [36] focused on transferring learned feasibility constraints from previous executions.

Similarly to the work in this Chapter, a few approaches have used a unified RRT in combined symbolic and geometric space. The main challenge is guiding the RRT towards the goal, which requires long-horizon planning. In TM-RRT [23] by Caccavale and Finzi, RRT is guided by a distance function that accounts for the cardinality of the difference of two symbolic states. In Planet [24] a unified RRT is guided using a heuristic which takes into account failures due to collisions, but unlike in the proposed PROTAMP-RRT the heuristic does not consider the pose of the movable objects. Another work [37] extracts rich information from the geometric planner as it identifies “culprits”, i.e., problematic sequences of actions. However, [37] operates on a discretized environment, and it defines culprits as logical statements. Similarly, [38] defines geometric constraints in a quantized representation of the parameters.

Some TAMP approaches focused on object re-arrangement in order to move or remove a single target object [39] [40] [41]. These methods can make additional

assumptions, such as heuristics based on regions [40], and removal of objects to reduce constraints [41]. While this thesis aims at finding the first geometrically feasible solution, other works [42] [43] [44] focus on finding an optimal trajectory, e.g. in terms of robot execution time, by applying complex optimization approaches. The work in [45] deals with advanced sampling approaches for geometric locations where symbolic transitions may take place. In [46] general sampling-based methods are proposed for TAMP, but unlike PROTAMP-RRT it focuses on sampling with dimensionality-reducing constraints.

2.2 Method

A planning problem is defined as the tuple $\langle \mathscr{E}, \mathscr{O}, \mathscr{S}, \mathscr{C}, \mathscr{M}, \mathscr{A}, \mathscr{R}, \mathscr{L}, \mathscr{G}, q_0, s_0 \rangle$, where $\mathscr{E}$ is the static environment, $\mathscr{O}$ is the set of manipulable objects, $\mathscr{S}$ is the symbolic state space of object predicates, $\mathscr{C} \subseteq \mathbb{R}^{\text{robot DoF}}$ is the robot configuration space, $\mathscr{M}$ are all possible *motions* connecting two robot configurations $q_1, q_2 \in \mathscr{C}$, $\mathscr{A}$ is the symbolic action space, $\mathscr{L}$ is a set of significant object poses, $\mathscr{R}$ is a set of regions where poses in $\mathscr{L}$ can be sampled, $\mathscr{G}$ is the task goal, $q_0 \in \mathscr{C}$ is the initial robot configuration, $s_0 \in \mathscr{S}$ is the initial symbolic state. A symbolic state $s \in \mathscr{S}$ may contain any predicate about objects. However, the work discussed in this Chapter deals with predicates that specify the poses of the objects. The set $\mathscr{L}$ contains a finite number of significant poses of objects, i.e. geometric poses that are also represented as symbols in the task planner. In each symbolic state s objects are either assigned to a significant object pose in $\mathscr{L}$, or attached to the robot gripper. When an object is in a significant object pose the symbolic state may also change if the object is grasped or released by the robot. When a grasped object is moved by the robot, in-between poses are not encoded as significant object poses. Each object in $\mathscr{O}$ contains information about its geometry, the available grasping poses, and the regions $\mathscr{R}$ where it can be placed by the robot, either as intermediate locations or goal locations. Elements of $\mathscr{R}$ are continuous regions where placement of objects is allowed, e.g. flat surfaces. A sample generator that adds new significant object poses to $\mathscr{L}$ is associated to each region. The generators sample poses independently of actions or of existing object poses

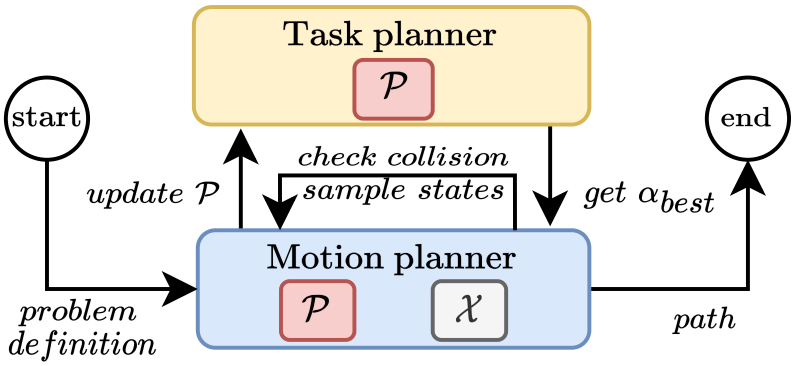


Figure 2.2: PROTAMP-RRT flowchart.

by extracting random poses in regions $\mathscr{R}$, with the requirement that at least one inverse kinematic solution exists to reach an object in that pose, but not that the pose is collision-free. Symbolic actions $a \in \mathscr{A}$ are operations that change the symbolic state, and in the proposed method involve moving objects between significant poses. Task goal $\mathscr{G}$ is a set of conditions that must be satisfied by the final symbolic state.

The flowchart of the proposed planner is shown in Fig. 2.2. The main component is the RRT-based motion planner, which operates in an unified state space $\mathscr{X} = \mathscr{C} \times \mathscr{S}$ (Section 2.2.1). The motion planner samples new unified states, checks them for collisions, and organizes them into a tree data structure $\mathscr{T}$. A probabilistic model $\mathscr{P}$ (Section 2.2.2) is updated whenever a new valid unified state is created or discarded due to collisions with the environment or other objects, in order to evaluate the probability that the motion plan will be able to progress towards the sub-goal for each action. If the probability of the current task plan becomes too low, the motion planner triggers the task planner (Section 2.2.3) to obtain a new task plan, i.e. the sequence of actions α_{best} that reaches the goal $\mathscr{G}$ with the highest probability. The motion planner descends $\mathscr{T}$ until it finds the first unreached sub-goal of α_{best}, and then it extends $\mathscr{T}$ towards this sub-goal (Section 2.2.4).

2.2.1 Unified state space

Each symbolic state $s \in \mathscr{S}$ is a set of predicates that describe the environment. In this Chapter work predicates describe mainly the poses of movable objects. Since PROTAMP-RRT does not incorporate any symbolic reasoning conventions within its task planner, the predicates in a symbolic state do not comply with formal representations, like in STRIPS. A symbolic state s contains a set of pairs c_i in the form $\langle o_i, l_{j(i)} \rangle$ that associates each manipulable object in $o_i \in \mathscr{O}$ to a (significant) object pose in $l_j \in \mathscr{L}$ (except for the grasped object, if any). The task planner operates on $\mathscr{S}$, where symbolic states are connected by abstract actions in $\mathscr{A}$. Each action $a \in \mathscr{A}$ has preconditions that must be verified before the action can be applied to the symbolic state, and post-conditions that describe the changes applied to the symbolic state. Postconditions include as a sub-goal the final configuration $q_{end}(a)$ of the robot after the action. Given a sequence of actions α and the initial symbolic state s_0, post-conditions can be applied consecutively to determine a sequence of symbolic states $\{s_0, \ldots, s_i, \ldots, s_{|\alpha|}\}$, where s_i is the symbolic state after action a_i. Some actions in $\mathscr{A}$ are based on action templates which depend on locations and/or objects, such as $Move(o, l_s, l_d)$ that moves object $o \in \mathscr{O}$ from l_s to l_d, where $l_s, l_d \in \mathscr{L}$ are significant object poses. When new significant object poses are sampled by the sample generators, new actions are added to $\mathscr{A}$ using the new poses.

Four examples of symbolic states are shown in Figure 2.3: initial symbolic state s_0 and states s_1, s_2, and s_3, where there is a different assignment of each box to the significant object poses $l_1, l_2, l_3, l_4 \in \mathscr{L}$. Two actions are defined: action $a_1 \in \mathscr{A}$ moves *Box* 1 from significant object pose l_1 to l_2, and action a_2 moves *Box* 2 from significant object pose l_3 to l_4. Symbolic state s_1 is generated by applying a_1 to the initial state s_0, and symbolic state s_2 by applying a_2. Both actions in any order lead to the same final symbolic state s_3.

The RRT motion planner operates on a unified state space $\mathscr{X} = \mathscr{C} \times \mathscr{S}$. Each unified state $x \in \mathscr{X}$ consists of a single robot configuration $q \in \mathscr{C}$ and a symbolic state $s \in \mathscr{S}$. The RRT based algorithm samples unified states $x \in \mathscr{X}$ and it organizes them in a tree data structure $\mathscr{T}$, with root at the initial unified state $x_0 = (q_0, s_0)$ and whose edges are motions $m \in \mathscr{M}$. The distance between two unified states, in the

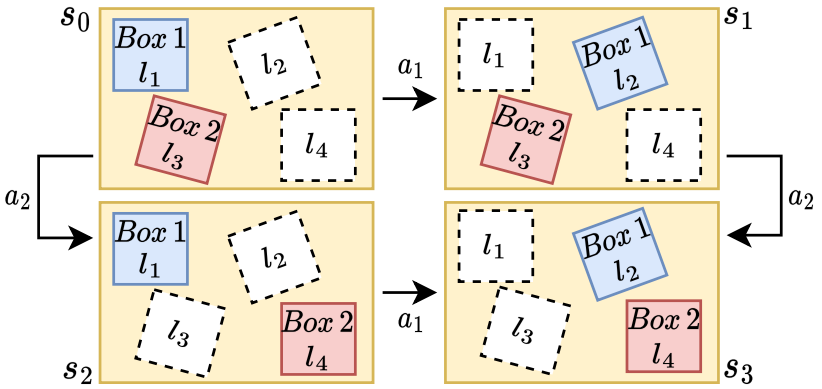


Figure 2.3: Example symbolic states generated by applying actions a_1 and a_2 in different order to s_0. In s_0 (top left), *Box* 1 is in pose l_1, and *Box* 2 is in pose l_3. Two initially unoccupied poses are also defined: l_2 and l_4 (dashed).

same symbolic state, is computed as the Euclidean distance in the robot joint space. Motions may also include changes in the symbolic state. Therefore, given a motion, the sequence of symbolic actions performed up to that motion can be retrieved by traversing the tree upwards to the root. In practice, as the planner needs to efficiently query the sequence of symbolic actions which lead to each motion, each motion holds a reference to the sequence of actions.

2.2.2 Probabilistic model

The purpose of the probabilistic model $\mathscr{P}$ is to evaluate the probability that the RRT planner will be able to progress towards the solution of an action $a \in \mathscr{A}$, given the knowledge about past performance of the motion planner on the same action. In particular, the probabilistic model is used to estimate the probability $\mathscr{P}(a|s)$ of action a in symbolic state s as the probability of successfully sampling a new robot configuration towards the action sub-goal $q_{end}(a)$. The new configuration is successfully sampled if it does not collide against the static environment or other movable objects. Therefore, under the assumption of independence of the c_i effects, $\mathscr{P}(a|s)$ may be

decomposed as:

$$\mathscr{P}(a|s) = \mathscr{P}_e(a)\ \mathscr{P}_{obj}(a|s) = \mathscr{P}_e(a) \prod_{c_i \in s} \mathscr{P}_c(a|c_i) \tag{2.1}$$

where $\mathscr{P}_e(a)$ is the probability that the new configuration does not collide with the static environment, and $\mathscr{P}_{obj}(a|s)$ is the probability that it does not collide with movable objects. In turn, $\mathscr{P}_{obj}(a|s)$ is the product of all $\mathscr{P}_c(a|c_i)$ where $c_i{=}\langle o_i, l_{j(i)}\rangle$, i.e. the probability that the new configuration does not collide with object o_i if the latter is in pose $l_{j(i)}$. As $\mathscr{P}_c$ in (2.1) depends on the action and only on a single pair c_i, it is used to compute $\mathscr{P}(a|s)$ for all symbolic states s where c_i is true. In summary, the probabilistic model $\mathscr{P}$ has three elements for each action a: an *environmental probability* $\mathscr{P}_e(a)$, a set of *object conditional probabilities* $\mathscr{P}_C(a){=}\{\mathscr{P}_c(a|c_i)\}$, and a set of *solved symbolic states* $\mathscr{P}_S(a)$.

Environmental probability

the environmental probability $\mathscr{P}_e(a)$ is the probability of not colliding with the static environment when sampling a robot configuration towards the action sub-goal $q_{end}(a)$. The probability is estimated as the fraction of configurations sampled towards the action sub-goal which did not collide with the environment:

$$\mathscr{P}_e(a) = \frac{N_e(a)}{D_e(a)} \tag{2.2}$$

where N_e and D_e are two positive integers. Initially, $N_e(a){=}N_0^e$, $D_e(a){=}D_0^e$, where N_0^e and D_0^e are positive integer constants and $N_0^e{<}D_0^e$. Whenever the RRT discards a sampled configuration due to a collision with the static environment $\mathscr{E}$, $\mathscr{P}_e(a)$ is reduced by incrementing D_e. Whenever the RRT successfully samples a configuration towards the action sub-goal $q_{end}(a)$, $\mathscr{P}_e(a)$ is incremented by incrementing both $N_e(a)$ and $D_e(a)$. After a successful sampling, if $\mathscr{P}_e(a){<}\frac{N_{min}}{D_{min}}$, $N_e(a)$ and $D_e(a)$ are reset to $N_e(a){=}N_{min}$ and $D_e(a){=}D_{min}$, where N_{min} and D_{min} are constants ($N_{min}{<}D_{min}$). This rule restores the probability to a higher value when the RRT samples a robot configuration towards an action sub-goal after many failures, e.g. when a narrow passage is found after many collisions.

Object conditional probabilities

the set of object conditional probabilities $P_C(a)$ for action a defines a probability $\mathscr{P}_c(a|c_i)$ for each object/pose pair $c_i = \langle o_i, l_{j(i)} \rangle$ in a set $C(a)$ of object/pose pairs. A $\mathscr{P}_c(a|c_i)$ represents the probability that a robot configuration does not collide with object o_i in significant pose l_j. Set $C(a)$ is initially empty, and it is filled with object/pose pairs that negatively affect action a as they cause collisions. Similarly to the environmental probability, the object conditional probability is estimated as the fraction of configurations extracted towards the action sub-goal that did not collide with the object o_i when it was at location $l_{j(i)}$:

$$\forall c_i \in C(a) \qquad \mathscr{P}_c(a|c_i) = \frac{N_c(a|c_i)}{D_c(a|c_i)} \tag{2.3}$$

where N_c and D_c are two positive integers. Whenever the RRT fails to sample a robot configuration due to a collision, all colliding objects and their poses are collected in a set of pairs $C_{new} = \{c_i\}$. For each $c_i \in C_{new}$, if c_i is not already in $C(a)$, it is added and $\mathscr{P}_c(a|c_i)$ is initialized with $N_c(a|c_i) = N_0^c$, $D_c(a|c_i) = D_0^c$. Otherwise, if c_i is in $C(a)$, $\mathscr{P}_c(a|c_i)$ is decreased by incrementing $D_c(a|c_i)$. Conversely, if the RRT succeeds in sampling a configuration towards $q_{end}(a)$, probability $\mathscr{P}_c(a|c_i)$ is increased by incrementing $N_c(a|c_i)$ and $D_c(a|c_i)$ for all $c_i \in C(a)$. Similarly to $\mathscr{P}_e(a)$, after a successful sampling, if $\mathscr{P}_c(a|c_i) < \frac{N_{min}}{D_{min}}$, then $N_c(a|c_i)$ and $D_c(a|c_i)$ are reset to $N_c(a|c_i)$=N_{min} and $D_c(a|c_i)$=D_{min}.

Solved symbolic states

a set of symbolic states $\mathscr{P}_S(a)$ where the action sub-goal $q_{end}(a)$ was reached for action a. That is, when the RRT reaches $q_{end}(a)$ in a symbolic state s, then s is added to $\mathscr{P}_S(a)$.

When the probabilistic model is queried to obtain probability $\mathscr{P}(a|s)$, first it checks if s is among solved symbolic states $\mathscr{P}_S(a)$, as in this case the action sub-goal has already been reached and probability is 1. Else, probability $\mathscr{P}(a|s)$ is computed according to (2.1), (2.2) and (2.3).

2.2.3 Task planner

The task planner searches for a task plan, i.e. a sequence of symbolic actions $\alpha_{best} = \{a_1, a_2, \ldots, a_{|\alpha_{best}|}\}$ that satisfies the task goal $\mathscr{G}$ while maximizing $\mathscr{P}(\alpha)$, i.e., the joint probability of performing the entire sequence of actions α:

$$\mathscr{P}(\alpha) = \prod_{i=1,\ldots,|\alpha|} \mathscr{P}(a_i|s_{i-1}) \tag{2.4}$$

where s_{i-1} is the symbolic state obtained after applying all actions of the task plan up to a_{i-1}. In PROTAMP-RRT, a tree search approach was used based on $A^\star$, which prioritizes the expansion of the node with the highest value of the heuristic:

$$F(\alpha') = \mathscr{P}(\alpha')\, H(s_{|\alpha'|}) \tag{2.5}$$

where α' is the partial plan up to that node, and $H(s_{|\alpha'|})$ estimates the probability to reach the task goal given the final symbolic state $s_{|\alpha'|}$ of α'. For each condition in $\mathscr{G}$ (e.g., an object must be in a given location) which is violated by state $s_{|\alpha'|}$, at least one additional action must be taken by the robot. Hence, given a constant P_H defining the a-priori probability of an action, heuristic $H(s_{|\alpha'|})$ is:

$$H(s_{|\alpha'|}) = (P_H)^{U(s_{|\alpha'|})} \tag{2.6}$$

where $U(s_{|\alpha'|})$ is the number of unsatisfied task goal conditions. A bounded approach was used to speed up the search after the first execution of the task planner, given the previous solution α_{prev}. Nodes where $\mathscr{P}(\alpha') < \mathscr{P}(\alpha_{prev})$ are pruned, as they cannot lead to an improvement over α_{prev}.

PROTAMP-RRT searches for a solution with a single execution of the unified RRT planner. The unified RRT motion planner iteratively extends $\mathscr{T}$ until a path to a symbolic state that satisfies the task goal $\mathscr{G}$ is found. The extension is guided by the most promising sequence of symbolic actions α_{best}.

The planning algorithm (Algorithm 1) has a main loop (lines 2-27) that consists of two parts, task planning and motion planning. The task planning part (lines 3-12) invokes the task planner to obtain α_{best}, while the motion planning part (lines 13-21) extends the RRT $\mathscr{T}$ guided by the symbolic plan.

Algorithm 1 PROTAMP-RRT planning algorithm

Require: $\mathscr{E}$, $\mathscr{O}$, $\mathscr{A}$, $\mathscr{R}$, $\mathscr{L}$, $\mathscr{G}$, q_0, s_0

Ensure: α_{best}, $\mathscr{T}$

$\mathscr{P} \leftarrow \varnothing$, $\mathscr{T} \leftarrow \{(s_0, q_0)\}$, $P_{best} = 1$
while true **do**
  if $\alpha_{best} = \varnothing$ or $\mathscr{P}(\alpha_{best}) < P_{best} P_{mult}$ **then** ▷ Task planning
    $\alpha_{best} \leftarrow TaskPlanner(\mathscr{O}, \mathscr{A}, \mathscr{L}, \mathscr{G}, s_0, \mathscr{P})$
    $r \leftarrow ReachableActions(\mathscr{T}, \alpha_{best})$
    while $\mathscr{P}_{obj}(\alpha_{best}) \leq P_{low}^{|\alpha_{best}|-(r-1)}$ **do**
      $\mathscr{L} \leftarrow \mathscr{L} \cup \{SampleNewObjectPose(\mathscr{R})\}$
      $\alpha_{best} \leftarrow TaskPlanner(\mathscr{O}, \mathscr{A}, \mathscr{L}, \mathscr{G}, s_0, \mathscr{P})$
      $r \leftarrow ReachableActions(\mathscr{T}, \alpha_{best})$
    end while
    $P_{best} \leftarrow \mathscr{P}(\alpha_{best})$
  end if
  $q_{goal} \leftarrow q_{end}(a_r)$ ▷ Motion planning
  $q_{new} \leftarrow SampleState(\mathscr{T}, q_{goal})$
  $C_{new} \leftarrow CheckCollisions(q_{new}, s_{r-1})$
  $\mathscr{P} \leftarrow UpdateProb(\mathscr{P}, \mathscr{E}, \mathscr{T}, a_r, q_{new}, q_{goal}, C_{new}, s_{r-1})$
  if $C_{new} = \varnothing$ **then**
    $\mathscr{T} \leftarrow Extend(\mathscr{T}, (s_{r-1}, q_{new}))$
    if $q_{new} = q_{goal}$ and $r = |\alpha_{best}|$ **then**
      return α_{best}, $\mathscr{T}$
    end if
    if $q_{new} = q_{goal}$ **then**
      $SetSolved(\mathscr{P}, a_r, s_{r-1})$
      $r \leftarrow r + 1$
    end if
  end if
end while

Algorithm 2 Probability update procedure *UpdateProb*

Require: $\mathscr{P}, \mathscr{E}, \mathscr{T}, a, q_{new}, q_{goal}, C_{new}, s$

Ensure: $\mathscr{P}$

1: **if** $q_{new} = q_{goal}$ and $C_{new} \neq \varnothing$ **then** ▷ Goal unreachable
2: $\quad \forall c_i \in C_{new}, \quad \mathscr{P}_c(a|c_i) \leftarrow 0$
3: **else**
4: $\quad$ **if** $C_{new} \neq \varnothing$ **then** ▷ Collision
5: $\quad\quad$ **if** $\mathscr{E} \in C_{new}$ **then** $Decrease(\mathscr{P}_e(a))$
6: $\quad\quad$ **end if**
7: $\quad\quad \forall c_i \in C_{new}, \ Decrease(\mathscr{P}_c(a|c_i))$
8: $\quad$ **end if**
9: $\quad$ **if** $C_{new} = \varnothing$ and ▷ Success
10: $\quad\quad \forall q|(q,s) \in \mathscr{T}, \ |q_{new} - q_{goal}| < |q - q_{goal}|$ **then**
11: $\quad\quad Increase(\mathscr{P}_e(a))$
12: $\quad\quad \forall c_i \in s \cap C(a), \ Increase(\mathscr{P}_c(a|c_i))$
13: $\quad$ **end if**
14: **end if**

2.2.4 Integrated task and motion planner

Task planning (line 4) is executed only if there is no symbolic plan α_{best}, or if its probability $\mathscr{P}(\alpha_{best})$ (equation 2.4) falls below a threshold $P_{best} P_{mult}$, where $P_{mult} \ll 1$ is a constant and P_{best} (line 11) is the probability of the plan when it was first generated. Once $\alpha_{best} = \{a_1, \ldots, a_{|\alpha_{best}|}\}$ is available, the symbolic states traversed by α_{best}, $\{s_0, \ldots, s_{|\alpha_{best}|}\}$, are known. The planner uses *ReachableActions* (line 5) to descend the motion tree $\mathscr{T}$ and to find the index r of the first action a_r whose sub-goal has not been reached yet by the RRT. Symbolic state s_{r-1} is obtained by applying all actions of α_{best} up to a_{r-1}. If the task planner cannot find a plan with probability greater than a threshold (line 6), a new significant object pose is sampled into $\mathscr{L}$ from each region in $\mathscr{R}$ (line 7), thus adding new actions until a new plan becomes feasible. As new actions are added with an environmental probability $\mathscr{P}_e < 1$, old

actions with high probability may still be preferred by the task planner. Sampling new object poses never influences environmental probability, hence the test at line 6 uses only object conditional probability of the plan, defined as:

$$\mathscr{P}_{obj}(\alpha_{best}) = \prod_{i=1}^{r} \mathscr{P}_{obj}(a_i|s_{i-1}) \tag{2.7}$$

where $\mathscr{P}_{obj}(a|s)$ is defined in (2.1). The dynamic threshold (line 6) is a constant P_{low} raised to the number of non-reached sub-goals of α_{best}. At each iteration, the motion planner first sets the current sub-goal q_{goal} to the sub-goal $q_{end}(a_r)$ of action a_r (line 13). Then, at line 14 a new configuration q_{new} is sampled by *SampleState*, randomly or towards the action sub-goal with probability P_{goal}. When *SampleState* is called and q_{goal} is different from the previous iteration of the main loop, *SampleState* always extracts q_{goal}, to check if the action is trivially feasible or infeasible. The collision check is performed at line 15 and a set of pairs $C_{new} = \{\langle o_i, l_{j(i)} \rangle\}$ of colliding objects and their locations are found. This set is used to update the probabilistic model $\mathscr{P}$ (*UpdateProb*, line 16), which in turn changes the symbolic plan probability $\mathscr{P}(\alpha_{best})$ for the next iteration. If no collisions are found (line 17) $\mathscr{T}$ is extended with q_{new} (line 18). If $\mathscr{T}$ has been extended to q_{goal} and r was the index of the last action of α_{best}, the algorithm terminates because the task goal $\mathscr{G}$ is reached (lines 19-21). Otherwise, if $\mathscr{T}$ has been extended to q_{goal}, but other actions must be performed, the symbolic state s_{r-1} is added to the set of solved symbolic states $\mathscr{P}_S(a_r)$, and $\mathscr{P}_e(a_r)$ is set to 1 in the probabilistic model (*SetSolved* at line 23). Also, as q_{goal} has been reached, the next action of α_{best} is reachable, and therefore r is incremented (line 25). Then, the algorithm continues with the next iteration of the main loop.

In *UpdateProb* (Algorithm 2) if the sampled state q_{new} is the action sub-goal and it is in collision (line 1), then the action is set as infeasible whenever any colliding objects/pose pairs are present in C_{new} (line 2). If sampled state q_{new} is not the action sub-goal and it is in collision (lines 4-7), then the obstacles are hindering the action and probabilities $\mathscr{P}_e(a)$ and $\mathscr{P}_c(a|c_i)$ are decreased as explained in Sections 2.2.2 and 2.2.2. Probabilities are increased (lines 9-12) if q_{new} is not in collision, and if it is the nearest configuration to q_{goal} among the configurations in $\mathscr{T}$ with symbolic state s.

The *UpdateProb* procedure provides a negative feedback if the RRT fails to expand towards the current action sub-goal. In particular, in Algorithm 2 (line 10) the probability is increased only if the new configuration is closer to the action sub-goal than each previous configuration. The probability of finding configurations which are closer to the action sub-goal decreases exponentially with the number of configurations in $\mathscr{T}$. While expansion towards the same action sub-goal is attempted and fails, the probability $\mathscr{P}(\alpha_{best})$ is reduced and it will eventually be lower than P_{mult} multiplied by its initial value. When this occurs, the task planner is called again, to find a (potentially) different plan with higher probability.

2.2.5 Probabilistic completeness

In this section, a proof is outlined demonstrating that PROTAMP-RRT (Algorithm 1) is probabilistically complete, i.e., that as the planning time increases, the probability that a solution is found, if one exists, approaches one. This proof applies only for a finite symbolic space, i.e. when sampling of new significant object poses is disabled and a finite set of significant object poses is pre-defined. Proving probabilistic completeness of the whole algorithm is likely more complex, as the symbolic space can grow without limit. It is also suspected that PROTAMP-RRT is not probabilistically complete, but that it may be made probabilistically complete with minor modifications. For example, it may be made probabilistically complete by randomly selecting a random symbolic action from time to time.

The proof of probabilistic completeness proceeds as follows. First, it is proven that if the currently selected plan α_{best} is infeasible, the task planner will eventually be called to find a new symbolic plan, with probability approaching one (Theorem 1). Then, it is proven that if there exists at least a feasible symbolic plan, then a feasible plan will be selected as α_{best} a number of times approaching infinity as the planning time increases (Theorem 2). Finally, it will be proven that, if the current plan α_{best} is feasible, then there is a nonzero probability that the motion planner finds a solution before the generation of a new plan is triggered (Theorem 3). Therefore, as the motion planner has potentially any number of attempts to find the solution, for planning time approaching infinity the probability that the solution is found approaches one.

Theorem 1. *If the currently selected plan α_{best} is infeasible, the task planner will eventually be called to find a new symbolic plan, with probability approaching one.*

Proof. If α_{best} is infeasible, then there exists an action a_r whose sub-goal $q_{end}(a_r)$ is unreachable in symbolic state s_{r-1}. As there must be an obstacle preventing the action from reaching the goal, there is always a nonzero probability of sampling a robot configuration which is in collision. The collision may be either with the environment (hence it affects the environmental probability) or a movable object (hence it affects the object conditional probability). As the two kinds of probabilities are updated in the same way, in the following it will be used the term "probability" for both. Whenever a colliding configuration is sampled, probability of that action in the probabilistic model is reduced according to Section III-B. Conversely, probability is increased only when the sampled configuration is not in collision and it is closer to the sub-goal than all other configurations in $\mathscr{T}$ with the same symbolic state s_{r-1}. The probability of finding a new non-colliding configuration closer to the sub-goal decreases exponentially as more configurations are added to $\mathscr{T}$, as the new configuration must be closer than all other already-sampled configurations. Therefore, as the number of sampled configurations increases, the expected number of non-colliding configurations sampled closer to the goal approaches a finite value, while the number of colliding configurations grows unlimitedly. Hence, the estimated probability of action a_r in state s_{r-1} in the probabilistic model tends to zero. Therefore, also $\mathscr{P}(\alpha_{best}) \to 0$, and in particular it will eventually be lower than $P_{best} P_{mult}$, so the task planner will be triggered to find a new plan (line 3 of Algorithm 1). □

Theorem 2. *If there exists at least a feasible symbolic plan, then a feasible plan will be selected as α_{best} a number of times approaching infinity as the planning time increases.*

Proof. If $\mathscr{L}$ contains a finite number of significant object poses, the symbolic state is finite. Hence, the number of symbolic plans is also finite, as each symbolic plan traverses each symbolic state at most once. Otherwise, if the same symbolic state was traversed at two points in the plan, a shorter plan with higher probability would

be found by removing the actions in-between. According to Theorem 1, if the current symbolic plan is infeasible the task planner is eventually called to obtain a new symbolic plan. Now it is proven that eventually the returned plan is one of the feasible plans. Let's assume by contradiction that, starting at some point in time, all plans returned by the task planner are from a subset which contains only infeasible plans. By definition, each of these plans has at least one infeasible action a_r, whose sub-goal cannot be reached by the motion planner in symbolic state s_{r-1}. As RRT is probabilistically complete, and the number of plans is finite, eventually it is able to reach the sub-goal of all actions of all plans in the subset, up to the first infeasible action. When this happens, for Theorem 1 the motion planner eventually calls the task planner with the probability of the infeasible action in symbolic state s_{r-1} reduced at least by a factor $1/P_{mult}$. This operation also reduces the probability of the same action in symbolic states different from s_{r-1} (which may belong to feasible plans), but by a lesser extent, as the same action in other symbolic states does not share all the object conditional probabilities. Hence, eventually the probability of all symbolic plans in the subset of infeasible plans will be below one of the feasible plans, and the task planner returns a feasible plan which is not in the subset. A contradiction is reached, hence it can be concluded that a feasible symbolic plan will be eventually returned starting from any point in time. Therefore, as planning time increases a feasible symbolic plan is selected a number of times approaching infinity. □

Theorem 3. *Given a feasible symbolic plan α_{best}, the motion planner has a nonzero probability of finding a corresponding motion trajectory before the task planner is triggered.*

Proof. As long as robot configurations which are in collision are not sampled, the probability is never reduced in the probabilistic model, and therefore the task planner cannot be triggered. Hence, the RRT has a nonzero probability of finding the trajectory which is at least the probability of sampling all the robot configuration of the trajectory without sampling a colliding state. □

2.3 Experiments

2.3.1 Experimental setup

The PROTAMP-RRT approach was evaluated in simulated manipulation tasks in which a robot moves boxes. The robot is a fixed Universal Robot UR10 manipulator (with a reach of 1300 mm). A parallelepiped of size $0.1 \times 0.1 \times 0.11\,\text{m}$ was used to simulate an OnRobot VGC10 vacuum gripper. In these thesis it was assumed that gripper is able to perform grasping by contact with any flat side of the objects, disregarding dynamic effects. The simulated experimental environment was implemented on top of ROS (Robot Operating System) and visualized in RViz. The software was implemented in C++, using the OMPL planning library [16]. The MoveIt planning framework and the Flexible Collision Library [19] were used to instantiate the geometry and check for collisions. The IKFast OpenRAVE module was used to create an analytical inverse kinematics solver for the UR10. The experimental evaluation ran on an Intel i9 10900 processor @ 2.80 GHz (32 GB of RAM, Ubuntu 20.04 LTS). After performing an evaluation of the hyperparameters, shown in section 2.3.3, PROTAMP-RRT was configured with $N_0^e = 45$, $D_0^e = 50$, $N_0^c = 49$, $D_0^c = 50$, $N_{min} = 30$, $D_{min} = 50$, $P_H = 0.815$, $P_{mult} = 0.1$, $P_{low} = 0.8$, $P_{goal} = 0.3$.

2.3.2 Manipulation Tasks

The evaluation of PROTAMP-RRT was perfomed with tests on simulated pick-and-place tasks. Two action templates were considered, i.e. *pick* and *place* (Fig. 2.4). For each action, the probabilistic model $\mathscr{P}$ defines the environmental probability, the object conditional probabilities, and the set of solved symbolic states. A *pick action* represents the action of approaching an object while the gripper is empty, and then attaching the object to the gripper. This action is parameterized by the target object o_t, the significant object pose l_g of o_t, and the robot configuration q_g to grasp the object in l_g. For each object o_t and pose l_g, multiple pick actions are generated with different configurations q_g, as the robot may grasp the object from multiple sides and each side may be reached with a different inverse kinematic solution. The preconditions imply

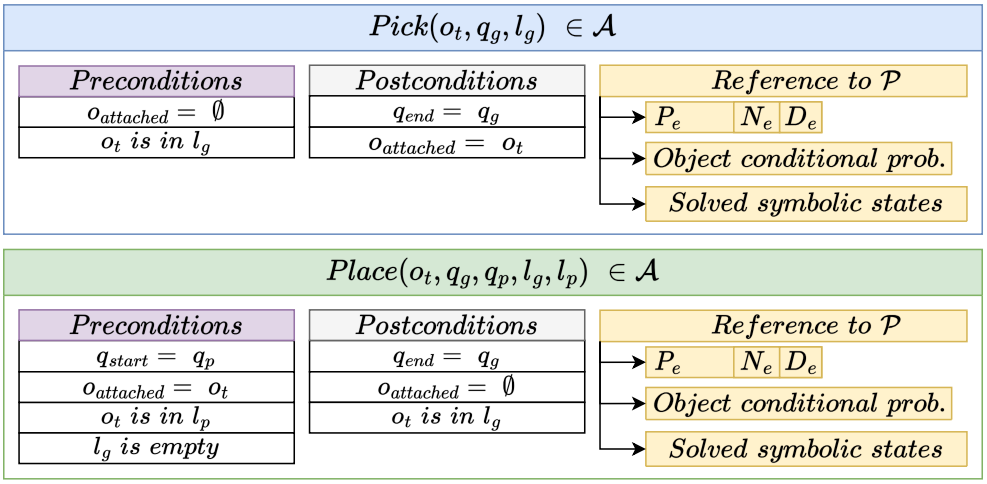


Figure 2.4: Definition of *Pick* and *Place* symbolic actions.

that no object is attached to the gripper ($o_{attached} = \emptyset$) and o_t is in l_g. As result of the action, the final robot configuration q_{end} is q_g and o_t is attached to the end effector. A *place action* represents the action of moving a grasped object to a new location, and then releasing the object. This action is parameterized by the target object o_t, the initial significant object pose l_p, the final significant object pose l_g, the initial robot configuration q_p, and the final robot configuration, q_g. As for the pick actions, for each o_t, l_p and l_g, multiple place actions are generated with different configurations q_p and q_g, taking into account multiple inverse kinematic solutions. Preconditions are that the initial robot configuration is q_p, o_t is grasped and it is in significant object pose l_p, while l_g pose must not be occupied by other objects. In the postconditions, the final robot configuration q_{end} is q_g, o_t is in l_g and no object is grasped.

In total, 6 manipulation tasks were simulated (Fig. 2.5, also shown in the video attached to [2]). In the *Transfer 1*, *Transfer 2* and *Transfer 3* tasks, red objects O_1, O_2 and O_3 must be relocated into significant object poses sampled from the *Place table* planar region. In *Transfer 1*, no obstacles are present between initial and goal locations, while in *Transfer 2* and *Transfer 3* walls have been added around the objects.

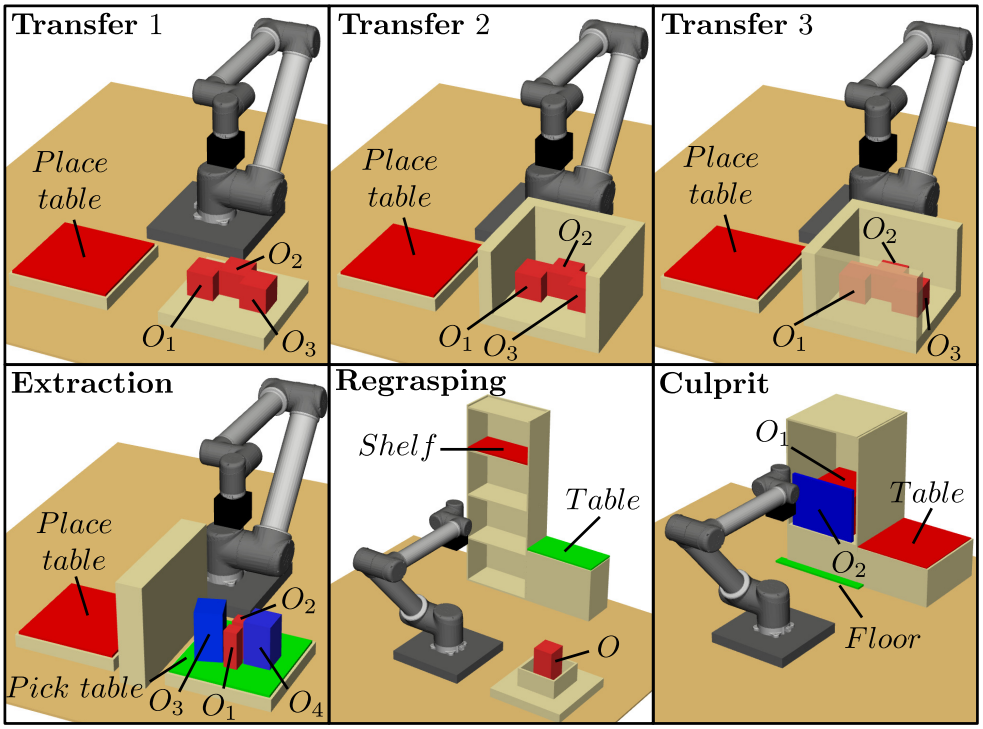


Figure 2.5: Simulated manipulation tasks, from the top left: *Transfer 1*, *Transfer 2*, *Transfer 3*, *Extraction*, *Regrasping* and *Culprit*.

The robot is allowed to grasp objects only from above and significant object poses can be sampled in the *Place table* region only. In the *Extraction* task (Fig. 2.6), red objects O_1 and O_2, must be relocated into the *Place table* region. Initially, grasping is infeasible due to the obstructing objects (blue boxes), O_2 and O_3, which have to be relocated. The robot can grasp each movable object from above or from the front side. Significant object poses for the red objects can be sampled in the *Place table* region, while for the blue objects they can be sampled in the *Pick table* region. In the *Regrasping* task the red box O must be relocated on top of the *Shelf* region. This task requires regrasping as initially the object can be picked only from above, and it can be

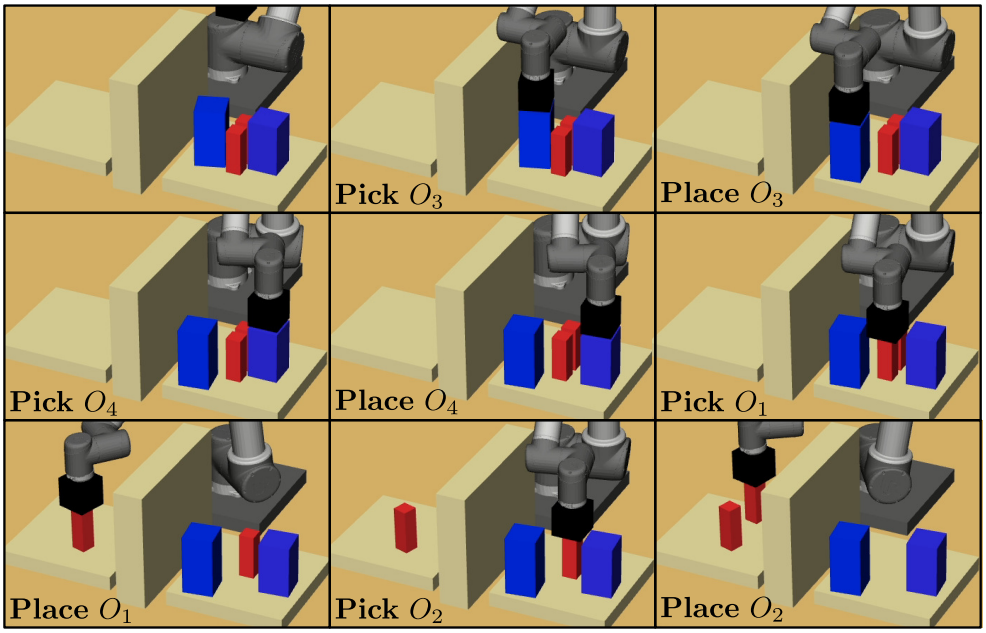


Figure 2.6: Example of *Extraction* task execution.

placed in the deposit region *Shelf* only if it is grasped from the side, due to obstacles. Movable objects can be grasped from above or from the front side. Significant object poses can be sampled in the *Shelf* region, and in the *Table* region where re-grasping may occur.

Finally, in the *Culprit* task red box O_1 must be extracted from the cupboard and relocated in the *Table* region. To do this, panel O_2 must be removed, as the opening is slightly too small otherwise. This task is challenging as the planner can be stuck in a strong local maximum: the robot is able to pick and move O_1 within the cupboard, and the planner can therefore successfully sample many robot configurations without moving O_2, however, none of these samples lead to a solution of the task. Significant object poses for the red object can be sampled in the *Table* region, and for O_2 they can be sampled in the *Floor* region. An example is shown in Fig. 2.1.

2.3.3 Hyperparameter evaluation

In this section, the hyperparameters in PROTAMP-RRT are discussed and their effect on the TAMP method performance are shown. In particular, there are ten configuration parameters in the approach proposed: N_0^e, D_0^e, N_0^c, D_0^c, N_{min}, D_{min}, P_H, P_{mult}, P_{low}, P_{goal}. These parameters are analyzed separately as two groups: parameters N_0^e, D_0^e, N_0^c, D_0^c, N_{min}, D_{min} affect the probabilistic model, while P_H, P_{mult}, P_{low}, P_{goal} affect the behavior of the task and the motion planner.

In the first group , N_0^e and D_0^e are the numerator and denominator of the initial environmental probability $P_0^e = N_0^e/D_0^e$, which represents the *a priori* environmental probability for a new action. Similarly, N_0^c and D_0^c are the numerator and denominator of the initial object conditional probability $P_0^c = N_0^c/D_0^c$, which represents the *a priori* conditional probability $\mathscr{P}_c(a|c)$ when a new object-pose pair c affecting action a is discovered. Parameters N_{min} and D_{min} are the numerator and denominator used to restore the probability to a higher value $P_{min} = N_{min}/D_{min}$ when the motion planner samples a new valid robot configuration towards an action sub-goal. This probability should be high if the geometric space has many dead ends (local minima) in which the motion planner can get stuck, so that the probability is reset to an higher value when the planner manages to escape one of these local minima. Changing the numerators and the denominators of the first group of parameters, while keeping P_0^e, P_0^c and P_{min} constant, changes the impact of a single probability increase or reduction on the probability value. When numerators and denominators are low, increments and decrements result in a larger impact, hence the probabilities change faster and, therefore, the task planner will be called more frequently if sampling of valid configurations fails. Hence, lower values are suited for simpler scene geometry, where just a few failures likely mean that the action is infeasible and that a different symbolic plan should be selected.

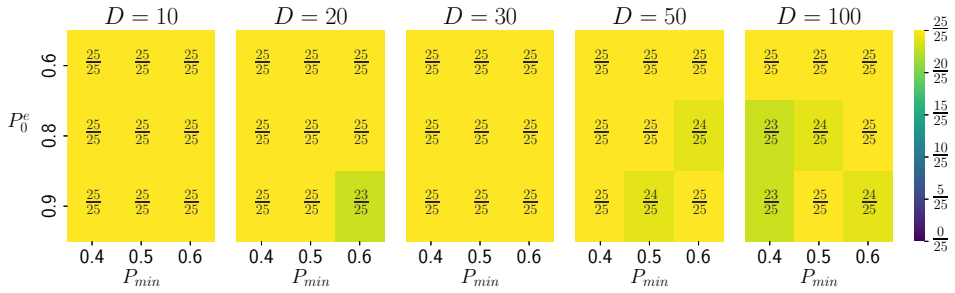
The parameters in the second group (P_H, P_{mult}, P_{low} and P_{goal}) affect the behavior of the task and motion planner. The first parameter P_H is used to compute the $A^\star$ heuristic:

$$H(s_{|\alpha'|}) = (P_H)^{U(s_{|\alpha'|})} \tag{2.8}$$

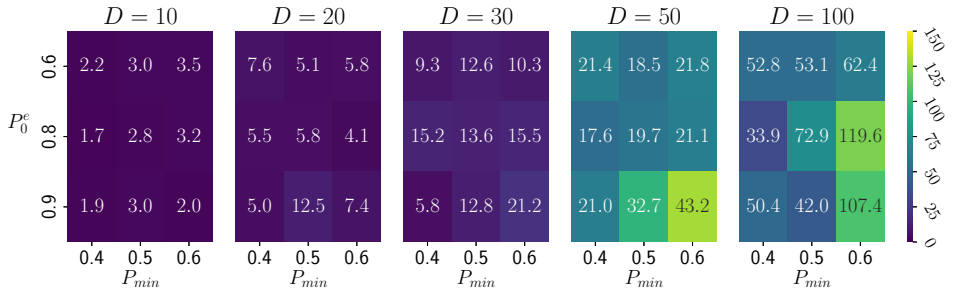
where $U(s_{|\alpha'|})$ is the number of unsatisfied conditions in the current symbolic state with respect to the task goal. Parameter P_{mult} is used to trigger replanning of the task plan. That is, when the probability of the current plan is reduced by less than P_{mult} multiplied by its original value, the task planner is called to potentially find a better symbolic plan. A lower value of P_{mult} means that the RRT is allowed more sampling failures before switching to a different task plan, and it is therefore suited for more complex scene geometry where the RRT needs more planning time. Parameter P_{low} is the threshold to sample new significant object poses: the higher P_{low}, the more frequently new poses are sampled. The last parameter, P_{goal}, is the probability that the RRT motion planner attempts an extension towards the action sub-goal, instead of sampling randomly.

The effect of the parameters on the planning algorithm are analyzed by executing a grid search for each group. During the evaluation of one group, the parameters of the other group were fixed as reported at the end of Section IV-A in the main article. Planning was executed 25 times for each parameter combination, on two tasks: *Transfer 2* and *Culprit*. The number of successes after 600 seconds and the average planning time were recorded.

For the first group of parameters (N_0^e, D_0^e, N_0^c, D_0^c, N_{min}, D_{min}), all denominators (D_0^e, D_0^c, D_{min}) were set equal to the same value D. Moreover, the numerator of the initial object conditional probability N_0^c was set to $N_0^c = D_0^c - 1$, so that P_0^c is only slightly less than 1. Indeed, when a new object conditional probability is initialized to P_0^c, the collision with that object-pose pair has occurred only once, and therefore the probability of the plan should not change significantly. These choices leave three degrees of freedom for the grid search: D, P_0^e and P_{min}. The following values were tested: $D \in \{10, 20, 30, 50, 100\}$, $P_0^e \in \{0.6, 0.8, 0.9\}$ and $P_{min} \in \{0.4, 0.5, 0.6\}$. Results are reported in Fig. 2.7 and Fig. 2.8. Generally, PROTAMP-RRT is rather robust against parameter changes, as most parameter combinations resulted in only a few failures. However, it can be observed that with $P_0^e = 0.6$ PROTAMP-RRT is unable to find a solution in the *Culprit* task, as probability of new actions is too low and they are never selected. Also, average planning time is lower for smaller D in both tasks, which means that in these particular tasks it is more efficient to change symbolic plan



(a) Success rate.

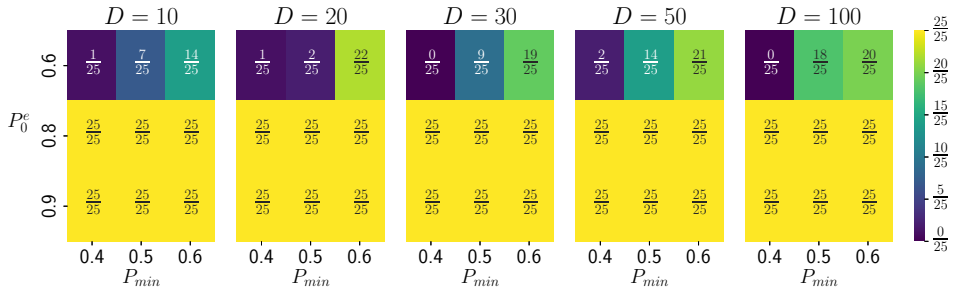


(b) Planning time (seconds).

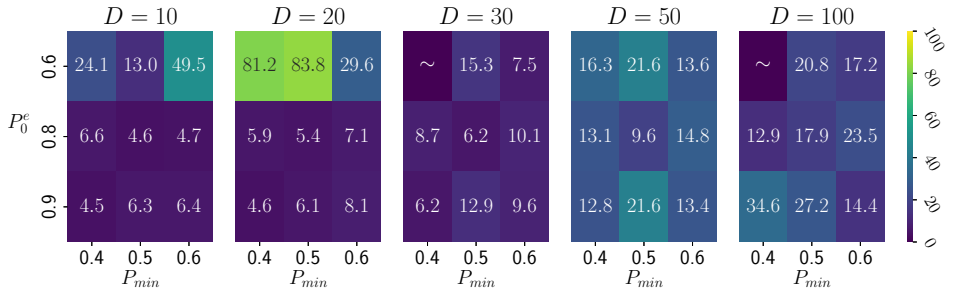
Figure 2.7: Success rate (a) and average planning time (b) for the first group of parameters, in the *Transfer 2* task. Values are displayed inside each cell.

after only a few sampling failures.

For the second group of parameters (P_{goal}, P_H, P_{mult} and P_{low}), parameter P_{goal} was fixed to 0.3 which is a common value for this parameter in RRT-based approaches (e.g., also used in TM-RRT by Finzi et al.). To provide a consistent heuristic for $A^\star$ (equation 2.8), parameter P_H should be set greater or equal than the joint probability of the actions needed to change an unsatisfied condition to a satisfied one. To select an approximate value for P_H, it is considered that, as in the proposed tasks conditions are represented by objects located in regions, the planner needs at least two actions (a pick and a place) to satisfy a goal condition. Hence, P_H is chosen as the squared a



(a) Success rate.



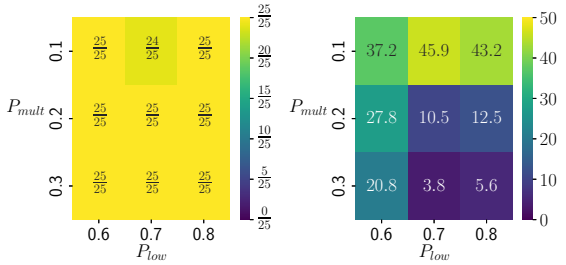
(b) Planning time (seconds).

Figure 2.8: Success rate and average planning time for the first group of parameters, in the *Culprit* task. Values are displayed inside each cell.

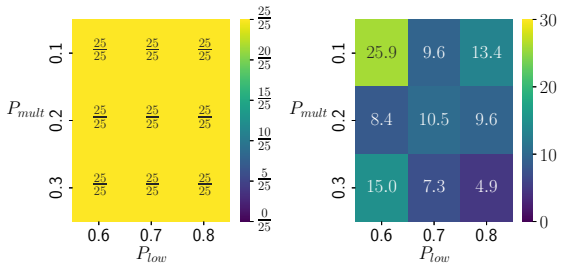
priori environmental probability $P_0^e = N_0^e / D_0^e$ for an action, increased by 0.5%:

$$P_H = 1.005 \cdot (P_0^e)^2 \tag{2.9}$$

A grid search was carried out for the two remaining parameters, P_{mult} and P_{low}, with the possible values: $P_{mult} \in \{0.1, 0.2, 0.3\}$ and $P_{min} \in \{0.6, 0.7, 0.8\}$. Results are shown in Fig. 2.9. The number of failures is very low for all tested combinations of the parameters and in both tasks. Generally, there is a noticeable planning time increase for low values of P_{low}, which triggers the sampling of new significant object poses less frequently. Similarly, high values of P_{mult} resulted in lower planning times. These observations confirm that in these particular tasks it is more efficient to change



(a) *Transfer 2* task



(b) *Culprit* task

Figure 2.9: Success rate (left) and average planning time in seconds (right) of the grid search for P_{mult} and P_{min}, in the *Transfer 2* and the *Culprit* tasks. Values are displayed inside each cell.

symbolic plan and sample new significant object poses more frequently.

2.3.4 Experiments in finite symbolic space

The proposed method was compared against two other unified RRT-based approaches, TM-RRT [23] and Planet [24], on the six manipulation tasks: *Transfer 1* (*T* 1), *Transfer 2* (*T* 2), *Transfer 3* (*T* 3), *Extraction* (*Extr.*), *Regrasping* (*Regr.*) and *Culprit* (*Culp.*). As TM-RRT can operate only on a finite symbolic space, a minimal finite set of significant object poses sufficient to solve the planning problem was pre-defined. Hence, sampling of new significant object poses in PROTAMP-RRT was disabled to ensure fair comparison. For the same reason, the goal of each task was not defined in terms

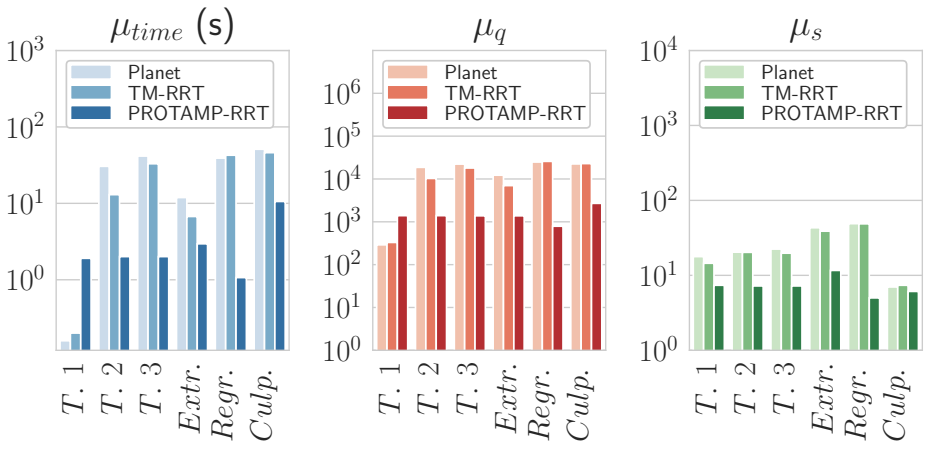


Figure 2.10: Avg. planning time (μ_{time}), avg. number of robot configurations (μ_q), avg. number (μ_s) of symbolic states (for Planet, TM-RRT, PROTAMP-RRT).

of regions, but as a specific goal pose for each object. Planet was also modified so that actions did not sample their sub-goal randomly, but selected it among the pre-sampled poses. For each task and each method, planning was executed 50 times.

The average planning time, the average number of robot configurations and the average number of symbolic states in $\mathscr{T}$ are in Fig 2.10. Planet and TM-RRT sampled more unified states in $\mathscr{X}$, as evidenced by the higher values of μ_q and μ_s. Instead, thanks to the probabilistic model $\mathscr{P}$, PROTAMP-RRT focused on the symbolic actions with highest probability, and therefore resulted in a lower planning time μ_{time}. The only exception is *Transfer 1* task, where no obstacles were present and motion planning was trivial. Although in *Transfer 1* PROTAMP-RRT still explored fewer symbolic states (lower value of μ_s), both Planet and TM-RRT resulted in a lower planning time μ_{time}. The main reason is that these algorithms use heuristics to guide the RRT instead of a long-horizon task planner.

The success rate as a function of planning time is shown in Fig. 2.11. Planet and TM-RRT success rate is generally lower than PROTAMP-RRT. At the maximum

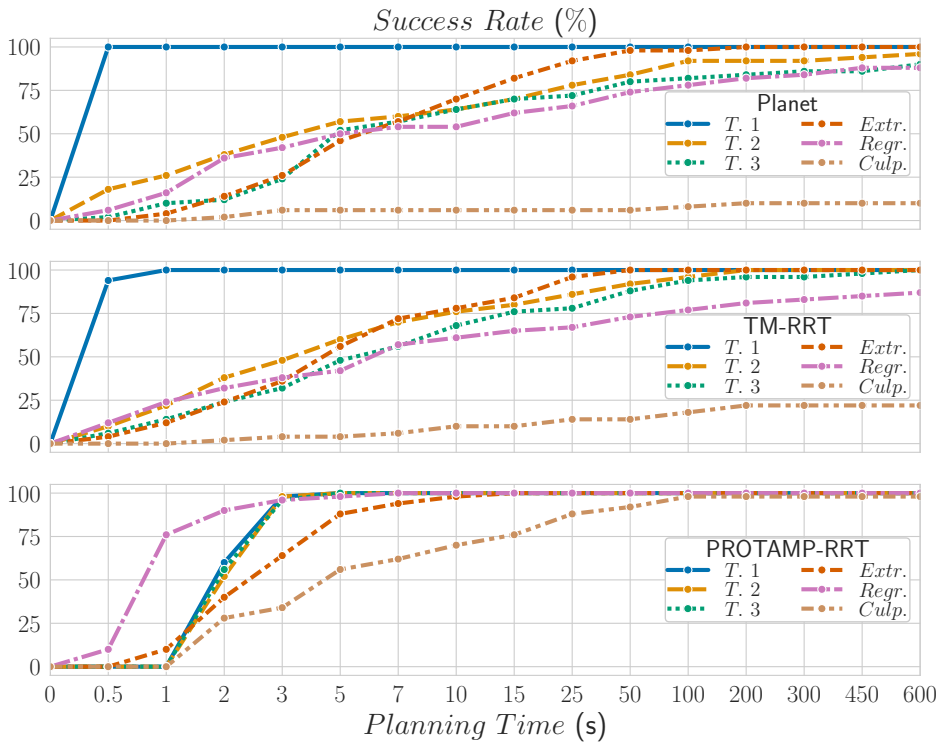


Figure 2.11: Planet, TM-RRT and PROTAMP-RRT average success rate with respect to planning time limit, with a finite symbolic space.

tested time limit of 600 seconds, Planet reached 100% success rate only in *Transfer 1* and *Extraction*, while TM-RRT only in *Transfer 1*, *Transfer 2*, *Transfer 3* and *Extraction*. In particular, Planet and TM-RRT struggled the most in the *Regrasping* and *Culprit* tasks. Instead, PROTAMP-RRT reached 100% success rate in all tasks except the *Culprit* task, where success rate was 98%, compared to the 22% of TM-RRT and 10% of Planet. The proposed probabilistic model is very effective in the solution of the *Culprit* task, as it is able to discover that, even if the red box can be picked and moved within the cupboard, it can not be extracted until the blue panel is moved.

2.3.5 Experiments in unlimited symbolic space

In this section, the full PROTAMP-RRT algorithm is evaluated in an unlimited symbolic space, without a pre-defined set of significant object poses, so that it sampled new significant object poses to reach the goal. Results are shown in Fig. 2.12. Compared to PROTAMP-RRT in Fig. 2.10, where the significant object poses were predefined, the higher complexity of the task resulted in an increased planning time μ_{time}, number of robot configurations μ_q and number of symbolic states μ_s, by about an order of magnitude. The only exception is the *Regrasping* task, where the results are similar.

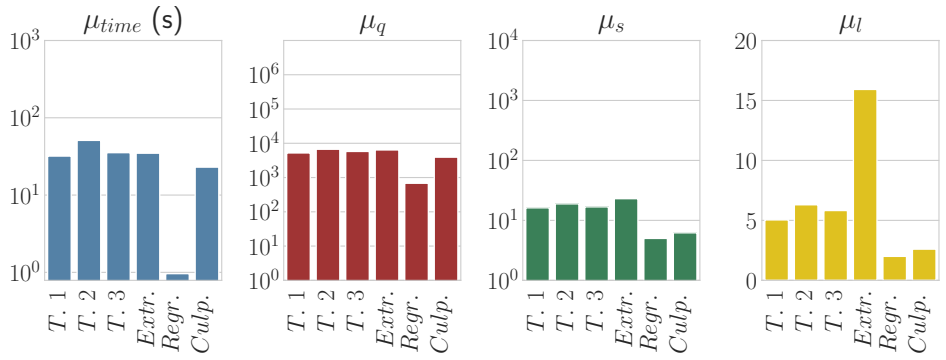


Figure 2.12: PROTAMP-RRT in unlimited symbolic space. Left to right: avg. planning time (μ_{time}), avg. number of robot configurations (μ_q), avg. number of symbolic states (μ_s), avg. number of sampled significant poses (μ_l).

Fig. 2.12 also reports the average number of sampled significant object poses (μ_l). As new significant poses are sampled only when task plan probability is very low, PROTAMP-RRT limited the expansion of the symbolic space. The task where more significant object poses were sampled is the *Extraction* task, where four movable objects were present. However, not all resulting symbolic states were explored and added to $\mathscr{T}$ as evidenced by μ_s, which is comparable to μ_s for the *Transfer* tasks. This

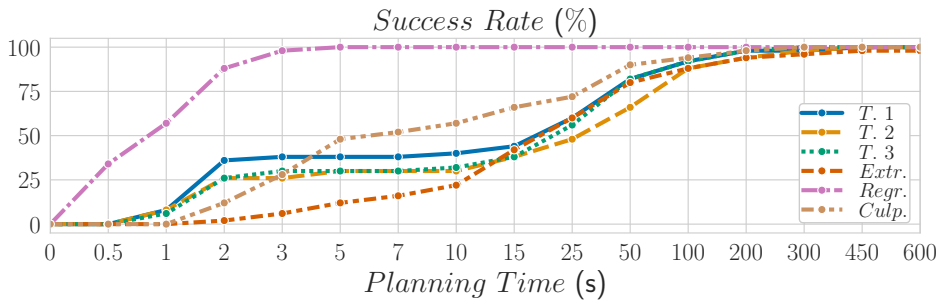


Figure 2.13: PROTAMP-RRT average success rate in unlimited symbolic space.

is likely an effect of the probabilistic model, that determined which symbolic states should be avoided. Figure 2.13 shows PROTAMP-RRT success rate as a function of planning time limit. Even if results are worse than in Fig. 2.11, PROTAMP-RRT successfully solved most of the trials after about 400 seconds. Results confirm that the *Extraction* task seems to be the most challenging.

2.3.6 Scalability

In this section experimental results about the scalability of PROTAMP-RRT are reported. The *Transfer 2* task was planned with a progressively increasing number of movable objects, up to 8 objects. For each number of objects, planning was repeated 25 times. In order to have enough space to relocate the objects, the area of the *Place table* region was increased proportionally to the number of the available movable objects. In particular, for n objects, the table side length was computed as:

$$l_{table} = \frac{l_0}{\sqrt{n_0}}\sqrt{n} \tag{2.10}$$

where $l_0 = 0.4$ and $n_0 = 3$ are, respectively, the initial table side length and the number of boxes in the *Transfer 2* task.

The results of the scalability test are shown in Figure 2.14, where the success rate is reported as a function of planning time for each number of objects. The success

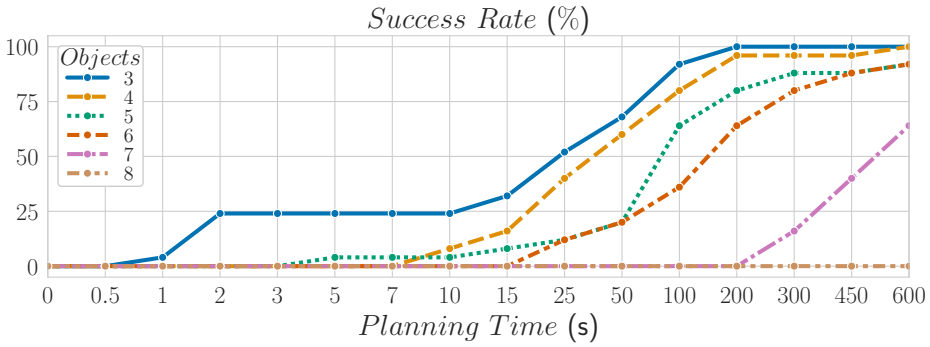


Figure 2.14: Scalability test in the *Transfer 2* task.

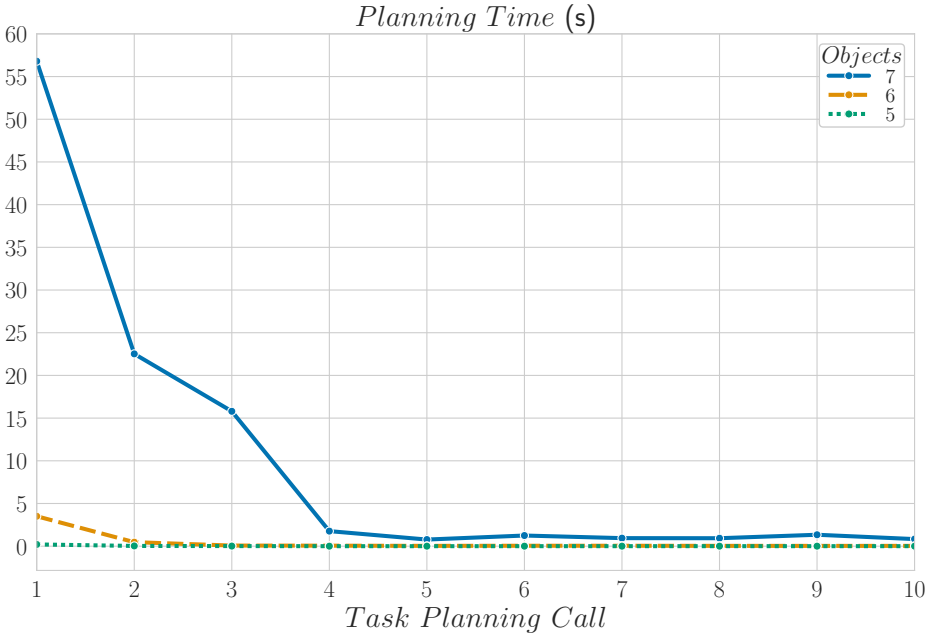


Figure 2.15: Average planning time of the first 10 task planning calls for 5, 6 and 7 objects.

rate decreases as the number of movable objects increases, reaching 0 for 8 objects. The main cause is that, for many objects, the $A^\star$ task planner has high planning time and memory usage, eventually filling the 32 GB of available RAM. The issue is most evident in Figure 2.15, in which the average planning times of the first 10 task planner calls are shown. At the beginning of the execution, when most actions have the same probability as only a few of them have been attempted by the motion planner, the task planner operates as a simple breadth-first search, and it is forced to enumerate all possible task plans with minimum length. Therefore, the first calls of the task planner require long planning times. It is hypothesized that better performance could be obtained with a different task planner which combines both breadth-first and depth-first search.

2.4 Discussion

An integrated TAMP approach based on a unified Rapidly-exploring Random Tree was proposed, that operates on both the geometric space and the symbolic space. The method introduces a probabilistic model that estimates the feasibility of symbolic actions by evaluating the probability that new collision-free configurations will be sampled. The PROTAMP-RRT method was evaluated in several manipulation tasks showing better performance than TM-RRT and Planet in terms of planning time and success rate. Moreover, the ability of PROTAMP-RRT to expand the symbolic space was evaluated indicating that it is able to solve planning problems by sampling a few new significant object poses.

Chapter 3

A Prototype Platform for Palletization and Depalletization Tasks using a Collaborative Robot

In the state-of-art regarding industrial depalletization tasks mobile robots are used to relocate entire pallets from their initial pose towards a depalletization area equipped with industrial manipulators. While the pallet is in the depalletization area, robotic arms systematically unload all the parcels one after the other. Once the unloading is completed, the pallet is relocated to its initial position, such as in a storage rack. As stated in Chapter 1, fixed manipulation stations are inefficient if a single object needs to be picked up from a pallet, while mobile manipulators represent a more flexible solution. Mobile manipulators are commonly used in research contexts and most of the existing commercials product are still in the prototype stage. In this Chapter a prototype platform for palletization and depalletization tasks is presented (Figure 3.1). It consists of a wheeled cart, a collaborative manipulator, a vacuum gripper and depth cameras. In sections 3.1, 3.2, 3.3 and 3.4 the hardware components are described in details. All the components used in the prototype, except for a few supports used to mount the devices, are available on the market. The collaborative manipulator and the gripper were selected to handle objects such as cardboard boxes and tissue items

(e.g., kitchen and toilet paper). The platform capabilities will be demonstrated in the next chapter, where the prototype will be used to relocate cardboard boxes.

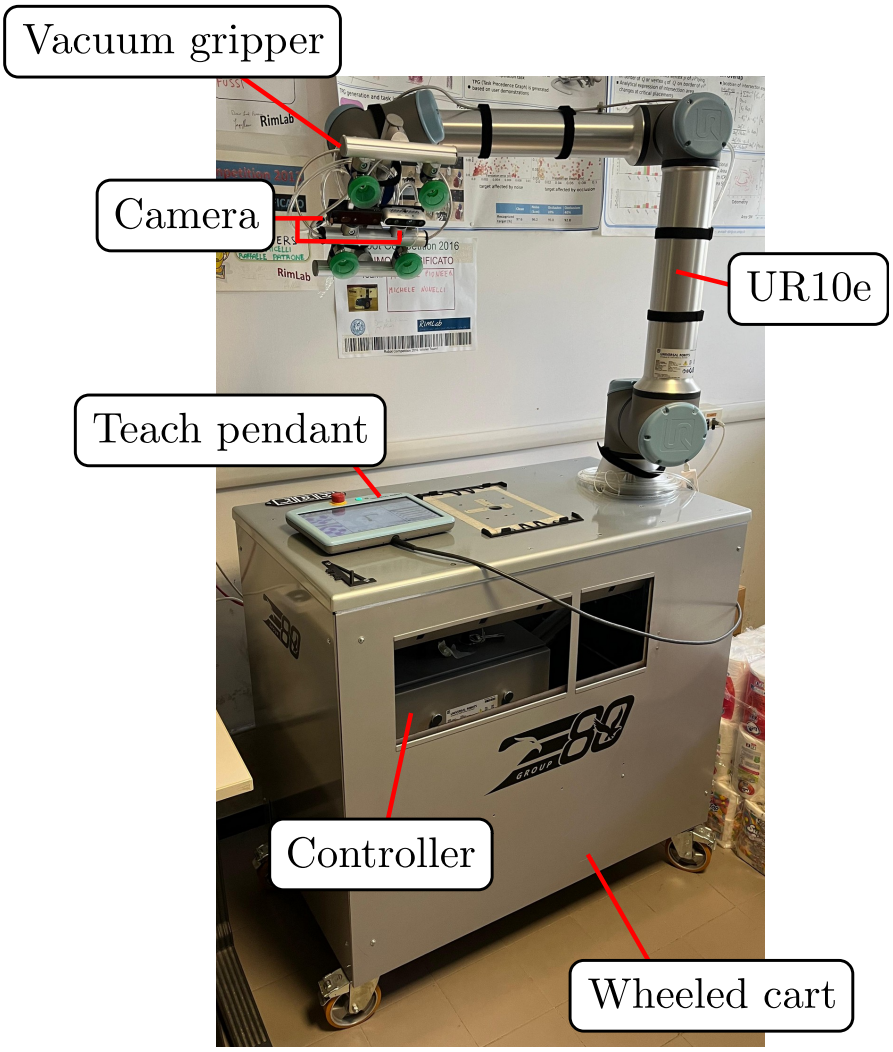


Figure 3.1: Collaborative prototype platform.

3.1 Mobile robot (Wheeled cart)

A mobile robot is capable of freely moving within a workspace. The primary component of a mobile robot is its locomotion system, which may include articulated bodies, legs, wheels, or tracks. The type of locomotion system to use must be determined based on the specific application domain in which the robot will be deployed. In outdoor scenarios with irregulars and slipping surfaces, legs and tracks may be the best choise. In indoor scenarios like warehouses, instead, wheels are the most commonly adopted solutions.

The design proposed in this thesis consists of a manipulator mounted on top of a wheeled cart. The cart weights over $250\,Kg$, its dimensions are $1.05 \times 0.61 \times 1.05\ m$ ($L \times W \times H$), it can be moved manually thanks to the four caster wheels with brakes, it is empty inside for inserting required equipment, e.g. the manipulator control box, and it was equipped with a frontal teach pendant holder. The non-mobile platform was adopted as an experimental testing platform for evaluating different manipulator mounting configurations and testing the manipulator capabilities in depalletization tasks. In Figure 3.2 the two possible mounting configuration for the collaborative manipulator are shown. The configuration A is located in the center of the cart, while

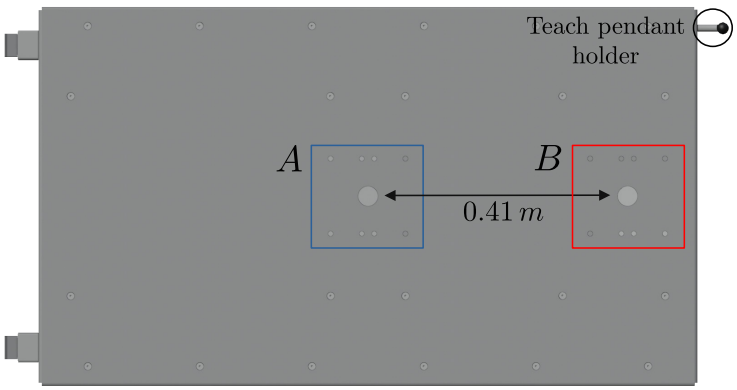


Figure 3.2: Top view of the wheeled cart CAD: A and B are the possible collaborative manipulator mounting configurations.

B is shifted of $0.41\,m$ towards the short edge (which is the mounting configuration used in 3.1).

The leading products of the italian company that supported this doctoral research, the E80 Group S.p.A., are LGVs and AGVs (Figure 3.3). In future, the cart may be replaced with the *Ant*, their smallest mobile robot with a payload of up to $1500\,Kg$ and a speed of $0.8\,m/s$.

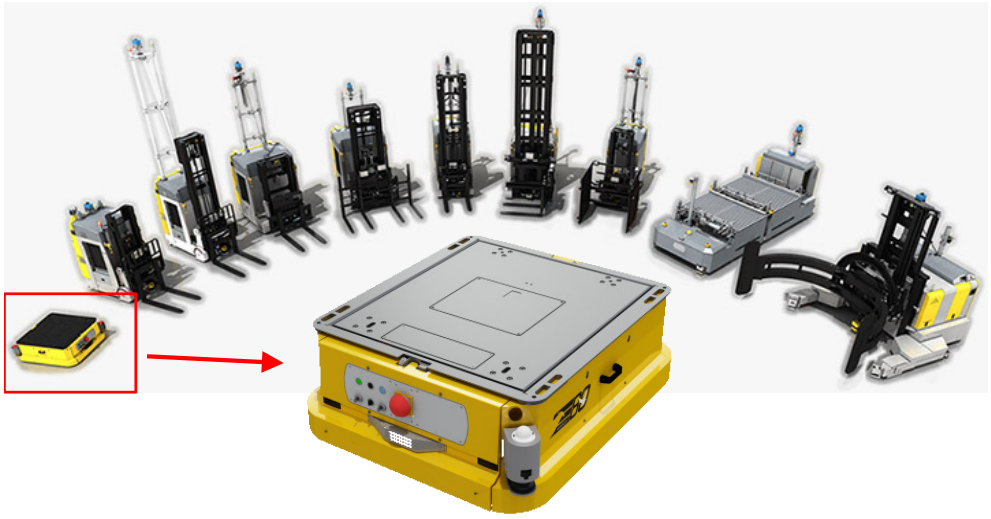


Figure 3.3: E80 Group S.p.A. AGVs and LGVs. The AGV highlighted in the center of the image is the *Ant*.

3.2 Collaborative Manipulator

In order to perform depalletization tasks a manipulator is required. Robot manipulators can be classified based on the number of degrees of freedom, the payload, the speed, the reach, the weight and the dimensions. Moreover, a further distiction can be made if a manipulator is collaborative, i.e. if it is designed to cohexist with human operator ensuring safety while moving. Non-collaborative robots, e.g. industrial

robots, are meant for high speed repetitive tasks. Typically, industrial robots are programmed for executing predefined trajectories and are surrounded by cages to ensure safety in the working environment. Collaborative manipulators, instead, are meant to be flexible. Thanks to smooth geometries, reduced speeds, lower payloads and internal sensors, cobots do not require cages and can be easily adopted for performing a larger variety of tasks, even in dynamic environments. In Figure 3.4 an industrial ma-

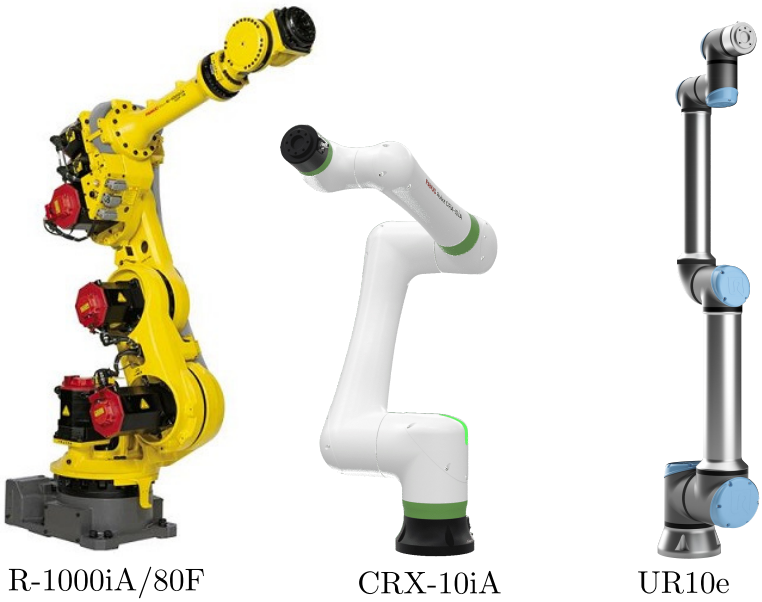


Figure 3.4: From left to right: industrial manipulator Fanuc R-1000iA/80F, cobot Fanuc CRX-10iA and cobot Universal Robot UR10e.

nipulator and two collaborative manipulators are shown. As it is possible to notice, cobots are designed with smoother edges and rounded contours compared to traditional industrial robots. This design feature enhances safety by minimizing the risk in case of accidental collisions with humans in shared workspaces, reflecting their collaborative nature.

The most renowned collaborative manipulators are manufactured by companies

such as Universal Robot, Fanuc, Doosan and Yaskawa, all of which were evaluated in simulation, as discussed in Chapter 1. The evaluation underscored the importance of selecting collaborative manipulators based on the specific depalletization tasks they need to perform. For instance, larger cobots are essential while depalletizing higher pallets, while smaller cobots are more suitable for lower pallets.

For the purposes of this dissertation, a Universal Robot manipulator was selected due to its favorable performance as demonstrated by the results of the simulated depalletization tasks in Chapter 1. Additionally, Universal Robots offers an interface known as RTDE for robot control, which is utilized by existing software to control Universal Robots manipulators with high-level programs, such as Python scripts. Section 3.2.1 will provide a detailed description of the robot used in this thesis, highlighting its capabilities and technical specifications, while Section 3.2.2 will focus on the external hardware provided with the robot.

3.2.1 Universal Robot UR10e, characteristics

The UR10e was selected over newer robots with larger payloads, despite their availability, due to its versatility. It offers adequate reach capabilities and a good payload within compact dimensions. The UR10e is a 6-axis robot, with a payload of $12.5\,Kg$, a reach of $1300\,mm$ and a weight of $33.5\,Kg$. In Figure 3.5 the robot joints and their position limits are shown. The first two joints (Base and Shoulder) have a maximum speed of $\pm 120^{\circ}/s$, while the remaining joints have a maximum speed of $\pm 180^{\circ}/s$. The flange of a UR10e robot arm includes input and output ports for communication with external devices and tools. The UR10e flange features include 2 digital inputs, 2 digital outputs, 2 analog inputs, and a tool I/O power supply voltage of $12/24\,V$, capable of supplying $2\,A$ (dual pin) or $1\,A$ (single pin).

The UR10e flange features an internal force-torque sensor. This sensor provides a force measurement range of up to $100\,N$ with a precision of $5\,N$ and an accuracy of $5.5\,N$. For torque, it can measure up to $10\,Nm$ with a precision of $0.2\,Nm$ and an accuracy of $0.5\,Nm$. The force-torque sensor allow the robot to sense and respond to forces applied during tasks such as assembly, handling delicate objects, or adjusting to varying payloads.

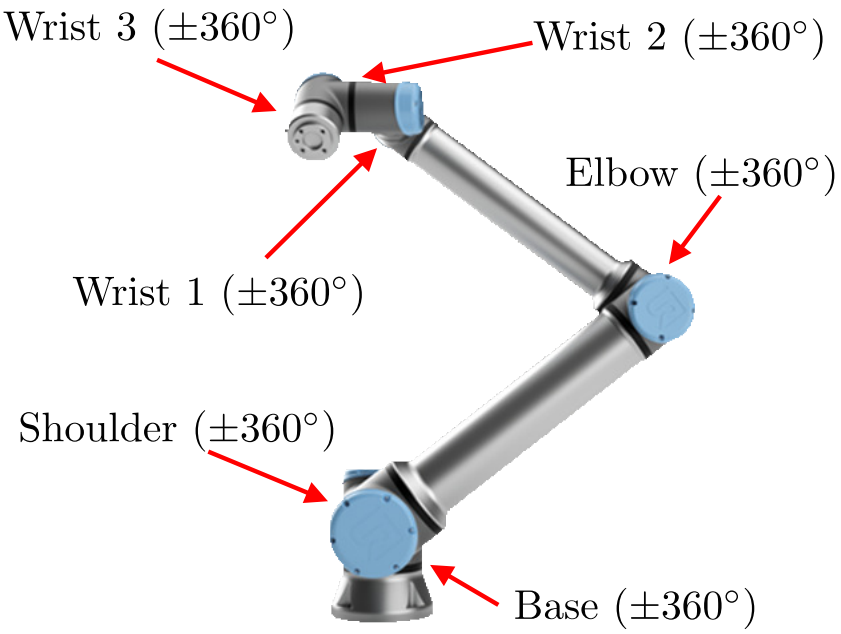


Figure 3.5: UR10e joints with minimum and maximum joint limits.

3.2.2 Universal Robot UR10e, supplementary hardware

An Universal Robot collaborative manipulator comes with a control box and the teach pendant (Figure 3.6). The Universal Robot control box is the core unit responsible for the robot functionality and movements. Inside of it are located the power supply, safety systems, and interfaces for connecting external devices. Instead, the teach pendant is a handheld device which can be used to interact with the robot. There are two types of control box: a standard type which require $100-240\,V$ AC,$(47-440\,Hz)$ as input voltage and an Original Equipment Manufacturer (OEM) version, which requires a $24-48\,V$ DC. Both version feature 16 digital inputs and outputs, 2 analog inputs and outputs, and 4 quadrature digital inputs, with an I/O power supply rated at $24\,V$ and $2\,A$. The prototype shown in this chapter adopts the standard type. However, for future applications where the wheeled cart will be replaced with a mobile robot, the OEM controller will be necessary.

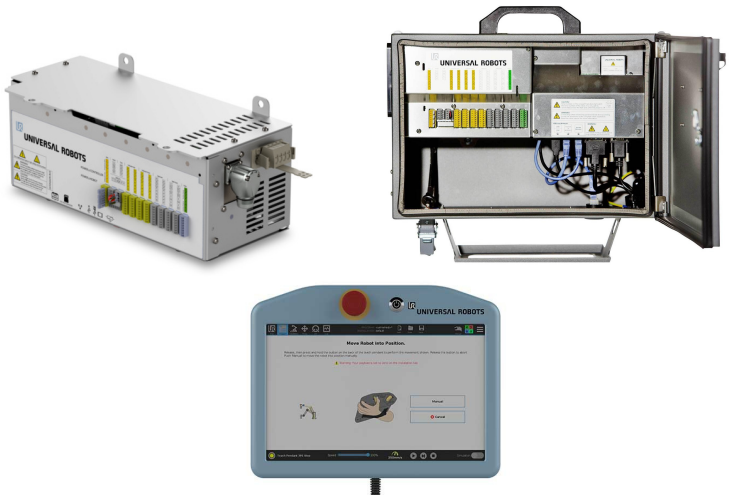


Figure 3.6: Universal Robots control box (OEM on the left, standard on the right) and teach pendant.

3.3 Vacuum-Based Gripping Systems

Objects can be manipulated using a gripper, i.e. a mechanical or robotic device typically mounted on the robot flange. A gripper is a device specifically designed to manipulate objects. Common gripper types include parallel jaw grippers, vacuum grippers, finger grippers, pneumatic grippers, electromagnetic grippers, and hydraulic grippers. These grippers differ on aspects such as the physical principles they use for manipulation, the number of degrees of freedom, and their energy consumption. Since they have become increasingly popular due to their versatility and efficiency in industrial automation, in this prototype a vacuum gripper was used. A vacuum gripper works by creating a suction force and lift objects. The vacuum is generated using cup or pad, e.g. foam pad, that seals against a surface of an object. The main advantages of using the vacuum principle are:

- Capability to generate significant suction forces and efficiently lift heavy objects with low power consumption.

- Ability to lift objects from exposed surfaces, such as grasping parcels on pallets from the side or above.
- Capability to handle a wide variety of soft and rigid objects differing in size, weight, and material.
- Versatility in handling various materials by adopting different types of suction cups or pads.

In Section 3.3.1 four models of vacuum grippers are compared, in Section 3.3.2 the design of the selected vacuum gripper and its components are discussed and in Section 3.3.3 the electrical connection between the gripper and the Universal Robot control box is shown.

3.3.1 Comparison of commercial vacuum grippers

An experimental evaluation of vacuum grippers was conducted testing four different models (Figure 3.7): Coval CVGL, OnRobot VGC10, OnRobot VGP20, and the Schmalz vacuum generator GCPi used with the PXT modular gripper system. The evaluation involved tests in which a human operator manually grasped cardboard boxes and tissue items from both their top and side faces. The purpose was to evaluate how effectively each gripper could handle different types of objects in varying orientations, providing insights into their practical usability, adaptability across different applications, ease of use and power consumption.

The Coval CVGL is a versatile gripper that can be customized in terms of dimensions, types of cups and pads, to suit different applications. It successfully grasped the target objects in all the tested orientations during evaluation. However, to generate vacuum, it requires a compressor that injects compressed air into a venturi system. High-power compressors are required to achieve better performance and they may not be optimal if the gripper is powered by the batteries of a mobile manipulator.

OnRobot VGC10 and VGP20, on the other hand, emerged as the optimal choice if the gripper must be powered by a mobile robot. These lightweight, plug-and-play vacuum grippers are designed for compatibility with collaborative manipulators, such

Coval CVGL

OnRobot VGC10

OnRobot VGP20

Schmalz PXT modular gripper system

Schmalz GCPi vacuum generator

Figure 3.7: Vacuum grippers evaluated grasping cardboard boxes and tissue items from both their top and side faces. The Schmalz PXT and GCPi systems were used together into a unified vacuum gripper.

as Universal Robot robots. They integrate the vacuum generator and can be power-supplied with the robot flange pins. Despite testing with multiple suction cups, OnRobot vacuum grippers were not able to grasp all the target objects effectively.

Good performance were obtained with the gripper composed of the PXT modular gripper system and the GCPi vacuum generator. This combined setup achieved successful grasping of all target objects from both top and side orientations. The Schmalz solution, similar to the Coval CGVL, requires an external component: the GCPi vacuum generator. However, unlike the Coval CGVL, the GCPi operates on a $24\,V$ DC power source, which is commonly available in mobile robot batteries, enhancing its suitability for mobile applications. Following the evaluation, the Schmalz vacuum generator GCPi and the PXT modular gripper system were selected to assemble the vacuum gripper for the prototype.

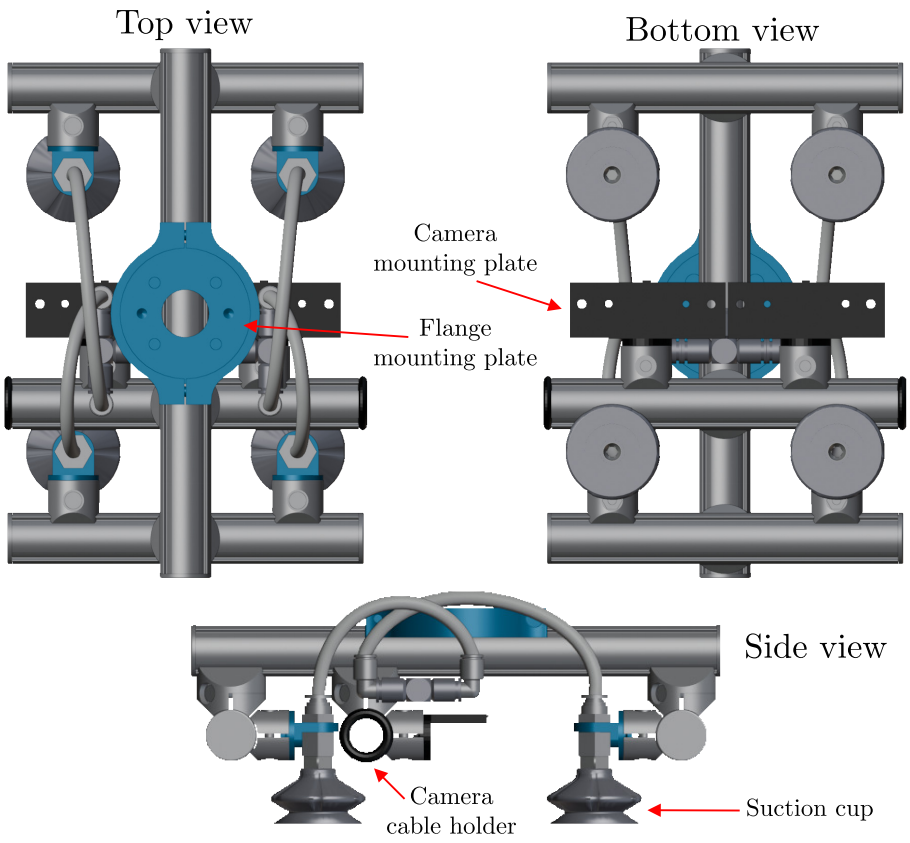


Figure 3.8: Vacuum-based gripping systems built with the Schmalz PXT modular gripper system.

3.3.2 Design and components of the Schmalz vacuum gripper

The prototype vacuum-based gripping systems is composed of the Schmalz vacuum generator GCPi and the PXT modular gripper system, as shown in Figure 3.8. The PXT kit consists of a mounting plate for the robot flange, aluminum profiles and suction cups, and it was used to build the gripper shown in Figure 3.8. The gripper is composed of a main aluminum profile of 30 cm and three aluminum profile of 20 cm. Suction cups were attached to the outer aluminum profiles, while camera mounting

plates were connected to the middle one (details about the camera mounting plates in Section 3.4). The active component of the gripper is the GCPi vacuum generator. The GCPi dimensions are $236 \times 102 \times 190\,mm$, it weights $3.22\,Kg$, it has a maximum pumping speed of $46\,l/min$ and it can be power-supplied with $24\,V$ DC. The generated vacuum is conveyed with a tube to the suction cups attached to the PXT. Schmalz offers a range of suction cups featuring various materials and dimensions. For the prototype described in this chapter, three specific types have been selected (Figure 3.9): SPB1-50, SFB1-40 and PSPF-33. Key considerations for suction cups include

Figure 3.9: Schmalz suction cups selected for the prototype.

material hardness, measured on the Shore A scale, and outer diameter. The main differences of the three types of suction cups are listed below.

- **SPB1-50**: it has an outer diameter of $50\,mm$, a material hardness of 65 Shore A and it is designed for cardboard boxes with intrinsic stability, i.e. cardboard boxes that ensure reliable protection and structural integrity during transporta-

tion and storage.

- **SFB1-40**: it has an outer diameter of 40 *mm*, a material hardness of 55 Shore A and ensures secure adhesion without damaging delicate material as flexible and thin cardboard.
- **PSPF-33**: it has an outer diameter of 33 *mm*, a material hardness of 30 Shore A and it is designed for thin, unstable, and liquid-filled packaging.

3.3.3 Gripper Electrical Connection Scheme

In Figure 3.10 the gripper electrical connection scheme is shown. The vacuum gener-

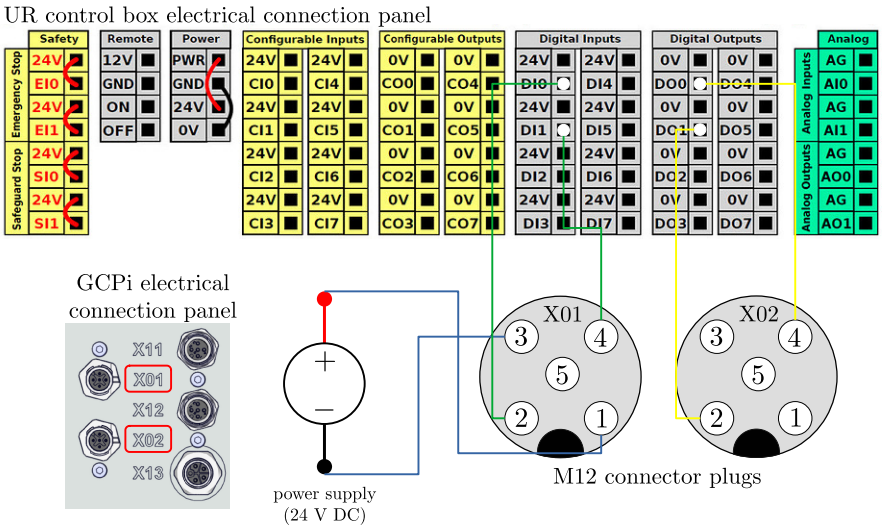


Figure 3.10: Electrical connection scheme for the GCPi vacuum generator.

ator electrical connection panel is composed of five M12 plug connectors. M12 cables were used to connect an external power supply (24 *V* DC) to the GCPi and interface GCPi digital I/O pins with the Universal Robot control box digital I/O pins. The X01 connector is responsible for supplying power to the GCPi and forwarding feedback signals to the control box. When DI0 I/0 digital input is set to 24 *V* it indicates a sys-

tem error. When DI1 is activated, it indicates that an object was successfully grasped by the vacuum gripper, i.e. the GCPi was able to generate enough vacuum on the object surface and no significant pressure loss were detected. Conversely, the X02 connector was used to send signals from the control box to the GCPi, allowing for the activation and deactivation of the vacuum generation. Sending $24V$ at DO0 the vacuum generation is activated. Instead, sending $24V$ at DO1 air is released through the vacuum tube, allowing the suction cups to detach from an object.

3.4 Vision System

Sensors enhance robot capabilities by enabling it to perceive the surrounding environment. Commonly used sensors include radars and lidars, for obstacle avoidance, as well as cameras, for object detection. Since the objective of the prototype is to solve depalletization tasks, requiring the detection of pallet objects, a vision system was adopted. In particular the protype was equipped with two depth cameras: Intel Realsense D435 and a Luxonis OAK-D Pro (Figure 3.11). The D435 has stereo cam-

Intel Realsense D435 Luxonis OAK-D Pro

Figure 3.11: Depth cameras mounted on the collaborative prototype platform.

eras, an RGB sensor and an infrared projector. Similarly, the OAK-D Pro features stereo depth cameras, an RGB sensor and an infrared illuminator. In addition, the OAK-D Pro integrates the Myriad X VPU allowing for on-device artificial intelligence processing.

The depth cameras were mounted directly on the gripper using mounting plates

specifically designed for the prototype described in this Chapter (Figure 3.12).

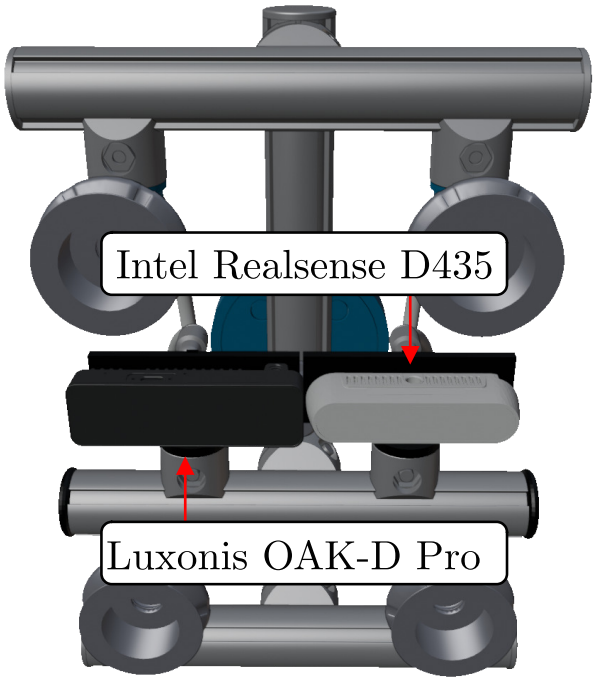


Figure 3.12: Gripper with Intel Realsense D435 and Luxonis OAK-D Pro.

The working principle of a camera mounting plate is shown in Figure 3.13. Inspired by the Schmalz PXT suction cups connectors, a camera support was designed to be printed with a 3D printer. The support is compatible with aluminum profile connector, allowing it to be attached to the gripper. This design allows for securely positioning one or more cameras along any aluminum profile and enables reconfiguration of the cameras at any time. For example, in Chapter 4 a single-camera collaborative prototype platform was used to evaluate an experimental strategy for minimizing placement error in palletization tasks. Althought the support shown was adapted to be compatible with the Intel Realsense D435 and the Luxonis OAK-d pro,

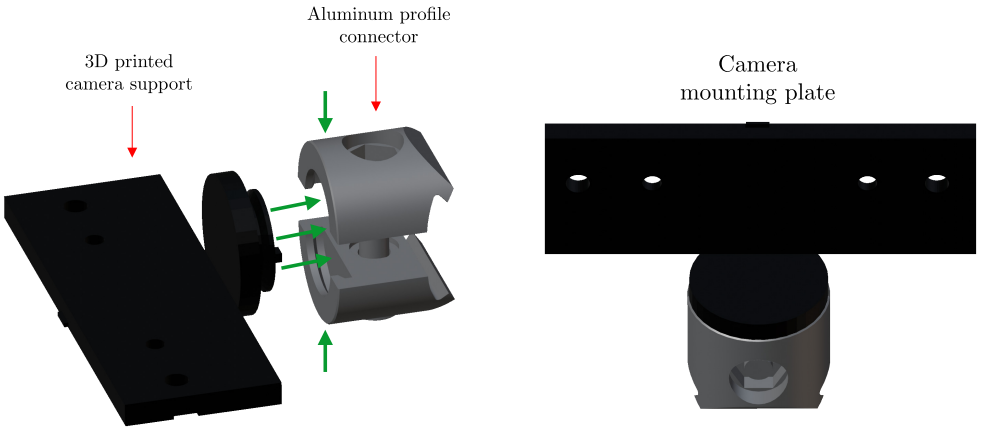


Figure 3.13: 3D printed camera support which will be inserted in an aluminum profile connector to compose the camera mounting plate.

the dimensions and holes of the support can be adjusted for different camera models.

Chapter 4

Contact-based in-Hand Package Pose Estimation Using a Collaborative Robot

One of the most important objectives in automated robot manipulation tasks is the maximization of accuracy while placing a grasped object. This research aims to minimize the placement error of cardboard boxes in manipulation tasks such as industrial palletization, using a collaborative robot. An accurate object placement in palletization tasks reduces the chance of collision between a grasped package and other packages already deployed on the pallet being assembled and, therefore, it reduces the required gap between boxes, maximizing space usage. Usually, object placement accuracy is affected by a displacement error between the expected and the actual grasp pose (Fig. 4.1). Approaches that deal with displacement errors can be subdivided into pre-grasp strategies, which operate before grasping the object, and post-grasp strategies, which operate while the object is already attached to the end-effector.

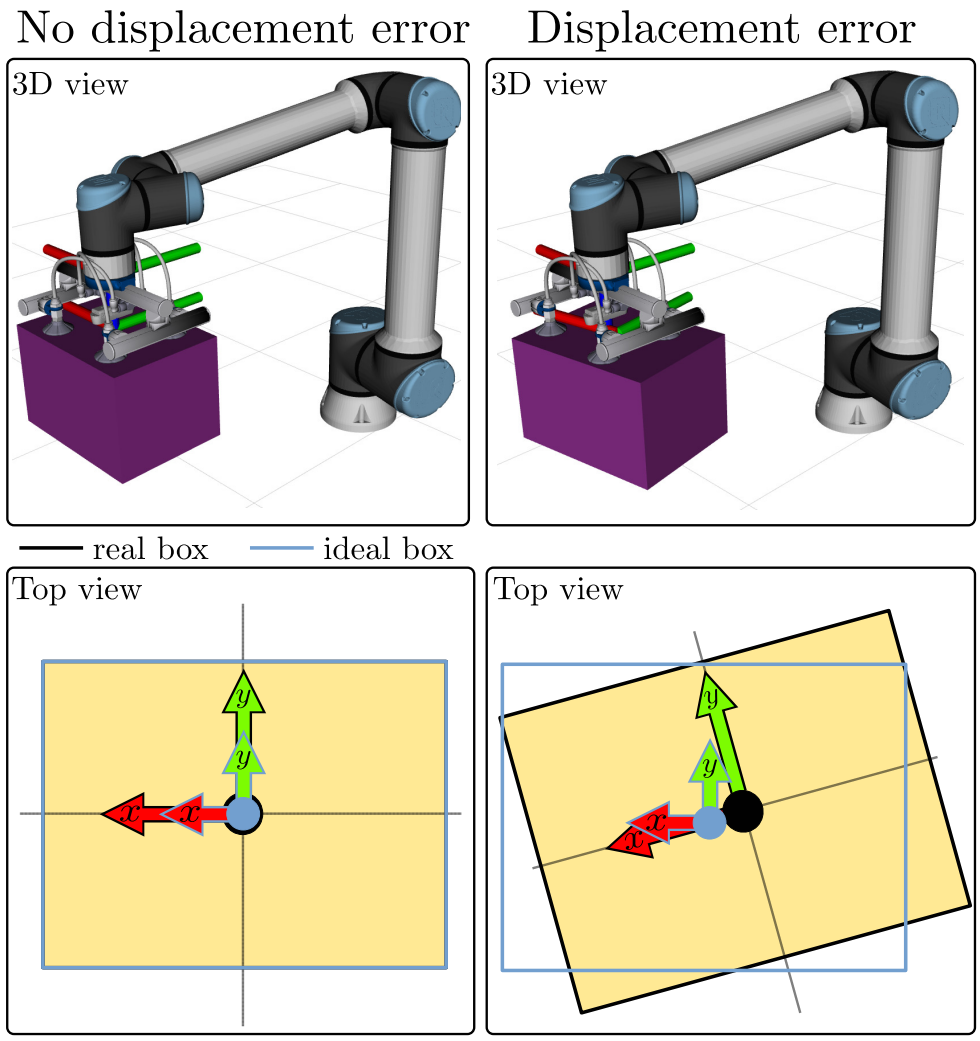


Figure 4.1: Example of a simulated robot grasping a cardboard box with no displacement error (left) and a non-zero displacement error (right). When displacement error is present, the real box (black rectangle as seen from the top) is misaligned with respect to the ideal planned configuration (light blue rectangle).

In pre-grasp strategies, standard cameras or depth sensors are often adopted to reduce the displacement error by estimating the object pose before grasping. However, although object pose estimation can be performed using highly accurate sensors and by applying advanced perception algorithms, uncertainties in the grasping process can still lead to a significant displacement error. Hence, "post-grasp" strategies have been proposed that estimate the displacement error while the target object is already grasped. In particular, vision-based, depth-sensing-based, tactile-based and sensorless techniques are promising post-grasp solutions that can be adopted. Inspired by human dexterity [47], other post-grasping approaches involve physical interactions between the object and fixed mechanical structures, called fixtures or jigs [48]. Such fixtures are used by the robot to constrain the movement of the object with respect to the environment in a repeatable and predictable manner. For example, in industrial environments a standard solution that works for boxes involves an inclined plane with a termination at the bottom end. The grasped box is dropped by the robot on the inclined plane and then it slides to the bottom due to gravity. Hence, the box always stops in a known configuration, where it is re-grasped by the robot. This solution has some drawbacks: it requires re-grasping and is not easily adaptable to packages of different size, as the mechanical structure requires a custom design. Moreover, the inclined plane is a bulky structure, and it may not work well for non-stationary robots like mobile manipulators [49].

In this Chapter a novel in-hand post-grasp approach for object displacement estimation is proposed where the robot induces collisions between the grasped object and a minimal footprint fixture, i.e. a single peg with known position with respect to the manipulator. In the proposed implementation, the peg has been manufactured as a small cylinder, affixed to the robot support platform. However, in principle any suitable corner in the environment may be used. Information from multiple induced collisions is used to estimate the displacement error between the grasped object and the robot gripper by solving an optimization problem. Collisions occur at slow speed to prevent the package, and its content, from being damaged. The system was evaluated using a collaborative robot, equipped with a force-torque sensor to detect collisions, in a palletization tasks with cardboard boxes of different weights.

4.1 Related Work

In literature there are several works that deal with the issue of displacement error of grasped objects.

The main problem of pre-grasp strategies is that the result depend on sensor calibration. Moreover, uncertainties during the grasping process are not taken into account. An example pre-grasp pose estimation algorithm for cardboard boxes was proposed in [50]. This approach was also used in this Chapter for initial pose estimation.

In post-grasp strategies the pose of the object is estimated after grasping. These methods can adopt vision, depth, and tactile sensors or they can exploit physical interactions between the grasped object and special environment fixtures. Furthermore, post-grasp strategies can be classified depending on whether sensors are mounted on the robot end effector or not. Finally, one can distinguish between error-correction strategies, which actively correct the displacement error, and error-estimation strategies, that only estimate the displacement while the object is attached to the gripper. Error-correction strategies often make use of regrasping as the object is first placed on a temporary support and then it is grasped again in a different pose. However, regrasping delays the task.

Examples of error-correction strategies were presented in [51], [52] and [53]. Hu et al. [51] presented a sensor-less strategy where the robot regrasps the object after dropping it on a triangular corner fixture. Von Drigalski et al. [52] proposed a sensorless regrasping strategy that uses interactions between the gripper, the target object and the environment. Using three actions: grasp, place and push, the target object pose is adjusted based on its known shape. An example of vision-based approach was proposed in [53]. First, the object is picked and placed in the desired placement location. Then, if a displacement error is detected by the camera, the object is pushed by the gripper. The process is repeated until the displacement error drops below a threshold.

Post-grasp strategies that estimate the displacement error while the target object is attached to the gripper may adopt tactile, depth and vision sensors. In [54] and [55] two custom grippers for displacement error estimation have been proposed.

Zhiyuan et al. [54] proposed a solution based on several monocular cameras mounted on the gripper. Fengchun et al. [55] proposed an approach based on a custom gripper equipped with eight lasers used in combination with a fixture, likewise endowed with eight lasers. Gao et al. [56] developed a custom gripper that combines RGB cameras and visuotactile sensors. The work by Galaiya et al. [57] collects data from compliant tactile sensors and it estimates the displacement orientation error with an LSTM neural network. The authors in [58] proposed a soft jig, that enables object pose estimation thanks to a soft membrane. Such custom equipment are very accurate but they are not very flexible and adaptable. Zhao et al. [59] proposed a method involving two convolutional neural networks to predict grasp robustness and post-grasp object displacements using depth images. In [60] an approach using an external vision system was adopted, as well as torque sensors. Contact-based estimation algorithms were presented in [61], [62], [63] and [64]. They estimate the displacement error by letting a target object collide with the environment while still attached to the gripper. Contacts information, simulated environments and particle filters were used to infer the displacement error. In particular, von Drigalski et al. [61] used a force-torque sensor to detect contacts during physical interactions. The approach is similar to the strategy proposed in this Chapter in that it uses contact information to determine displacement error of the object. However, a particle filter and a collision library were applied, which add a significant computation overhead compared to the closed-form approach proposed in this thesis. Moreover, in [61] the experimental evaluation considered seven sequential contacts of the object with the environment. Instead, in this dissertartion the approach is evaluated with as few as one contact, to minimize cycle time. In [62] the approach by von Drigalski et al. [61] was extended to include an external camera. Pankert et al. [63], instead of using sensor-fusion, adopted reinforcement learning to decide how the object should interact with the environment. Sipos et al [64] adopted two particle filters, the first estimates contact locations while the second estimates the object pose.

The proposed approach does not require simulations, additional sensors and complex fixtures. It is an in-hand post-grasping displacement estimation strategy which requires a minimal footprint fixture and a collision-detection system.

4.2 Method

Given a destination pallet, a robot manipulator $\mathscr{R}$, a gripper $\mathscr{G}$ and a set of manipulable box-shaped objects $O = \{o_1, o_2, .., o_n\}$, a palletization task is defined as the process of relocating each object $o_i \in O$ from its initial pose to a final deposit location on the pallet. As shown in Fig. 4.2, robotic palletization tasks were considered

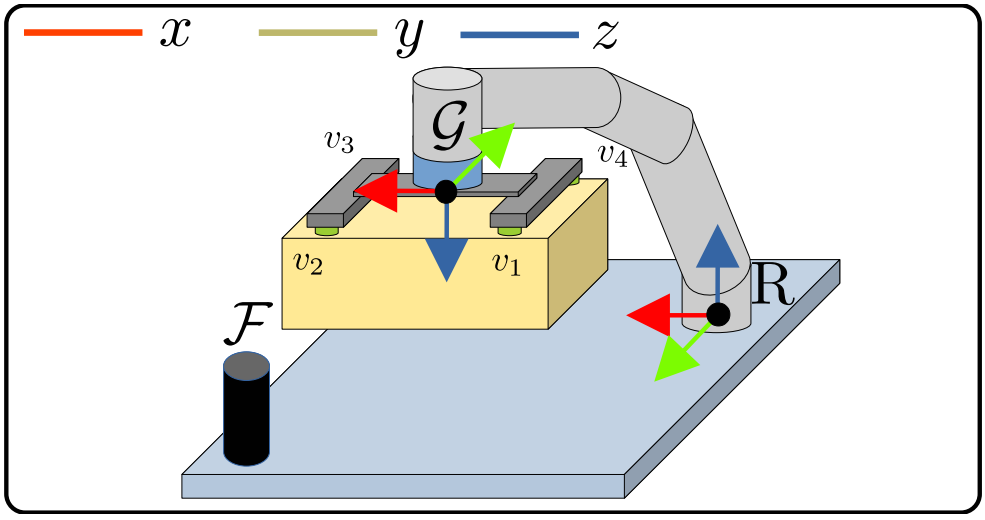


Figure 4.2: Simulated industrial scenario where the robot is grasping a package from the top face. The gripper reference frame is located on the end-effector, while the robot reference frame is at the robot base.

where 1) the gripper is able to grasp an object o_i from the top face, 2) the gripper is aligned parallel to o_i top face, and 3) there is no vertical displacement error. Under these assumptions the displacement error D_E is defined as a relative rigid transformation between the actual and the expected pose of object o_i when attached to $\mathscr{G}$, i.e.

$$D_E(e_x, e_y, e_\theta) = \begin{bmatrix} \cos(e_\theta) & -\sin(e_\theta) & e_x \\ \sin(e_\theta) & \cos(e_\theta) & e_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

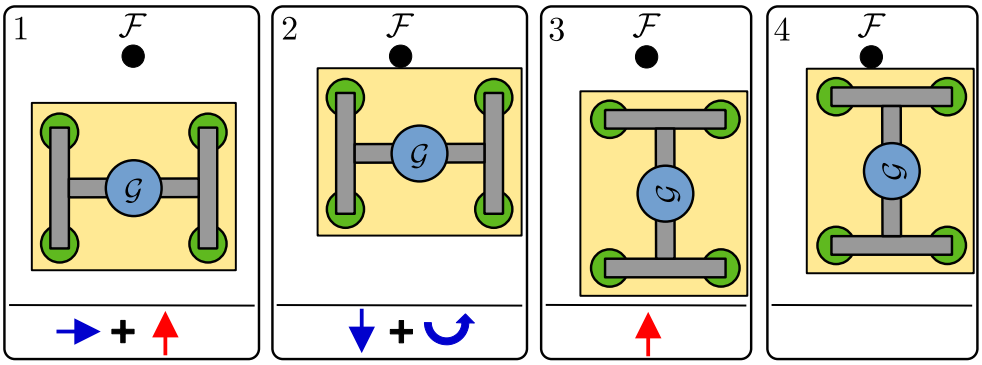


Figure 4.3: Images of sequential robot actions (from left to right) applied to induce collisions between a fixed peg $\mathscr{F}$ and the grasped object o_i (top view). Arrows represent robot actions being executed: *Move* and *Rotate* (blue arrows), *Collide* (red arrow).

As reported in equation 4.1, the displacement error is a roto-translation matrix expressed in the gripper reference frame with three unknowns: the translation component, described by e_x and e_y, and the rotation angle e_θ of o_i around the yaw axis. In the proposed method the robot is able to estimate D_E inducing collisions of the grasped object with a minimal footprint fixture $\mathscr{F}$. The algorithm requires knowledge about the position of $\mathscr{F}$ in the robot frame and the vertices of o_i in an ideal grasp pose with respect to the gripper reference frame. Moreover, the system must be provided with a collision-detection system. In the proposed approach, the fixture $\mathscr{F}$ is approximated with a single point ${}^{\mathscr{R}}\vec{p}_{\mathscr{F}} \in \mathbb{R}^3$. The approximation requires a calibration procedure as explained in the experiments (Section 4.3.1).

4.2.1 Induced collisions

Collisions between the grasped object and the peg fixture are induced by moving the robot, and consequently o_i, towards the fixture $\mathscr{F}$. Due to the symmetrical shape of the peg fixture, the object can collide with $\mathscr{F}$ from any direction around its principal

axis. In order to reduce the execution time, the robot was programmed to approach $\mathscr{F}$ along a predefined direction and to keep the object orientation horizontal. Three robot predefined movement, defined as actions, are applied to induce collisions: *Move*, *Rotate* and *Collide*. The *Move* action can be used to move o_i forward, backward and laterally, without changing its orientation. *Rotate* can be used to rotate o_i around the yaw axis. *Collide*, instead, can be used to move the grasped object o_i towards $\mathscr{F}$ until a collision occurs. Contact between the grasped object and the fixed peg is determined when force-torque data on the gripper exceed pre-defined thresholds. An example action sequence that induces two contacts is shown in Figure 4.3. Starting from the initial configuration (image 1), the robot moves the grasped object to the right and then forward, resulting in the configuration illustrated in image 2 where a collision occurs. Subsequently, the object is moved backward and then it is rotated. Finally, o_i is moved forward again until another collision occurs.

When a collision is detected, the transformation matrix ${}^{\mathscr{G}}_{\mathscr{R}}M$ between $\mathscr{R}$ and $\mathscr{G}$ is used to convert the collision point ${}^{\mathscr{R}}\vec{p}_{\mathscr{F}}$ in the gripper reference frame: ${}^{\mathscr{G}}\vec{p}_{\mathscr{F}} = {}^{\mathscr{G}}_{\mathscr{R}}M\,{}^{\mathscr{R}}\vec{p}_{\mathscr{F}}$. The estimation algorithm, which will be discussed in Section 4.2.2, solely relies on x and y coordinates of ${}^{\mathscr{G}}\vec{p}_{\mathscr{F}}$. These coordinates represent a *contact point*, which will be indicated as $c_p \in \mathbb{R}^2$ from now on.

4.2.2 Displacement error estimation

The algorithm requires as input the set of contact points C=$\{c_{p_1}, c_{p_2}, ..\} \subseteq \mathbb{R}^2$ resulting from the sequence of induced collisions, and the vertices of the grasped object in the ideal (planned) configuration (expressed in the gripper reference frame). For simplicity, ignoring the object vertical dimension, the set of four vertices of the ideal grasped object configuration are defined as $V = \{v_1, v_2, v_3, v_4\} \subseteq \mathbb{R}^2$, where v_k is a point along x and y axes in the gripper reference frame. Given V, it is also possible to define the set of four edges $S = \{\overline{v_1v_2}, \overline{v_2v_3}, \overline{v_3v_4}, \overline{v_4v_1}\}$ of the ideal object configuration. Each contact point $c_{p_j} \in C$ can be associated with an object edge $s_j \in S$ based on their point-segment distance. For each object edge $s_k \in S$, the nearest point to c_{p_j}, denoted as p_k, is found. The closest object edge s_j is then determined as the edge whose p_k is closest to c_{p_j}. However, if a non-zero displacement error is present,

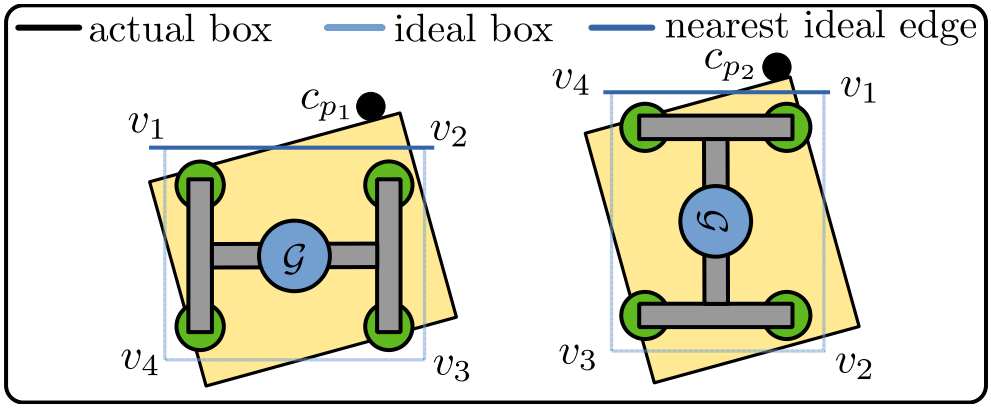


Figure 4.4: Actual contact points (c_{p_1}, c_{p_2}) and expected vertices (v_1, v_2, v_3, v_4) of the ideal (planned) package configuration in the gripper reference frame. Due to a non-zero displacement error the contact points do not perfectly lie on the nearest expected edges of the ideal package configuration.

$c_{p_j} \neq p_k$ and thus the actual contact point will not perfectly lie on the ideal edge s_j (as shown in Fig. 4.4).

In this thesis, the displacement error is estimated by solving a non-linear least squares optimization problem. Given a *line* function (Eq. 4.2), which provides the three coefficients of the line equation that passes through two points $v_k, v_l \in \mathbb{R}^2$, i.e.

$$\text{line}(\overline{v_k v_l}) = \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \frac{v_{k,y} - v_{l,y}}{d} \\ \frac{v_{l,x} - v_{k,x}}{d} \\ \frac{v_{k,x} v_{l,y} - v_{l,x} v_{k,y}}{d} \end{bmatrix} \tag{4.2}$$

where $d = \sqrt{(v_{k,y} - v_{l,y})^2 + (v_{l,x} - v_{k,x})^2}$, the optimization problem can be defined as

$$\arg\min_{e_x, e_y, e_\theta} \sum_{j \in [1,..,|C|]} \left(\text{line}(s_j)^T D_E \begin{bmatrix} c_{p_j} \\ 1 \end{bmatrix} \right)^2 \tag{4.3}$$

The operation $(\text{line}(s_j)^T D_E \, [c_{p_j} \, 1]^T)$ computes the distance between a line that passes through the s_j vertices and a contact point transformed into the grasped object

reference frame by D_E. Basically, the optimization problem aims to find the optimal parameters e_x, e_y, e_θ that, if applied to C, minimize the sum of the squared distance of each contact point in C to the corresponding closest edge of the ideal grasped object configuration.

The optimization problem is solved using a non-linear optimizer. In under-determined problems, i.e. with less than three contact points, it is necessary to add a consistency error term to Eq. 4.3. That is, the algorithm adds to the objective function the term $(\varepsilon_\theta e_\theta)^2 + (\varepsilon_x e_x)^2 + (\varepsilon_y e_y)^2$, where ε_θ, ε_x, ε_y are positive real values close to 0. This consistency error term ensures convergence of the under-determined problems by penalizing solutions too different from an identity transformation, under the assumption that the object has not been grasped too far from the ideal pose.

4.3 Experiments

Experiments were conducted in palletization tasks where a robot manipulator (Fig. 4.5) relocates cardboard boxes (Fig. 4.6) from a source pallet to a destination pallet (Section 4.3.3).

The experimental setup includes a Universal Robot UR10e equipped with a force-torque sensor, a vacuum gripper (Schmalz GCPi vacuum generator and PXT modular gripper system), and an eye-in-hand camera (Intel RealSense D435) for object detection. All cardboard boxes have a size of $0.315 \times 0.23 \times 0.2$ m ($L \times W \times H$).

The software was implemented on top of the ROS2 framework in Ubuntu 22.04 LTS. Motion planning was performed by the MoveIt planning framework. The motion of the robot was programmed using the ur_ros_rtde[1] ROS2 interface for ur_rtde[2]. The optimization problem (Eq. 4.3) was solved by the Levenberg-Marquardt algorithm with trust regions calculated by the Ceres optimization library. The consistency error terms were configured as ε_x=0.001, ε_y=0.001 and ε_θ=0.001. The software ran on a notebook connected to the robot via Ethernet and equipped with an AMD Ryzen 7 5800HS@2.80 GHz (16 GB of RAM). A calibration procedure of the contact de-

[1] A. Saccuti, "ur_ros_rtde", `https://github.com/SuperDiodo/ur_ros_rtde`, 2024
[2] A. P. Lindvig, "ur_rtde", `https://gitlab.com/sdurobotics/ur_rtde`, 2018

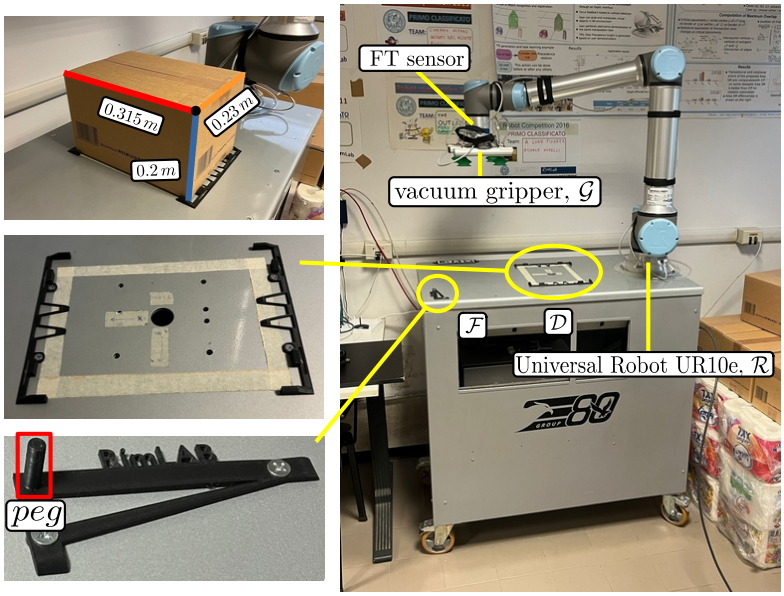


Figure 4.5: Universal Robot UR10e ($\mathscr{R}$) equipped with a force-torque sensor and a vacuum gripper ($\mathscr{G}$). Peg fixture ($\mathscr{F}$) and the deposit fixture ($\mathscr{D}$) used for measuring the ground truth displacement error.

tection system was carried out as explained in Section 4.3.1.

Specific experiments were carried out to evaluate the accuracy of the proposed displacement error correction method (Section 4.3.2). Ground truth data were measured through manual guidance thanks to a deposit fixture $\mathscr{D}$ (Fig. 4.5) of known pose. In particular, after computing the displacement error of a grasped box the robot relocated the box above $\mathscr{D}$ and paused. Then, the robot was moved manually (by a human operator) in free drive mode to place the box exactly inside $\mathscr{D}$. Hence, the ground truth displacement error (translation and rotation around the yaw axis) was computed from the applied manual correction. An example of a box before and after the manual correction procedure is shown in Fig. 4.7.

A video of the palletization task and of the displacement error correction is available at http://rimlab.ce.unipr.it/%7ermonica/saccuti/ETFA%5F2024.mp4.

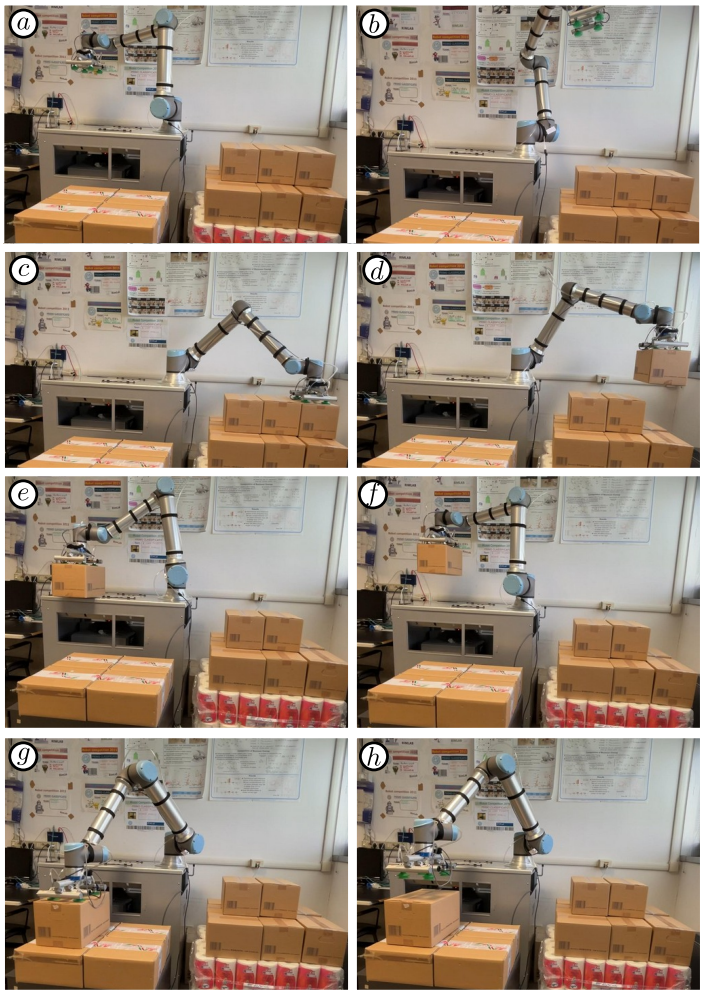


Figure 4.6: Example pick-and-place operation of a palletization task. Starting from a resting configuration (a), the robot moves above a source pallet and then it detects a box on the top layer using the eye-in-hand camera (b). The box is then picked up and moved close to the peg (c, d, e). Afterwards, the displacement error estimation procedure is performed by inducing collisions against the peg. The box is finally placed on the destination pallet (f, g, h).

Figure 4.7: Placement of a cardboard box above the deposit fixture $\mathscr{D}$ without displacement error correction (left) where it can be observed that the gripper is aligned to the deposit fixture, while the box is not. Placement of a cardboard box inside the deposit fixture $\mathscr{D}$ with manual error correction (right image) to compute the ground truth error correction.

4.3.1 Calibration of the contact detection algorithm

Contact detection between the grasped object and the peg fixture, induced to estimate the displacement error of the grasped object, was performed thanks to the robot force-torque sensor. For each collision event the robot moves the box towards the peg fixture until the torque read by the sensor exceeds a threshold $\tau_{\max}$ (Fig. 4.8). Experiments were carried out with boxes of two different weights: empty boxes (with negligible weight) and heavy boxes (weighting about 1.6 Kg). Hence, two different threshold values for τ_{max} were used: 1 Nm for experiments with empty boxes and 3 Nm for experiments with heavy boxes.

Since the *Collide* action finishes when enough reaction torque is applied to the robot, the box may be tilted when in collision with the peg fixture. As shown in Fig. 4.9, due to the inclination of the package the gripper advances slightly forward even after the collision has occurred. Hence, the contact point c_p may be incorrectly estimated on the inside the box. This effect was canceled out by scaling down the size of the box in the optimization algorithm. The scaling factors for the x and y dimensions, named f_x and f_y were estimated by measuring the position difference of

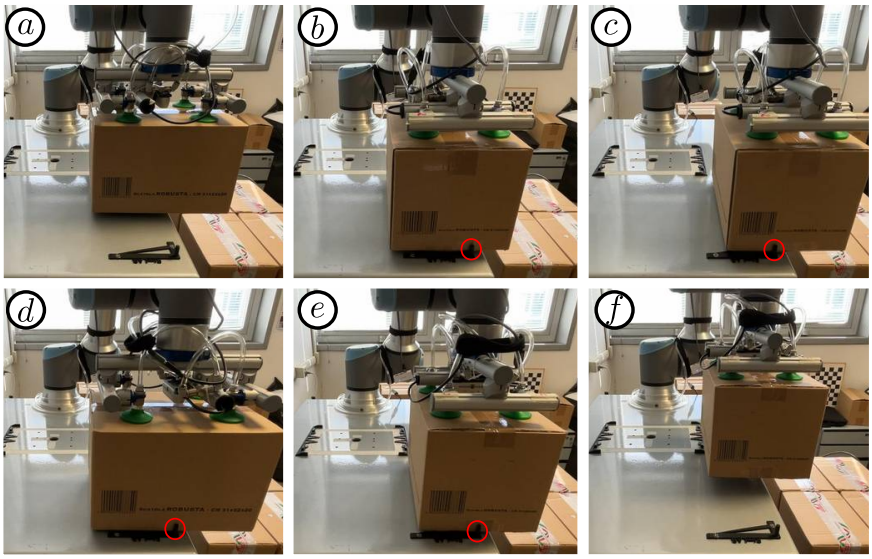


Figure 4.8: Example of displacement error estimation process. Starting from a predefined configuration (image *a*, which is also illustrated in Fig. 4.6*e*), the robot executes *Rotate* and *Move* actions to induce collisions between the box and the peg fixture. Contact points are highlighted in red.

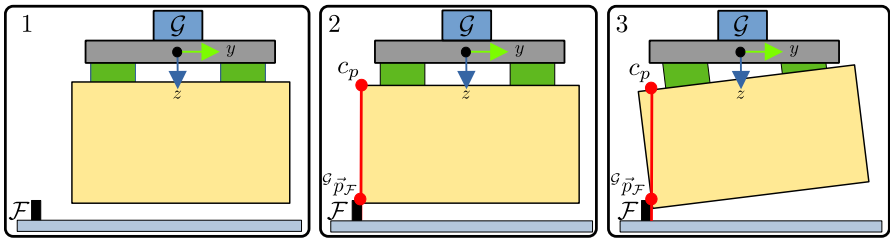
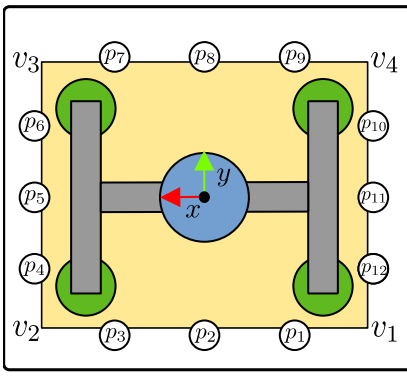


Figure 4.9: The robot, starting from a non-colliding state (image 1), induces a collision (image 2) at point ${}^{\mathscr{R}}\vec{p}_{\mathscr{F}}$. However, if the object tilts when pushed against the peg (image 3) the contact point c_p, without calibration, would be incorrectly estimated as lying inside the box (c_p is drawn for convenience on the top side of the box).

a contact point while the box was not tilted (Fig. 4.9-2) and with tilting (Fig. 4.9-3). In particular, the scaling factor f_x was estimated using two contact points on the long edge of the box, while f_y using two contact points on the short edge of the box, resulting in (f_x=6.5mm, f_y=5.5mm) for the empty boxes and (f_x=10.0mm, f_y=8.5mm) for the heavy boxes. The reduced boxes size was set to $0.315 - 2f_x \times 0.23 - 2f_y \times 0.2$ m.

4.3.2 Accuracy evaluation

The accuracy of the proposed method for displacement error estimation of grasped objects was evaluated in four modes (**E-DET**, **E-MAN**, **H-DET**, **H-MAN**) using empty or heavy boxes, and with or without the support of an in-hand perception system for object detection [50] that reduces the displacement error. manually attached



	$p_i \in P$ indexes		
M_1	2	M_{13}	$2,4,10$
M_2	5	M_{14}	$1,3,5,8$
M_3	$2,5$	M_{15}	$4,6,8,11$
M_4	$2,8$	M_{16}	$2,5,8,11$
M_5	$5,11$	M_{17}	$1,3,4,6$
M_6	$2,5,8$	M_{18}	$1,3,7,9$
M_7	$2,5,11$	M_{19}	$4,6,10,12$
M_8	$1,3,5$	M_{20}	$2,4,6,9,11$
M_9	$1,3,8$	M_{21}	$1,3,4,6,11$
M_{10}	$4,6,11$	M_{22}	$1,3,4,6,7,9$
M_{11}	$2,4,6$	M_{23}	$1,3,4,6,7,9,10,12$
M_{12}	$1,5,8$	M_{24}	$1,2,3,4,5,6,7,8,9,10,11,12$

Figure 4.10: Reference contact points $\{p_1,..,p_{12}\}$ used for the experimental evaluation (left). The 24 tested configurations of contact points $\{M_1,..,M_{24}\}$ (right) including the indices of the contact points belonging to each configuration.

to the gripper. In particular, in the **E-DET** mode empty boxes were used and the robot grasped the object by automatically aligning the gripper with the object, after initial automatic object detection. In the **E-MAN** mode empty boxes were used and a human operator attached the object to the gripper by hand, without automatic ob-

ject detection, thus leading to a large displacement error. In the **H-DET** mode heavy boxes were used with object detection. Finally, in the **H-MAN** mode heavy boxes were

Twelve reference contact points $P = \{p_1, .., p_{12}\} \subseteq \mathbb{R}^2$ were considered (Fig. 4.10, left) on the rectangular section of the box, with three points for each side, and 24 different configurations $\{M_1, .., M_{24}\}$ of contact points in P (Fig. 4.10, right) were defined for testing. Each configuration was tested in 14 trials as follows: **E-DET** (3 trials), **E-MAN** (4 trials), **H-DET** (3 trials), **H-MAN** (4 trials).

In each trial the robot induced all the collisions of the grasped object against the peg as specified by the selected configuration, by executing a sequence of *Move*, *Rotate* and *Collide* actions leading to a set of contact points close to the reference contact points.

The average ground truth displacement error for each mode is reported in Table 4.1. The displacement error was estimated using the proposed method for each trial and compared against the ground truth. The average position and rotation residual displacement errors are shown in Table 4.2, for each configuration M_i.

Table 4.1: Ground truth average displacement error for each execution mode.

	E-DET	**H-DET**	**E-MAN**	**H-MAN**
$Avg.\ GT_{x,y}\ (mm)$	7.04	3.92	28.05	31.62
$Avg.\ GT_{\theta}\ (deg)$	0.55	0.76	6.17	7.84

In general, it can be observed that increasing the number of reference points decreases both the translation ($r_{x,y}$) and rotation (r_{θ}) residual displacement errors. In particular, the highest error is obtained in under-determined estimation problems with less than 3 contact points (M_1, M_2, M_3, M_4 and M_5). For example, configuration M_2 contained only a single contact point which was located along the y axis, hence only e_x could be estimated. Nonetheless, even with a single contact point, the displacement error was reduced (i.e., 9.46 mm in **E-MAN** instead of 28.05 mm as reported in Table 4.1). In some configurations with two contact points, such as M_4 and M_5, the contact points lie on opposite sides of the box and, therefore, the estimation of the

Table 4.2: Average residual position error (mm) and average residual orientation error (deg) for each experimental configuration.

$\|M_i\|$	$M_i \in P$	**E-DET**		**H-DET**		**E-MAN**		**H-MAN**	
		$r_{x,y}$	r_θ	$r_{x,y}$	r_θ	$r_{x,y}$	r_θ	$r_{x,y}$	r_θ
1	M_1	6.23	0.55	4.50	0.76	26.55	6.17	25.53	7.84
	$\mathbf{M_2}$	**6.16**	**0.55**	**2.19**	**0.76**	**9.46**	**6.17**	**18.47**	**7.84**
2	$\mathbf{M_3}$	4.96	**0.55**	2.51	**0.76**	**7.33**	**6.17**	**6.02**	7.84
	M_4	**4.00**	0.58	3.46	0.89	24.63	6.30	24.72	10.99
	M_5	6.14	2.23	**2.22**	7.22	9.09	7.42	16.87	**4.93**
3	M_6	2.68	0.67	1.40	0.64	**2.47**	4.78	**3.75**	11.04
	M_7	5.10	2.30	3.38	7.23	5.91	4.96	4.35	13.15
	$\mathbf{M_8}$	5.30	0.68	3.21	0.68	7.95	**0.29**	4.09	1.32
	M_9	5.14	0.48	4.01	**0.61**	25.13	1.74	24.81	**0.87**
	M_{10}	6.14	1.01	2.19	1.43	8.99	2.02	16.89	2.12
	M_{11}	4.97	1.02	2.67	1.15	8.34	0.63	4.34	1.97
	M_{12}	**2.35**	8.57	**0.77**	4.90	4.79	9.80	4.73	4.13
	M_{13}	4.93	**0.31**	2.63	1.58	7.81	4.86	4.63	6.29
4	M_{14}	3.82	0.48	2.27	0.61	4.48	1.74	3.61	0.87
	$\mathbf{M_{15}}$	**0.88**	1.01	**0.65**	1.43	3.15	2.02	**2.40**	2.12
	M_{16}	2.48	0.96	1.65	2.68	**2.07**	6.20	2.88	2.36
	M_{17}	5.28	**0.28**	3.37	0.73	7.86	**0.38**	4.17	0.48
	M_{18}	4.58	0.60	3.61	**0.44**	24.67	1.43	24.76	0.72
	M_{19}	6.15	0.44	2.19	0.92	8.95	1.29	16.87	**0.57**
5	$\mathbf{M_{20}}$	**3.36**	1.45	**1.51**	**0.80**	**3.26**	2.72	**2.49**	**0.82**
	M_{21}	5.25	**0.26**	3.22	0.80	6.88	**1.31**	3.95	0.90
6	$\mathbf{M_{22}}$	**3.01**	**0.26**	**1.86**	**0.49**	**3.39**	**1.22**	**3.52**	**0.38**
8	$\mathbf{M_{23}}$	**3.01**	**0.30**	**1.62**	**0.41**	**1.93**	**1.37**	**2.84**	**0.55**
12	$\mathbf{M_{24}}$	**2.96**	**0.28**	**1.62**	**0.42**	**2.02**	**1.78**	**2.84**	**0.57**

displacement error does not improve when compared to configurations that contain a single contact point. Hence, it can be concluded that both the number and the location of the contact points affect the algorithm. Overall, when 3 contact points were used the best results were achieved by configuration M_6 in the first three modes, and M_8 for **H-MAN**. In general, over-determined configurations with more than 3 points produced the lowest residual displacement errors. Sub-degree orientation errors were obtained with M_{17} (4 points) in all the four modes. In particular, M_{15} achieved sub-millimeter accuracy in the **E-DET** and **H-DET** modes.

For each group of configurations, grouped by their number of reference points, the one that obtained the best average residual errors across all the four modes was highlighted in bold. When 3 contact points are induced the two configurations that achieved the best results were M_6 (avg. $r_{x,y}$=2.58 mm, avg. r_θ=4.28°) and M_8 (avg. $r_{x,y}$=5.13 mm, avg. r_θ=0.74°), while for 4 points they were M_{14} (avg. $r_{x,y}$=3.55 mm, avg. r_θ=0.93°) and M_{15} (avg. $r_{x,y}$=1.77 mm, avg. r_θ=1.65°). Configuration M_8 can be considered better than M_6 due to the higher orientation error of M_6 in the **H-MAN**. Similarly, configuration M_{15} can be considered better than M_{14} because of the sub-millimeter accuracy in the **E-DET** and **H-DET** modes.

4.3.3 Palletization tasks and execution time

Palletization tasks were carried out for each configuration $M_i \subseteq P$ highlighted in bold in Table 4.2, in order to analyze the task execution time. In each task, the robot had to relocate 3 boxes from their initial pose to deposit locations on a destination pallet. Initially, in the first step, the robot moves the end-effector to an observation pose and it detects the 3 boxes poses using the eye-in-hand camera and the algorithm in [50]. Then, each box is grasped and the displacement error is estimated using the proposed approach. Finally, the box is placed on the destination pallet taking into account the displacement error. Motion planning was performed by MoveIt, which was configured so that the box was kept horizontal while attached to the gripper. The planned robot paths were executed at the maximum speed allowed by the UR10e in ur_ros_rtde.

The execution time of each pick and place task was split into the total time spent

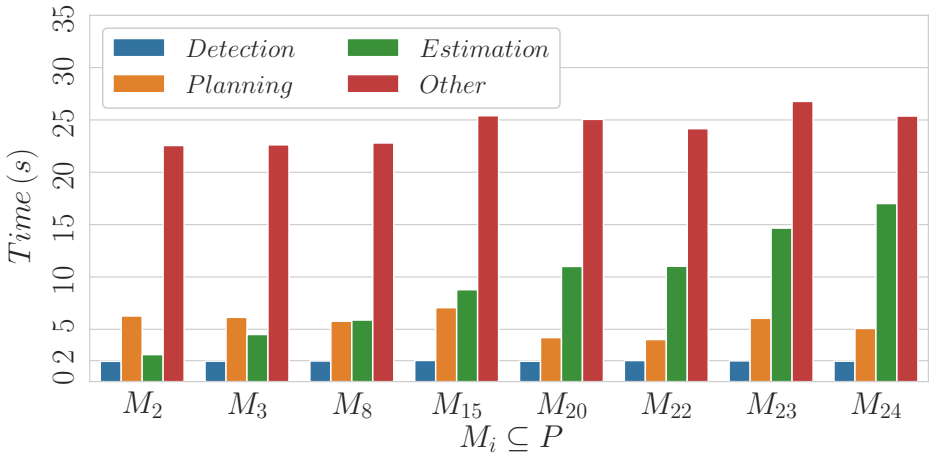


Figure 4.11: Total time spent in *Detection*, *Planning*, *Estimation* and *Other* steps divided by the number of boxes.

for the initial *Detection* step, displacement error estimation (*Estimation*), motion planning (*Planning*), and other operations (*Other*) such as grasping and releasing the box and robot movements to lift and place the box. All times were divided by the number of boxes. The results are shown in Fig. 4.11. Displacement error estimation includes mainly robot movements to collide with the peg, while processing time is negligible. The highest execution time was obtained with the subset M_{24} with 12 points (17.01 seconds). Conversely, the overhead of M_2 is only 2.58 seconds.

Table 4.3 shows the *Cycle time*, i.e. the total time divided by the number of boxes and the time spent for displacement error *Estimation*. Similar estimation times were obtained with M_{20} (11.01 seconds) and M_{22} (11.02 seconds). Configuration M_{20} contains 5 points, which are placed on all four sides of the box. Instead, M_{22} contains 6 points, two for each of three sides of the box. These results can be explained by

Table 4.3: Average total palletization cycle time and average displacement error *Estimation* time.

M_i	$\|M_i\|$	*Cycle time* (s)	*Estimation* (s)
M_2	1	33.34	2.58
M_3	2	35.20	4.50
M_8	3	36.41	5.88
M_{15}	4	43.26	8.78
M_{20}	5	42.21	11.00
M_{22}	6	41.21	11.02
M_{23}	8	49.46	14.66
M_{24}	12	49.41	17.01

observing that the *Rotate* action is more time consuming than the *Move* action, and thus inducing multiple collisions within the same side of the box is more efficient.

4.4 Discussion

In this Chapter an in-hand post-grasp displacement error estimation strategy has been proposed where the robot induces collisions between the grasped object and a minimal footprint fixture. The results show its effectiveness in estimating the displacement position and orientation error of cardboard boxes. The number of induced collisions and their position along the box edges affect the results. The computation time increases with the number of contact points. Overall, the best results were obtained with at least 3 contact points, achieving sub-millimeter accuracy for the position and sub-degree accuracy for the orientation. However, under-determined configurations with few contact points still reduce the displacement error.

Chapter 5

Conclusions

In this thesis novel design and motion planning strategies for collaborative robots in automated warehouses were presented. Manipulators and vacuum grippers were evaluated to design a prototype platform for handling cardboard boxes and tissue items. Focus was placed on cobots due to their advantages over non-collaborative industrial robots, including flexibility, safety, ease of use and the tendency to integrate with mobile manipulators. The contributions proposed in this dissertation are related to the handling of goods in palletization tasks. The challenges included choosing a feasible sequence of actions to manipulate objects, which may prove useful in palletization tasks, and minimizing placement errors.

In Chapter 1 collaborative manipulators, mounted on a mobile base, were compared in simulated depalletization tasks. The aim of this comparison was to evaluate the cobots capabilities in mobile depalletization tasks within industrial scenarios where a pallet is positioned inside a storage rack. All the cobots, of different dimensions, were evaluated in the same simulated industrial scenarios, testing various mounting configurations, pallet sizes, and mobile base placement strategies. The results have shown that the mobile manipulator must be designed accordingly to the task to solve. For example, larger cobots are better suited for depalletizing higher pallets.

In Chapter 2 an integrated task and motion planning approach based on a unified

Rapidly-exploring Random Tree was proposed. The main contribution in PROTAMP-RRT is the probabilistic model that can be used to estimate the feasibility of symbolic actions as a probability of sampling new collision-free configurations. The proposed planning algorithm was evaluated in several manipulation tasks and has shown better performace than the compared state-of-art TAMP algorithms. In addition, PROTAMP-RRT demonstrated its ability to solve task and motion planning problems by expanding the symbolic space when needed. If a set of known pick and place actions contains only unfeasible options significant poses are sampled and new pick and place actions are generated accordingly.

A novel collaborative prototype platform for palletization and depalletization tasks was presented in Chapter 3. The prototype is composed of a wheeled cart, a UR10e collaborative manipulator, a vacuum gripper and depth cameras. An evaluation of vacuum grippers and details about the hardware devices were provided. Most of the hardware devices employed in the prototype are components available on the market. Moreover, in future, the wheeled cart may be replaced with an AGV produced by E80 Group S.p.A. italian company.

In Chapter 4 an in-hand post-grasp displacement error estimation strategy has been presented. In the proposed strategy a collaborative robot induces collisions between an object attached to the gripper and a minimal footprint fixture. Collision information are used to estimate the displacement error solving a non-linear least squares optimization problem. Experiments were conducted in which the prototype of Chapter 2 had to relocate cardboard boxes. The results shown the strategy effectiveness in estimating the displacement error varying the number and the position of the induced collisions. Even with a single contact point the proposed method was able to reduce the displacement error. However, the best results were obtained with at least 3 induced collisions.

Future works

The progress outlined in this thesis set the stage for future works for both the prototype and the presented algorithms.

The primary direction for future work on the prototype platform is to replace the wheeled cart with a mobile robot. This prototype update will require a careful redesign of the component configuration on the mobile platform, including optimized space allocation for devices such as batteries.

PROTAMP-RRT can be extended by adopting more advanced symbolic planners to evaluate logical statements other than object/pose pairs. Actions other than *pick* and *place* can also be evaluated. For example, in palletization and depalletization tasks with a mobile manipulator, the mobile base could be controlled using the *move* action, which can be combined with existing pick and place actions. PROTAMP-RRT can also be beneficial for planning manipulation paths in case of narrow passages and tight spaces like palletization and depalletization of packages close to storage racks, where objects may hinder manipulations and the sequence in which they are manipulated is crucial. Of course, the high computational time of a planner like PROTAMP-RRT must also be considered for real world industrial applications, and speedup techniques can be investigated to improve planning efficiency.

The strategy proposed in Chapter 4 could be generalized to objects of shape other than a box. Moreover, in future work, the optimization problem could be adapted to rely on collision data collected from a single side of the object. As seen in the results, rotating the object is an expensive operation that should be avoided to reduce the cycle times in palletization tasks.

The comparative analysis performed in Chapter 1 highlighted planning challenges that novel algorithms may address. Robot paths were planned for the manipulator considering the placement of the mobile base separately. Moreover, while a box was attached to the gripper, RRTConnect with orientation constraint was used to keep the object horizontal.

Bibliography

[1] A. Saccuti, R. Monica, and J. Aleotti, "A comparative analysis of collaborative robots for autonomous mobile depalletizing tasks," in *2022 Sixth IEEE International Conference on Robotic Computing (IRC)*, pp. 34–38, 2022.

[2] A. Saccuti, R. Monica, and J. Aleotti, "Protamp-rrt: A probabilistic integrated task and motion planner based on rrt," *IEEE Robotics and Automation Letters*, vol. 8, no. 12, pp. 8398–8405, 2023.

[3] A. Saccuti, R. Monica, and J. Aleotti, "Contact-based in-hand package pose estimation using a collaborative robot," in *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2024.

[4] F. Sherwani, M. M. Asad, and B. Ibrahim, "Collaborative Robots and Industrial Revolution 4.0 (IR 4.0)," in *International Conference on Emerging Trends in Smart Technologies (ICETST)*, pp. 1–5, 2020.

[5] M. A. Roa, D. Berenson, and W. Huang, "Mobile manipulation: Toward smart manufacturing [tc spotlight]," *IEEE Robotics Automation Magazine*, vol. 22, no. 4, pp. 14–15, 2015.

[6] F. Ferraguti, A. Pertosa, C. Secchi, C. Fantuzzi, and M. Bonfè, "A methodology for comparative analysis of collaborative robots for industry 4.0," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1070–1075, 2019.

[7] M. Yang, E. Yang, R. C. Zante, M. Post, and X. Liu, "Collaborative mobile industrial manipulator: A review of system architecture and applications," in

25th International Conference on Automation and Computing (ICAC), pp. 1–6, 2019.

[8] F. Vitolo, A. Rega, C. Di Marino, A. Pasquariello, A. Zanella, and S. Patalano, "Mobile Robots and Cobots Integration: A Preliminary Design of a Mechatronic Interface by Using MBSE Approach," *Applied Sciences*, vol. 12, no. 1, 2022.

[9] F. D'Souza, J. Costa, and J. N. Pires, "Development of a solution for adding a collaborative robot to an industrial AGV," *Industrial Robot: the international journal of robotics research and application*, vol. 47, pp. 723–735, Jan 2020.

[10] J. Aleotti, A. Baldassarri, M. Bonfè, M. Carricato, D. Chiaravalli, R. Di Leva, C. Fantuzzi, S. Farsoni, G. Innero, D. Lodi Rizzini, C. Melchiorri, R. Monica, G. Palli, J. Rizzi, L. Sabattini, G. Sampietro, and F. Zaccaria, "Toward future automatic warehouses: An autonomous depalletizing system based on mobile manipulation and 3d perception," *Applied Sciences*, vol. 11, no. 13, 2021.

[11] S. Thakar, P. Rajendran, A. M. Kabir, and S. K. Gupta, "Manipulator Motion Planning for Part Pickup and Transport Operations From a Moving Base," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 1, pp. 191–206, 2022.

[12] J. Xu, Y. Domae, T. Ueshiba, W. Wan, and K. Harada, "Planning a minimum sequence of positions for picking parts from multiple trays using a mobile manipulator," *IEEE Access*, vol. 9, pp. 165526–165541, 2021.

[13] C. Wang, Q. Zhang, Q. Tian, S. Li, X. Wang, D. Lane, Y. Petillot, and S. Wang, "Learning mobile manipulation through deep reinforcement learning," *Sensors*, vol. 20, no. 3, 2020.

[14] A. Makhal and A. K. Goins, "Reuleaux: Robot Base Placement by Reachability Analysis," in *Second IEEE International Conference on Robotic Computing (IRC)*, pp. 137–142, 2018.

[15] D. Di Marco, O. Zweigle, and P. Levi, "Base pose estimation using shared reachability maps for manipulation tasks," in *13th International Conference on Autonomous Robot Systems*, pp. 1–6, 2013.

[16] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[17] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, pp. 995–1001 vol.2, 2000.

[18] P. Beeson and B. Ames, "TRAC-IK: An open-source library for improved solving of generic inverse kinematics," in *IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pp. 928–935, 2015.

[19] J. Pan, S. Chitta, and D. Manocha, "FCL: A general purpose library for collision and proximity queries," in *IEEE International Conference on Robotics and Automation*, pp. 3859–3866, 2012.

[20] I. A. Şucan and S. Chitta, "Motion planning with constraints using configuration space approximations," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1904–1910, 2012.

[21] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annu. rev. control robot. auton. syst.*, vol. 4, no. 1, pp. 265–293, 2021.

[22] S. M. LaValle, "Rapidly-exploring random trees: a new tool for path planning," *The annual research report*, 1998.

[23] R. Caccavale and A. Finzi, "A rapidly-exploring random trees approach to combined task and motion planning," *Robotics and Autonomous Systems*, vol. 157, 08 2022.

[24] W. Thomason and R. A. Knepper, "A unified sampling-based approach to integrated task and motion planning," in *International Symposium on Robotics Research (ISRR)*, pp. 48–76, 2019.

[25] N. T. Dantam, S. Chaudhuri, and L. E. Kavraki, "The task-motion kit: An open source, general-purpose task and motion-planning framework," *IEEE Robotics & Automation Magazine*, vol. 25, no. 3, pp. 61–70, 2018.

[26] C. R. Garrett, T. Lozano-Pérez, 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.

[27] C. R. Garrett, T. Lozano-Perez, and L. P. Kaelbling, "PDDLStream: Integrating Symbolic Planners and Blackbox Samplers via Optimistic Adaptive Planning," in *International Conference on Automated Planning and Scheduling*, 2018.

[28] C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, and B. Nebel, "Semantic attachments for domain-independent planning systems," in *19th International Conference on Automated Planning and Scheduling (ICAPS)*, p. 114 – 121, 2009.

[29] P. M. U. Eljuri, G. A. G. Ricardez, N. Koganti, J. Takamatsu, and T. 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.

[30] A. M. Wells, N. T. Dantam, A. Shrivastava, and L. E. Kavraki, "Learning feasibility for task and motion planning in tabletop environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1255–1262, 2019.

[31] B. Kim and L. Shimanuki, "Learning value functions with relational state representations for guiding task-and-motion planning," in *Proceedings of the Conference on Robot Learning* (L. P. Kaelbling, D. Kragic, and K. Sugiura, eds.), vol. 100 of *Proceedings of Machine Learning Research*, pp. 955–968, PMLR, 30 Oct–01 Nov 2020.

[32] Z. Jiao, Y. Niu, Z. Zhang, S.-C. Zhu, Y. Zhu, and H. Liu, "Sequential manipulation planning on scene graph," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8203–8210, 2022.

[33] M. Khodeir, B. Agro, and F. Shkurti, "Learning to search in task and motion planning with streams," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 1983–1990, 2023.

[34] D. Xu, A. Mandlekar, R. Martín-Martín, Y. Zhu, S. Savarese, and L. Fei-Fei, "Deep affordance foresight: Planning through what can be done in the future," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6206–6213, 2021.

[35] T. Ren, A. I. Cowen-Rivers, H. Bou-Ammar, and J. Peters, "Learning geometric constraints in task and motion planning," *arXiv preprint arXiv:2201.09612*, 2022.

[36] B. Kim, Z. Wang, L. P. Kaelbling, and T. Lozano-Pérez, "Learning to guide task and motion planning using score-space representation," *Int J Rob Res*, vol. 38, no. 7, pp. 793–812, 2019.

[37] F. Lagriffoul and B. Andres, "Combining task and motion planning: A culprit detection problem," *The International Journal of Robotics Research*, vol. 35, no. 8, pp. 890–927, 2016.

[38] T. Lozano-Pérez and L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," in *IEEE/RSJ Intl Conference on Intelligent Robots and Systems*, pp. 3684–3691, 2014.

[39] C. Nam, S. H. Cheong, J. Lee, D. H. Kim, and C. Kim, "Fast and resilient manipulation planning for object retrieval in cluttered and confined environments," *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1539–1552, 2021.

[40] R. Wang, K. Gao, D. Nakhimovich, J. Yu, and K. E. Bekris, "Uniform object rearrangement: From complete monotone primitives to efficient non-monotone

informed search," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6621–6627, 2021.

[41] A. Krontiris and K. Bekris, "Trade-off in the computation of minimum constraint removal paths for manipulation planning," *Advanced Robotics*, vol. 31, pp. 1–12, 09 2017.

[42] N. Castaman, E. Tosello, and E. Pagello, "Conditional task and motion planning through an effort-based approach," in *IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pp. 49–54, 2018.

[43] M. Toussaint and M. Lopes, "Multi-bound tree search for logic-geometric programming in cooperative manipulation domains," in *IEEE Intl Conference on Robotics and Automation*, pp. 4044–4051, 2017.

[44] W. Thomason, M. Strub, and J. Gammell, "Task and Motion Informed Trees (TMIT*): Almost-Surely Asymptotically Optimal Integrated Task and Motion Planning," *IEEE Robotics and Automation Letters*, vol. 7, pp. 11370–11377, 10 2022.

[45] J. Ortiz-Haro, V. N. Hartmann, O. S. Oguz, and M. Toussaint, "Learning efficient constraint graph sampling for robotic sequential manipulation," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 4606–4612, 2021.

[46] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Sampling-based methods for factored task and motion planning," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1796–1825, 2018.

[47] C. Eppner, R. Deimel, J. Álvarez Ruiz, M. Maertens, and O. Brock, "Exploitation of environmental constraints in human and robotic grasping," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 1021–1038, 2015.

[48] F. Fiedler, J. Ehrenstein, C. Höltgen, A. Blondrath, L. Schäper, A. Göppert, and R. Schmitt, "Jigs and fixtures in production: A systematic literature review," *Journal of Manufacturing Systems*, vol. 72, pp. 373–405, 2024.

[49] J. Aleotti, A. Baldassarri, M. Bonfè, M. Carricato, D. Chiaravalli, R. Di Leva, C. Fantuzzi, S. Farsoni, G. Innero, D. Lodi Rizzini, C. Melchiorri, R. Monica, G. Palli, J. Rizzi, L. Sabattini, G. Sampietro, and F. Zaccaria, "Toward future automatic warehouses: An autonomous depalletizing system based on mobile manipulation and 3D perception," *Applied Sciences*, vol. 11, no. 13, 2021.

[50] R. Monica, J. Aleotti, and D. L. Rizzini, "Detection of parcel boxes for pallet unloading using a 3d time-of-flight industrial sensor," in *Fourth IEEE International Conference on Robotic Computing (IRC)*, pp. 314–318, 2020.

[51] Z. Hu, W. Wan, K. Koyama, and K. Harada, "Reducing uncertainty using placement and regrasp planning on a triangular corner fixture," *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 1, pp. 652–670, 2024.

[52] F. von Drigalski, K. Kasaura, C. C. Beltran-Hernandez, M. Hamaya, K. Tanaka, and T. Matsubara, "Uncertainty-aware manipulation planning using gravity and environment geometry," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11942–11949, 2022.

[53] B. Kreis, R. Menon, B. K. Adinarayan, J. de Heuvel, and M. Bennewitz, "Reactive correction of object placement errors for robotic arrangement tasks," in *International Conference on Intelligent Autonomous Systems (IAS)*, 2023.

[54] Z. Chen and T. Li, "Precise geometry and pose measurement of in-hand objects with simple features using a multi-camera system," *Manufacturing Letters*, vol. 35, pp. 40–48, 2023. 51st SME North American Manufacturing Research Conference (NAMRC 51).

[55] F. Li, Y. Jiang, T. Li, Y. Feng, and S. Chen, "Design of a robot end effector with measurement system for precise pick-and-place of square objects," *Procedia Manufacturing*, vol. 48, pp. 172–180, 2020. 48th SME North American Manufacturing Research Conference, NAMRC 48.

[56] Y. Gao, S. Matsuoka, W. Wan, T. Kiyokawa, K. Koyama, and K. Harada, "In-hand pose estimation using hand-mounted rgb cameras and visuotactile sensors," *IEEE Access*, vol. 11, pp. 17218–17232, 2023.

[57] V. R. Galaiya, M. Asfour, T. E. Alves de Oliveira, X. Jiang , and V. Prado da Fonseca, "Exploring tactile temporal features for object pose estimation during robotic manipulation," *Sensors*, vol. 23, no. 9, 2023.

[58] T. Sakuma, T. Kiyokawa, T. Matsubara, J. Takamatsu, T. Wada, and T. Ogasawara, "Jamming gripper-inspired soft jig for perceptive parts fixing," *IEEE Access*, vol. 11, pp. 62187–62199, 2023.

[59] J. Zhao, J. Liang, and O. Kroemer, "Toward precise robotic grasping by probabilistic post-grasp displacement estimation," in *Field and Service Robotics* (G. Ishigami and K. Yoshida, eds.), pp. 131–144, Springer Singapore, 2021.

[60] M. Pfanne, M. Chalon, F. Stulp, and A. Albu-Schäffer, "Fusing joint measurements and visual features for in-hand object pose estimation," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3497–3504, 2018.

[61] F. von Drigalski, S. Taniguchi, R. Lee, T. Matsubara, M. Hamaya, K. Tanaka, and Y. Ijiri, "Contact-based in-hand pose estimation using bayesian state estimation and particle filtering," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7294–7299, 2020.

[62] F. von Drigalski, K. Hayashi, Y. Huang, R. Yonetani, M. Hamaya, K. Tanaka, and Y. Ijiri, "Precise multi-modal in-hand pose estimation using low-precision sensors for robotic assembly," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 968–974, 2021.

[63] J. Pankert and M. Hutter, "Learning contact-based state estimation for assembly tasks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5087–5094, 2023.

[64] A. Sipos and N. Fazeli, "Simultaneous contact location and object pose estimation using proprioception and tactile feedback," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3233–3240, 2022.