Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in Computer Engineering

Bimanual robotic manipulation based on potential fields

Master Candidate

Alberto Paiusco

Student ID 2006674

Supervisor

Prof. Emanuele Menegatti

University of Padova

Co-supervisors

Prof. Gloria Beraldo

Alberto Gottardi

Academic Year 2022/2023

July 13, 2023

Abstract

Dual manipulation is a natural skill for humans but not so easy to achieve for a robot. The presence of two end effectors implies the need to consider the temporal and spatial constraints they generate while moving together. Consequently, synchronization between the arms is required to perform coordinated actions (e.g., lifting a box) and to avoid self-collision between the manipulators. Moreover, the challenges increase in dynamic environments, where the arms must be able to respond quickly to changes in the position of obstacles or target objects. To meet these demands, approaches like optimization-based motion planners and imitation learning can be employed but they have limitations such as high computational costs, or the need to create a large dataset. Sampling-based motion planners can be a viable solution thanks to their speed and low computational costs but, in their basic implementation, the environment is assumed to be static. An alternative approach relies on improved Artificial Potential Fields (APF). They are intuitive, with low computational, and, most importantly, can be used in dynamic environments. However, they do not have the precision to perform manipulation actions, and dynamic goals are not considered. This thesis proposes a system for bimanual robotic manipulation based on a combination of improved Artificial Potential Fields (APF) and the sampling-based motion planner RRTConnect. The basic idea is to use improved APF to bring the end effectors near their target goal while reacting to changes in the surrounding environment. Only then RRTConnect is triggered to perform the manipulation task. In this way, it is possible to take advantage of the strengths of both methods. To improve this system APF have been extended to consider dynamic goals and a self-collision avoidance system has been developed. The conducted experiments demonstrate that the proposed system adeptly responds to changes in the position of obstacles and target objects. Moreover, the self-collision avoidance system enables faster dual manipulation routines compared to sequential arm movements.

Sommario

La manipolazione duale è un'abilità naturale per gli esseri umani, ma non è così facile da svolgere per un robot. La presenza di due manipolatori implica la necessità di considerare i vincoli temporali e spaziali che essi generano mentre si muovono insieme. Di conseguenza, la sincronizzazione tra le braccia è necessaria per eseguire azioni coordinate (ad esempio, sollevare una scatola) ed evitare l'autocollisione tra i manipolatori. Inoltre, le difficoltà aumentano negli ambienti dinamici, in quanto le braccia devono essere in grado di rispondere rapidamente ai cambiamenti di posizione degli ostacoli o degli oggetti target. Per soddisfare queste esigenze, approcci come gli optimization-based motion planner o l'imitation learning possono essere utilizzati, ma essi presentano limitazioni come elevati costi computazionali o la necessità di creare un ampio dataset. I sampling-based motion planners possono essere una valida alternativa grazie alla loro velocità e ai bassi costi computazionali ma, nella loro implementazione di base, l'ambiente è considerato solo statico. Un altro possibile approccio si basa sugli improved Artificial Potential Fields (APF). Sono intuitivi, con un basso costo computazionale e, soprattutto, possono essere utilizzati in ambienti dinamici. Tuttavia, non hanno la precisione necessaria per eseguire azioni di manipolazione e i target dinamici non sono considerati. Questa tesi propone un sistema di manipolazione robotica bimanuale basato su una combinazione di improved Artificial Potential Fields (APF) e del sampling-based motion planner RRTConnect. L'idea di base è quella di utilizzare gli improved APF per portare i manipolatori vicino alla loro posizione obiettivo, reagendo al contempo ai cambiamenti dell'ambiente circostante. Solo allora RRTConnect viene attivato per eseguire il compito di manipolazione. In questo modo, è possibile sfruttare i punti di forza di entrambi i metodi. Per migliorare questo sistema, gli APF sono stati estesi per considerare target dinamici ed è stato sviluppato un sistema di anticollisione tra le braccia. Gli esperimenti condotti dimostrano che il sistema proposto risponde abilmente ai cambiamenti di posizione degli ostacoli e degli oggetti target. Inoltre, il sistema di self-collision avoidance consente di velocizzare le routine di manipolazione duale rispetto a movimenti sequenziali delle braccia.

Contents

1 Introduction

1.1 Bimanual manipulation

Dual arm systems were among the first robotic manipulators to be developed, back in the 1940s and 1950s. Such systems were mostly teleoperated and designed to operate in environments prohibitive to humans. Some examples include the manipulators produced by Goertz for handling radioactive materials [9], and dual-arm teleoperation setups used for deep sea exploration [8]. Following this initial research, there has been a proliferation of studies exploring the use of single-arm manipulators, which have been widely adopted as the norm, particularly during the 1980s and 1990s. Since the 2000s, advances in the field of humanoid robotics, as well as research into learning by imitation, have created new opportunities for the use of dual-arm setups.

In [34] a broad overview of dual manipulation is presented, along with the main factors motivating the study of dual arm setups. Some of such factors are:

- **Similarity to the operator**: since humans are used to performing tasks using both hands, this skill can be transferred to dual-arm robots.
- **Manipulability**: the ability to perform tasks that require at least two hands (e.g. screw assembly or folding laundry).
- **Flexibility**: the possibility of using and combining different kinds of end effectors, for example, a gripper can be great for stabilizing a bottle while a human-like robotic hand removes the cap.
- **Human form factor**: compared to two single-arm units, dual-arm systems take up less space and incur lower costs. Moreover, it is possible to substitute human labor without the need to modify the workspace.

A wide range of robotic platforms are available for dual-arm manipulation. While some researchers focus on utilizing two single-arm manipulators that share the same workspace, others dedicate their efforts to creating dual-arm platforms. Some of these configurations have a torso-like structure and are mounted on a mobile base, allowing them to combine manipulation skills with the ability to navigate their surroundings (Willow Garage PR2, PAL Robotics TIAGo). On the contrary, other configurations are fixed, designed specifically for assembly line tasks in industrial environments. These fixed robots are optimized for high precision, speed, and accuracy (Yaskawa Motoman SDA10D).

While many dual-arm systems are primarily used for research purposes to develop and evaluate technologies and principles, there are also practical applications for these systems. Domestic applications such as folding laundry have been widely studied; vision is used to detect folds and corners, and different manipulation methods are employed depending on the task. For example, a system for autonomous towel folding has been demonstrated using stereo vision and a grasp point detection algorithm on the PR2 robot [21] (Fig. 1.1). Other examples of domestic applications include attendant care for the elderly [11], making pancakes [2], and serving tea using preprogrammed motions and teaching by demonstration [23]. Dual-arm systems are commonly used in industrial settings for parts assembly and material reshaping. For example, in [36] it is presented a preprogrammed gearbox assembly, and in [20] a dual-arm system capable of folding cartons into predetermined shapes. Other application scenarios are in agriculture to help harvest plants, in logistics for sorting and packing goods in warehouses, and also in the medical field.

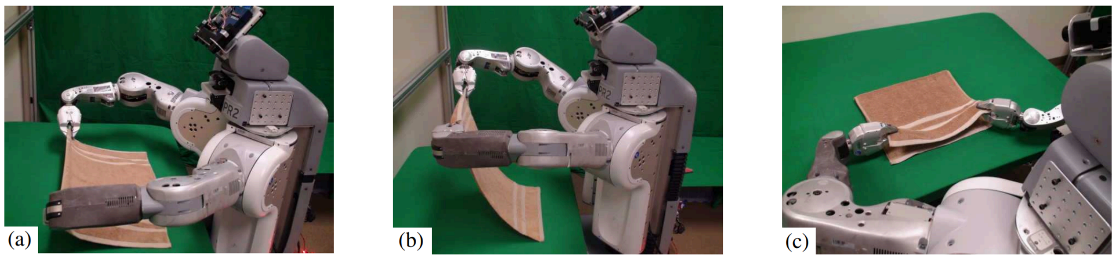


Figure 1.1: Some of the steps performed in [21] to fold towels

Challenges

To understand which are the main challenges and constraints of bi-manual robotic manipulation, it is useful to analyze the various types of dual manipulation that can occur. Bi-manual manipulation can be divided into two main categories (showed in Fig. 1.2): uncoordinated and coordinated, based on whether there is any kind of spatial or temporal coordination between the hands [16].

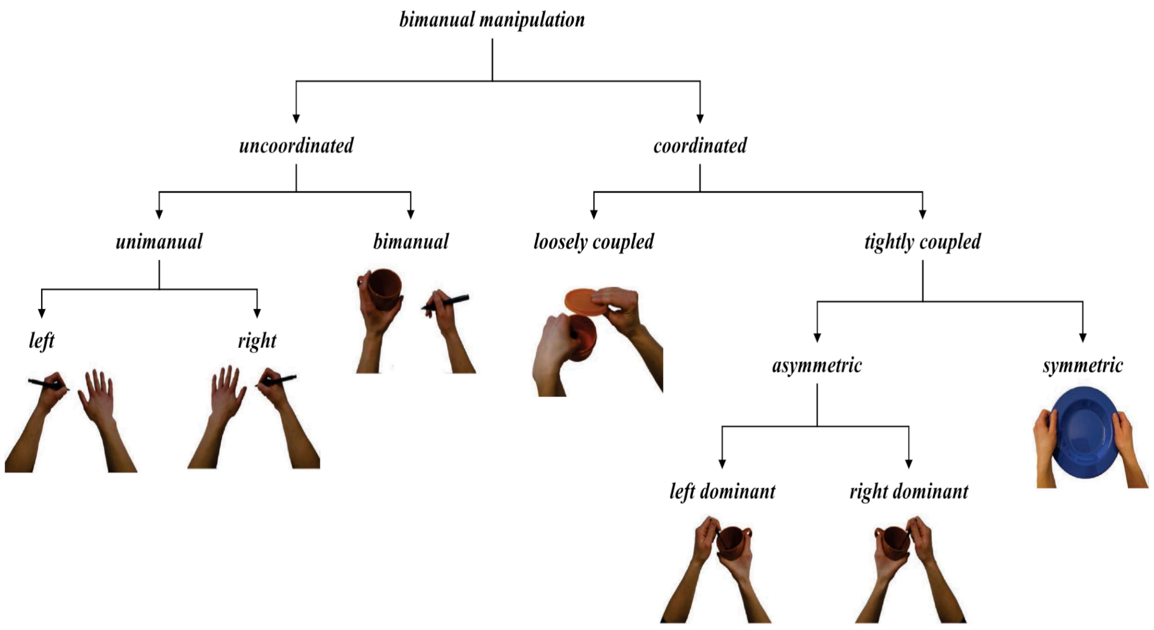


Figure 1.2: Bimanual manipulation taxonomy proposed in [16]

In the uncoordinated scenario, the two arms perform completely unrelated tasks that can be seen as simultaneously executed uni-manual actions. Since there are no time constraints, the space coordination is confined to preventing clashes between the arms. An example of this type of manipulation can be picking up a bottle while the other manipulator takes a pen. The two picks are mutually independent.

In the coordinated case, the hands work together to achieve a common goal. If the cooperation is only at a logical level (i.e., there is no contact with a common object) the task falls into the coordinated loosely coupled case and the constraints are the same as the uncoordinated bimanual scenario. An example of the previous case can be "clear the table", both manipulators cooperate to achieve the common logical goal, but this does not impose any time constraint, only space coordination is required. Conversely, if the hands act on the same item, such coordinated manipulation is called tightly coupled; in this situation, both spatial and temporal constraints are present, and the manipulators must avoid collisions with each other and act within a precise time window. In addition, constraints imposed by forces must be taken into account; the objects must be grasped not too lightly or too strongly to avoid the risk of dropping or breaking the item. The tightly coupled category can be further subdivided into symmetric and asymmetric by considering the role of the hands. The asymmetrical scenario occurs when one hand, called non-dominant, has a stabilizing role and provides the reference frame for the dominant hand. Instead, the symmetrical case occurs when both hands play the same role. An example of asymmetric manipulation can be stirring the liquid inside a cup, the non-dominant hand holds the cup steady while the dominant hand stirs. An example of symmetric manipulation can be picking and lifting a box.

In order to have robotic personal assistants capable of working in unstructured environments, dual manipulation systems are often installed on mobile bases [33]. This type of system is called Mobile Manipulation (MM) and imposes an additional constraint: the positioning of the base. In fact, the robot must be positioned congruently with the bi-manual action to be performed.

1.2 Dynamic manipulation

As described in the previous section, bimanual robots can manipulate objects within their workspace. However, if that area is uncertain, additional difficulties arise. The target objects might change position over time, potentially causing the manipulation system to fail in grasping them. Moreover, the arms may collide with newly introduced items in the workspace. Dynamic manipulation allows to deal with such scenarios by employing online motion planning. In online motion planning, a real-time process receives a stream of data about the robot's surroundings. As the robot moves, new plans are created in response to environmental changes. In contrast, offline motion planning is a one-time calculation that is performed before any motion is executed. In order to use this method, all relevant data must be known beforehand, making it impractical for environments that are constantly changing.

Dynamic manipulation systems impose new challenges compared to static configurations [5]:

1. Perception: robots require accurate and fast perception of the object location as well as the ability to adapt to changes in lighting and surroundings

2. Grasping: the robot must be able to adjust its grasp in real time according to changes in the position of the object
3. Safety: as robots interact with moving objects, the possibility of accidents and damage increases. Therefore, safety and reliability are key aspects in dynamic manipulation and require the use of sophisticated detection, planning, and control algorithms to avoid accidents and ensure safe operation

Dynamic manipulation is essential in several contexts. An example is MM since the operation of robots in unstructured human environments poses significant obstacles to their efficiency. Manipulation in these places involves handling unfamiliar objects and dealing with cluttered workspaces, different lightings, and imprecise sensor data. Another important application of dynamic manipulation is medical robotics, where robots are used to assist surgical procedures [37]. In this context, to reach high levels of autonomy, robots must be able to manipulate delicate tissues and organs without causing damage and to respond to unexpected movements or changes in surgical conditions. Dynamic manipulation also plays an important role in the industry as collaborative robots are increasingly present. They are designed to work alongside humans in a shared workspace and thus require dynamic skills to interact with humans properly. Another application is in search and rescue operations, where robots are used to navigate through rubble and debris [22]. In this application, dynamic manipulation allows robots to remove debris while being able to react to obstacles or changes in the environment. To sum up, dynamic robotic manipulation is an important capability for autonomous robots, which must be able to perform complex tasks in unstructured and changing environments. As technology continues to advance, we can expect to see this capability applied to a growing range of applications in industry, medicine, and beyond.

1.3 Thesis objective

To develop a dynamic bimanual manipulation system, various approaches can be employed, such as optimization-based motion planners and imitation learning. However, these approaches come with significant drawbacks, such as high computational costs or the requirement of a large dataset. An alternative solution, therefore, relies on sampling-based motion planners, specifically RRTConnect. Thanks to its simplicity, it is fast and with low computational cost but, in its basic implementation, it considers the environment as static. For this reason, it cannot be applied directly but must be integrated with something else to be robust to dynamic goals and obstacles. With this purpose, we focus on Artificial Potential Fields (APF) [14]. APF perform well in dynamic environments and can be used to bring the end effectors close to the target items while reacting to changes in the surrounding scenario. Anyhow, one of the main limitations of APF are local minima: an end effector might get stuck near an obstacle without being able to overcome it. Hence, this thesis builds upon the work presented in [10] where APF have been improved in order to solve such a problem. The so-called "improved APF" enables an easier escape from local minima through the dynamic generation of escape points around obstacles. This approach has been successfully tested in a shared control scenario where the user controlled a single robotic arm via different external devices. However, only the dynamism of the obstacles was considered while the target goals remain static. This thesis, therefore, aims to extend this work to have a fully dynamic bimanual manipulation system, capable of performing both coordinated and uncoordinated tasks. Moreover, a control system has been developed to detect potential arm collisions and generate dynamic motion plans that allows the end effector to reach the respective goal safely.

1.4 Thesis structure

The content of this thesis is divided into the following chapters:

- Chapter 2 describes the current state of the art. It provides an overview of sampling-based motion planners, optimization-based motion planners, imitation learning approaches, and APF. In addition to highlighting their strengths, this chapter examines the limitations of each approach
- Chapter 3 illustrates the tools used in the development of this thesis. After introducing the TIAGo++ robot, ROS, MoveIt, and Gazebo are described
- Chapter 4 introduces the proposed solutions, defining the problem and outlining the process of developing the bimanual manipulation system. It includes block diagrams and detailed descriptions of the main ROS nodes used in the system
- Chapter 5 presents the results of experiments conducted with the proposed bimanual manipulation system. It describes the setup and provides pictures and tables for each experiment, illustrating the capabilities of the approach
- Chapter 6 draws conclusions about the proposed bimanual manipulation system and discusses potential future works

2 State-of-the-Art

To perform bimanual actions in dynamic environments, different approaches can be employed. This chapter is devoted to presenting an overview of the most common approaches for bimanual manipulation. First, sampling-based motion planners are presented, despite their simplicity they are still widely used to plan manipulation routines (e.g., pick and place). Next, optimization-based motion planners are described; they are more complex than sampling-based methods and require more computational power. The following section is dedicated to imitation learning. Thanks to recent developments in the field of deep learning, in fact, it is possible to learn bimanual actions via human demonstrations; nevertheless, this method still has several limitations. Finally, manipulation with APF is discussed. Although it is not a recent methodology, thanks to some improvements it is still effective, especially in dynamic environments.

2.1 Sampling-based motion planners

Sampling-based motion planners generate paths by sampling the configuration space (C-space[1]). These planners do not guarantee to find a solution in cases where one exists, which is known as *completeness*. Instead, they offer a less certain form of completeness, namely *probabilistic completeness*. In other words, given enough time, these algorithms should be able to provide a solution if one

[1]Space of possible positions that the robot may attain

exists [6]. However, the solution found is probabilistic, meaning that it may not be the best one. It takes time to optimize it or look for the optimal solution directly. The two most common types of sampling-based motion planners are:

- **Probabilistic Road Map (PRM)**: comprising of two stages, the first part involves a learning phase. During this stage, the C-space is sampled for a specific interval, where samples from the obstacle space are excluded while others are retained. This is followed by a query phase, in which a roadmap is created by linking the initial configuration with the final one using collision-free neighboring samples [6]
- **Rapidly-Exploring Random Tree (RRT)**: this algorithm involves the gradual expansion of a tree from the initial configuration. Configurations within the C-space are randomly selected, and if they are in a free space, an attempt is made to connect them with the tree. As the algorithm continues to sample new configurations, the tree grows and explores more of the environment finding possible paths towards a goal position [6]

Some extensions of RRT and PRM have been developed in order to apply such algorithms to dual-arm manipulation. Some examples are Bi-RRT [18] and SBL [31], which allow both arms to reach their respective targets without colliding with each other. Moreover, many other optimizations have been studied, like the Probabilistic Road Map with Obstacles (PRMwO) planner [30]. In a two-arm scenario, the presence of another manipulator can be an obstacle and thus impose limitations on the possible grasping points of an object, but it might also be a powerful tool. In fact, it is possible to exploit one of the two arms to remove potential obstacles and ensure an unobstructed path to grasp the object of interest. The PRMwO planner not only generates paths for each arm to grasp the target objects but, at the same time, returns the set of obstacles that must be removed. This method is based on the generation of a graph where the root node symbolizes the target object. The motion planning algorithm is then used to search for a trajectory to reach and grasp the item of interest. If a path without collisions is discovered, it is immediately returned. Otherwise, the graph is enlarged by adding child nodes that represent the objects that must be removed to clear the path for the parent node, thus launching the process into a repeating cycle. An upgrade of RRT instead is the Bimanual Grasp-RRT planner [35]. The Grasp-RRT planner integrates the primary actions required to pick an object, such as: identifying a valid grasp, resolving inverse kinematics, and finding an unobstructed path to bring the hand into the grasp position. In the bimanual scenario, the planner launches two Grasp-RRT planners, one for each hand, which are run simultaneously to find feasible grasps. The main thread gathers the grasps and their corresponding trajectories for both hands and evaluates them to determine the best bimanual solution. Whenever a new grasping trajectory is discovered, all possible bimanual combinations with the grasps from the other hand are calculated and their quality is evaluated. If the bimanual score exceeds a certain threshold and there is no self-collision, the combined solution with the grasping details is returned and can be executed.

Anyhow, even if the random procedure of sampling-based planners offers advantages in solving challenging problems quickly, a drawback is that the resolutions are commonly perceived as suboptimal, as the movements may be unnecessarily large and not human-like. Moreover, when there are many obstacles in the considered area, the strategies employed by sampling-based planners to move through narrow spaces may be unnecessarily complicated and extra steps must be taken to eliminate any rough or unnecessary movements from the routes created by these planners. Moreover, in their basic bimanual implementation, PRM and RRT are not suitable for dynamic scenarios since they assume that the environment is static and therefore roadmaps are generated offline. In fact, a strength of the improved APF over sampling-based methods is the online generation of the motion, thus the ability to react to changes in target position or obstacles.

2.2 Optimization-based motion planners

Optimization-based motion planners use mathematical optimization techniques to plan the robot's movements [38]. The process of these planners involves breaking up the C-space into finite cells using a consistent pattern (usually a grid). This is followed by building a graph that includes a vertex for each cell, with edges connecting nearby vertices that can be linked by a feasible motion. Only collision-free region vertices and edges are included. Along the path, all edges may be assigned a cost which allows for the minimisation of a cost function in addition to finding any path. The strength of optimization-based planners lies precisely in the cost function since it can take into account several factors such as obstacle avoidance, kinematic and dynamic constraints, end effector pose, orientation constraints, energy consumption, etc. This can lead to smooth trajectories, high adaptability to a wide range of environments, and applications that provide significant savings in battery life and operating costs.

Two popular optimization-based motion planners are Covariant Hamiltonian Optimization for Motion Planning (CHOMP) [27] and Stochastic Trajectory Optimization for Motion Planning (STOMP) [13]. Although many high-dimensional motion planners divide the trajectory generation process into separate planning and optimization stages, CHOMP exploits covariant gradient and functional gradient techniques in order to create a motion planning algorithm that solely relies on trajectory optimization. Starting from a naïve trajectory, CHOMP strategically reacts to the environment generating a trajectory that avoids any possible collisions, while simultaneously fine-tuning the cost function. STOMP is based on creating trajectories with random perturbations to investigate the surroundings of a starting trajectory, even if it may be infeasible. These perturbed trajectories are merged to generate an improved path with a lower cost. A cost function, consisting of both obstacle and smoothness costs, is selected for optimization in every step. The optimization algorithm that is used does not require gradient information, allowing for the inclusion of general costs, such as those related to constraints and motor torques, for which derivatives may not be accessible.

Since CHOMP and STOMP also check for self-collisions, they can be used directly for dual-arm manipulation. For example, in [32] a bimanual robotic system is presented for harvesting aubergines that uses STOMP as motion planner. The aim of the study was to improve the efficiency of aubergine harvesting while minimizing damage to the fruit. Results showed that the robotic system was effective in increasing the speed of harvesting and reducing fruit damage compared to manual harvesting methods. Another setup in which the STOMP planner is exploited is presented in [24] for autonomous dual-arm manipulation of familiar objects. The system uses a 3D camera to perceive the environment and generate a point cloud, which is then utilized by the STOMP algorithm to plan the manipulation actions. In [26] a motion planning approach is proposed for dual-arm assembly of ring-shaped elastic objects that rely on CHOMP. An energy-based functional objective is established by incorporating a component related to the potential energy of the elastic entity to the cost function. The result is a path that minimizes the deformation of the elastic object. In addition, the cost function for collisions is modified to allow the robot to get as close as possible to the assembly parts without causing collisions.

However, optimization-based motion planners have some drawbacks. Compared with improved APF, tuning the parameters might not be trivial. For example, the default parameters of CHOMP are effective in obstacle-free environments, but if the scenario contains several obstacles, it might get trapped in a local minima. Therefore, it becomes essential to fine-tune the parameters according to the specific scenario. Nevertheless, due to the numerous parameters involved, this task can be challenging and non-intuitive to perform. In contrast, improved APF parameters are less and much more intuitive. In addition, STOMP and CHOMP can be computationally expensive, as they require a finely discretized trajectory to perform collision checks with obstacles and ensure a smooth solution. Improved APF, in contrast, thanks to their simplicity can be applied also to systems with low computational capabilities.

2.3 Imitation Learning

Imitation Learning (IL) [1] is a type of machine learning where an agent learns to perform a task by imitating an expert's behavior. The central element of IL is the environment where the expert acts, which can be described as a Markov Decision Process (MDP). Therefore it has a set of states S, a set of actions A, a transition model $T(s'|s,a)$ (probability that taking action a in the state s will result in a transition to state s') and an unknown reward function $R(s,a)$. Whereas in Reinforcement Learning (RL) it may be necessary to manually design a reward function that meets the desired behavior, in IL the expert provides a series of demonstrations $\tau = \{(s_0,a_0),(s_1,a_1),...\}$ that the agent imitates to learn the optimal policy (mapping from states to actions) π*. In the most basic form of imitation learning, Behavior Cloning (BC), the expert's demonstrations $D = \{\tau_1, \tau_2, ..., \tau_n\}$ are treated as independent and identically distributed state-action pairs and the policy is learned via supervised learning minimizing the loss $L(a^*, \pi(s))$ that depends on the application. An alternative to BC is inverse reinforcement learning (IRL), in which the idea is to learn the reward function of the environment through expert demonstrations and then employ RL to determine the optimal policy that maximizes such reward function. This process is repeated until the policy found is satisfactory.

The application of imitation learning for bimanual manipulation tasks remains challenging [7] even if very promising. To generate the expert demonstrations, two categories of methods can be employed: indirect teaching and direct teaching [17]. Indirect teaching methods involve the use of wearable devices and visual systems to directly capture human movement data, which can then be used by robots to perform anthropomorphic tasks. An example of this is presented in [39]: using stereo cameras and two data-gloves with magnetic trackers and tactile sensors are recorded actions demonstrations. These acquisitions are then classified into coordinated symmetric, coordinated asymmetric and uncoordinated actions by considering the spatial relationship between the trajectories of the hands. Symmetrical coordination is detected by analyzing closed kinematic chains and a set of heuristics is then applied to differentiate between asymmetrical and uncoordinated actions. The actions are then executed using a synchronization framework that is based on Petri nets [25]. However, since the operational characteristics of the robot are not taken into account and indirect teaching is done separately, demonstrations cannot be guaranteed to be of high quality. On the other hand, direct teaching methods obtain demonstrations directly from the robot and can be divided into kinesthetic teaching and teleoperation teaching. Kinesthetic teaching involves physical interaction between the operator and the robot to accomplish a specific task. During such operation, the robot autonomously gathers information about the movement. Nevertheless, robots that are appropriate for this kind of interaction must be passively controllable and require physical contact. Therefore, this method is often unsuitable, especially for dual-arm robots. Teleoperation teaching allows to extract demonstrations by operating the robot with remote control devices such as joysticks, touch sensors, wearable devices, etc. Usually the operator wears a head-mounted display that provides the robot's point of view. Even if teleoperation teaching provides many advantages like high safety, wide application range and high-quality teaching samples, most of the current teleoperations merely focus on instructing the robot on how to move its body or follow a certain path, neglecting crucial details on the necessary force required for precise operations. This limitation presents a challenge for the robot to execute intricate tasks effectively.

In recent years, there has been growing interest in exploring deep imitation learning methods for bimanual manipulation. The paper [15] presents a possible approach in this area. A state $s \in S$ can be composed by the user's gaze position, the foveated image embedding of the target object, and the left and right arm kinematic states. However, due to the large size of the concatenated kinematic states, distractions (i.e., irrelevant information that can interfere with the learning process) can occur, leading to poor policy generations. For example, if the robot employs its right arm to reach for an item, the kinematic state of the left arm is not useful to calculate the policy and becomes a distraction. Unlike visual attention, which can be improved by analyzing the direction in which the eyes are looking, there are no equivalent teaching cues for somatosensory information. To solve this problem the authors of the paper exploit the Transformer. Self-attention mechanisms like the Transformer allow to evaluate the relationships between elements on a sequence determining which features are of interest and which are not, this method can therefore be applied to the sensory input sequence. Different types of bi-manual tasks are performed (uncoordinated, loosely coupled, tightly coupled) numerous times, generating the sensory input dataset. Such dataset is then fed to the Transformer which generates a model capable of predicting the actions for both arms and a binary signal open/close for the gripper.

However, the main downside of IL and deep learning-based methods, in general, is the generation of the dataset. In order to be effective, deep learning methods require a lot of data, information that must consider different scenarios. While for a priori known environments this might not be a problem, the size of the dataset needed for a model that can generalize across different scenarios could be enormous. Moreover, the time and hardware needed for the training phase must be taken into account. In contrast, improved APF, are a simple, low-cost algorithm that can be applied directly. Although in the future, deep learning-based methods are likely to become very efficient at performing bimanual actions, for now, improved APF are a good compromise between effectiveness and simplicity.

There are methods that attempt to solve the problem of the large amount of data required in RL, but they have some limitations. For example, [12] proposes the Boosted Hybrid Reinforcement Learning (BHyRL). This approach is designed for learning reaching and fetching tasks in MM with reachability behavior priors while considering hybrid action spaces (a combination of both discrete and continuous actions that a robot can perform within its environment). First of all, Hybrid RL is designed to model the distribution of discrete actions utilizing the Gumbel-Softmax reparameterization. Afterward, it is trained a reachability prior (a model that determines the positions that a robot can reach) by utilizing data obtained from the robot's working area. Finally, is derived BHyRL, an algorithm that benefits from representing Q-functions as a combination of residuals. When presented with a new task, the acquired residuals are transferred and the task-specific component of the Q-function is learned. This approach ensures that the structure of the task is maintained from previous behaviors. Anyhow, the drawback of this study is that the agent is trained to maximize the likelihood of finding IK (Inverse Kinematics) solutions. As a result, optimal actions are learned only in relation to reachability, rather than manipulability. It is also not optimized for bimanual tasks and working in cluttered environments.

2.4 Artificial Potential Fields

The basic idea behind APF is that the manipulator moves in a field of forces, where the target location is an attractive pole and the obstacles are repulsive surfaces, the final movement of the manipulator is controlled by the resultant force [14]. Let p be the position of the end effector in the workspace. The resulting artificial potential field, to which the end effector is subjected, is:

$$U_{APF}(p) = U_A(p) + U_R(p)$$

where $U_A(p)$ is the attractive potential field generated by a goal and $U_R(p)$ the repulsive potential field generated by obstacles. Attraction and repulsion can be expressed in several ways, a possible formulation (and the one used in this thesis) is the following. $U_A(p)$ depends on a parameter k_g that determines the attraction of the goal, and is calculated as:

$$U_A(p) = \frac{1}{2}k_g(p - p_g)^2$$

where p_g is the goal position. $U_R(p)$ depends on three parameters: η (costant gain), ρ (shortest distance to the obstacle) and ρ_0 (limit distance of the potential field influence). It is calculated as:

$$U_R(p) = \begin{cases} \frac{1}{2}\eta(\frac{1}{\rho} - \frac{1}{\rho_0})^2 & \text{if } \rho \leq \rho_0 \\ 0 & \text{if } \rho > \rho_0 \end{cases}$$

Due to its straightforward mathematical formulation, minimal hardware demands, and quick planning capabilities, this method has found extensive use in robot navigation and obstacle avoidance. Nevertheless, the APF technique is subject to certain drawbacks, such as the possibility of becoming trapped in local minima, the target being inaccessible in closed proximity to obstacles, and oscillations occurring when navigating through obstacles or narrow passages.

To tackle some of these problems a solution based on decision trees has been proposed in [19]. Decision trees are commonly employed in machine learning for classification tasks. They serve as a prediction model that maps object attributes to object values. The benefits of decision trees in rule extraction and expression have been leveraged to create an enhanced path planning model based on APF. This new model incorporates decision tree-based methods to facilitate real-time and precise identification of current behaviors, enabling quick decision-making for subsequent path-planning steps. Even if this method shows good results, it has not been tested for dual-arm manipulation.

[4] proposes a system for the execution of uncoordinated and loosely coupled coordinated bimanual tasks, based on APF. For uncoordinated actions, each end effector is assigned to the closest target (considering the Euclidean distance) and then starts moving toward it via APF. However, in some cases, the manipulator paths might intersect due to the position of obstacles in the environment. In this scenario "goal configuration sampling" is performed to search for a configuration for both arms that avoids arm crossing. If, despite attempts, no valid configuration can be found, the tasks cannot be performed simultaneously and the arms must move consecutively. For loosely coupled coordinated tasks (e.g., passing an object from one manipulator to another, assembly of two parts) a method called "goal space sampling" is used. The space in which a common goal can be achieved is sampled to search for a suitable position for the manipulators to perform the coordinated task. If such a position is found the end effectors start moving towards it via APF. The process of identifying a mutually valid target is also capable of handling the discovery of new obstacles in real-time. If the target position cannot be reached anymore, due to an obstacle, a new target position is sought starting from the ones adjacent to the previous target. A difference with the method developed in this thesis lies in the scenario where the arms would cross. If in the previous work, the end effectors would move one at a time, in this thesis both arms continuously check online for self-collision; instead, this check is performed online. If a self-collision risk is detected, a recovery method is executed and the arms move consecutively from then on. In this way, the execution time of the action can be reduced. However, the main difference between the two approaches consists of the improved APF formulation since this thesis starts from the work shown in [10]. In order to reduce the risk of getting stuck in a local minima the dynamic generation of escape points around the obstacles is performed. Such points are attractive poles that the end effector

can follow to overcome the obstacles more easily; the process of deciding which point to pursue is made in real-time by solving a soft-constrained problem (CSP) [3] that optimizes both obstacle avoidance and goal achievement. In the chosen point, located in p_e, an attractive force is then applied according to the attractive constant k_e:

$$U_A^e(p) = \frac{1}{2}k_e(p - p_e)^2$$

However, this method has only been applied for single-arm shared control, and it is, in fact, the aim of this thesis to extend this work to a double-arm setup and to deal with the related challenges of spatial and temporal coordination.

3 Technologies used

3.1 TIAGo++

TIAGo++ is a mobile manipulator robot designed by PAL robotics[1]. It combines perception, navigation, manipulation, and human-robot interaction skills that make it a great mobile manipulation robot for research in fields such as ambient-assisted living, healthcare, or light industry.

Starting from the top of the robot (Fig. 3.1), the first component is a moving head containing an RGB-D camera. The head is fixed on a base that can be rotated and tilted by two motors. These movements allow to adjust the camera's viewing range expanding the perception area without repositioning the robot. Directly below the head, at the base of the neck, the robot has a stereo microphone that can be used for interaction and a speaker capable of playing different sounds.

Moving down to the torso, the main components are the seven Degrees Of Freedom (DOF) arms attached to the sides of the robot. The first joint allows the shoulder to rotate over the ground plane, like a 7 DOF inverted industrial arm. The arm end effector includes a force/torque sensor to monitor the load capacity at the arm end. The left arm is equipped with a parallel gripper, while the right arm features a five-finger hand as end effector. On the back of the torso there is a NVIDIA® Jetson™ TX2 GPU.

Continuing downward, there is a prismatic joint on the upper section of the

[1]https://pal-robotics.com/robots/tiago/

mobile base. This link can be lengthened or shortened to adjust the height of the robot by raising or lowering the torso. The mobile base is a differential drive equipped with all the essential elements for navigation. It incorporates a laser sensor for distance detection, two rear sonar sensors and a programmable LED belt that can display various lights and patterns (which may also signal a warning status). Inside the base it's also installed the internal computer that includes a Intel i7-7700 3.60GHz CPU, 16GB of RAM and a 500GB HDD. The operating system is Ubuntu LTS 64-bits.

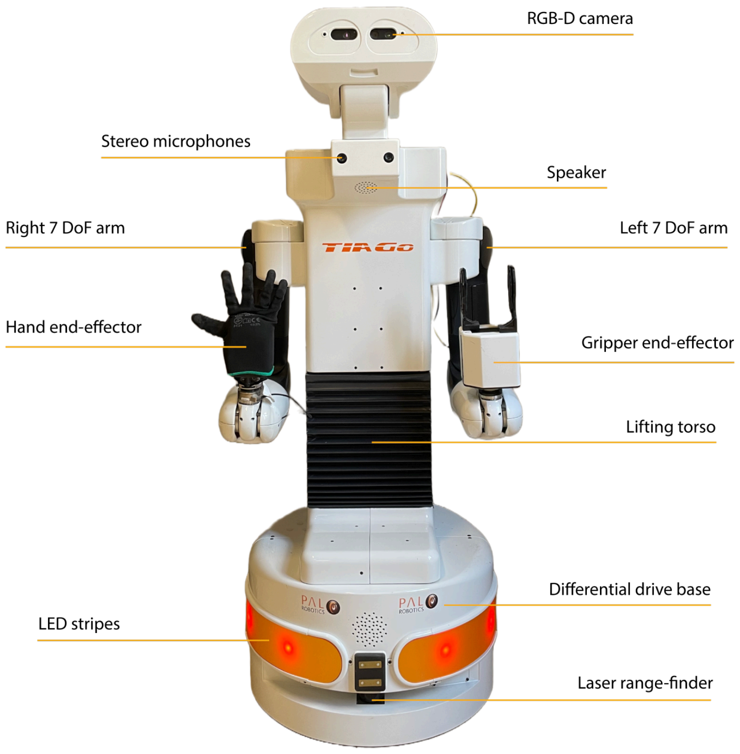


Figure 3.1: The TIAGo++ robot

3.2 Robot Operating System (ROS)

TIAGo relies on ROS[2] as middleware between the software layer and the Ubuntu OS. A robotics middleware functions as an interface that enables the communication between the software applications developed by the user and the different drives of the operating system that manage the robot.

[2]https://www.ros.org/

"*The Robot Operating System (ROS) is a set of software libraries and tools that help you build robot applications*" [29]. In fact, ROS is an open-source layer that provides operating system services such as hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management.

The origin of this middleware can be traced back to Eric Berger and Keenan Wyrobek, two doctoral students from Stanford University who identified deficiencies in the technical expertise of software developers, which hindered their ability to fully exploit the capabilities of robotic hardware [28]. Back in 2007, Eric Berger and Keenan Wyrobek became a part of Willow Garage, a recently established enterprise by Scott Hassan. During that year, Willow Garage released an initial version of ROS on SourceForge. Within three years, the company managed to produce a preliminary robot model capable of navigating inside an office and manipulating doors. After publishing comprehensive documentation on the project, the entire framework gained widespread acceptance and became a standard de facto in the field of robotics.

ROS Architecture

Packages: ROS software is arranged in packages. Typically, every package is assigned a particular operation to execute. This method guarantees that each package has a well-defined function, simplifying its use and debugging process. ROS packages are designed in adherence to the "Goldilocks" principle: they should provide sufficient functionality to be valuable, but not so much that the package becomes cumbersome and challenging to use in other software.

Nodes: processes that perform the operations necessary for the execution of a task. These units are assembled into graphs and communicate with each other through various means such as topics, services, and actions. As an example, in the management of a small robot, one node would oversee the wheel control, another would determine the optimal path to take, another would be devoted to collision avoidance, and so forth. The use of nodes in ROS offers a number of advantages for the system as a whole:

- They enhance fault tolerance: nodes operate independently, so if any one of them malfunctions, it only affects a single unit rather than the entire system

- They simplify code complexity in contrast to a monolithic structure that lacks modularity
- Implementation specifics are confidential since nodes utilize a severely restricted API to communicate with the rest of the graph

Topics: serve as the medium through which nodes communicate with one another. Essentially, they are buses that enable nodes to exchange messages. Messages can be defined as data structures comprising primitive data types such as integers, floating-point numbers, booleans, and arrays thereof. Moreover, messages may include nested structures. Each topic is capable of conveying a singular category of information, meaning that it can only transmit pre-determined data. The topics maintain a policy of anonymity for both the senders and receivers involved. For instance, if Node 1 publishes data in a certain topic, Node 2 must subscribe to that topic if it wishes to access the information being transmitted. Through the exchange, no details are communicated regarding who sent or received the data, ensuring complete anonymity. Additionally, it is worth noting that each topic may have multiple publishers and subscribers.

Services: another method of enabling the exchange of information between nodes. Although the topic-based model is highly adaptable for communication purposes, its many-to-many one-way transport system is not suitable for request/response interactions that are often required in distributed systems. To address this, request/response is facilitated through a service, which consists of a pair of messages; one for the request and the other for the reply. To provide a service, a ROS node defines the service server under a string name, a client will trigger the service by sending the request message to the server and waiting for a response. Clients may establish a persistent connection to a service to achieve better performance, but this comes at the expense of reduced resilience to changes in service providers.

Actions: in case a service requires significant processing time, users may prefer to have options such as canceling the request while it's running or receiving feedback on its progress at intervals. To achieve this, the *actionlib* package offers server tools for executing long-term goals that can be preempted. Furthermore, it provides clients with an interface for submitting requests to the server. For the client and server to exchange information, a set of messages that they can use to communicate must be established. This is accomplished through an action specification, which outlines the Goal, Feedback, and Result messages that will serve as the communication medium between the two parties.

- **Goal:** To execute tasks through actions, it is introduced the concept of a goal that can be transmitted to an ActionServer by an ActionClient. In the case of base movement, the goal would be represented by a PoseStamped message containing details of the robot's target location in the world
- **Feedback:** allows server developers to update an ActionClient regarding the gradual advancement of a goal. In the case of base movement, this could mean reporting the robot's present position along its path
- **Result:** upon accomplishing a goal, the ActionServer transmits a result to the ActionClient. This is distinct from feedback, as it is dispatched only once. This feature is particularly beneficial when the objective of the action is to supply some form of data

For this thesis, the ROS Melodic version was used.

3.3 Gazebo

Gazebo[3] is an open-source 3D robotics simulator that is natively supported by ROS. It offers a wide range of functionalities, such as its own format for describing robot models and environments (the SDF format), numerous freely available models and plugins, and efficient physics engines. With Gazebo, it is possible to simulate every aspect of the real world in detail, including gravity, wind, controllers dynamics, sensor behavior (taking into account noise), and inertia and friction distribution for any model. Gazebo shares many characteristics with ROS as an open-source product, including a supportive community, active development, and efficient support forums.

3.4 MoveIt

MoveIt[4] is an open-source platform for designing and operating various types of robots. It facilitates the planning of motion trajectories within an environment, and offers collision control capabilities to prevent obstacles from interfering with

[3]https://gazebosim.org/
[4]https://moveit.ros.org/

robot movements. Moveit enables the resolution of inverse and direct kinematics, as well as the execution of joint trajectories using a variety of controllers and hardware interfaces. Additionally, it integrates seamlessly with Gazebo and ROS control to create a comprehensive development platform. The Rviz Motion Planning plugin enables users to engage in interactive trajectory planning by relocating the end effector to the desired location, and to visualize the trajectory plan within the space. The primary ROS node of MoveIt is `move_group`, which provide users with a collection of ROS services and actions (Fig. 3.2). MoveIt provides three distinct methods for accessing the features and services provided by the `move_group` node:

1. `move_group_interface` package to interface to the node in C++
2. `moveit_commander` package to interface with python
3. Rviz Motion Planning plugin with the GUI

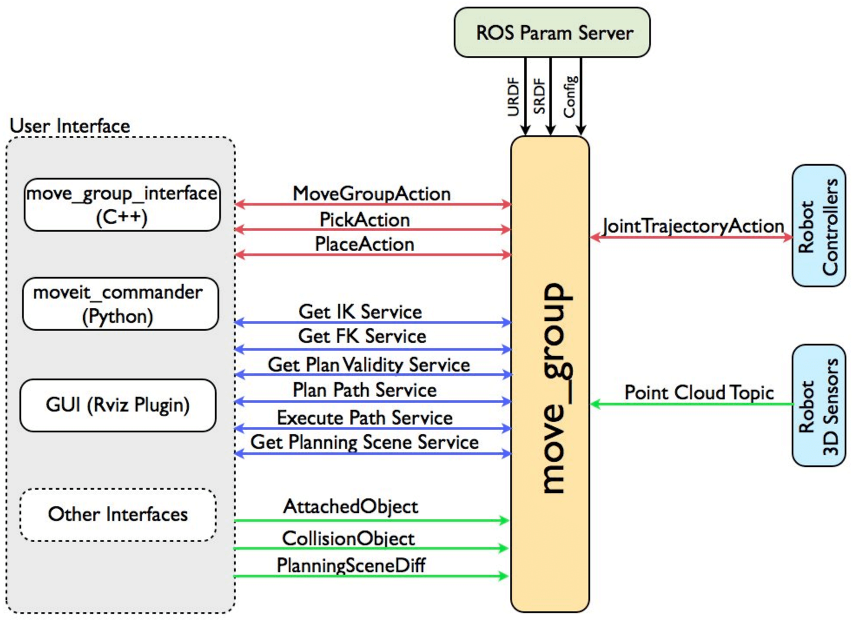


Figure 3.2: MoveIt move_group

The `move_group` node is configured via the ROS parameter server, which provides the node with the robot's URDF model, the SRDF file produced via the MoveIt Setup Assistant, as well as additional information such as joint limits, kinematics, motion planning, and perception. Typically, the `move_group` node communicates via ROS topics and actions, gathering information regarding the robot's current state and other general data, and dispatching orders to the controllers through the FollowJointTrajectoryAction[5] interface, which is a ROS action interface. MoveIt interacts with various motion planners through a plugin interface that employs a ROS Action/Service provided by the `move_group` node. The default motion planners for `move_group` are based on the Open Motion Planning Library (OMPL). Such library contains many samplig-based motion planning algorithms like PRM, RRT, RRTConnect, etc. When requesting a motion plan, MoveIt asks the motion planner to relocate the robot to a specified target pose or a different position of the joints, after which the `move_group` node generates the desired trajectory in order to move the robot.

3.5 AprilTag

AprilTags[6] are visual markers that are commonly used for various computer vision applications such as robotics, augmented reality, and camera calibration. These markers have a distinctive black and white pattern that is designed to be easily identifiable by a camera and includes a unique identification code. The pattern is similar to a 2D barcode and is built to be resilient to lighting changes, orientation, and scaling.

AprilTags were developed by the robotics research group at the University of Michigan and are released under an open-source license. They have become increasingly popular among robotics experts because of their accuracy, efficiency, and user-friendliness. AprilTags can be detected and monitored in real-time by a camera using specialized algorithms, which is why they are commonly used for tasks such as object recognition, pose estimation, and localization.

AprilTags are subdivided into families. Each family refer to a group of AprilTag designs that share similar properties, such as the size of the tag, the number of bits used to encode the ID, and the error correction capabilities. These families are identified by a number, such as 36h11 or 16h5, where the first number represents the size of the tag and the second number represents the number of bits used to encode the ID. Each AprilTag family is designed for a specific purpose, such as tracking large objects or small objects, working with close or far distances, or performing well under different lighting conditions.

[5]https://ros-planning.github.io/followjointtrajectory

[6]https://github.com/AprilRobotics/apriltag

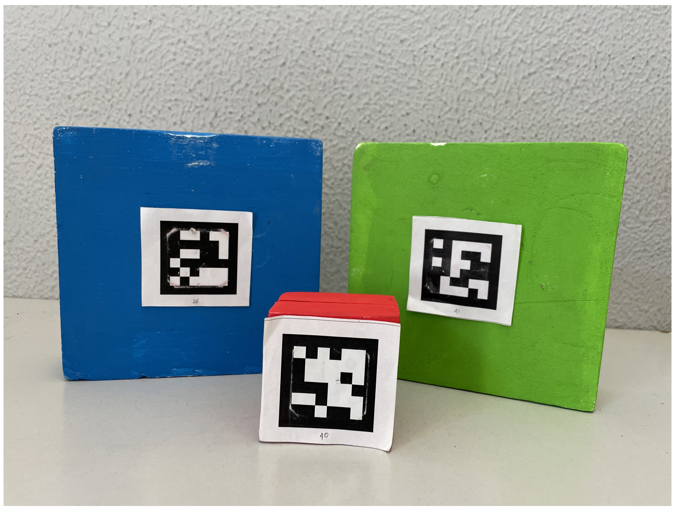


Figure 3.3: Examples of 36h11 apriltags

4 Solution & System Implementation

To develop a bimanual manipulation system based on APF, capable of operating effectively in dynamic environments, there are several important considerations that need to be taken into account. The first section of this chapter delves into formal definitions and presents a mathematical formulation of the constraints that must be considered. The second section explains the proposed solutions at a high level, outlining the architecture of the developed system. Lastly, in the final section of this chapter, a more detailed analysis of the proposed approach is presented. Specifically, it describes the ROS structure that has been employed and the algorithm utilized for the self-collision avoidance system.

4.1 Problem definition

4.1.1 Basic concepts

First of all, it is necessary to describe the space in which a bimanual robotic system can act. In robots such as PAL Tiago++, where the arms are positioned symmetrically with respect to the torso, the workspace W can be divided into three main areas: a safe zone S, a common zone C, and an unreachable zone U (Fig. 4.1).

$$W = S \cup C \cup U$$

The safe zone can be divided into two types: a safe zone for the left arm S_L and a safe zone for the right arm S_R. The main characteristic of this area is the possibility of end effectors to move freely, without the risk of colliding with the other arm and only obstacles can be considered. Objects positioned in S_L are unreachable for the right arm and vice versa.

$$S = S_L \cup S_R \qquad S_L \cap S_R = \emptyset$$

The common zone C is defined as the intersection of the workspaces of the two arms. It indicates the area where both arms are capable of operating. It is a risky area, where the task of picking or placing an object can be further complicated by the possibility of collision between the arms. The unreachable zone U, represents the area outside the workspaces of the left and right arms. Therefore, it indicates an area where neither arm can operate due to kinematic constraints. For instance, an object placed very close to the torso or even behind is likely to be unreachable.

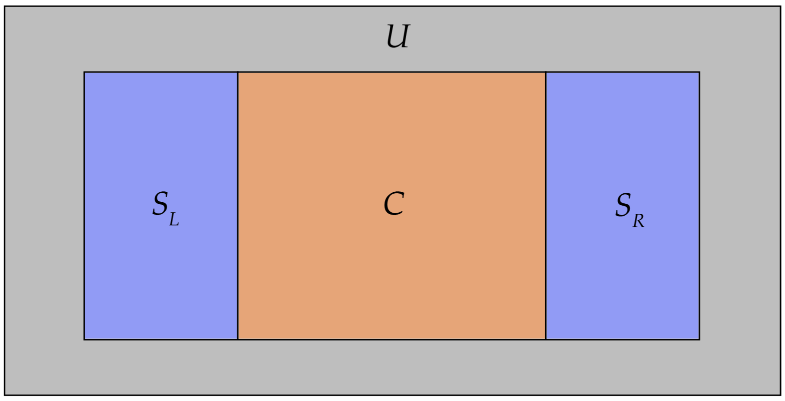


Figure 4.1: Workspace schema. S_L and S_R are the safe zones for the left and right arm, respectively. C is the common area and U is the unreachable zone

To determine the positions of the end effectors in the workspace over time, the reference frame set at the base of the robot is considered. The x-axis points to the right, the y-axis points upward, and the z-axis points outward from the origin (Fig. 4.2). Therefore, the position of a generic end effector e_i can be defined as $e_i = (x_i, y_i, z_i)$. According to the previous workspace description, $e_L \in \{S_L, C\}$ and $e_R \in \{S_R, C\}$. Moreover, to make the experiments uniform, it has been decided that at the beginning and at the end of a manipulation task, e_L and e_R must return to the starting position within the respective safe zones.

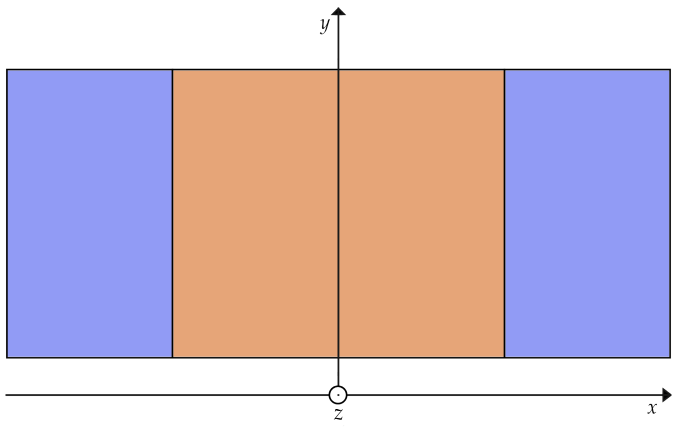


Figure 4.2: Workspace with reference frame

Some considerations regarding the objects in the workspace are also necessary. Let O be the set of all objects in the workspace. Such a set can be divided into two categories: items that can be grasped with only one end effector (O_{H1}) and items that must be picked up using both arms synchronously (O_{H2}). O_{H1} is further divided into objects that can be grasped only by the left end effector O_{HL} and only by the right end effector O_{HR}. If an object can be grasped by both manipulators will be in both O_{HL} and O_{HR}. An object may end up in O_{H1} or O_{H2} not only based on its size or weight (e.g. a large object necessarily requires both effectors to be grasped), but the logic behind the grasp also plays a role. For example, a plate can be picked up with only one arm, but in order to have greater stabilization and a more human-like form, both effectors should be employed. Objects in O_{H2} must require both arms to be used for grasping. For instance, a bottle can be picked up with both the end effectors at the same time, but it would be unnecessary and hinder proper grasping and subsequent movement. In order to grasp objects in O_{H2}, two grasping points have to be defined: g_L for the left arm and g_R for the right manipulator. Such points should be positioned as symmetrical as possible in order to be easily reached by the respective end effectors (Fig. 4.3).

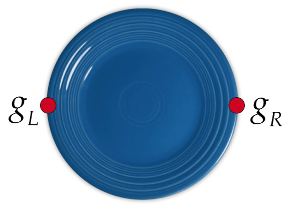


Figure 4.3: Example of grasping points for picking a plate

4.1.2 Constraints

To define the constraints of bimanual manipulation in a dynamic environment, two scenarios need to be considered: uncoordinated bimanual and symmetric coordinated bimanual. The asymmetric coordinated case (e.g., mixing a liquid in a cup) is unnecessary given that, in the pick phase, it does not differ from the uncoordinated case. After picking the two objects, the asymmetric coordinated action can be easily performed with a static motion planner.

Uncoordinated bimanual

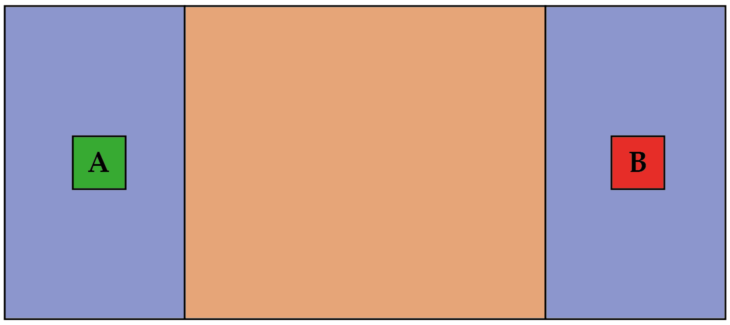


Figure 4.4: Uncoordinated bimanual setting. $A \in S_L$ can be grasped only by e_L, $B \in S_R$ can be grasped only by e_R

Assume a setting like the one in Figure 4.5. Two objects $A, B \in O_{H1}$ are positioned in S_L and S_R respectively. Due to their positions, A can only be picked up by e_L and B by e_R. In such a scenario, the arms cannot collide with each other during the path towards their respective goal and the main challenges are to avoid obstacles and handle the dynamic position of the objects in the workspace. Then, a mathematical definition of these points is provided. Consider an obstacle $Q = (x_Q, y_Q, z_Q)$ with a regular base, where r represents the maximum distance between the center and the edge of the base (e.g., in a cylindrical object, r is the radius of the circular base). Then, the following relationship must be respected in order to avoid collisions between the end effectors and the obstacles:

$$d(e_i, Q) > r + d_s$$

where $d(\cdot, \cdot)$ is the Euclidean distance between two points and d_s represents a small value that ensures the end effectors maintain a safe distance from obstacles.

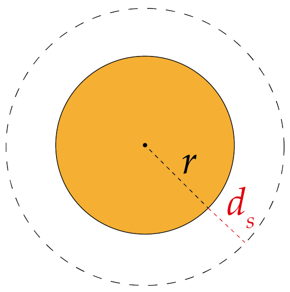


Figure 4.5: Safe distance from an obstacle

Assume now that A_1 and A_2 respectively represent the position of the object A at times t_1 and t_2. Then, the position of a generic end effector e_i, capable to reach A_2 at the instant of pick t_{pick} (with $t_{pick} > t_2$), is:

$$e_i(t_{pick}) = A_2 + d_p$$

where d_p is the distance that allows the end effector to grasp the object correctly (e.g. the offset to grasp the object in the gripper).

The previous considerations remain valid even if A and B are in the C zone, as long as they are respectively on the left and right side of the common area. The situation becomes different when the uncoordinated bimanual scenario is like the one in Figure 4.6. If the left end effector is in charge of picking up B and the right end effector is assigned the task of picking up A, no problems arise since the arm trajectories do not overlap. However, suppose instead we stick with e_L associated with A and e_R associated with B. This choice may be motivated by the need to use a specific end effector for a certain object. For example, a bottle can be grasped with either a hand-style end effector or a gripper, but the latter allows for greater stabilization and is therefore preferable in most cases.

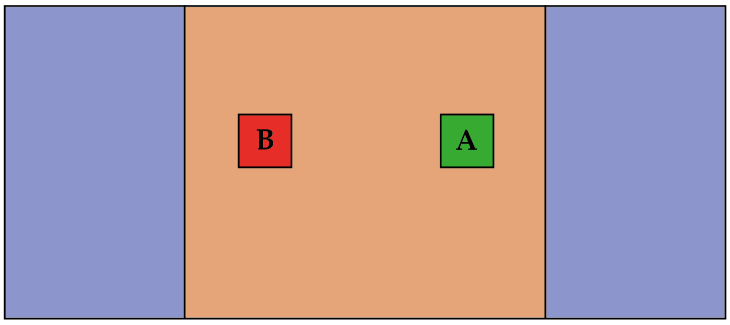


Figure 4.6: Uncoordinated bimanual setting that requires crossing arms to grasp A with e_L and B with e_R

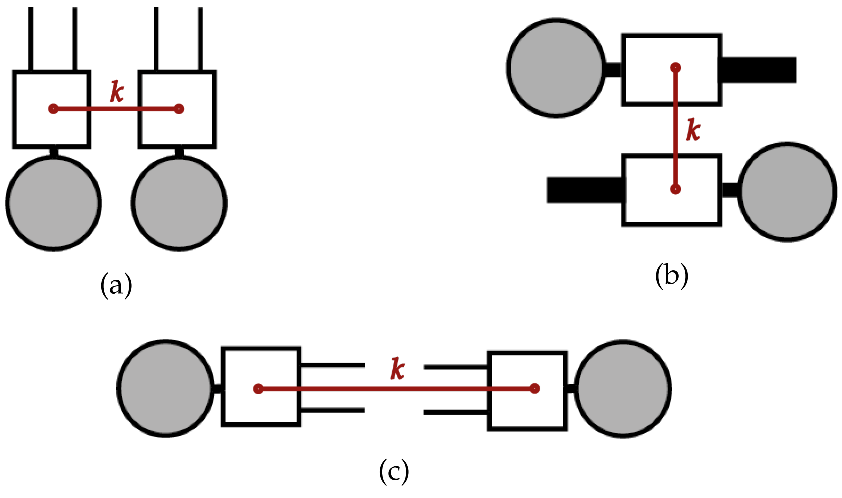


Figure 4.7: Different orientations of the end effectors

In this scenario, the probability of self-collision between the arms is very high. It is therefore necessary the implementation of a method that prevents this. It is hypothesized that the following two relationships must be respected to avoid arm-crossing and self-collision:

$$\begin{cases} x_L < x_R \\ d(e_L, e_R) > k \end{cases} \tag{4.1}$$

where x_L and x_R are the x coordinates of the left and right end effectors and k is the miniumum safe distance that must exist between the arms. Only the x coordinates are considered since movements along the y and z axis cannot cause the arms to cross. Even if $d(e_L, e_R) > k$ would be enough to prevent self collisions, the arm crossing constraint ($x_L < x_R$) adds an additional level of security to the grasping system. The k value should not be a constant but should vary according to the orientation of the end effectors. For example, if the manipulators are positioned as shown in Figure 4.7a, the value of distance k must be different compared to the setup depicted in Figure 4.7c. Assigning the correct value of k is critical to the self-collision system, as a value that is too low could result in collisions, while a value that is too high could cause slow and inefficient movements.

Coordinated Bimanual

Consider the coordinated bimanual scenario depicted in Figure 4.8. To pick the object $C \in O_{H2}$, e_L should reach the left grasping point g_L and e_R the right grasping point g_R.

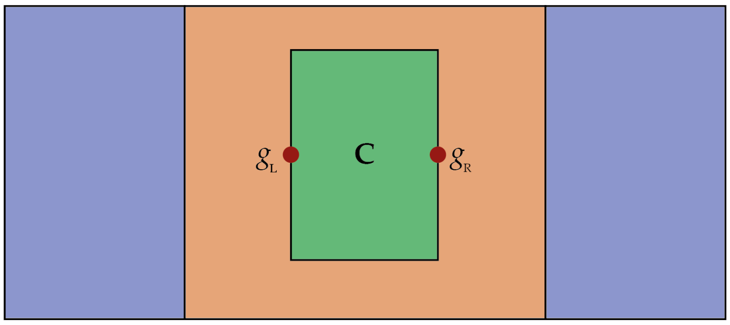


Figure 4.8: Coordinated bimanual setting. C is an O_{H2} object, g_L and g_R are the left and right grasping points

In addition to react to obstacles and changes in the position of objects as before, the manipulation system must perform a synchronous pick and move in a symmetric manner from the moment of grasping until the object is released. If one manipulator moves a certain amount along the three axes, the other manipulator must also move accordingly in the same way. Assume that in t_1 the object is picked and that in t_2 is released, then $\forall t_i : t_1 \leq t_i \leq t_2$ the following relationship must apply ($\vec{l}$ indicates a shift along the three axes):

$$e_L^i = e_L^{i-1} + \vec{l} \longleftrightarrow e_R^i = e_R^{i-1} + \vec{l} \tag{4.2}$$

To sum up, a dynamic bimanual manipulation system should avoid obstacles, react to changes in the positions of objects in the workspace, have a self-collision system and be capable to perform synchronous symmetric movements. The next section presents the APF-based solution proposed in this thesis to deal with such requirements.

4.2 Proposed system

A simple approach to implementing a manipulation system in robotics involves a two-step process: object detection followed by motion planning 4.9. The user chooses the object that should be grasped, for example, indicating the apriltag number associated with the object, and with which end effector to perform the grasp. This request is forwarded to an object detection system that uses vision techniques to determine the position and orientation of all objects in the field of view of the camera. At this point, if the selected object has been detected, the motion planner brings the end effector to a grasping position for such an item.

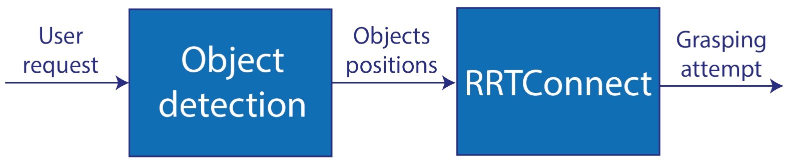


Figure 4.9: RRTConnect-only manipulation system

For this purpose, the RRTConnect motion planner can be employed. After defining the planning scene, i.e., the collision objects for all items in the workspace, it is then possible to search for a path that leads the chosen end effector to the respective object to be picked. RRTConnect can also be used to perform dual grasping. For example, in the uncoordinated scenario without crossing arms in Figure 4.5, it is possible to bring the two manipulators e_L and e_R simultaneously to their respective pick positions for the target objects $A, B \in O_{H1}$. In the coordinated case in Figure 4.8, RRTConnect can be used to perform the grasping of the object $C \in O_{H2}$ by bringing the manipulators to the corresponding grasping points (g_L for e_L and g_R for e_R) and then creating routines to lift/lower the object as described in the Equation 4.2. Although RRTConnect allows for such uncoordinated and coordinated grasps, it suffers from the problems associated with the basic implementations of sampling-based motion planners, described in Chapter 2. The main limitation is the lack of dynamism, i.e., online generation of the path to the goal. For this reason, if objects in the workspace are moved, or new ones appear, the probability of collisions and failure to pick the selected object is very high. Moreover, considering the uncoordinated case with crossing arms, RRTConnect is unable to find a solution. This happens because the motion plans inevitably end up overlapping thus causing the planner to fail. To overcome this problem, it is necessary to move only one arm sequentially. This results in a suboptimal behavior since moving the arms together could reduce the total time of the grasping routine in addition to having a more human-like and intelligent execution. For such reasons, this thesis proposes the architecture depicted in Figure 4.10.

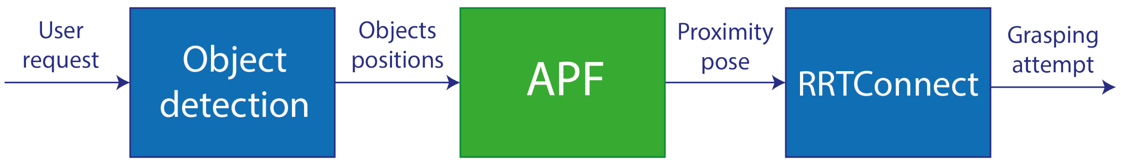


Figure 4.10: APF+RRTConnect manipulation approach

The object detection phase and the RRTConnect phase remain the same as the previous system. The difference is the addition of APFs as support for RRTConnect. As described in Section 2.4, APFs suffer from some problems including the risk of the agent getting stuck in a local minimum. For this reason, it was decided to use the improved APFs version described in [10] which allows this risk to be reduced through the dynamic generation of escape points around obstacles. Such escape points are attractive points that allow the agent to avoid obstacles more easily.

Thanks to APFs, it is possible to move the end effector toward the goal exploiting the ability to react to changes in the surrounding environment. Therefore, APFs can bring the manipulator to a position near the target object (named proximity pose), from which RRTConnect can be triggered to perform the grasp. APFs in fact do not have the precision required to pick up an object, but it is necessary to rely on RRTConnect. Even in this system, if there are major changes in the workspace during the RRTConnect phase, the pick is likely to fail. In any case, since the RRTConnect phase is much shorter than before, such changes are less likely to occur in a real-world scenario. Figure 4.11 depicts such reasoning:

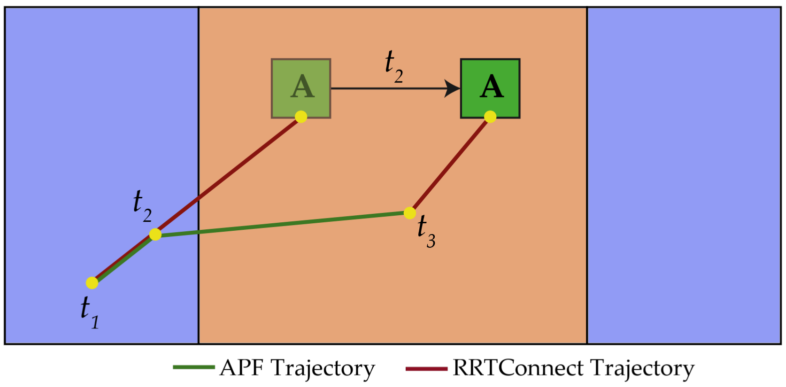


Figure 4.11: Illustrative representation of the functioning of the RRTConnect and APF+RRTConnect approaches

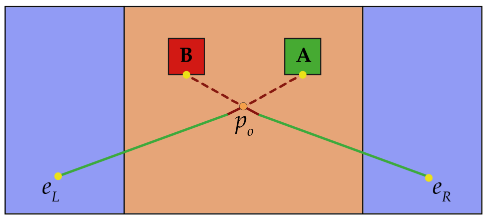
To better understand the concept, assume that the object to be picked up is $A \in O_{H1}$ which at instant t_1 is at position p_1. At instant $t_2 > t_1$ such object is moved from p_1 to a new position p_2. Suppose that the motion planning of the RRTConnect-only approach is executed at t_1, then the end effector will reach p_1 after a time t_p. If $t_p > (t_2 - t_1)$ then A is no longer there, thus missing the pickup. In contrast, in the APF+RRTConnect approach, the arm starts moving toward A at time t_1 adjusting its trajectory at time t_2 when A is moved. When the end effector arrives at a position close to the target object (at time $t_3 > t_2$), RRTConnect brings the manipulator to p_2 where A is present.

However, the previous example considered only a single arm. This thesis expands the system to two arms, necessitating a method to prevent self-collision between the arms.

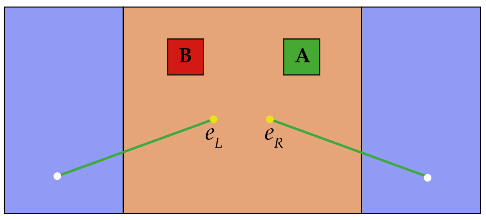
4.2.1 Self-collision avoidance

The core concept of the developed self-collision avoidance system is relatively straightforward. This system is designed to monitor the positions and orientations of end effectors over time. If a potential collision risk is detected, a recovery method is activated. This system employs a prioritization approach, favoring the manipulator closest to the goal. The other end effector is required to initially retreat by a certain distance (according to the context), providing ample space for the prioritized arm to safely reach the target object. Subsequently, it remains stationary until further notification. Once the arm with priority completes its routine, the waiting end effector can resume its movement towards its goal, thereby concluding the grasping routine.

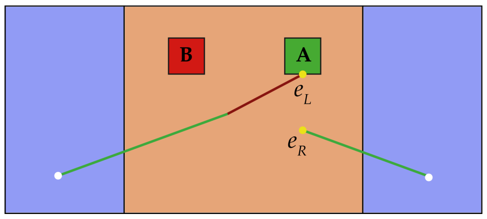
Consider, for instance, the application of such system to an uncoordinated bimanual case with arm crossing like the one in Figure 4.6. Without the self-collision avoidance system, the paths of the two end effectors would overlap at point p_o, resulting in an arm collision, as depicted in Figure 4.12a. On the other hand, by implementing the previously described control system, the arms would halt upon detecting the risky situation (Figure 4.12b). Assuming that e_L is closer to its goal (A) compared to e_R, the left manipulator is given priority. As a result, the right arm moves backward to create sufficient space for the left arm to complete its grasping routine (Figure 4.12c). Finally, e_R can reach point B without encountering any issues (Figure 4.12d).



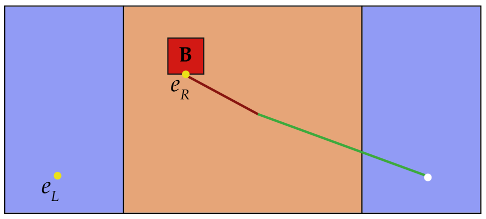
(a) The trajectories of the end effectors would overlap, colliding in p_o



(b) e_L and e_R must stop when a potential collision is detected



(c) Priority given to the end effector closer to the goal (e_L)



(d) Once e_L has finished its routine, e_R can start

Figure 4.12: Self-collision avoidance system

4.3 Implementation

The entire solution presented in the previous section was implemented in ROS using C++. The code developed in [10] was used as starting building block.

One of the main nodes of the system is `robot_controller`. As the name suggests, it handles the robot's movements via RRTConnect. In fact, using the */move_srv* service, it is possible to request the pick or release of a specified object, bring the end effector to a certain position, adjust the arm joints, and perform predefined routines such as bringing the robot to an approach or home position. In addition, the `robot_controller` node is responsible for managing the dynamic environment. It is subscribed to the `apriltag_ros` node that constantly publishes the positions and orientations of the objects in the field of view of the camera. Such data are subsequently used to dynamically modify the planning scene[1] (i.e. the collision objects). However, a problem arises: when an end effector moves, it may hinder the reading of apriltags (i.e., occlusion). As a result, some objects may not be added to the planning scene, leading to potential

[1]It is assumed that the dimensions of the objects in the workspace as well as the grasping points for the items in O_{H2} are known a priori

collisions. To solve this, the Algorithm 1 is proposed. The basic idea is that, for the whole duration of the manipulation routine, if an object "disappears" (i.e., is no longer in O) it is not deleted from the scene but remains in its last known position, which is recorded in R.

Algorithm 1 Object recorder

```
Require: $O$: vector of current detected objects; $P$: vector of picked objects;
  $R$: vector of objects detected so far
Ensure: $O_f$: $O$ with missing objects
  for all $o \in O$ do
    if $o \in R$ then
      $R$.update($o$) {update the position of $o$}
    else
      $R$.push($o$)
    end if
    $O_f$.push($o$)
  end for
  for all $r \in R$ do
    if $r \notin O$ and $r \notin P$ then
      $O_f$.push($r$)
    end if
  end for
  return $O_f$
```

Once RRTConnect is ready to be used it is then time to start the APFs. If the user specifies only one apriltag code and which end effector to use, then the single-arm manipulation routine is triggered. The `apf_control` constantly receives information from the `robot_controller` node about which goal to pursue and its position, the location of obstacles and their respective escape points. By leveraging such information, it is therefore possible to calculate the attractive and repulsive force acting on the considered end effector, moving it towards its target. For proper operation of APFs, certain parameters need to be tuned, some of the most influential are:

- **Attractive potential fields gain**: parameter defining the attractive force of the target object
- **Repulsive potential fields gain**: parameter defining the repulsive force of the obstacles
- **Repulsive force threshold distance**: parameter defining the distance from obstacles where the repulsive force begins to act

- **Escape gain**: parameter defining the attractive force of the escape points
- **Goal radius**: distance from the target object that the end effector must reach. For example, a value of 0.10 indicates that the end effector uses APFs up to ~10cm away from the target, from that point the grasp is performed by RRTConnect

In summary, the nodes `apf_control` and `robot_controller` allow single-arm actions to be performed in a dynamic environment with APFs+RRTConnect. Figure 4.13 depicts the ROS structure described so far.

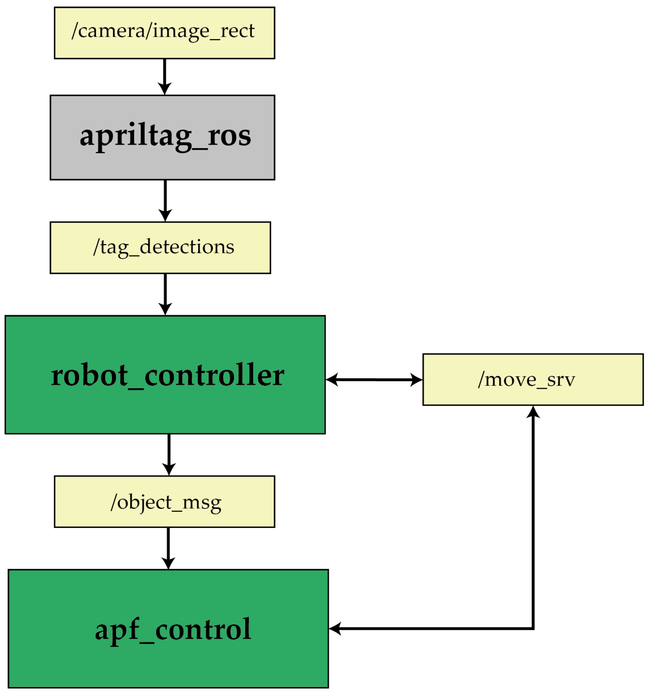


Figure 4.13: Single-arm APF+RRTConnect ROS structure

If the user specifies two apriltag codes and the respective end effector to be used, the dual-arm manipulation routine is triggered. Two `apf_control` nodes are started, one for each arm (named `left_apf_control` and `right_apf_control`). There are a few differences in the ROS structure compared to Figure 4.13. First of all, `robot_controller` publishes the messages */object_msg_left* and */object_msg_right* instead of the single */object_msg*, but the structure remains the same. It was decided to differentiate the messages since the goal for the left end effector must be considered as an obstacle by the right end effector, and vice versa.

Further considerations are also needed for performing synchronous or asynchronous grasps. For the latter, it was decided to adopt the following methodology: assuming that the left manipulator e_L reaches its goal via APF before the right end effector, the left arm will initiate the pick of its target object via RRTConnect. If the right manipulator reaches its target item during this phase, it will remain idle and wait until the left arm completes its routine. Once e_L has grasped the object and returned to its safe position, e_R can proceed to carry out the pick of its target item. Such a synchronization is achieved through an exchange of data via services (*/left_arm_feedback* and */right_arm_feedback*) between the left arm node and the right arm node. Instead, for the execution of synchronous grasps, if e_L reaches its APF goal before e_R, the left end effector waits until the right manipulator has also reached its goal. Once this condition is met, the synchronous grasp is executed via RRTConnect. The synchronous pick is performed by the `parallel_pick` node, with which the left and right arm nodes constantly communicate their current state. The decision on which behavior to perform depends on the value assigned to the *parallel_pick* parameter. If set to *true*, a synchronous pick is executed. If set to *false* (or left unspecified), an asynchronous pick is performed. If the pick of an object in O_{H2} is requested, it is not necessary to specify this parameter, as it will be assigned the value *true* by default.

4.3.1 Self-collision avoidance

The self-collision avoidance system, described in Section 4.2.1, is implemented in the `bimanual_coordinator` node in the package of the same name. This node is launched whenever a bimanual movement is requested. The algorithm can be subdivided into two phases: arm crossing control and a coordination routine. The arm-crossing control algorithm iteratively checks whether the end effectors are in danger of colliding with each other until both arms have finished the pick and returned to the safe position. In order to work properly three parameters need to be tuned: k, γ, and μ. k is the minimum safe distance that must exist between the two end effectors. As mentioned in Section 4.1, the distance k cannot be a fixed value but must depend on the orientation of the end effectors. This dependence is determined by the parameters γ and μ. As shown in the Algorithm 2, the RPY orientation of the end effectors are considered, μ and γ are the weights for the *sum_pitch* and *diff_yaw* variables, respectively. The values assigned to these parameters have been determined empirically by considering various configurations of the Tiago++ robot arms.

Algorithm 2 Arms crossing control

```
Require: $r_y$, $l_y$: position along the y-axis of the right and left end effector;
  $r_{yaw}$, $l_{yaw}$: yaw angle of the RPY orientation of the right and left end effector;
  $r_{pitch}$, $l_{pitch}$: pitch angle of the RPY orientation of the right and left end effector;
  $k$: minimum safe distance between end effectors;
  $\gamma$, $\mu$: weights
Ensure: true if arm crossing detected
  $d \leftarrow | r_y - l_y |$
  $d_{yaw} \leftarrow | r_{yaw} - l_{yaw} |$
  $s_{pitch} \leftarrow | r_{pitch} + l_{pitch} |$
  $K \leftarrow \max(k, k + ((\gamma \cdot d_{yaw}) - (\mu \cdot s_{pitch})))$
  if $d < K$ then
    return true
  end if
  return false
```

In case a risk situation is detected, the arm crossing control returns *true*, and the coordination algorithm is triggered (Algorithm 3). First of all, the arms are stopped via the topics */start_and_stop_left* and */start_and_stop_right*. Then, if it is the first stop since the start of the manipulation routine, the recorded movement vectors of the left and right arms are saved. These arrays contain the joint values of the respective arms during the APF trajectories (rm_L and rm_R). However, they are not sent to the coordination algorithm (via the */recorded_movement_left* and */recorded_movement_right* topics) in their entirety. Instead, they are sampled based on a parameter ϕ. For example, in Figure 4.14 are depicted the four positions recorded in rm_L. The parameter ϕ determines how much the arm without priority needs to retract to allow the leading arm to reach its goal. If the value of ϕ is too low, it can lead to continuous blockages and jerky movements. Conversely, if the value is too high, there would be no significant time advantage over moving the arms individually one at a time. After saving the recorded movements, it is checked which end effector is closest to the goal. If the left manipulator has the priority, the right arm returns to its previous position in rm_R and the left end effector will be given the green light to start (vice versa if the right arm has priority). If further collision risks occur, the right arm move back further and further until it might return to the safe starting position.

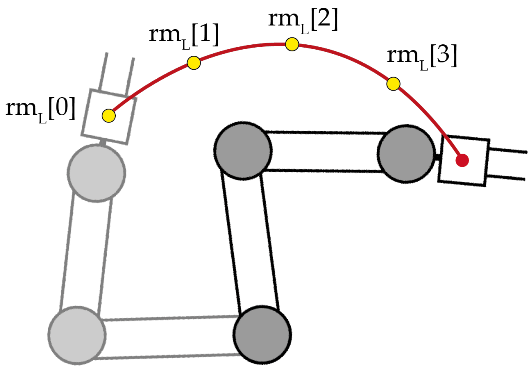


Figure 4.14: Example of recorded movement vector

Algorithm 3 Coordination routine

```
Require: distance_left: distance between the left end effector and its goal
  distance_right: distance between the right end effector and its goal
Ensure: self collision avoidance
  counter ← counter + 1
  stopArmLeft()
  stopArmRight()
  if counter == 0 then
    $rm_L$ ← getRecordedMovementLeft()
    $rm_R$ ← getRecordedMovementRight()
  end if
  if distance_left > distance_right then
    pos ← $rm_L$.size() - 1 - counter
    moveArmLeft($rm_L$[pos])
    startArmRight()
  else
    pos ← $rm_R$.size() - 1 - counter
    moveArmRight($rm_R$[pos])
    startArmLeft()
  end if
```

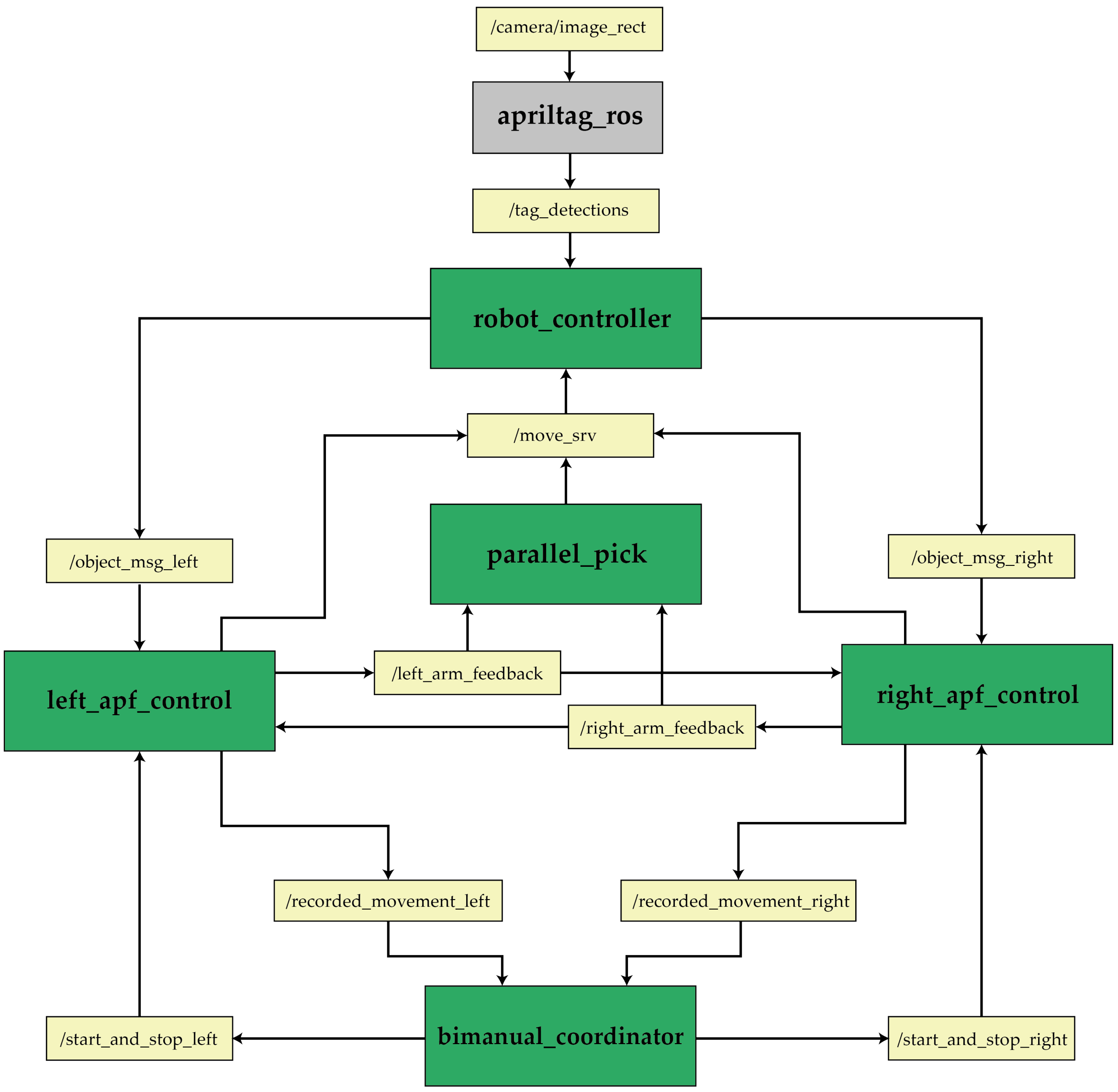


Figure 4.15: APF+RRTConnect manipulation system

5 Experiments

This chapter presents an evaluation of the performance and effectiveness of the proposed dual arm manipulation system based on APF. To assess the capabilities and limitations, it has been tested in the execution of uncoordinated and coordinated bimanual actions. Moreover, the advantages of using the self-collision avoidance system are shown.

5.1 Setup description

The workspace on which the robot can operate consists of a table measuring 124x90x71 cm. The TIAGo++'s torso is at a distance of 34 cm from the edge of the table which allows the robot to observe nearly the entire workspace. As a result, the camera remained stationary throughout all the experiments, without requiring the additional detection phase in which the head is moved to scan the entire table. The objects used for manipulation are the following:

- Blue polystyrene cube (7x7x7 cm) with apriltag code 29, associated with the left end effector and used in uncoordinated bimanual experiments
- Red polystyrene cube (7x7x7 cm) with apriltag code 42, associated with the right end effector and used in uncoordinated bimanual experiments
- Cardboard box (30x40x40 cm), with apriltag code 0, that requires both end effectors to be grasped and used in the coordinated bimanual experiments

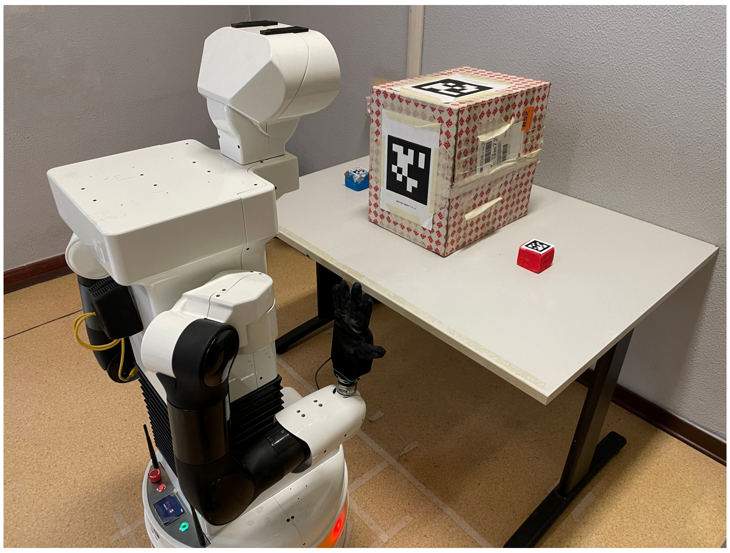


Figure 5.1: Photo of the workspace used in the experiments

5.2 Experimented modalities

To test the dynamic capabilities of the proposed APF+RRTConnect manipulation system, it has been compared with an RRTConnect-only approach in performing uncoordinated and coordinated actions. Moreover, another experiment has been performed to show the utility of the proposed self-collision avoidance system. The comparison is conducted between moving the arms simultaneously and moving the arms sequentially, in an uncoordinated bimanual scenario, using the proposed APF+RRTConnect system. The RRTConnect-only system was not taken into account because, as explained in Chapter 4, it is incapable of performing the bimanual task with crossing arms by moving both end effectors simultaneously. Even if a comparison can be performed, moving one arm at a time, the RRTConnect-only system would consistently miss the pick due to the dynamic nature of the environment. Therefore, given the meaningless of this condition, this has not been tested. The modalities are evaluated based on the metrics described in Section 5.3.

- **Uncoordinated bimanual:** the left end effector must reach and grasp the blue cube, instead the right manipulator has to pick up the red cube. The pick is asynchronous, meaning that the end effectors do not perform the pick simultaneously. Instead, it has been decided to prioritize the arm that reaches its target object first. The other manipulator must wait for the priority arm to finish the manipulation before proceeding. Three different initial positions were chosen for both objects to test the adaptability of the system. For each of these positions, twenty test runs are performed with

APF+RRTConnect and twenty test runs with RRTConnect-only (i.e., 60 per modality in total)

- **Coordinated bimanual:** the left end effector must reach the left grasping point of the box and the right manipulator the right grasping point. After grabbing the object, the box must be raised. If the box does not fall and the base remains parallel to the table, the pick is considered successful. Sixty test runs are performed with APF+RRTConnect and sixty test runs with RRTConnect-only
- **Uncoordinated bimanual with crossing arms:** as the uncoordinated bimanual case described above but with the need to coordinate the arms to avoid a self-collision. A total of sixty test runs were carried out by moving the arms simultaneously and another sixty by moving the arms sequentially

5.3 Metrics description

To perform the comparison analysis between the proposed APF-based system and the RRTConnect-only approach, the following metrics are considered for each test run and for each end effector:

- **SR[1] APF**: determines the ability of APFs to reach the point at which RRTConnect is triggered. For each run, a value of 1 is assigned if the end effector reaches such point, 0 if for any reason during the movement the system fails or crashes
- **SR RRTConnect**: determines the ability of RRTConnect to perform the requested pick motion. For each run, it is assigned value 1 if the movement is executed (even if the object is eventually dropped), 0 if the motion planner fails for any reason
- **SR Pick**: determines the ability to grasp the requested object. For each run, a value of 1 is assigned if the object is successfully grasped and remains stable for the entire duration of the picking routine, otherwise, a value of 0 is assigned

These metrics are considered separately for the left and right arms.

In the coordinated scenario (i.e. grasping an object in O_{H2}), in addition to the previous metrics, the **grasping offset** is also considered. In order to ensure a proper grasp of the box, it is essential for the end effectors to accurately reach the left grasping point g_L and the right grasping point g_R. This precision is

[1]Success Rate

necessary to prevent any backward or forward sliding of the box when it is lifted. For this reason, the grasping offset metric indicates the distance between the actual grasping position reached by the end effectors and the expected ones.

In the uncoordinated case with crossing arms, in addition to the three initial metrics, the **routine times** are also considered, namely the time to perform the routine. The pick routine for each arm consists of reaching the pick point via APF, grasping the item, and returning to the safe starting position.

5.4 Results

5.4.1 Uncoordinated bimanual

In this experiment, the cubes (i.e., the target objects) are moved during the arm motion. The results show that the proposed APF+RRTConnect system reacts to such position change (Figures 5.3a, 5.4a, 5.5a, 5.6a, 5.7a, 5.8a) and is therefore capable to perform the pick in most cases (59/60 for the left arm, 52/60 for the right arm).

It should be kept in mind that the SR pick of the right arm is also affected by the different end effector. In fact, the hand of the TIAGo++ is not always capable of ensuring a secure grip, resulting in instances where the cube has slipped. In such cases, the pick was considered to have failed, even though APF and RRTConnect behaved correctly. In contrast, the gripper on the left arm is much more reliable. Unlike the APF+RRTConnect system, the RRTConnect-only approach does not react to changes in position of the target objects (Figures 5.3b, 5.4b, 5.5b, 5.6b, 5.7b, 5.8b). As a result, it is unable to successfully grasp them. Furthermore, when using APF as the first step, RRTConnect never failed, unlike when using RRTConnect directly (52/60 SR RRTConnect Left).

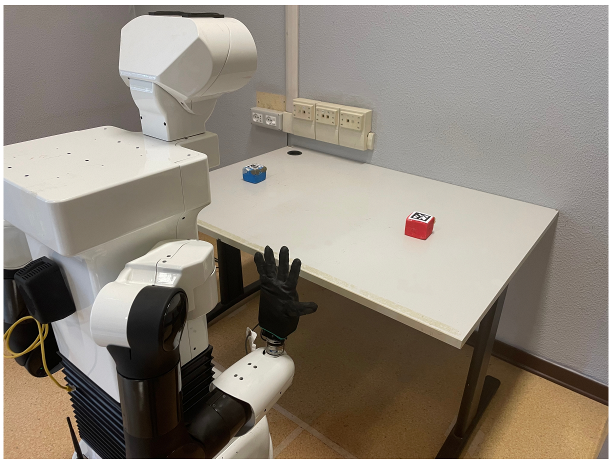
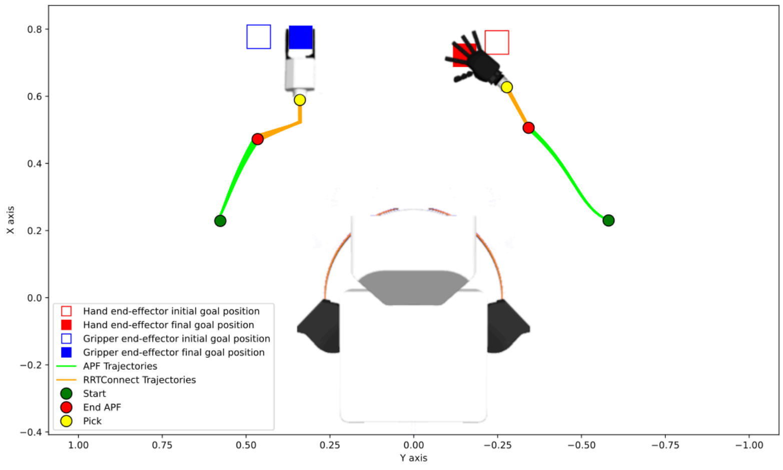


Figure 5.2: Uncoordinated bimanual setup

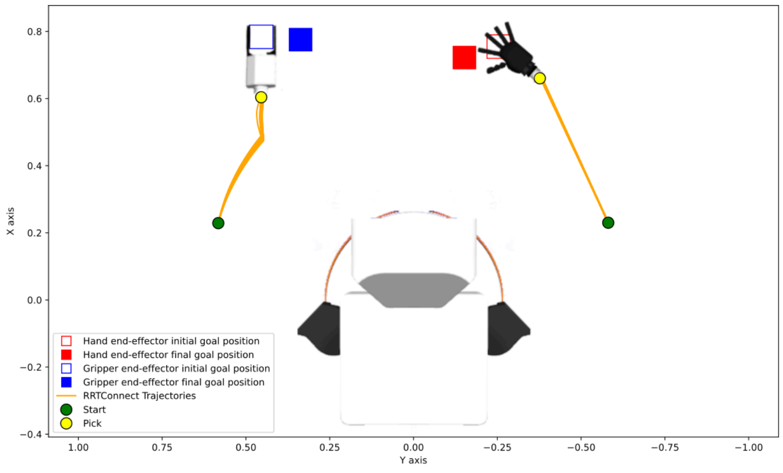
Position 1

	APF+RRTConnect	RRTConnect
SR APF Left	20/20	-
SR APF Right	20/20	-
SR RRTConnect Left	20/20	15/20
SR RRTConnect Right	20/20	20/20
SR Pick Left	20/20	0/20
SR Pick Right	15/20	0/20

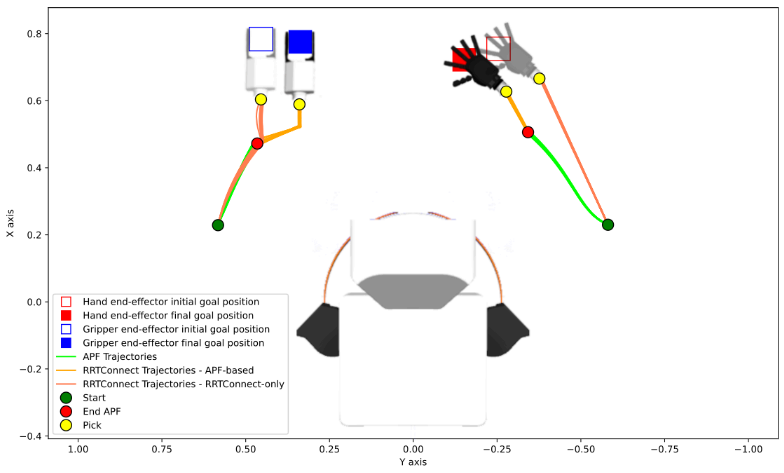
Table 5.1: Uncoordinated bimanual success rates - Position 1



(a) APF+RRTConnect behavior in 2D. The arms react to the change in position of the target objects and thus are able to grasp them

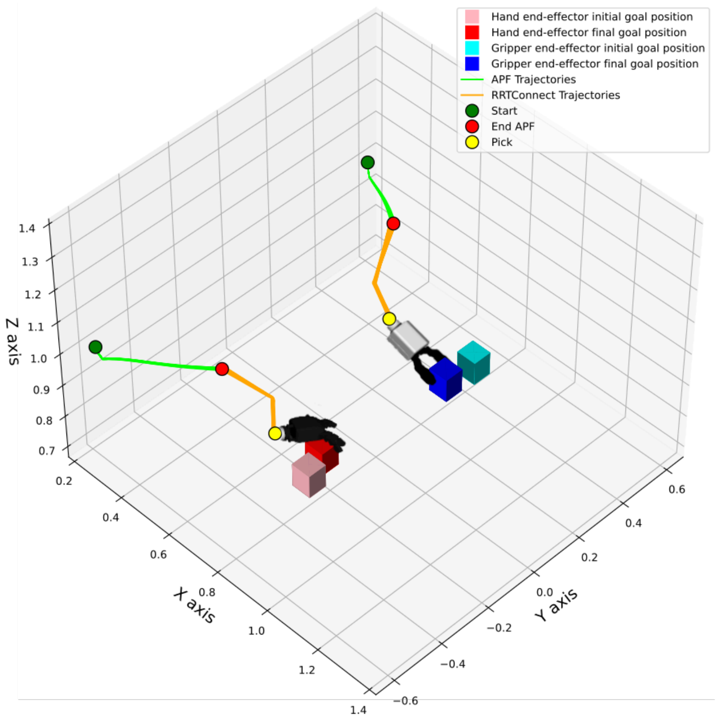


(b) RRTConnect-only behavior in 2D. The arms do not react to the change in position of target objects and thus fail the grasp

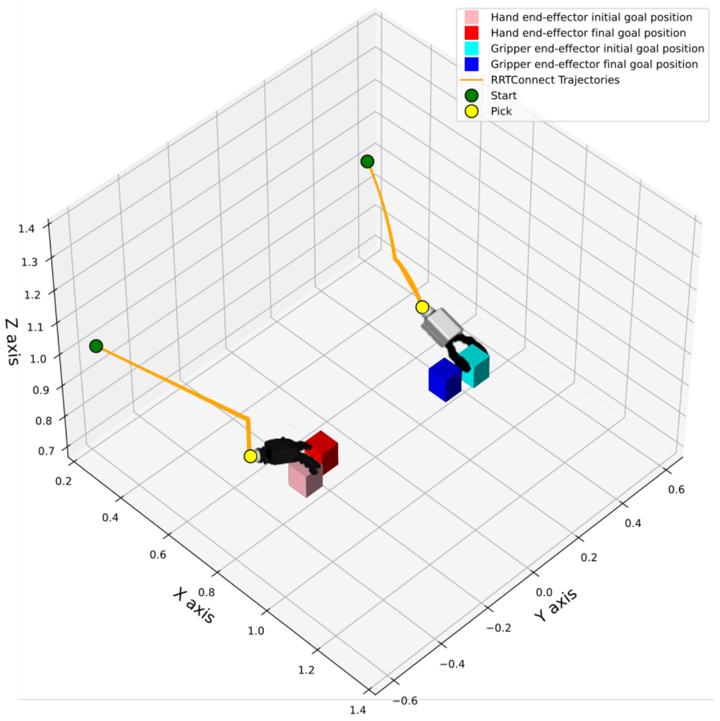


(c) Merged graphs in 2D

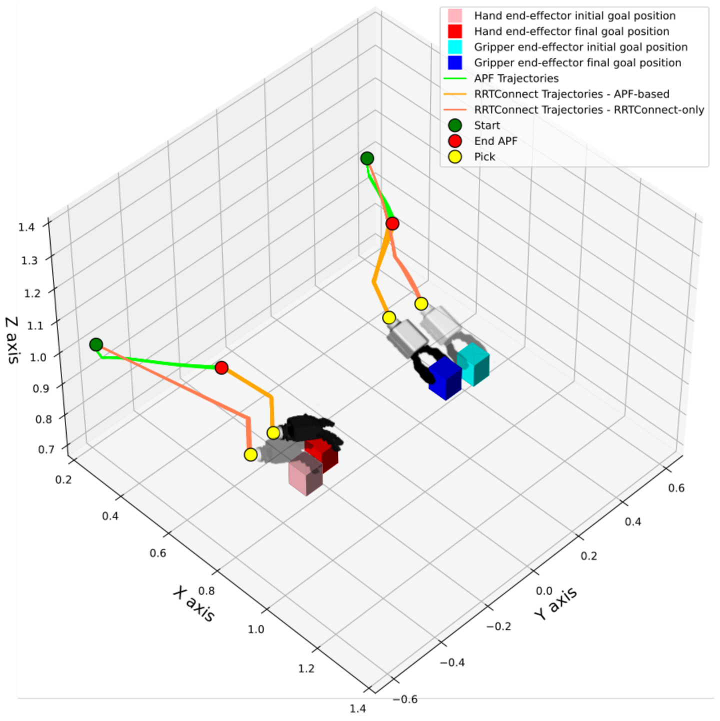
Figure 5.3: Uncoordinated bimanual - Position 1 - 2D



(a) APF+RRTConnect behavior in 3D. The arms react to the change in position of the target objects and thus are able to grasp them



(b) RRTConnect-only behavior in 3D. The arms do not react to the change in position of target objects and thus fail the grasp



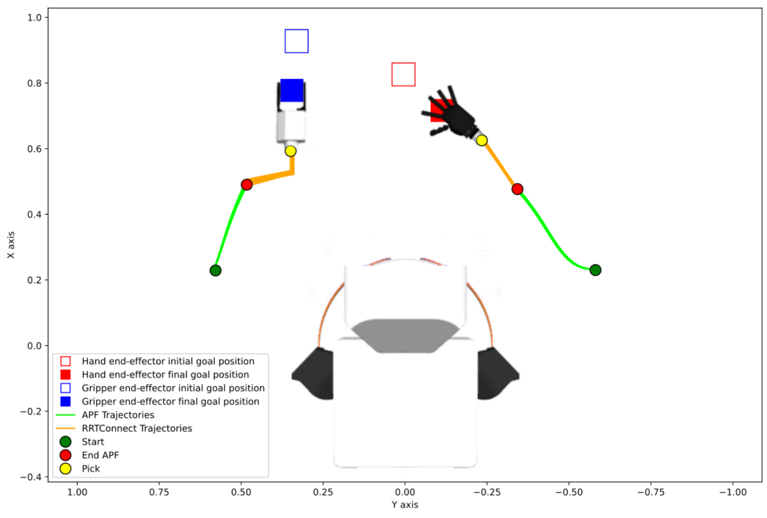
(c) Merged graphs in 3D

Figure 5.4: Uncoordinated bimanual - Position 1 - 3D

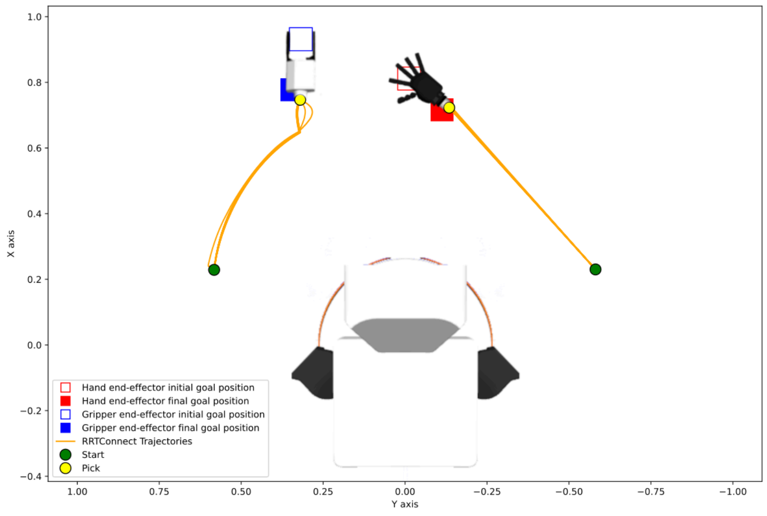
Position 2

	APF+RRTConnect	RRTConnect
SR APF Left	20/20	-
SR APF Right	20/20	-
SR RRTConnect Left	20/20	18/20
SR RRTConnect Right	20/20	20/20
SR Pick Left	20/20	0/20
SR Pick Right	20/20	0/20

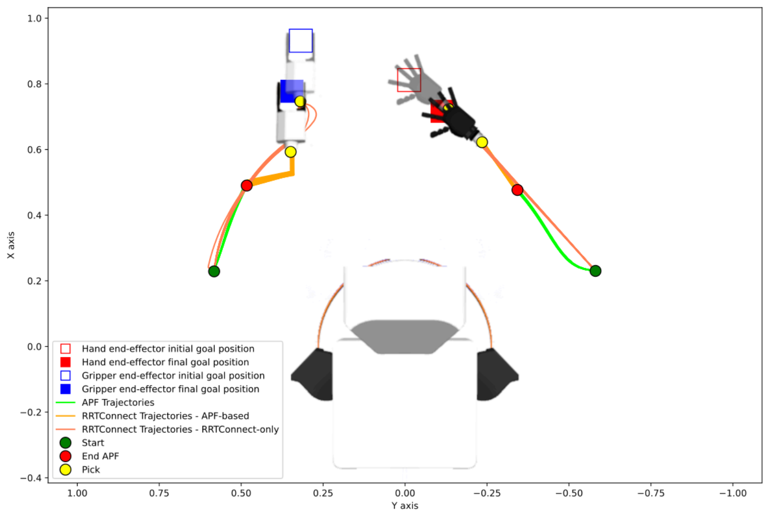
Table 5.2: Uncoordinated bimanual success rates - Position 2



(a) APF+RRTConnect behavior in 2D

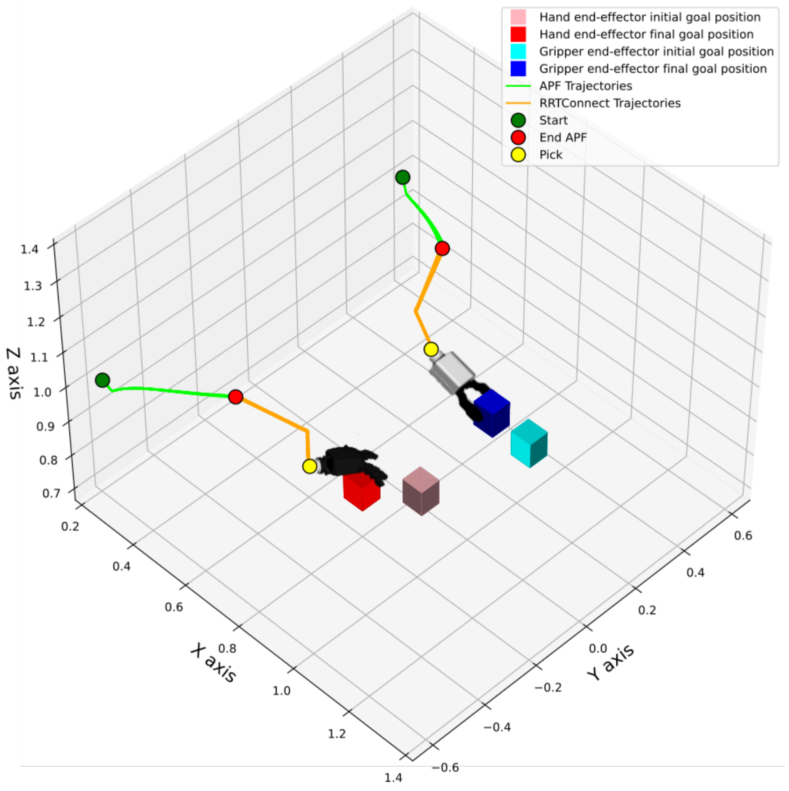


(b) RRTConnect-only behavior in 2D

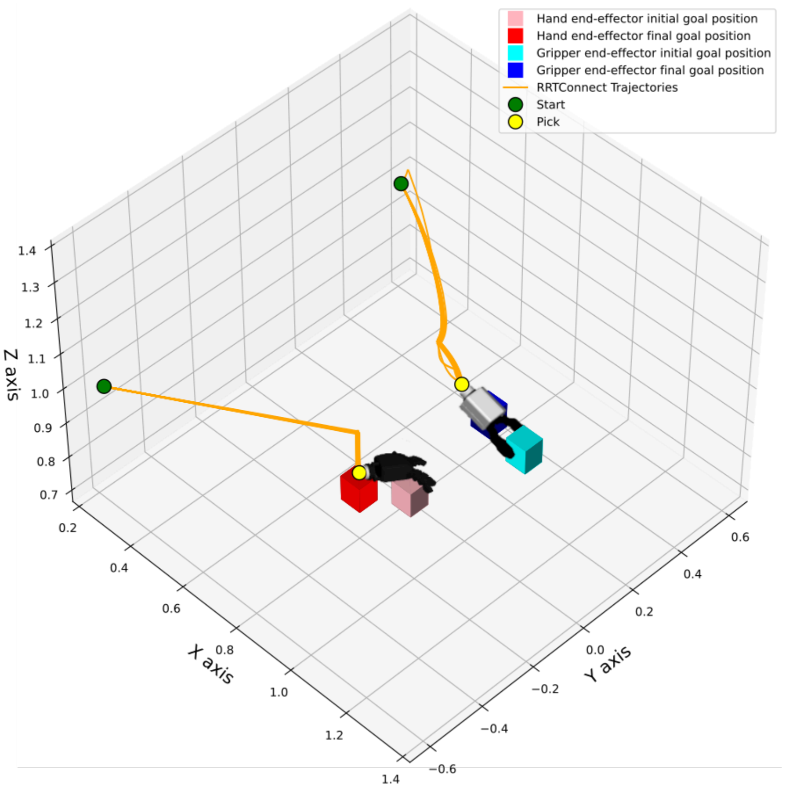


(c) Merged graphs in 2D

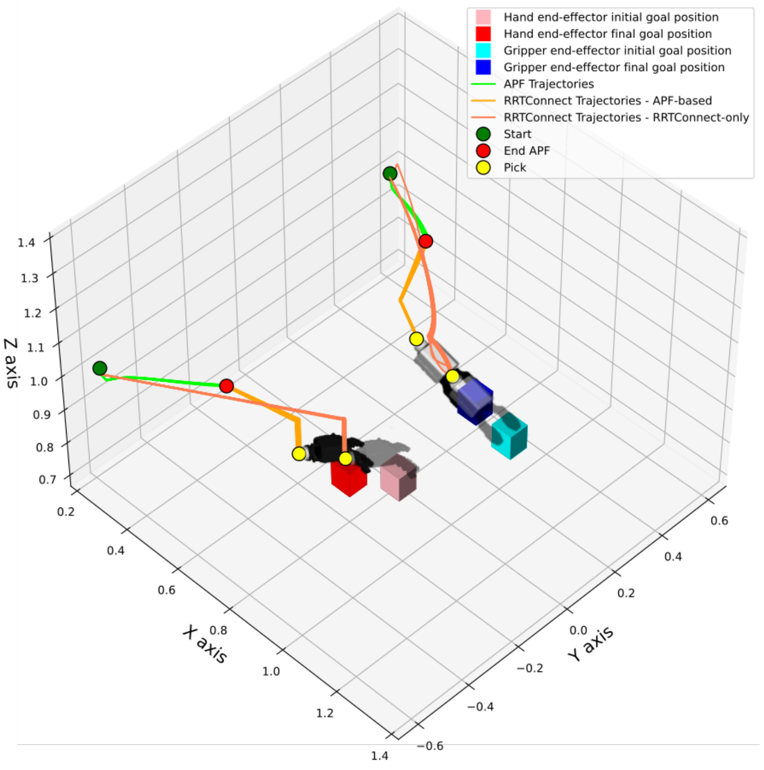
Figure 5.5: Uncoordinated bimanual - Position 2 - 2D



(a) APF+RRTConnect behavior in 3D



(b) RRTConnect-only behavior in 3D



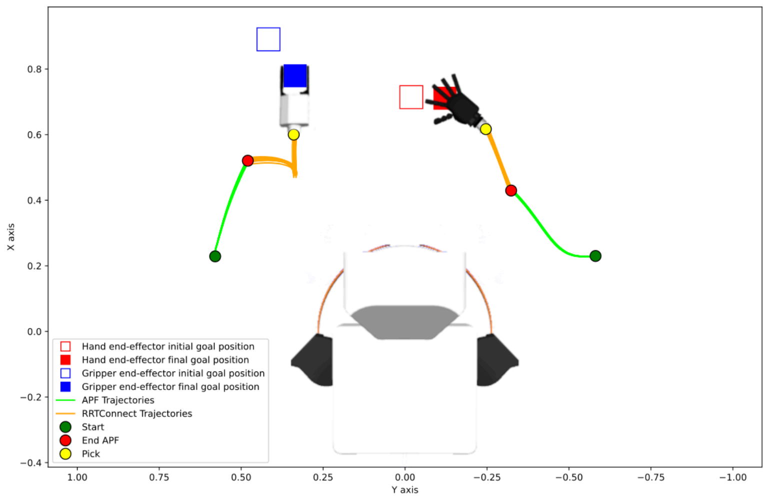
(c) Merged graphs in 3D

Figure 5.6: Uncoordinated bimanual - Position 2 - 3D

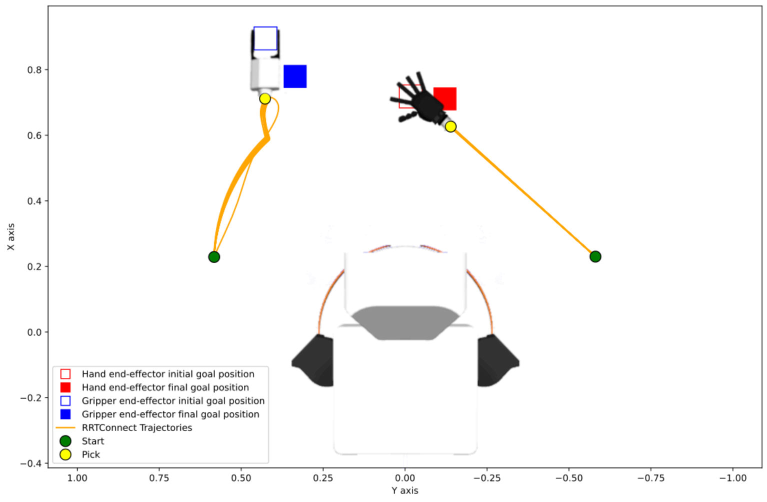
Position 3

	APF+RRTConnect	RRTConnect
SR APF Left	20/20	-
SR APF Right	20/20	-
SR RRTConnect Left	20/20	19/20
SR RRTConnect Right	20/20	20/20
SR Pick Left	19/20	0/20
SR Pick Right	17/20	0/20

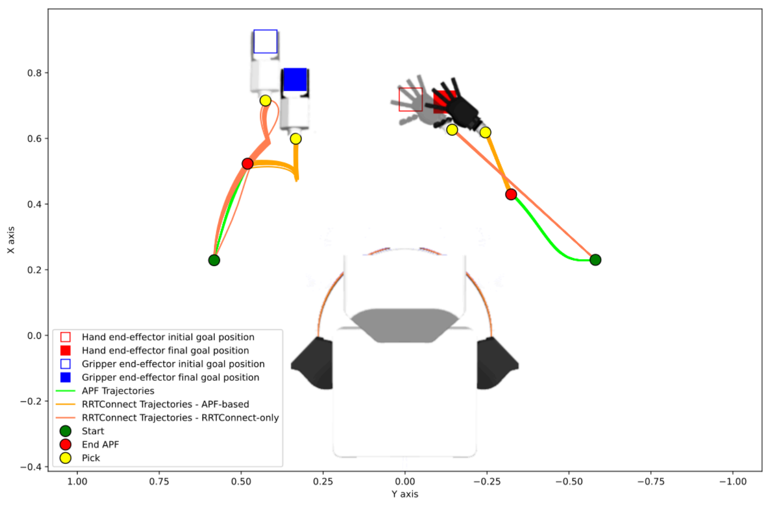
Table 5.3: Uncoordinated bimanual success rates - Position 3



(a) APF+RRTConnect behavior in 2D

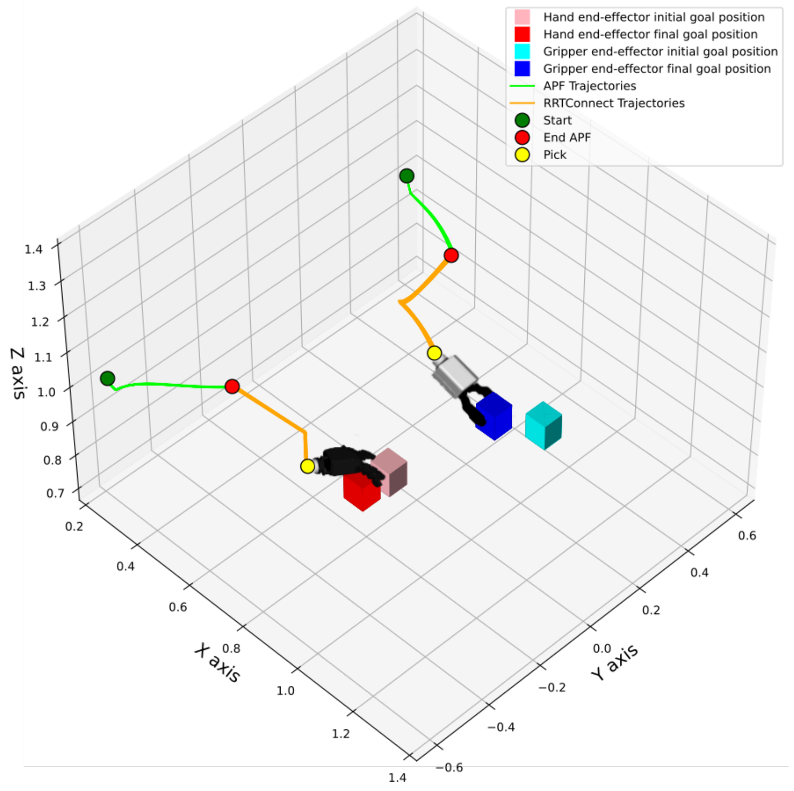


(b) RRTConnect-only behavior in 2D

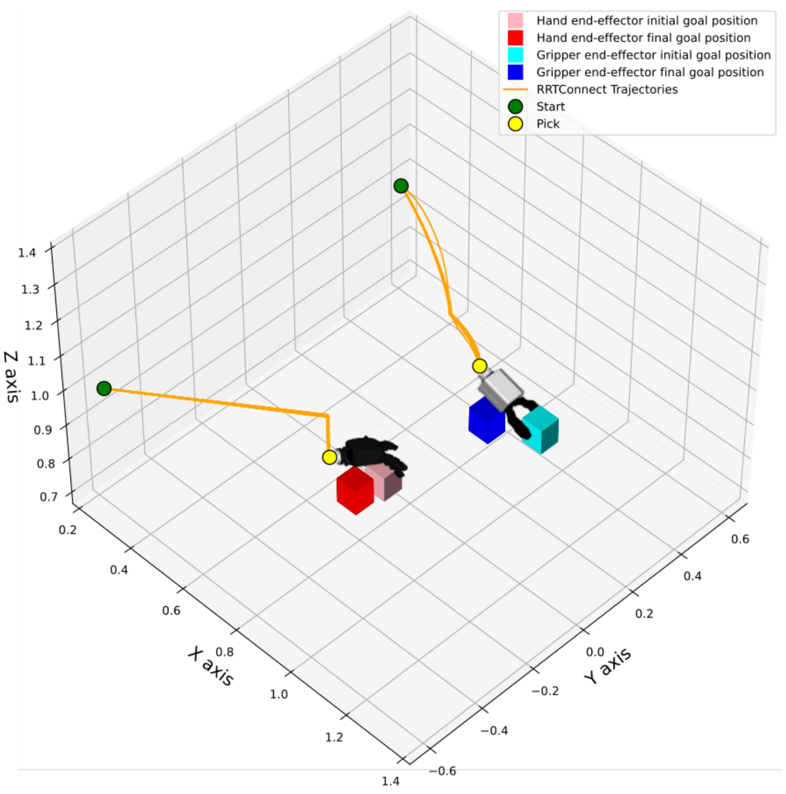


(c) Merged graphs in 2D

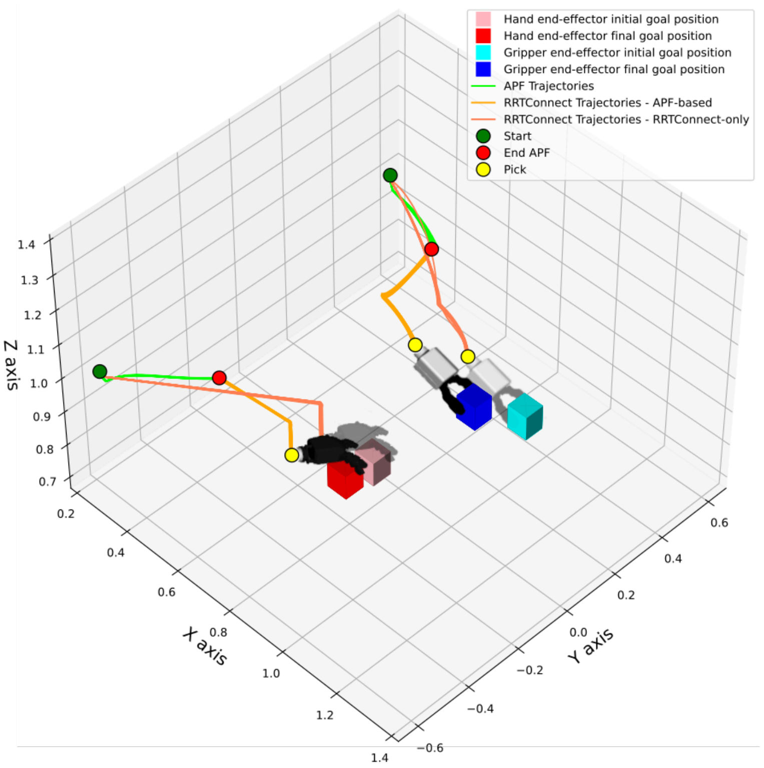
Figure 5.7: Uncoordinated bimanual - Position 3 - 2D



(a) APF+RRTConnect behavior in 3D



(b) RRTConnect-only behavior in 3D



(c) Merged graphs in 3D

Figure 5.8: Uncoordinated bimanual - Position 3 - 3D

5.4.2 Uncoordinated bimanual with crossing arms

In this experiment, an uncoordinated bimanual action is considered as before, but with the additional requirement of avoiding self-collisions. In fact, the placement of the red cube on the right side of the table and the blue cube on the left side leads the trajectories of the arms to inevitably cross each other during the APF phase. As described in Chapter 4, the self-collision avoidance system is launched every time a dual manipulation routine is requested. When the potential collision is detected (Figure 5.10a) it is checked which arm is closer to its goal. In this scenario, the right end effector is closer to its target object and therefore has priority. The left arm moves backward[2] (Figure 5.10b) and the right end effector can restart moving towards its goal. Once the right arm has grasped the object (Figure 5.10c) and has returned to its safe position, no more obstacles hinder the movement of the left end effector. Consequently, it can resume its motion towards the blue cube (Figure 5.10d).

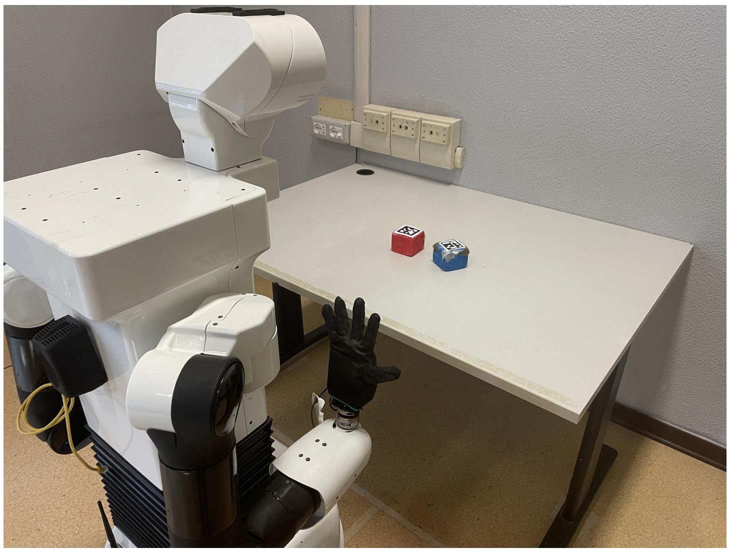
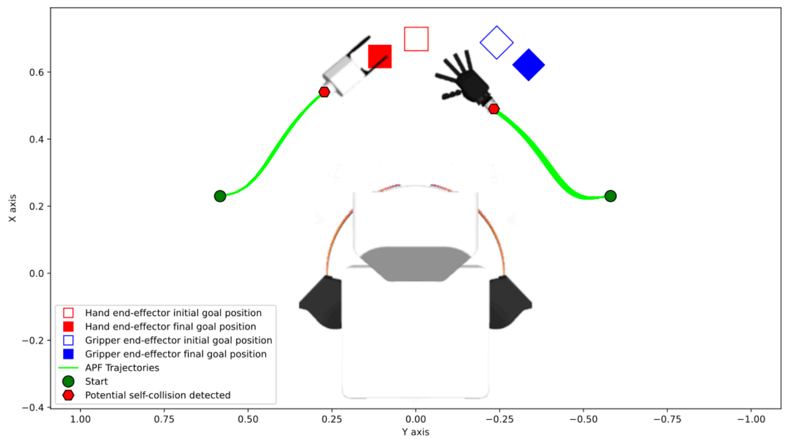
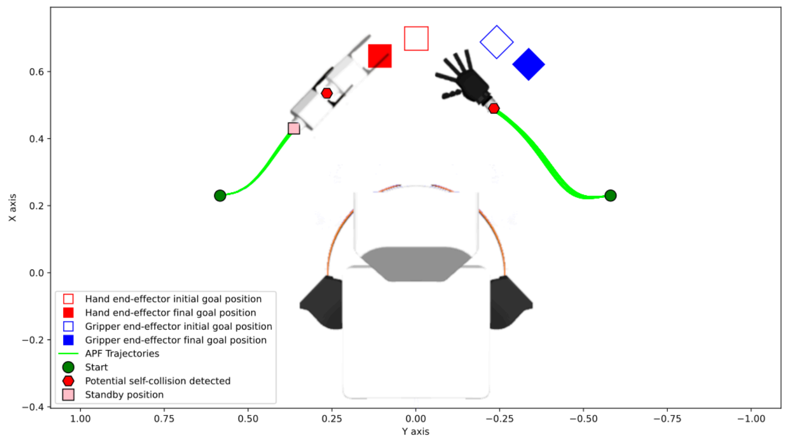


Figure 5.9: Uncoordinated bimanual with crossing arms scenario

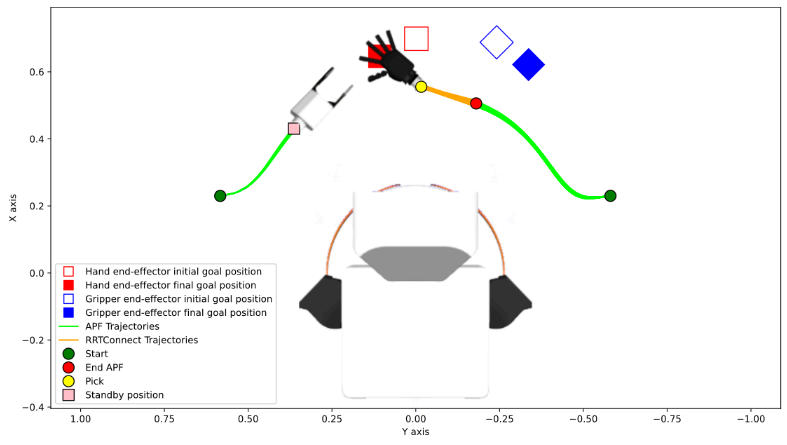
[2]In the previous position stored in the recorded movement vector



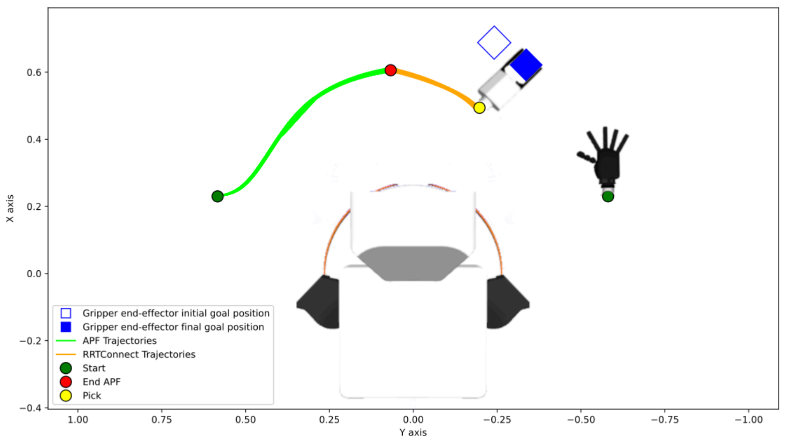
(a) Potential collision detected



(b) The left arm moves backward



(c) The right arm reaches its goal, grasps the object, and returns to a safe position



(d) The left arm picks the right object completing the routine

Figure 5.10: Self-collision avoidance behavior

	Dual APF+RRTConnect	**Sequential APF+RRTConnect**
SR APF Left	60/60	59/60
SR APF Right	60/60	60/60
SR RRTConnect Left	60/60	59/60
SR RRTConnect Right	60/60	60/60
SR Pick Left	59/60	58/60
SR Pick Right	58/60	60/60

Table 5.4: Uncoordinated bimanual with crossing arms - Success rates

Moving the arms simultaneously or sequentially the behavior is very similar (Table 5.4). The main difference lies in the total time of the manipulation routine (Table 5.5). The term "APF time" refers to the time required by an end effector to reach the APF goal, i.e. the distance from the target object where RRTConnect is triggered to perform the grasp. "Pick time" represents the time required for grasping and lifting an object. "Home time" refers to the time an end effector needs to return to the initial safe position after the pick. To calculate the total time for the dual APF+RRTConnect approach, only "Avg. APF left time", "Avg. pick left time", and "Avg. safe left time" are summed since "Avg. APF left time" includes the partial times for the right arm. As the results show, the self-collision avoidance system reduces the total time by 5.1%.

	Dual APF+RRTConnect	**Sequantial APF+RRTConnect**
Avg. APF left time	175.01s	86.92s
Avg. pick left time	31.35s	31.27s
Avg. safe left time	32.38s	33.08s
Avg. APF right time	*64.15s*	50.74s
Avg. pick right time	*18.20s*	18.22s
Avg. home right time	*26.57s*	31.40s
Avg. total time	**238.74s**	**251.63s**

Table 5.5: Uncoordinated bimanual with crossing arms - Times

5.4.3 Coordinated bimanual

In this experiment, the box (i.e., the target object for both end effectors) is moved during the arm motion. The results show that the proposed APF+RRT Connect system reacts to such position change (Figures 5.12a, 5.13a) and is therefore capable to perform the pick in most cases (53/60). In the seven failed cases, the arms were always able to pick and lift the box by grasping it at the expected grasping points, but the box slipped backward or forward. Although not flawless, the APF+RRTConnect system is certainly an improvement compared to the RRTConnect-only system. Since the RRTConnect-only approach is unable to react to the change in position of the box, it inevitably ends up grabbing it in the wrong position (Figures 5.12b, 5.13b). This results in the box sliding backward, causing the pick to fail. Moreover, the trajectories generated by the APF+RRTConnect method exhibit greater compactness, with a higher degree of similarity to each other, in contrast to the trajectories obtained with the RRTConnect-only system. Lastly, similar to the uncoordinated bimanual case, when using APF as the first step, RRTConnect never fails, unlike when using RRTConnect directly.

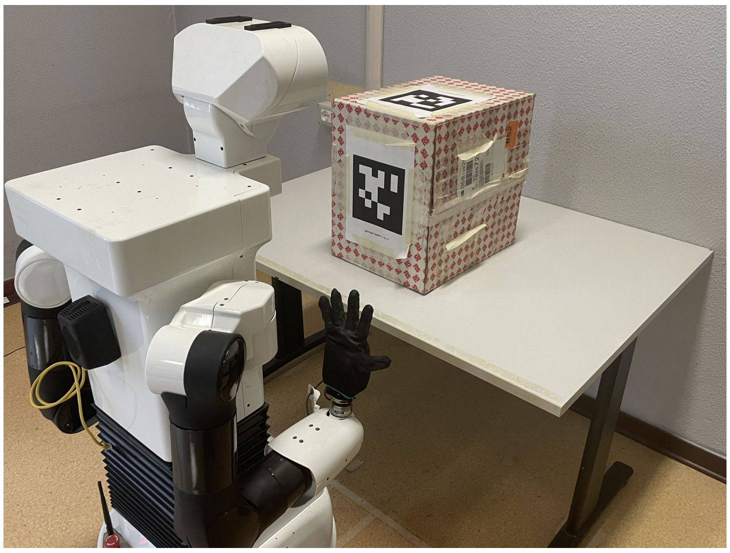


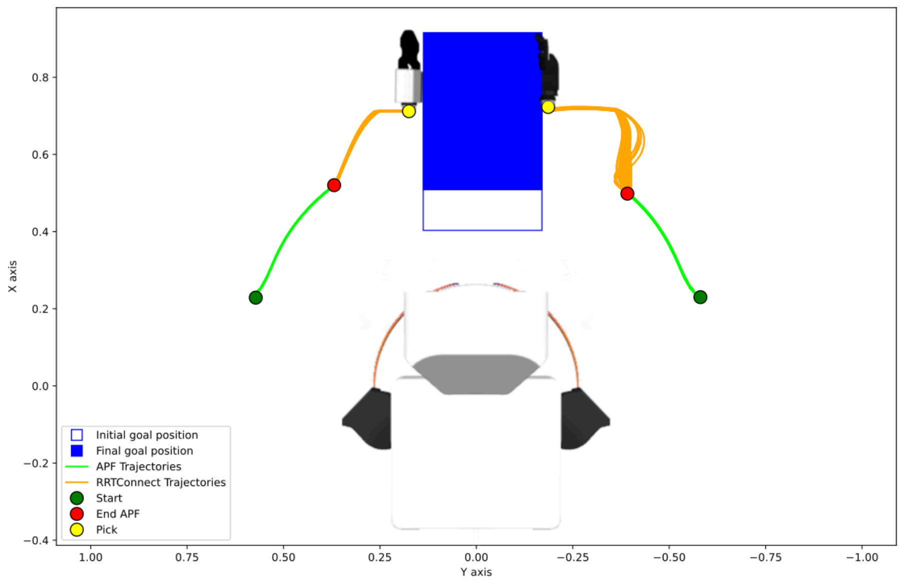
Figure 5.11: Coordinated bimanual scenario

	APF+RRTConnect	RRTConnect
SR APF Left	60/60	-
SR APF Right	60/60	-
SR RRTConnect Left	60/60	57/60
SR RRTConnect Right	60/60	57/60
SR Pick Left	53/60	0/60
SR Pick Right	53/60	0/60

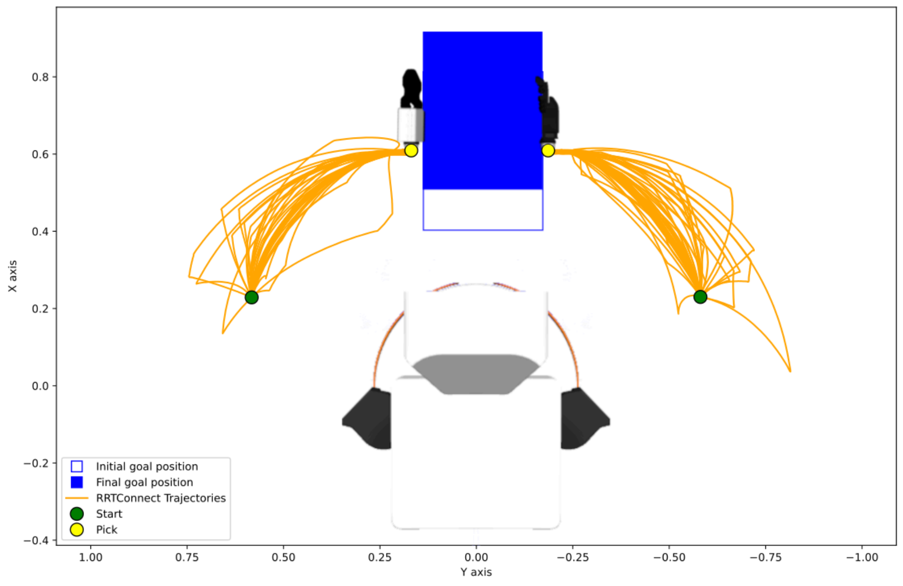
Table 5.6: Coordinated bimanual - Success rates

	Avg. Grasping offset left	Avg. Grasping offset right	SD. Grasping offset left	SD. Grasping offset right
APF+RRTConnect	0.00088	0.00090	0.06419	0.08251
RRTConnect	0.10950	0.10956	0.12623	0.12624

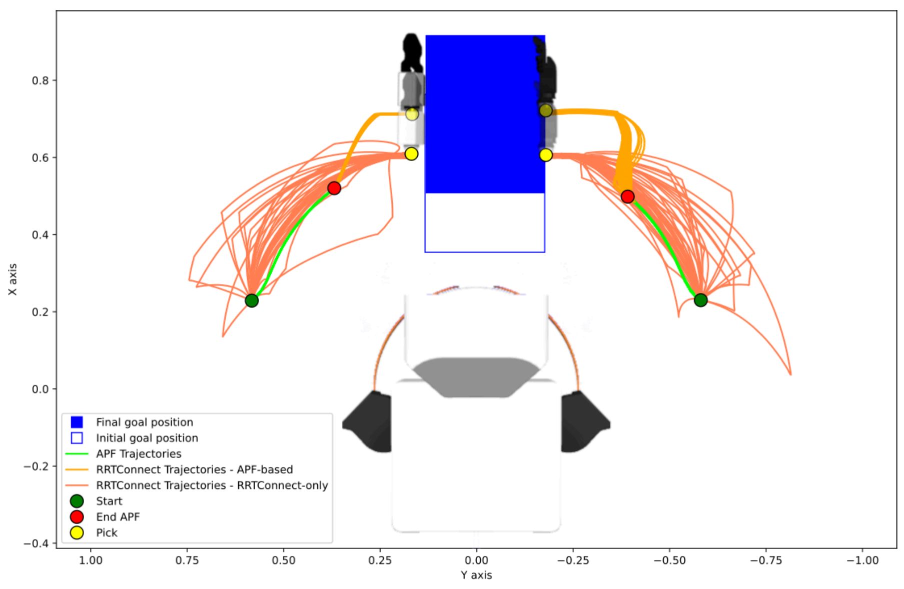
Table 5.7: Coordinated bimanual - Grasping offset



(a) APF+RRTConnect behavior

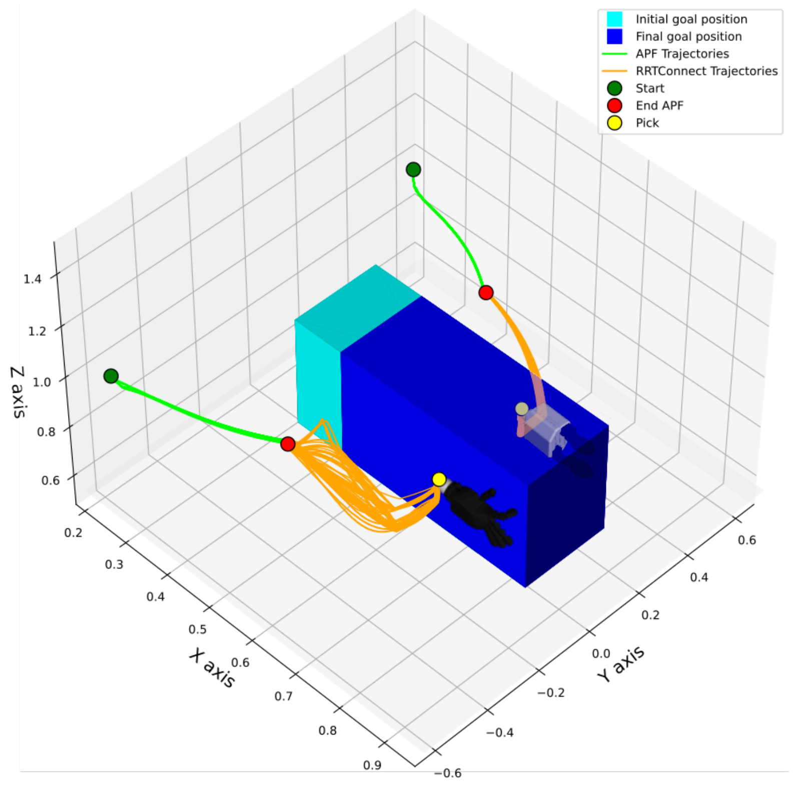


(b) RRTConnect-only behavior

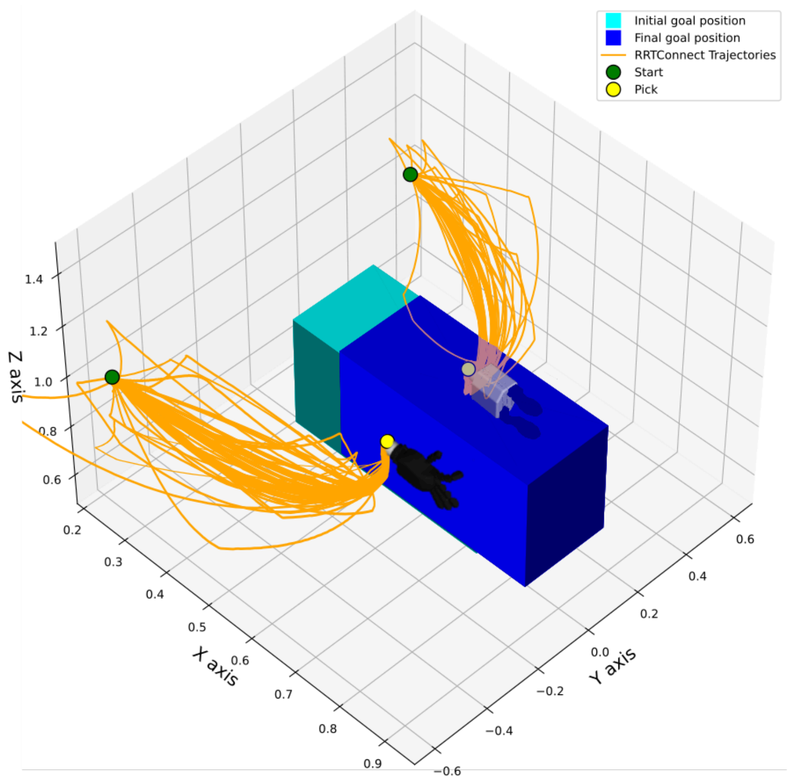


(c) Merged graphs

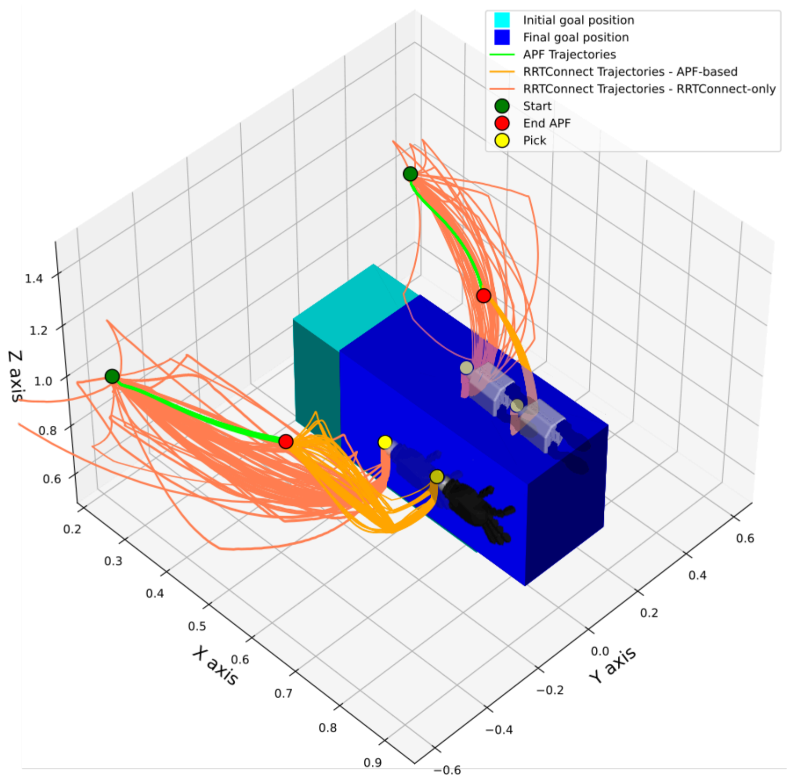
Figure 5.12: Coordinated bimanual - 2D



(a) APF+RRTConnect behavior



(b) RRTConnect-only behavior



(c) Merged graphs

Figure 5.13: Coordinated bimanual - 3D

5.5 Discussion

From the experiments performed, several considerations can be made. First of all, analyzing the APF success rate across all the experiments, it is observed that it has never failed (478/480, 99.6%). Therefore, the improved APFs demonstrate a significant level of robustness, as they consistently bring the end effectors to the requested goal position without encountering any problems. The same consideration can also be made for RRTConnect since, considering its success rate (705/720, 97.9%), it proves to be capable of generating grasping plans very often. Analyzing the pick success rates, it is noticed how the APF+RRTConnect combination allows to grasp the objects in most cases (452/480, 94.2%). On the contrary, the RRTConnect-only approach, because of its static nature, is not capable of picking objects in dynamic environments (0/240, 0%).

Considering the grasping offset metric, in the coordinated bimanual experiment, it can be seen how the RRTConnect-only approach positions the end effectors nearly 11 cm away from the correct grasping positions. Even if it manages to catch the object, because of the offset the box ends up sliding backward in most cases. If for a closed box this might not be a significant issue, in a real-world scenario the consequences could be more serious. If a robot needs to pick up and lift a box containing objects, an incorrect pick could result in the items inside the box falling out, with the risk of suffering or causing damage. On the contrary, the APF+RRTConnect system reacts to the changing position of the box and set the manipulators in the correct location. Therefore the object is raised correctly 53 times out of 60. Another advantage of the APF+RRTConnect system can be deduced by comparing Figures 5.12a and 5.12b. With the APF+RRTConnect approach, both trajectories via APF and RRTConnect tend to be "compact". Instead, when relying on the RRTConnect-only system, the generated trajectories become significantly more diverse and broad. Consequently, these movements are much more inefficient and less human-like.

The uncoordinated bimanual with crossing arms experiment highlights the utility of the self-collision avoidance system. This security method allows to exploit the two arms of the robot to reduce the total time of the picking routine. Moving the arms sequentially in fact results in a 5.1% slowdown of the routine (as well as not being a human-like movement). While this percentage may seem small, in a setup where a robot must constantly perform bimanual actions in a dynamic environment, it leads to considerable time savings.

6 Conclusions and Future Works

This thesis is born from the idea of creating a dual manipulation system capable of working in dynamic environments. This means that such a system must be able to perform both uncoordinated and coordinated bimanual tasks while effectively adapting to changes in the surrounding scenario as humans do. Several approaches, such as optimization-based motion planners or imitation learning, can be used for this purpose. However, due to drawbacks like high computational cost or the need to create a large dataset, it was decided to opt for the simpler sampling-based motion planners; in particular RRTConnect. Such motion planner allows to execute bimanual actions but, in its basic implementation, it consider the environment as static and therefore cannot be applied directly in dynamic scenarios. For this reason, the dual arm manipulation approach proposed in this thesis relies on the combination of APF and RRTConnect. APF, thanks to their simplicity and ability to react to changes in the surrounding environment, are a useful support tool for RRTConnect. In fact, instead of generating static motion plans when the user asks to pick an object, APF generates motion plans in an online manner, bringing the end effector to an intermediate position, close to the target item, while reacting to changes in the scenario. Only after the APF phase RRTConnect is triggered, in this way it have to generate a much shorter plan resulting in a lower risk of changes in the environment and in a higher probability of grasping the target object. Since with APF the end effectors might get trapped in local minima, the so-called "improved-APF" implementation has been adopted. Such improvement, described in [10], enables an easier escape from local minima through the dynamic generation of escape points around obstacles. However, they lacked the ability to react to changes in the goal position and it was necessary to develop an additional extension to address this limitation. Moreover, the APF consider the two arms as separate entities, meaning that during their movements they might collide with each other. For this reason, it has been developed a self-collision avoidance mechanism that allows the detection of risky situations over time and, consequently, coordinates the arms in a way that allows safe movements. The experiments of this thesis showed that the APF+RRTConnect approach is an effective dynamic dual manipulation system. Thanks to the APF and the self-collision avoidance method, it can safely grasp non-static objects in the workspace, in a coordinated or uncoordinated way. It also leverages the use of two arms to minimize the overall routine time required for picking two objects. For these reasons, the proposed dual manipulation system can be applied in different real-world scenarios like in logistics for sorting and packing goods in warehouses or as support assistants in industrial and medical applications. However, the current version of the APF+RRTConnect manipulation system is not without some limitations and problems. One of the issues concerns the speed during the APF phase, as you get progressively closer to the goal the attractive force decreases thus causing a reduction in the velocity of the end effectors. Moreover, the movement via APF is jerky (at least with TIAGo++) so further improvements and parameter optimizations will be needed in order to achieve a smoother behavior. To apply the proposed manipulation system in real-world scenarios would also be better to avoid using apriltags since every object that needs to be manipulated must be tagged. Instead, relying on a computer vision algorithm capable of detecting objects in the environment and estimating their position would be a better solution. Furthermore, to perform the experiments, only a few grasping points have been defined for each object. To use the APF+RRTConnect system in real-life scenarios it should be integrated with a method capable of automatically generating several grasping points for every object that must be picked, according to its shape. Finally, another upgrade regards the RRTConnect phase: if the vision system detects that the target objects changed position during the RRTConnect phase, the current motion plan must be aborted and the system should return to the APF phase. These considerations will be all taken into account to improve the proposed manipulation system in future works. This thesis can therefore be seen as a starting point for a full dynamical and bimanual manipulation system applicable in a wide variety of scenarios.

References

[1] SmartLab AI. *A brief overview of imitation learning*. Sept. 2019. URL: https://smartlabai.medium.com/a-brief-overview-of-imitation-learning-8a8a75c44a9c.

[2] Michael Beetz et al. "Robotic roommates making pancakes". In: *2011 11th IEEE-RAS International Conference on Humanoid Robots* (2011). DOI: 10.1109/humanoids.2011.6100855.

[3] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. "Semiring-based constraint satisfaction and Optimization". In: *Journal of the ACM* 44.2 (1997), pp. 201–236. DOI: 10.1145/256303.256306.

[4] Steven Byrne, Wasif Naeem, and Stuart Ferguson. "Improved APF strategies for dual-arm local motion planning". In: *Transactions of the Institute of Measurement and Control* 37.1 (2014), pp. 73–90. DOI: 10.1177/0142331214532002.

[5] Sachin Chitta et al. "Mobile Manipulation in Unstructured Environments: Perception, Planning, and Execution". In: *IEEE Robotics and Automation Magazine* 19.2 (2012), pp. 58–71. DOI: 10.1109/MRA.2012.2191995.

[6] Mohamed Elbanhawi and Milan Simic. "Sampling-based Robot Motion Planning: A Review". In: *IEEE Access* 2 (2014), pp. 56–77. DOI: 10.1109/access.2014.2302442.

[7] Bin Fang et al. "Survey of imitation learning for robotic manipulation". In: *International Journal of Intelligent Robotics and Applications* 3.4 (2019), pp. 362–369. DOI: 10.1007/s41315-019-00103-5.

[8] T. Fletcher. "The undersea mobot". In: (Jan. 1960).

[9] R.C. Goertz. "Fundamentals of general-purpose remote manipulators". In: *Nucleonics (U.S.) Ceased publication* Vol: 10, No. 11 (Nov. 1952), pp. 36–45. URL: https://www.osti.gov/biblio/4394310.

[10] Alberto Gottardi et al. “Shared control in robot teleoperation with improved potential fields”. In: *IEEE Transactions on Human-Machine Systems* 52.3 (2022), pp. 410–422. DOI: 10.1109/thms.2022.3155716.

[11] Hiroyasu Iwata and Shigeki Sugano. “Design of human symbiotic robot TWENDY-ONE”. In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 580–586. DOI: 10.1109/ROBOT.2009.5152702.

[12] Snehal Jauhri, Jan Peters, and Georgia Chalvatzaki. “Robot learning of mobile manipulation with Reachability Behavior Priors”. In: *IEEE Robotics and Automation Letters* 7.3 (2022), pp. 8399–8406. DOI: 10.1109/lra.2022.3188109.

[13] Mrinal Kalakrishnan et al. “STOMP: Stochastic trajectory optimization for motion planning”. In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 4569–4574. DOI: 10.1109/ICRA.2011.5980280.

[14] Oussama Khatib. “Real-time obstacle avoidance for manipulators and mobile robots”. In: *Autonomous Robot Vehicles* (1986), pp. 396–404. DOI: 10.1007/978-1-4613-8997-2_29.

[15] Heecheol Kim, Yoshiyuki Ohmura, and Yasuo Kuniyoshi. “Transformer-based deep imitation learning for dual-arm robot manipulation”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2021). DOI: 10.1109/iros51168.2021.9636301.

[16] Franziska Krebs and Tamim Asfour. “A bimanual manipulation taxonomy”. In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 11031–11038. DOI: 10.1109/lra.2022.3196158.

[17] Vikash Kumar et al. *Learning Dexterous Manipulation Policies from Experience and Imitation*. 2016. arXiv: 1611.05095 [cs.LG].

[18] Steven Michael LaValle. *Planning algorithms*. Cambridge University Press, 2014.

[19] Xin Lin, Zhan-Qing Wang, and Xu-Yang Chen. “Path planning with improved artificial potential field method based on decision tree”. In: *2020 27th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS)* (2020). DOI: 10.23919/icins43215.2020.9134006.

[20] Honghai Liu and Jian Dai. “An approach to carton-folding trajectory planning using dual robotic fingers”. In: *Robotics and Autonomous Systems* 42.1 (2003), pp. 47–63. DOI: 10.1016/s0921-8890(02)00312-3.

[21] Jeremy Maitin-Shepard et al. "Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding". In: *2010 IEEE International Conference on Robotics and Automation*. 2010, pp. 2308–2315. DOI: 10.1109/ROBOT.2010.5509439.

[22] Robin Murphy. *Disaster robotics*. The MIT Press, 2014.

[23] Hiroyuki Nakai et al. "Development of Dual-Arm Robot with Multi-Fingered Hands". In: *ROMAN 2006 - The 15th IEEE International Symposium on Robot and Human Interactive Communication*. 2006, pp. 208–213. DOI: 10.1109/ROMAN.2006.314419.

[24] Dmytro Pavlichenko et al. "Autonomous dual-arm manipulation of familiar objects". In: *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)* (2018). DOI: 10.1109/humanoids.2018.8624922.

[25] James L. Peterson. "Petri Nets". In: *ACM Comput. Surv.* 9.3 (Sept. 1977), pp. 223–252. ISSN: 0360-0300. DOI: 10.1145/356698.356702. URL: https://doi.org/10.1145/356698.356702.

[26] Ixchel G. Ramirez-Alpizar, Kensuke Harada, and Eiichi Yoshida. "Motion planning for dual-arm assembly of ring-shaped elastic objects". In: *2014 IEEE-RAS International Conference on Humanoid Robots* (2014). DOI: 10.1109/humanoids.2014.7041423.

[27] Nathan Ratliff et al. "Chomp: Gradient optimization techniques for efficient motion planning". In: *2009 IEEE International Conference on Robotics and Automation* (2009). DOI: 10.1109/robot.2009.5152817.

[28] *Robot operating system Story*. URL: https://en.wikipedia.org/wiki/Robot_Operating_System.

[29] *Robot operating system Wiki*. URL: http://wiki.ros.org/.

[30] Carlos Rodríguez, Andrés Montaño, and Raúl Suárez. "Planning manipulation movements of a dual-arm system considering obstacle removing". In: *Robotics and Autonomous Systems* 62.12 (2014), pp. 1816–1826. DOI: 10.1016/j.robot.2014.07.003.

[31] Gildardo Sánchez and Jean-Claude Latombe. "A single-query bi-directional probabilistic roadmap planner with lazy collision checking". In: *Springer Tracts in Advanced Robotics* (2003), pp. 403–417. DOI: 10.1007/3-540-36460-9_27.

[32] Delia Sepulveda et al. “Robotic Aubergine harvesting using dual-arm manipulation”. In: *IEEE Access* 8 (2020), pp. 121889–121904. DOI: `10.1109/access.2020.3006919`.

[33] Bruno Siciliano and Oussama Khatib. “Mobility and manipulation”. In: *Springer Handbook of Robotics*. Springer, 2016.

[34] Christian Smith et al. “Dual arm manipulationa survey”. In: *Robotics and Autonomous Systems* 60.10 (2012), pp. 1340–1353. DOI: `10.1016/j.robot.2012.07.005`.

[35] Nikolaus Vahrenkamp et al. “Integrated grasp and motion planning”. In: *2010 IEEE International Conference on Robotics and Automation* (2010). DOI: `10.1109/robot.2010.5509377`.

[36] Y. Yamada, S. Nagamatsu, and Y. Sato. “Development of multi-arm robots for automobile assembly”. In: *Proceedings of 1995 IEEE International Conference on Robotics and Automation*. Vol. 3. 1995, 2224–2229 vol.3. DOI: `10.1109/ROBOT.1995.525592`.

[37] Guang-Zhong Yang et al. “Medical Roboticsregulatory, ethical, and legal considerations for increasing levels of autonomy”. In: *Science Robotics* 2.4 (2017). DOI: `10.1126/scirobotics.aam8638`.

[38] Yajue Yang, Jia Pan, and Weiwei Wan. “Survey of optimal motion planning”. In: *IET Cyber-Systems and Robotics* 1.1 (2019), pp. 13–19. DOI: `10.1049/iet-csr.2018.0003`.

[39] R. Zollner, T. Asfour, and R. Dillmann. “Programming by demonstration: Dual-arm manipulation tasks for humanoid robots”. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)* (). DOI: `10.1109/iros.2004.1389398`.

Acknowledgments

First, I would like to thank my supervisors, Prof. Emanuele Menegatti, Prof. Gloria Beraldo and Alberto Gottardi. Thanks to their support, it was possible to undertake and realize this thesis. In addition, I would like to extend thanks to the entire IAS-Lab for being a welcoming and stimulating environment.
I also want to thank my family: my mom Maria Grazia for tenacity, my dad Luciano for dedication, my sister Anna for serenity, my aunt Maria Grazia for motivation. Thanks also to my grandparents Paola, Rosalia, Francesco and Cesare for always being with me, in one way or another.
Finally, I would like to thank my friends, the ones I have known all my life, the new ones and and even those who are no longer part of my journey. Each of you has contributed to shaping who I am today.

Innanzitutto, vorrei ringraziare i miei supervisori, Prof. Emanuele Menegatti, Prof. Gloria Beraldo e Alberto Gottardi. Grazie al loro supporto è stato possibile intraprendere e realizzare questa tesi. Inoltre, vorrei estendere il ringraziamento a tutto lo IAS-Lab per essere stato un ambiente accogliente e stimolante.
Desidero ringraziare anche la mia famiglia: mia mamma Maria Grazia per la tenacia, mio papà Luciano per la dedizione, mia sorella Anna per la serenità, mia zia Maria Grazia per la motivazione. Grazie anche ai miei nonni Paola, Rosalia, Francesco e Cesare per essere sempre stati con me, in un modo o nell'altro.
Infine, vorrei ringraziare gli amici, quelli che conosco da sempre, quelli nuovi e anche quelli persi. Ognuno di voi ha contribuito a formare la persona che sono oggi.

Padova, Luglio 2023

Alberto Paiusco