UNIVERSITÀ DI GENOVA

MASTER'S DEGREE IN ROBOTICS ENGINEERING

A Behavior-Tree Framework for Object Searching, Approaching and Grasping with a Quadruped Manipulator

Academic Year 2021 / 2022

Candidate:	Riccardo Parosi	
Supervisors:	Carmine Recchiuto	Università di Genova
	Mattia Risiglione	Dynamic Legged Systems Lab (IIT)
	Victor Barasuol	Dynamic Legged Systems Lab (IIT)
	Claudio Semini	Dynamic Legged Systems Lab (IIT)

Acknowledgements

I would like to thank the Professors of Università di Genova and Ècole Centrale de Nantes, for their lectures and for their advices. I would like to thank my UniGe supervisor Carmine Recchiuto for his availability. I would like to thank my IIT supervisors especially Mattia Risiglione for his help and tips, Victor Barasuol, Claudio Semini and all the people that I met during this journey. Finally, I would like to thank my parents and all my friends for their support.

Abstract

Legged robots represent a versatile platform due to their ability to locomote over rough terrains thanks to their legs. A wide variety of applications can be performed, such as search and rescue operations, logistics, and maintenance. Recently, robust locomotion strategies and availability of reliable quadruped platforms has pushed the legged locomotion community to add manipulation capabilities to these systems by equipping them with a manipulator arm. Additionally, in order to make this type of platform, commonly referred as legged manipulator, fully autonomous, vision is required. In this context, this thesis proposes a control pipeline that uses images and point clouds from an Eye-in-hand camera to identify, approach to and grasp a target object using a quadruped manipulator. First, the camera, mounted on the arm's end-effector, scans the environment and seen objects are classified and localized in the camera frame by a neural network (NN). Once the target object is found, visual feedback is used to center the object in the camera and approach to it. During this phase, the arm control is split into two part: the joints controlling the links with lower mass are used for keeping the object centered in the camera, while the links controlling higher mass are used to place the arm aligned with the object. Finally, an off-the-shelf software package determines the grasping pose, and the arm is controlled through visual and proprioceptive information to grab it. A Behavior-Tree framework dictates the sequence of actions and allows to modularly encode additional tasks. The approach is validated through a set of simulations using the HyQ quadruped platform equipped with a 7-DoF manipulator arm.

Keywords: Behavior-Tree, Visual Servoing, Loco-manipulation, Quadruped robots

Notations

Visual Servoing

$\boldsymbol{e}$ error

$\boldsymbol{s}$ current value of the features

$\boldsymbol{s}^*$ desired value of the features

$\boldsymbol{v_c}$ velocity of the camera expressed in the frame of the camera

$\boldsymbol{L_s}$ Iteraction matrix related to $\boldsymbol{s}$

$\widehat{\boldsymbol{L_s}}$ estimated Iteraction matrix related to $\boldsymbol{s}$

Abbreviations

BT Behavior Tree

CoM Center of Mass

CV Computer Vision

DLS Dynamic Legged Systems

DoF Degrees of Freedom

FSM Finite State Machine

GRFs Ground Reaction Forces

HSV Hue, Saturation, Value

IBVS Image-based visual servoing

IIT Istituto Italiano di Tecnologia

IoU Intersection over Union

mAP mean Average Precision

MPC Model Predictive Control

RGB Red, Green, Blue

RCF Reactive Control Framework

RGBD Red, Green, Blue, Depth

RL Reinforcement Learning

ROS Robot Operating System

SAG Search, Approach and Grasp

TO Trajectory Optimization

w.r.t. with respect to

WBC Whole-Body Controller

List of Figures

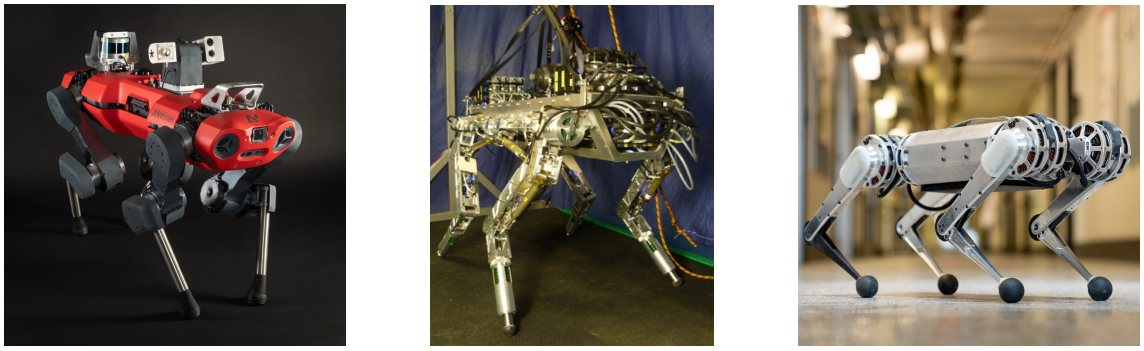
Contents

Chapter 1

Introduction

1.1 Motivation

One of the big purposes of robots is to be deployed in environments which are dangerous for human operators. An example of that can be an area after a disaster both natural (e.g. earthquakes) and human-made (e.g. Fukushima explosion). Another example is the exploration of new planets which present inhospitable conditions.



(a) ANYmal C [5]. (b) IIT's HyQ [6]. (c) Mini Cheetah [7].

(d) HyQReal [8] (e) Spot [9].

Figure 1.1: Examples of commercial and research quadrupedal robots.

Most of these scenarios present an uneven ground formed by rubble and boulders, on which it is hard to move with a wheeled or biped robot. Indeed quadruped robots are more suited to this type of context, because they have more mobility than a robot with wheels and a definitely bigger support area than two-legged ones. In the last decades several quadrupeds robot have been developed. Examples of them are: MIT's Mini Cheetah [7], Anybotics's ANYmal C [5], IIT's HyQ [6], IIT's HyQReal [8], Boston Dynamics's Spot [9]. A major drawback of this type of robots is the lack of proper manipulation capabilities, which affect deeply the range of feasible tasks. In previous years researchers have tried different solutions in order to deal with this problem. There are quadrupeds which use part of their legs for manipulation [10], there are others that have special grippers attached to their legs [11]. However these solutions perturb significantly the stability of the robot and, more importantly, do not allow to perform manipulation and locomotion at the same time. The system which solves these problems is a quadruped with an additional arm mounted on the top of it (generally in front) and it is named as quadruped manipulator. We report in Fig. 1.2 examples of this type of platform in literature.

(a) ANYmal B with a Kinova robotic arm. [12].

(b) HyQReal with INAIL-IIT robotic arm [13].

Figure 1.2: Examples of quadruped manipulators.

To fully enlarge the number of tasks legged manipulators can execute, vision is required. The use of vision has shown successful results for locomotion [14], allowing the robot to select safe footholds and to increase the traversability of difficult terrains. When it comes to quadruped robots with arm, many of the manipulation tasks (e.g opening doors, pick and place of objects) has been done assuming to know the environment. Hence, to enhance the autonomy of legged manipulators, vision need

to be taken into account.

1.2 Contributions

To fill the gap, this thesis proposes and verifies a control strategy using vision to search, approach and grasp (SAG) an object. The different operations are managed by a behavior tree, that selects the action the robot has to take and to produce the high level commands to the robot. In more details, references to control a quadruped manipulator are generated, to locomote the base and to center the object through visual feedback. It is important to mention that for the visual control only an Eye-in-Hand RGBD camera placed on the gripper of a robotic arm is used.

1.3 Outline

This document is organised in the following way: Chapter 2 describes the related works and introduces the reader to the problem. Chapter 3 explains the tackled problem and gives to the reader the mathematical background used. Chapter 4 illustrates the metholodogy and describes the developed control approach. In Chapter 5, simulation results validates the approach and implementation details are provided. Finally, Chapter 6 ends the manuscripts with a summary of the work, considerations and suggestions about future improvements of this work.

Chapter 2

Related works

In order to build the framework, many components which come from different robotics fields are required. One of them is the control, used to generates command signals to track the desired motions of the robot. Another necessary component for this project is an object detection strategy, since most of the robot behaviors are dependent to the target object. In order to achieve the final goal of the SAG problem, there is the necessity to understand the grasping position and orientation of the object with respect to the robot. Lastly, all the modules derived from these fields need to be managed and synchronized by an higher component. Regarding the latter, a Behavior Tree is used.

2.1 Control of Quadruped Manipulators

This new type of robots has brought new challenges. One example is the robustness against the internal and external disturbances both for the two systems. Indeed, the walking or trotting of the quadruped can perturb the task of the manipulator and vice versa. A specific case, regarding a small quadruped, is a possible lack of balance when the arm is performing manipulation tasks or is carrying a heavy load. This is caused by the fact that the Center of Mass (CoM) of the quadruped can be shifted significantly. Generally, for multi-degree of freedom (DoF) floating-base systems, such as quadrupeds, optimization-based techniques are prominently adopted for whole-body control (WBC) design [15], [16], [17]. Such control schemes optimize control objectives for multiple tasks, while handling physical constraints, such as actuation limits, friction cone constraints, etc. Generally, these optimization problems generates accelerations and contact forces, which are used lately through inverse dynamics in order to compute the torques of the joints. Some of the approaches tend

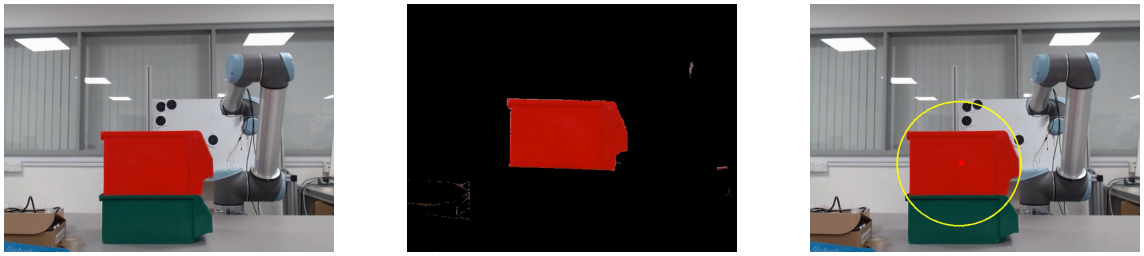
to consider the robotic manipulator as a disturbance and model the controller to stabilize the base [18]. Other approaches, as in [12], generates together the motion for base and manipulator respecting the zero-moment point constraints. Other approaches use Reinforcement Learning (RL) policies for locomotion and Model Predictive Control (MPC) for controlling the arm and predicting, based on future arm motion, the disturbance on base [19]. Some other approaches use whole-body planners to generate a reference for the system. One example of that is [20], in which a whole-body MPC planner is used to avoid self and environment collisions which may happen during a locomanipulation task.

2.2 Object Detection

Object detection is a family of algorithms which are able to recognize an object in an image. Since in this thesis such type of algorithm is used to detect a specific object in the environment in real-time, two characteristics are needed: real-time detection speed and a high accuracy of the result. Two categories of algorithms which satisfy the two above requirements are Conventional Computer Vision and Deep Learning for Computer Vision.

Starting from the first category, a very important algorithm is Color-based Object Detection. It is a good choice if the scenarios are simple which means if the color of the background is very different from the color of the target object. That because that algorithm extracts from an image only the regions which have a high amount of a precise chosen color. Therefore all the background is removed and the output is an image with only the target object. After that, in order to avoid noise interference, is extracted the largest pixel block in the connected area. As last step, is computed the position of the centroid of the pixel block which is considered the position of the center of the target object. An important consideration on this method is that the great part of the cameras are RGB (Red, Green, Blue) and it is very sensitive to illumination. To solve this problem is possible to transform into HSV (Hue, Saturation, Value) the color space of the image.

To be more precise, Hue contains the color information, Saturation contains the intensity of the color and Value contains the brightness of the color. So the color of the target object is known, the Hue component is the one to take into account. HSV is also coherent with the form in which humans perceive colors. Another interesting property of doing HSV-based object detection is that it is able to adapt to the change



(a) Initial image generated by an RGB sensor

(b) Red color segmentation with HSV.

(c) Output of the algorithm

Figure 2.1: [1] Example of Conventional CV algorithm: Color-based Object Detection.

of color which is provoked by different illuminations. An example of that algorithm can be observed in Fig.(2.1).

The second category is Deep Learning for Computer vision. It is a neural-based approach to object detection. It is a good choice if the scenarios are complex, which means that there are a lot of colors in the background, and also if the target object has no color characteristics (note that in both cases the color-based object detection doesn't work but it is significantly less computationally expensive). In particular an efficient object detection algorithm of this category is called YOLOv3 [2]. It is based on neural network and it has two features. The first is the use of bounding boxes on the image to determine the positions of the detected objects. The second feature is the classification of each bounding box, in order to know which type of object is inside them. An example of these two features can be seen in Fig.(2.2).

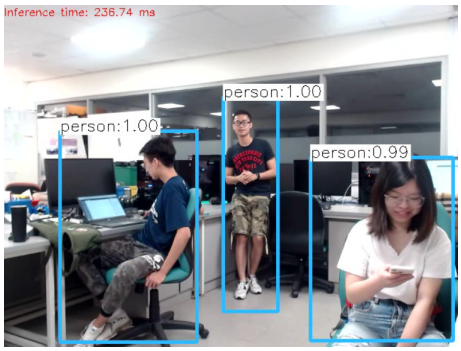


Figure 2.2: [1] Example of bounding boxes and class predictions computed by YOLOv3.

To be more precise the input is an image and the output is a tensor which contains the class predictions and the bounding boxes. The latter is composed by five elements (x, y, w, h, c), where x, y are the coordinates of the center of the box, w and h are respectively width and height of the box and c is the confidence that the network

has in that box. YOLOv3 has a network structure based on DarkNet-53, which is formed by 53 convolutional layers. In particular, YOLO has 106 layers in total, 53 are used for feature extraction and the other 53 for detection. YOLOv3 has more precision and is faster than other similar models, this can be seen in Fig.(2.3).

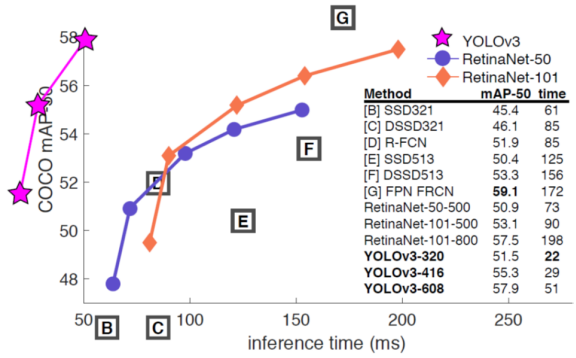


Figure 2.3: [2] Deep Learning CV algorithms comparison, using mAP (mean Average Precision) at 0.5 IoU (Intersection over Union).

Now, is time to describe the other algorithm which is in the title of this section, the Object Tracking algorithm. It allows to locate, in videos (so over multiple frames), the same chosen objects. It can be seen as the step after object detection, because when a target object is detected it continuous to track it. But when there are more than one object in the same image is difficult to understand which is the target one. To solve this problem is possible to compute the Euclidean distance between the target object of the previous frame and every object of the new frame, after that the minimum Euclidean distance determines the target object, as is explained in [1].

2.3 Search and Approach of Objects in Mobile Manipulators

The goal of mobile manipulation is to allow a robotic arm to perform tasks on objects which, initially, are not in its workspace and/or in its field of view. That definition includes all the manipulators which are able to move in the environment, this includes wheel bases and legged bases. In the literature it is possible to find a large quantity of papers which describe the problem of mobile manipulation. As we already explained, some approaches use a whole-body controller such as [21], [12]. Instead others deal with that kind of robot as two different systems, such as [22], [23]. Recently, with the growth of reinforcement learning, some policies to achieve this behavior have been

developed, examples are [24] and [25]. However most of them focus their contribution on the manipulation, using for the navigation a trajectory planner. This brings to a lack of variety and novelty in the methods of searching and approaching the object. One research which addresses that problem from the detection of the object to the grasping is [22]. In particular they use a Toyota Human Support Robot (HSR) [26], which is a wheeled robot with a 4 DoF manipulator and it has one RGBD camera in the head (which is actuated by a 2 DoF mechanism) and one RGB camera in the parallel gripper. They implemented the search phase moving the head camera through a series of search poses, but they made the assumption that the target object is in front of the camera. Once the object is detected they build a map of all the detected objects and they use visual servoing with the camera on the gripper to reach a desired pose relative to the target object. They implemented a learning strategy to estimate correctly the depth from the camera to the object, since their Eye-in-Hand camera is RGB and does not present a stereo-vision module. After that their grasp phase uses visual servoing to center the object which is approximated to a cylinder to perform the grasping. The assumptions in that work are quite strong, since they did not implement a real search behavior and the grasping part is very simplified also because they always point down the end-effector camera, so that approach will work only for objects which are on the ground. Another interesting paper is [21]. Their method uses a whole-body trajectory optimization which is able to create trajectories for a locomanipulation task. Their search phase strategy is to move the base slowly forward while the Eye-in-Hand camera is scanning the area in the front from right to left and vice versa. This implies that also in this case they made the assumption of the target object in front of the robot. Instead, in the approach phase their TO is able to track and follow the object firstly with the arm and after slowly with the base. This allows the base to don't perform aggressive maneuvers. This is used also for the grasping phase, and it is done adopting the 3-D position of the target object as equality constraint. In addition, the just presented papers, use a simple finite state machine (FSM) to change between states.

2.4 Object grasping

The grasping of objects is still an open problem in research and is becoming increasingly important. That can be demonstrated by the number of publications on IEEExplore which include the keywork "6DoF" in the document and "Grasping" in

the metadata, Fig.(2.4).

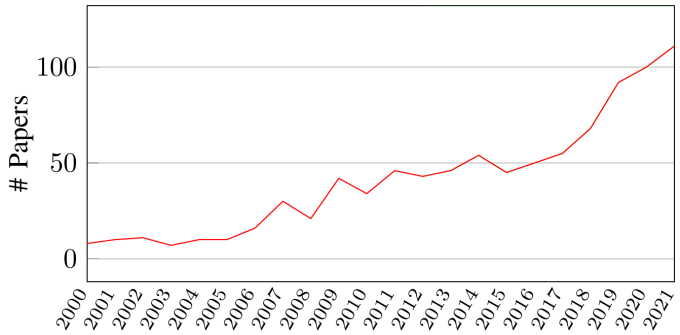


Figure 2.4: [3] Number of publications on IEEExplore including "6DoF" in the document and "Grasping" in the metadata.

Starting from the beginning, a "grasp pose" defines the position and orientation of the end-effector in order to achieve a successful grasp. The procedure of finding that pose is called grasp synthesis and there are mainly two types: 4DoF and 6DoF. The first class of methods is characterized by having the desired pose of the gripper composed by a 3-D position and only one orientation which is around the axis perpendicular to the object. So usually it used with the camera perpendicular to a table and the end-effector simply moves up and down. That is the reason why such approaches are also referred as Top-Down. Obviously, is easy to understand that it is a strong assumption and it is cannot be applied in many real cases. For that reason researchers invented the 6DoF methods, which as can be imagined, have as output both position and orientation in the three dimensional space. For both this two types, it is possible to make a further categorisation regarding the gripper which is in use. In fact, some implementations are for dexterous hands, like human hands with five fingers, and the others are for parallel hands, two fingers grippers. Talking about techniques for finding these grasping poses, there are two big categories: analytical approaches and data-driven approaches. The first category has been developed until the beginning of the 21st century and the goal was to model and estimate physical conditions of the grasping, in [27] the reader can find a survey about them. However these methods were very complex and not applicable in real-time. For that reason researchers started to develop the second class of methods which use features extracted from a group of known object to find the right grasp pose for them but also for unknown ones. At the beginning the features where hand-designed, for example searching which part of the object fits in the gripper, more information on that are available in [28]. From the second decade of the century and to the present, the growth of

machine learning and neural networks has influenced also this field, allowing the researchers to use learned features to detected the correct grasp. The number of these approaches is very large and for that reason it is possible to divide them into four main methodologies. The first type of algorithms is grasp pose sampling, they generate one or more grasp samples and after they evaluate each of them individually using a learned quality function. Another approach is to perform a direct regression considering all the possible grasp globally and learning a function to predict only good ones. After that there are the reinforcement learning (RL) methods, which are based on the maximization of a cumulative reward function which depends on the actions of the robot. Last but not the least there is the category of examplar methods which generate grasps from an existing database using a similarity metric. In addition, depending on the precise implementation of the method, which could be one of the presented four, the input can be different and usually is one of the following four: RGB image, RGB image, PointCloud, Voxel Grid. For the sake of this thesis there is no reason to go deeper on that topic, however for gaining more information that survey [3] can be used.

2.5 Behavior Trees

A behavior tree (BT) is a hierarchical node structure which controls the switching between the finite states of a system. BTs were born in the gaming industry as an evolution of Finite State Machine (FSM) in order to solve their drawbacks. In particular, it is difficult to determine who was the real inventor of BTs since many key aspects were shared in conferences and workshops, but it is possible to underline some milestones by Andrew Stern and Michael Mateas [29] and Damian Isla [30]. After some years, the first journal paper on that topic was published [31] and subsequently also the ones related to robotics [32], [33]. The key word for BTs is modularity, which is the main problem of FSMs. In particular the latters encode the transitions inside the state and that means each state has to know about the existence and capabilities of the other states. Instead, a BT node has just to know if it succeeded or not; each node in the tree can return three values: success, running, failure. In conclusion FSMs can bring to complexity and difficulty in extending, adapting and reusing the states. While, with BTs is possible to reuse a tree by inserting it as a sub-tree in another tree, their graphical representation is easy to read and understand, and finally there are many types of nodes which can create a more complex control flow.

The functioning of BTs is very simple, starting from the root node the graph is traversed (tick) until a leaf node, the one which really contains a control action. Depending on the result of that node, the tree can be traversed more or it may ends returning back to the root and restarting again the procedure. If the leaf node return running, that node is continuously tick until it returns one of the other two states. An example of BT is shown in Fig.(2.5).

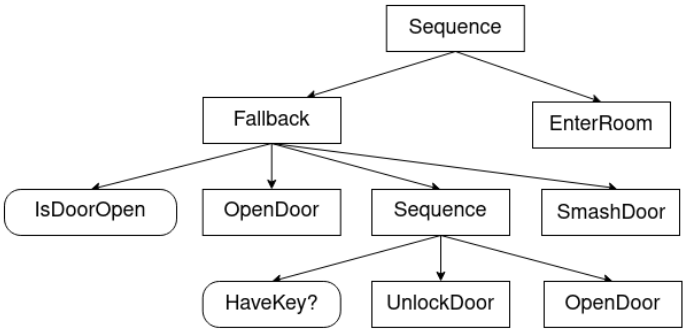


Figure 2.5: Example of Behavior Tree [4].

There are three main types of nodes: Composite, Decorator and Leaf. Composite nodes are the only ones which can have more than one child node. The ones of the second type can have only single child and can alter its result or tick it more than one time. The last category is straightforward: it contains nodes which do not have a child. In the following lines, the most important nodes will be explained. Starting from the Composite ones, the most used is Sequence. It is also the simplest one and it will tick all the children from the left to the right until the last returns success or until one of them returns failure. That node can also be seen as a logical AND. The other very important component of that category is the Fallback and it is also known as Selector. That second name is very explanatory, in fact it ticks the children from right to left until one returns success, so if the first child is ticked and it returns success, the Fallback exits returning success. This can also be seen as a logical OR. The next category, Decorator nodes, is the biggest one, because it contains all the nodes which can modify the behaviour of its child. One of them is the Inverter node, it reverts the result of the child, so if the latter returns success the Inverter return failure, and the other way round. Other nodes are Force Success, Force Failure, Repeat, Retry, Keep Running Until Failure. The name of all these nodes already explain their behavior and there is no need to explicitly describe them. Last but not least, the category of Leaf nodes. They can be only of two types: Action and Condition. The main difference is that Conditions cannot return running, while

Actions can. That is done because the Conditions have to be instantaneous, like "isDoorOpen" in the example, while Actions are meant as real interaction of the agent with the environment, which in the large part they need some time to complete, like "OpenDoor" in the example. Before concluding, it is important to say that BTs have another strong advantage. It is possible to modify their structure also online, because it is written in a textual representation, like XML, which does not need to be compiled.

Chapter 3

Problem Formulation

From the previous chapter it is depicted a lack in the state of the art regarding a complete pipeline which allows a quadruped manipulator to search, approach and grasp (SAG) a general object in the environment. To accomplish this task, in literature many assumptions have been generally used: the position of the object is known a priori [34], it does not move and stays in front of the camera [22], and it has a basic shape [21]. However, in [21], the authors implements something closer to a SAG behavior, but without explaining their search phase and using only color segmentation and basic shape object (a red ball). This thesis has the objective to fill that gap, implementing a complete SAG pipeline which can be generalised to any object, using trained object detection and grasping pose generator neural networks. In order to achieve that, there is the need to define the problem in a proper way including also the principals on which the solution is based. In particular first we define the dynamic model of such type of robot, which will be used to formulate the torque controller. After there is an explanation about visual servoing which is used to generate a control strategy based on images. At the end of this chapter our solution to the problem is described briefly.

3.1 Dynamic Modelling of a Quadruped Manipulator

Here, the dynamic model of the quadruped manipulator is split into the model of the quadruped itself and the model of the arm. First, the former is presented. A quadruped robot is a particular case of legged robot with four legs. Indeed we can consider it as a free-floating base, which can be referred as B, with a set of limbs attached to it. In order to describe the motion of the system (base B plus four legs)

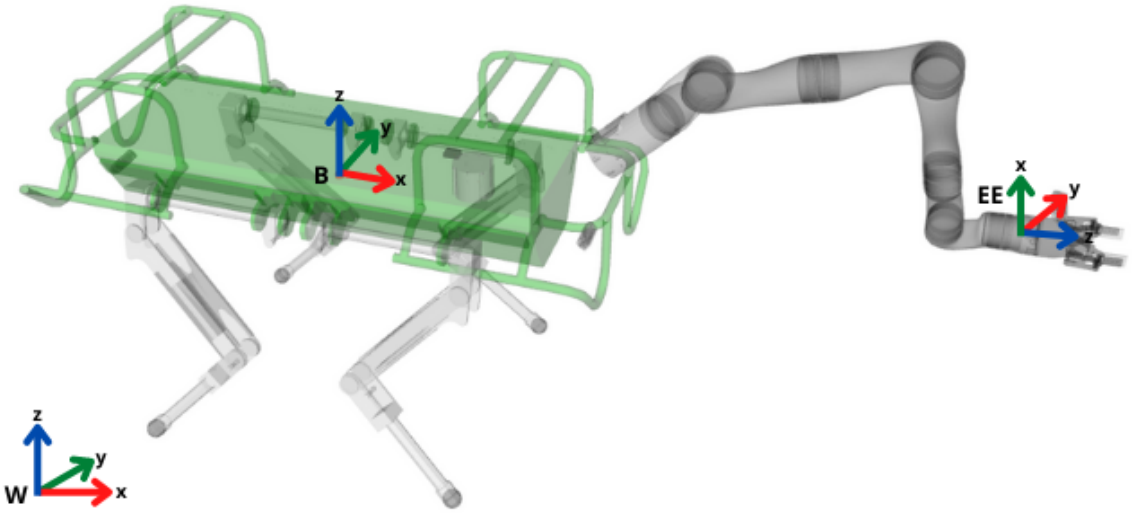


Figure 3.1: Quadruped manipulator model with the inertial frame W, base frame B and end-effector frame EE.

it is important to define a fixed inertial frame, which is referenced as world W. Now is possible to define the position of B in the inertial frame w.r.t. (with respect to) W as ${}_W\boldsymbol{p}_{WB} \in \mathbb{R}^3$. Instead, the orientation of B w.r.t. W can be defined by the roll, pitch and yaw angles (in XYZ convention), denoted as $\boldsymbol{q}_{WB}$. Regarding the limbs, the total number of joints (four legs) is n_j and their positions (joint angles) are stacked in the vector of generalized coordinates defined as $\boldsymbol{q}_j \in \mathbb{R}^{n_j}$. The position of the base, its orientation and the joint legs are stacked in the vector of generalized coordinate denoted by $\boldsymbol{q}$ as follows:

$$\mathbf{q} = \begin{bmatrix} {}_W\mathbf{p}_{WB} \\ \mathbf{q}_{WB} \\ \mathbf{q}_j \end{bmatrix} \in \mathbb{R}^{n_s} \tag{3.1}$$

where $n_s = 6 + n_j$ being the total number of degrees of freedom of the quadruped subsystem.

As it has been explained, the other subsystem is the manipulator arm with n_a joints. The position of all its joints is described by the generalized coordinates vector denoted by $\mathbf{q}_a \in \mathbb{R}^{n_a}$.

3.1.1 Robot Dynamics

To describe the evolution in time of our system's dynamics. In order to do that, a model of the system has to be developed using a mathematical description that takes

into account the evolution in time of the system components. That description is referred as dynamic model. In this work, the base and the arm are decoupled, the consequence is that there are two different dynamic models. Starting from the one of the quadruped. It is possible to describe the dynamic model of a floating base as follows:

$$\mathbf{M}(\mathbf{q})\ \ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{J}_{st}^{\top}\mathbf{F}_g + \boldsymbol{\tau} \tag{3.2}$$

where $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n_s \times n_s}$ is the inertia matrix, $\ddot{\mathbf{q}}$ is the generalized acceleration vector and it is composed as follows $\ddot{\mathbf{q}} = [{}_W\dot{\boldsymbol{v}}_{WB}^{\top}\ \ {}_W\dot{\boldsymbol{\omega}}_{WB}^{\top}\ \ \ddot{\mathbf{q}}_j^{\top}]^{\top}$, with ${}_W\boldsymbol{v}_{WB}$ and ${}_W\boldsymbol{\omega}_{WB}$ the linear and angular velocities of the floating base B in the world frame w.r.t. W. Instead, $\mathbf{h} \in \mathbb{R}^{n_s}$ is the vector of Coriolis, centrifugal and gravity terms, while $\dot{\mathbf{q}}$ is the generalized velocity vector and it is structured as follows $\dot{\mathbf{q}} = [{}_W\boldsymbol{v}_{WB}^{\top}\ \ {}_W\boldsymbol{\omega}_{WB}^{\top}\ \ \dot{\mathbf{q}}_j^{\top}]^{\top}$. On the other side of the equation the first $\mathbf{J}_{st}^{\top}\mathbf{F}_g \in \mathbb{R}^{n_s}$ term accounts the ground forces, in particular $\mathbf{F}_g \in \mathbb{R}^3$ via the contact points Jacobian $\mathbf{J}_{st}^{\top} \in \mathbb{R}^{n_s \times 3}$. Last, $\boldsymbol{\tau} \in \mathbb{R}^{n_s}$ is the vector of joint torques. As is written before, that formulation includes both the floating base and the joints of the legs. So it is possible to write it in a more explicit way by dividing the two parts as follow:

$$\begin{bmatrix} \mathbf{M}_b & \mathbf{M}_{bl} \\ \mathbf{M}_{lb} & \mathbf{M}_l \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\xi}}_b \\ \ddot{\mathbf{q}}_l \end{bmatrix} + \begin{bmatrix} \mathbf{h}_b \\ \mathbf{h}_l \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{st,b}^{\top} \\ \mathbf{J}_{st,l}^{\top} \end{bmatrix} \mathbf{F}_g + \begin{bmatrix} \mathbf{0}_{6\times 1} \\ \boldsymbol{\tau}_l \end{bmatrix} \tag{3.3}$$

where $\mathbf{M}_a \in \mathbb{R}^{6\times 6}$ and $\mathbf{M}_{bl} \in \mathbb{R}^{6\times j}$ are the inertia matrix of the base and the inertia coupling matrix between the base and the legs, while $\mathbf{M}_{lb} \in \mathbb{R}^{j\times 6}$ and $\mathbf{M}_l \in \mathbb{R}^{j\times j}$ are the inertia coupling matrix between the legs and the base, and the inertia matrix of the legs. Instead $\dot{\boldsymbol{\xi}}_b = [{}_W\dot{\boldsymbol{v}}_{WB}^{\top}\ \ {}_W\dot{\boldsymbol{\omega}}_{WB}^{\top}]^{\top} \in \mathbb{R}^6$ is the total acceleration of the base (both linear and angular) and $\ddot{\mathbf{q}}_l$ is the acceleration of the leg joints. Regarding the Coriolis, centrifugal and gravity terms for the base, those terms are denoted by $\mathbf{h}_b \in \mathbb{R}^6$, while for the limbs these terms are denoted by $\mathbf{h}_l \in \mathbb{R}^j$. Contact Jacobians are denoted for the base and for the legs respectively as $\mathbf{J}_{st,b}^{\top} \in \mathbb{R}^{n_s\times 3}$ and $\mathbf{J}_{st,l}^{\top} \in \mathbb{R}^{l\times 3}$. As is possible to see, since the base is unactuated, $\boldsymbol{\tau}$, while the torques for the legs are denoted by $\boldsymbol{\tau}_l$.

Instead, for the robotic manipulator the following model is used:

$$\mathbf{M}_a(\mathbf{q}_a)\ \ddot{\mathbf{q}}_a + \mathbf{h}_a(\mathbf{q}_a, \dot{\mathbf{q}}_a) = \boldsymbol{\tau}_a \tag{3.4}$$

where $\mathbf{M}_a(\mathbf{q}_a) \in \mathbb{R}^{n_a \times n_a}$ is the inertia matrix of the arm and $\ddot{\mathbf{q}}_a \in \mathbb{R}^{n_a}$ is the acceleration vector of the manipulators' joints. Instead, $\mathbf{h} \in \mathbb{R}^{n_a}$ is the vector of Coriolis, centrifugal and gravity terms, while $\dot{\mathbf{q}}_a$ is the velocity of the joints of the manipulator or. Concluding, on the other side of the equation there is $\boldsymbol{\tau}_a \in \mathbb{R}^{n_a}$ which is the vector of joint torques. It is important to note that, in this model other external forces are not taken into account.

3.2 Visual Servoing

Visual servoing [35] is a category of control methods which use the vision data in order to control the motion of a robot. In particular, vision data can be given by a camera which is mounted on a manipulator, which is called Eye-in-Hand, by one which looks at the manipulator and is called Eye-to-Hand, or a camera mounted on a mobile robot.The aim of all vision-based control schemes is to minimize an error $\boldsymbol{e}(t)$:

$$\boldsymbol{e}(t) = \boldsymbol{s} - \boldsymbol{s}^* \tag{3.5}$$

Where $\boldsymbol{s}$ are the current states of the image features seen by the camera and $\boldsymbol{s}^*$ are the goal states of them. With these features it is possible to design a velocity controller. Indeed, the time variations of s are related to the camera velocity $\boldsymbol{v}_c$ by the following relation:

$$\dot{\boldsymbol{s}} = \boldsymbol{L}_{\boldsymbol{s}} \boldsymbol{\xi}_c \tag{3.6}$$

Where $\boldsymbol{L}_s \in \mathbb{R}^{k \times 6}$ (k is the number of used features) is called the interaction matrix related to $\boldsymbol{s}$ and it links how the features vary if the camera moves. That matrix has always to be estimated $\widehat{\boldsymbol{L}_{\boldsymbol{s}}}$ because it depends on the calibration of the camera and on quantities which are measured.

Considering $\boldsymbol{v}_c$ as the input to the robot controller and the objective to ensure an exponential decoupled decrease of the error (i.e., $\dot{\boldsymbol{e}} = -\lambda \boldsymbol{e}$), is possible to obtain:

$$\boldsymbol{\xi}_c = -\lambda \widehat{\boldsymbol{L}_{\boldsymbol{s}}^+} \boldsymbol{e} \tag{3.7}$$

where $\widehat{\boldsymbol{L}_{\boldsymbol{s}}^+} \in \mathbb{R}^{6 \times k}$ is the Moore-Penrose pseudoinverse of the estimated interaction matrix. Applying substitutions in the previous equations is obtained the time variation of the error $\dot{\boldsymbol{e}}$:

$$\dot{\boldsymbol{e}} = \boldsymbol{L}_s \boldsymbol{v}_c = -\lambda \boldsymbol{L}_{\boldsymbol{s}} \widehat{\boldsymbol{L}_{\boldsymbol{s}}^+} \boldsymbol{e} \tag{3.8}$$

where $\boldsymbol{L_s}\widehat{\boldsymbol{L_s^+}}$ is used to study the stability, since the sufficient condition is:

$$\boldsymbol{L_s}\widehat{\boldsymbol{L_s^+}} > 0 \tag{3.9}$$

There are different approaches to design $\boldsymbol{s}$, the most commons are image-based visual servo control (IBVS) and position-based visual servo control (PBVS).

3.2.1 Image-based visual servo control (IBVS)

Regarding IBVS (Image-based visual servoing) that error is generated from the difference between the desired image features and the current image features. The simplest features are points, while some more articuled are polar coordinates of a 2-D line or a more complex 2-D shape. An important remark is that these 2-D points are expressed in the projection plane. To map from image space (pixels) to projection plane space the intrinsic parameters of the camera are used as follows:

$$\begin{cases} x = \dfrac{(x_p - c_x)}{f_x} \\ y = \dfrac{(y_p - c_y)}{f_y} \end{cases}$$

where c_x and c_y are coordinates of the principal point, f_x and f_y are the focal lenght on the two axes, x_p and y_p are the pixel coordinates. Another useful mapping is between the 3-D space and the projection plane space, which is as follows:

$$\begin{cases} x = \dfrac{X}{Z} \\ y = \dfrac{Y}{Z} \end{cases}$$

where X, Y and Z are the 3-D coordinates. For example the interaction matrix of a 2-D point is the following:

$$\boldsymbol{L_{s2D}} = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix} \tag{3.10}$$

Rotational motions of the camera are the most problematic for IBVS. In this case the camera parameters influence the trajectory performed by the robot but will never affect the accuracy of the position reached. On the other hand, when the movement is long, the camera may reach a local minima or may cross a singularity

of the interaction matrix.

3.3 Solution of the SAG problem

As previously mentioned, the SAG problem deals with the search, approach and grasp of a target object in the environment. Here, we do not lower the complexity assuming the object to be known a priori. However, we rely on an existing work [36] for determining the grasping pose, being this still an open problem in robotics, and for the object detection we rely on YOLOv5 neural network [37].

In this section, we state the objectives for each of the three above mentioned phases. More specifically, while searching, the aim is to reduce the time required to find the object, penalizing base displacements and avoiding camera occlusions. The first penalization is motivated by the arm having lower inertia compared to the base. In other words, moving the arm can lead to faster motion; hence a lower time to scan the surroundings. In contrast, the second penalization is related to the fact that the quadruped can occlude the view of the camera, causing the impossibility to examine a portion of the environment and potentially to have fault detections. Differently, the objective of the approach task is to align the heading of the base in order to point the object and to reach a desired distance between the arm's end-effector and the object. Here, no assumption are made about the object being fixed in the world. Finally, during the grasping task, the robot's goal is first to position the arm's end-effector within the object's workspace and subsequently to grab the object.

Chapter 4

Methodology

In this thesis, to solve the SAG problem a dedicated pipeline is implemented. That means the need of controllers to generate the desired motions. Indeed, in this work are developed several controllers for the manipulator. While, regarding the base the controllers are external modules. In this chapter, first is explained the main idea behind the proposed control strategy for the arm. After, the pipeline and the controllers are described.

4.1 A decoupled control approach

Manipulators are composed by joints which are connected between them with links. The latters, usually are heavier at the beginning of the arm and they become lighter at the end. This is done because the first links have to hold all the others and also to enlarge the workspace of the robotic arm. This difference between the initial and final links of a manipulator allows to categorize them based on their inertia. In particular the first links can be seen as high inertia ones and the last links as low inertia ones. An example of that, it is the manipulator used in this work, a Kinova Gen3 7-DoF [38]. In fact, as it is possible to observe in Fig.(4.1), the first four links (blue ellipse) of that robotic arm have more inertia than the last three (yellow ellipse).

In other words, this grouping of links suggests to use the last ones to perform motion which have to be fast and reactive. The developed control strategy here presented relies on this concept. The reason is that in our case the robotic arm is mounted on top of a quadruped robot. The latter creates disturbances when moving, and these perturbations have to be compensated. To be more precise, a key aspect in a SAG pipeline is to keep visual on the object, even if the robot is moving. That task is

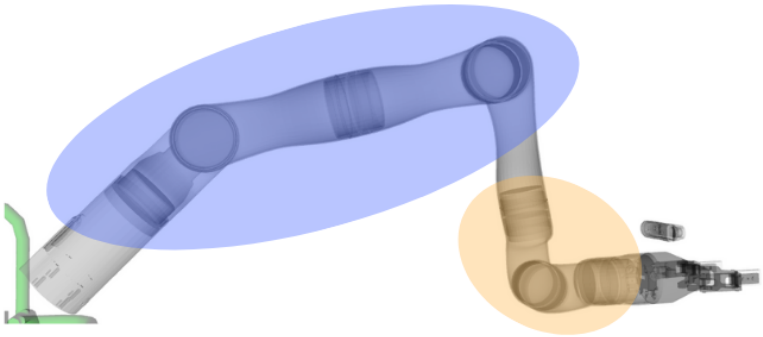


Figure 4.1: Representation of our manipulator split into two groups: in blue, high inertia links, while in yellow low inertia links.

very sensible to disturbance and a reactive control is needed.

4.2 Control Pipeline

Regarding the control strategy, the quadruped manipulator is controlled with force control. This brings several advantages to the system. One of them is the possibility to implement compliant control strategy, in presence to external disturbances. Instead with other control methods the system has not that possibility, for example with velocity or position control the joints remain stiff. Having that type of compliant behavior, is a key factor for these types of robots, since nowadays they have also the purpose to operate in environments where humans are present. In this project the two subsystems, quadruped and robotic arm, are controlled separately. In particular, the manipulator is directly controlled by our framework, since it generates as output the torques, $\boldsymbol{\tau}_a$, for its joints. Instead, concerning the quadruped, our work outputs a reference twist for the base, $\boldsymbol{\xi}_b$, which is composed by the desired linear and angular velocities. That is taken by the Reactive Control Framework (RCF) [39] which generates the desired wrench for the base, $\boldsymbol{\zeta}_b$, which is composed by the forces and moments. After that, as last module, there is the Trunk Controller [40] which starting from the desired wrench for the base outputs the torques for the legs' joints, $\boldsymbol{\tau}_l$. The graphical description of this pipeline is represented in Fig.(4.2).

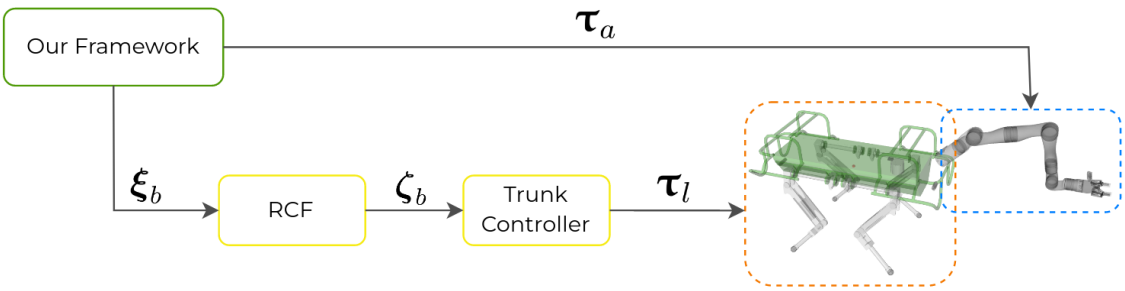


Figure 4.2: Graphical representation of the overall control pipeline.

4.2.1 RCF: Reactive Control Framework

The reactive control framework [39] is a controller that generates a desired wrench to stabilize the base and references for swing legs based on desired footholds. Regarding the balancing of the robot, a desired wrench is computed based on the error between desired and measured linear and angular acceleration, using a PD control law such as:

$$ {}_W\dot{\boldsymbol{v}}_{WBd} = \mathbf{K}_p \left({}_W\mathbf{p}_{WBd} - {}_W\mathbf{p}_{WB} \right) + \mathbf{K}_d \left({}_W\dot{\boldsymbol{v}}_{WBd} - {}_W\dot{\boldsymbol{v}}_{WB} \right) \tag{4.1} $$

$$ {}_W\dot{\boldsymbol{\omega}}_{WBd} = \mathbf{K}_p \left(\mathbf{q}_{WBd} - \mathbf{q}_{WB} \right) + \mathbf{K}_d \left({}_W\boldsymbol{\omega}_{WBd} - {}_W\boldsymbol{\omega}_{WB} \right) \tag{4.2} $$

where ${}_W\dot{\boldsymbol{v}}_{WBd} \in \mathbb{R}^3$ is the desired linear acceleration of the base, ${}_W\mathbf{p}_{WBd}$ and ${}_W\mathbf{p}_{WB}$ are the desired and current base trajectories, while ${}_W\dot{\boldsymbol{v}}_{WBd}$ and ${}_W\dot{\boldsymbol{v}}_{WB}$ are the desired and current base linear velocities. All the $\mathbf{K}_* \in \mathbb{R}^{3\times3}$ are the PD gains. Instead, ${}_W\dot{\boldsymbol{\omega}}_{WBd} \in \mathbb{R}^3$ is the desired angular acceleration of the base, $\mathbf{q}_{WBd} \in \mathbb{R}^3$ and $\mathbf{q}_{WB} \in \mathbb{R}^3$ are Euler angles representing the desired and actual orientation of the base with respect to the World reference frame, ${}_W\boldsymbol{\omega}_{WBd}$ and ${}_W\boldsymbol{\omega}_{WB} \in \mathbb{R}^3$ are the desired and current angular velocities of the base.

4.2.2 Trunk Controller

Once the desired wrench to stabilize the base is calculated from the RCF, another module called Trunk Controller finds the ground reaction forces to exert the desired wrench on the base. To avoid slipping, friction constraints are imposed to bound the tangential and lateral component of the ground reaction forces. Hence, an optimization problem is solved where a cost function expressed in the form of a wrench tracking is optimized, and the decision variables, i.e contact forces, are found imposing the above-mentioned friction constraints. In more details, the ground

reaction forces (GRFs, $\mathbf{f}_e$) are computed using the base dynamics model:

$$\underbrace{\begin{bmatrix} \mathbf{I} & \dots & \mathbf{I} \\ \mathbf{S}(\mathbf{p}_1) & \dots & \mathbf{S}(\mathbf{p}_{nc}) \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \mathbf{f}_{e,\,1} \\ \vdots \\ \mathbf{f}_{e,\,nc} \end{bmatrix}}_{\mathbf{f}_e} = \underbrace{\begin{bmatrix} \boldsymbol{M}_{bW}\dot{\boldsymbol{v}}_{WBd} \\ \mathbf{I}_{b\ W}\dot{\boldsymbol{\omega}}_{WBd} \end{bmatrix} + \boldsymbol{h}_b}_{\boldsymbol{b}} \tag{4.3}$$

with $\mathbf{p}_* \in \mathbb{R}^{3\times n_c}$ the contact positions and n_c is the number of contacts points. $\boldsymbol{M}_b$ and $\boldsymbol{I}_b$ being respectively the mass and inertias of the trunk. $\boldsymbol{S}(\mathbf{p}_*)$ denotes a skew-symmetric matrix used for calculating vector products between the contact point and contact force with the ground. At each control loop, the following quadratic programming problem is solved as:

$$\begin{aligned} \mathbf{f}_e^d = &\, argmin(\mathbf{A}\mathbf{f}_e - \mathbf{b})^\top \mathbf{S}(\mathbf{A}\mathbf{f}_e - \mathbf{b}) + \alpha \mathbf{f}_e^\top W \mathbf{f}_e \\ &\text{s.t. } \underline{\mathbf{d}} < \mathbf{C}\mathbf{f}_e < \overline{\mathbf{d}} \end{aligned} \tag{4.4}$$

where $\mathbf{S} \in \mathbb{R}^{6\times 6}$ and $\mathbf{W} \in \mathbb{R}^{3n_c\times 3n_c}$ are positive-definite weight matrices, $\alpha \in \mathbb{R}$ weighs the secondary objective, $\mathbf{C} \in \mathbb{R}^{p\times 3n_c}$ is the inequality constraint matrix that defines friction constraints and upper and lower bounds in the z direction, $\underline{\mathbf{d}}$, $\overline{\mathbf{d}} \in \mathbb{R}^p$ the lower/upper bound respectively, with p being the number of inequality constraints and n_c the number of contact points.

Subsequently, the found ground reaction forces are converted to joint torques through the Jacobian. This torque component is referred as feedforward component and it is obtained as $\boldsymbol{\tau}_{ff}$:

$$\boldsymbol{\tau}_{ff} = -\mathbf{J}_c^\top \mathbf{f}_e \tag{4.5}$$

where $\mathbf{J}_c \in \mathbb{R}^{n_c\times n_s}$ is the stacked jacobian of the contact points, $\mathbf{S} = [\mathbf{I}_{n_j\times n_j}\ \mathbf{0}_{n_j\times 6}]$ is a selection matrix that selects the actuated joints and n_j is the number of joints. For safety reasons and for generating the motion of the swing legs, a second component is computed using a PD joint-position controller with low gains. This second term, $\boldsymbol{\tau}_l$, is referred as feedback term and is calculated as:

$$\boldsymbol{\tau}_l = \boldsymbol{\tau}_{ff} + PD(\mathbf{q}_j, \dot{\mathbf{q}}_j, \mathbf{c}_{st}) \tag{4.6}$$

where $\mathbf{c}_{st}$ is a boolean vector that represent the stance condition of the legs. The overall torque is sent to the low-level controller that closes the loop in torque and track the desired joint torques. While the trunk controller runs at 250 Hz, the

low-level controller runs at 1 kHz, providing an higher stabilization to the robot.

4.3 Developed Framework

In the previous section has been introduced the framework which is the result of this thesis. It is meant as the core that guides the robot in all the actions which are necessary in order to achieve a SAG task. In fact, the main idea is to include all the different control tools in a single space and manage them with a Behavior Tree. Since the SAG problem can be split into three parts, also this framework can be divided in that manner. However, it is also possible to introduce a general structure which is shared by all the three.

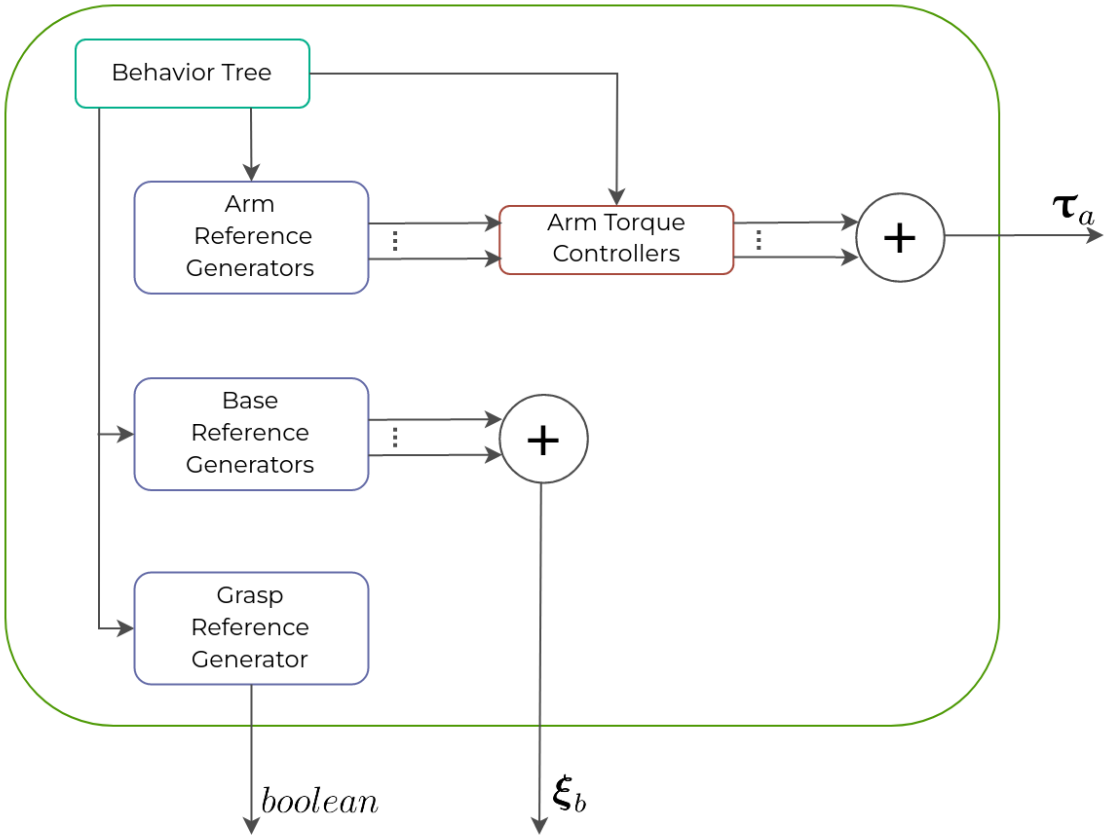


Figure 4.3: Graphical representation of the general structure of the framework, that is shared between all the three phases (search, approach and grasp).

The first element of this structure is the Behavior Tree (BT), which is connected to all the other modules in order to synchronize them. After that, there are the reference generators, which are split into three, one for the manipulator, one for the

base, and the last for the gripper. The latter is treated as another subsystem, since usually it is not included in the arm joints. The purpose of such type of elements is to group different methods to compute the desired states for the subsystems starting from the their current states. In each control iteration the methods are chosen by the BT. The generated references depend on the chosen methods, but for the base they are always twists, for the gripper is always a boolean, instead for the arm there are different possible types which will be explained in details in the next sections. Since there is only one reference generator for the gripper and it simply outputs if the gripper has to be open (0) or close (1), it will not be included in the next sections. At that point, regarding the base, the references are summed to obtain the final one which is the first output of the framework. Concerning the base the boolean is directly the second output of the framework. Instead, regarding the arm, the references are sent to another element which contains all the torque controllers. As before, in each control loop the controllers are chosen by the BT. Lastly, the torques generated from these controllers are summed together to obtain the final torques which are the second and last output of our framework. Concerning the torque controllers they are more precisely impedance ones and they use the following control law (that is in Cartesian space but the one in joint space can be derived by substituting the term $\mathbf{J}_a^\top[\mathbf{K}_p\ \mathbf{e}_p + \mathbf{K}_d\ \dot{\mathbf{e}}_d]$ with a PD controller which has the errors in joint space):

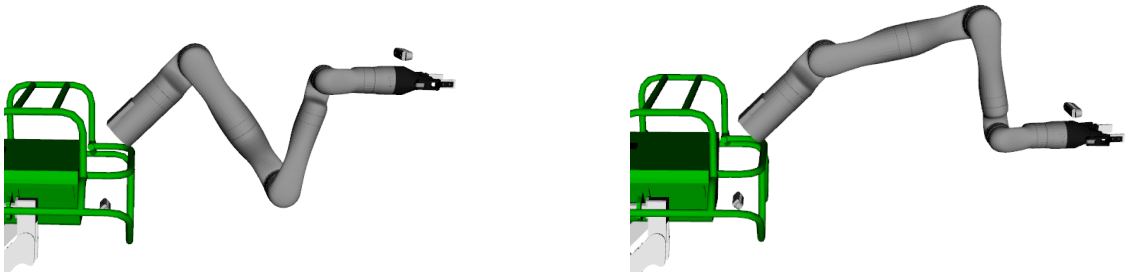
$$\boldsymbol{\tau}_a = \mathbf{J}_a^\top[\mathbf{K}_p\ \mathbf{e}_p + \mathbf{K}_d\ \dot{\mathbf{e}}_d] + \mathbf{h}_a \tag{4.7}$$

which is derived from the manipulator's dynamic model, described by the equation 3.1.1 in section 3.1, making some approximations (e.g. the derivative of the Jacobian can be approximated to zero). All the $\mathbf{K}_* \in \mathbb{R}^{3\times3}$ are the PD gains. As last thing, in our work the maximum number of degrees of freedom needed by all the tasks is six, since our manipulator is a 7-DoF one, has been decided to keep the third joint always in the same position. The latter is the reason why in the following section the reader will not find the torques for the third joint.

4.3.1 Control for searching the object

In order to completely understand the control strategy developed for the search phase, it is fundamental to recall the objective which has been fixed for that task: reduce the time required to find the object, penalizing base displacements and avoiding camera occlusions. To achieve it, a policy which avoid singular configurations and

self collisions for the robot arm is developed. The main idea is to first scan what is placed in front of the robot base, and lately what is behind it.



(a) Manipulator home posture with elbow down.

(b) Manipulator home posture with elbow up.

Figure 4.4: Examples of manipulator home postures with joints far from the base.

First, the arm is brought to a home configuration where the joints are kept not in the proximity of the base, as shown in Fig.(4.4). Then, a circular motion for the arm's end-effector is generated. In particular, collisions between the base and the arm are avoided moving the latter around a circle with center the arm's base. While, to avoid singularities, the circle's diameter is chosen lower than the maximum allowable distance for the arm. That circumference is chosen because the circle defined with center the robot's base has a too large radius that doesn't allow to move enough without reaching singularities and also because the arm moves only in the front of the robot and not also on the sides.
The next space to scan is behind the quadruped, this has to be divided into two steps, one on the left and one on the right of the base, this is done to avoid image occlusions caused by the robot being in the field of view of the camera. To perform these tasks two circular trajectories for the end-effector can be used. In that case the center of the circles has to be the wrist of the robotic arm. To avoid collisions and singularities also during these motions, it is better to perform them only with the last joints since the others are not necessary. Unifying the above-mentioned pipeline with the fact that in the next phases a decoupled approach is necessary, it is possible to use the high inertia links to perform the first circular trajectory and the low inertia links to perform the other two. In addition, to uniform the three motions, the first circle can be computed for the wrist instead of the end-effector. With that strategy it is possible to finish the first circular trajectory having the wrist already in a right position to start immediately the second one. A graphical representation of these three circles trajectories can be found in Fig.(4.5). In the particular case of our robotics arm, the wrist is the sixth joint. Currently, the only thing missing

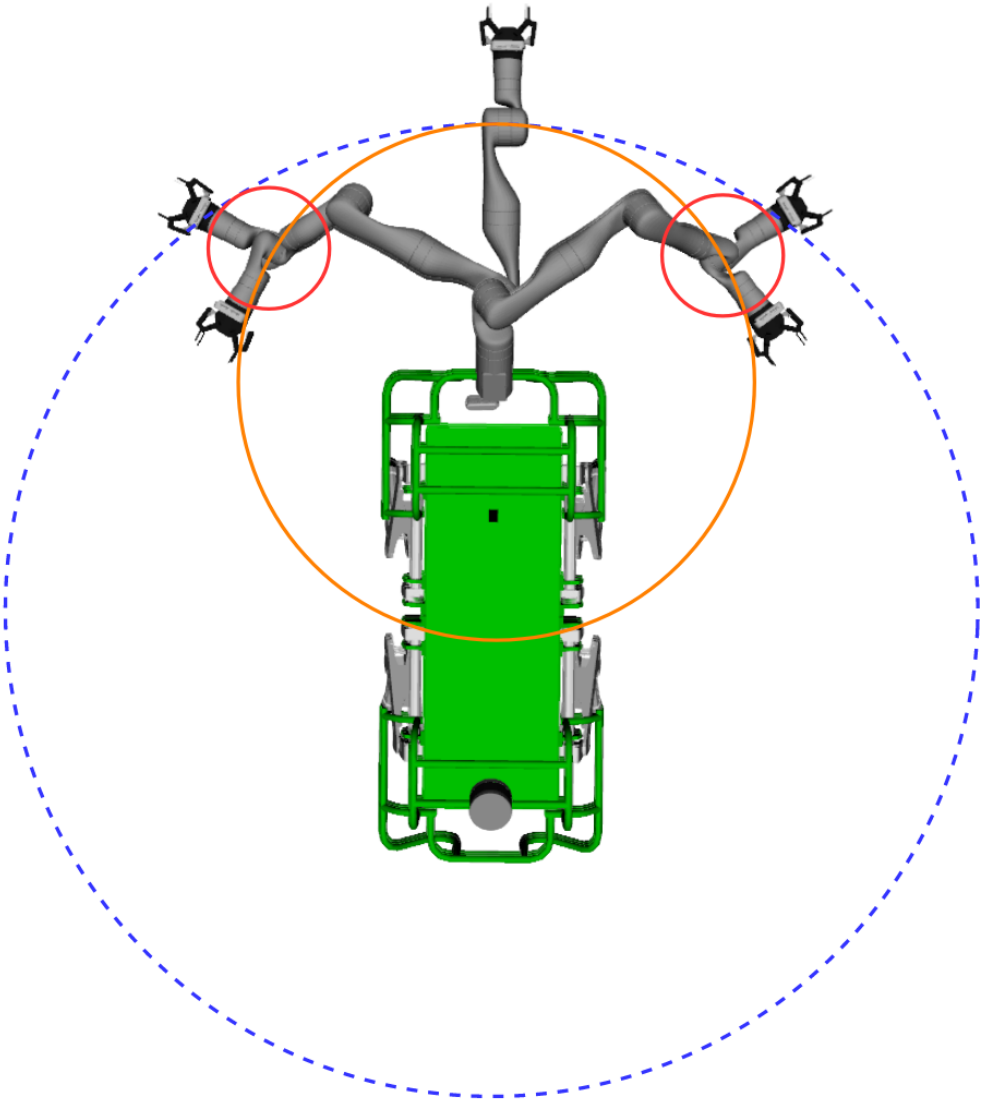


Figure 4.5: Graphical representation of the trajectories used in the search phase (with solid lines), the one with the dashed line indicates the non feasible one with center in the quadruped's base).

is a way to compute the left and right extreme points on the first circumference, the ones used as centers for the other two circles. In order to do that the following assumptions are made:

- the home posture is not in the proximity of the singularities, as already shown in Fig.(4.4)
- the height of the wrist along the first trajectory is constant and equal to the one in the home posture

- the radius of the first circumference, centered in the base of the arm, is the distance between the base and the wrist when the robotic arm is in the home posture
- the arm is placed in the front of the robot

Using these assumptions and considering that the wrist cannot move after the line passing through the arm's base, since in that case there will be collision with the quadruped, the circular trajectory is always inside the workspace of the manipulator's wrist. The just made observation is done because the arm is mounted with a certain roll angle with respect to the base plane, that implies the workspace of the robot, which is centered in the shoulder, to be shifted from the arm's base. So, to define the left and right limits is possible to use a desired distance from the base of the arm in order to be sure that the manipulator doesn't collide with the quadruped's base. That distance is shown with the double grey arrow in Fig.(4.6).

With the described pipeline, there is the little part on the hind of the quadruped which is not scanned to avoid camera occlusions. To solve that, if the object is not found during the three circular trajectories, the quadruped base will rotate around the yaw of π and after the pipeline is repeated. In other words, the quadruped moves only in a particular case and that confirms the fixed objective.

Regarding the control structure, in the search phase there are five arm reference generators and two base reference generators. Let's start describing briefly the role of each generator starting from the ones for the arm. The generator "Home Posture Arm Joints" computes the desired position of the joints, $\mathbf{q}_{ad} \in \mathbb{R}^{n_a}$, in order to reach the home configuration. Regarding the wrist, there is "Desired Position Circle Arm Wrist", which compute the next desired position on the circle for the wrist, ${}_B\mathbf{p}_{B\text{Wrist}} \in \mathbb{R}^3$, in order to reach the desired point of the circular trajectory. Instead "Desired Position Arm Wrist" computes the reference to bring the wrist in a desired cartesian point. For the end-effector, there are two generators, the first "Height and Roll Arm EE" solves the problem to maintain the height and the roll of the end-effector when the wrist moves on the circle. The last reference generator for the robotic manipulator computes the next desired position on the circle for the end-effector, ${}_B\mathbf{p}_{BEE} \in \mathbb{R}^3$, in order to reach the desired yaw orientation needed to look backwards. Concerning the reference generators for the base, "Maintain Pose Base" generates a zero twist, $\boldsymbol{\xi}_b \in \mathbb{R}^6$, in order to don't change the current state of the base. The other "Rotate Yaw Pi Base" generates a twist which has only a z angular component, in order to achieve a rotation of π relative to the orientation of the robot before start

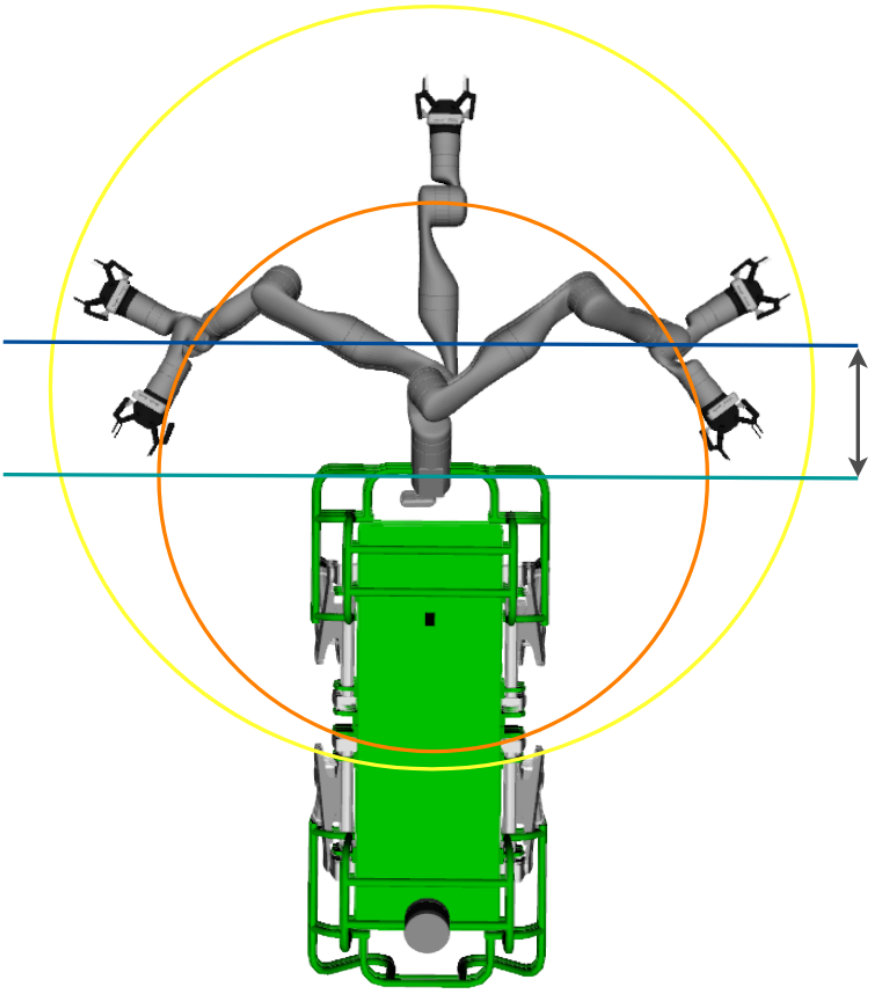


Figure 4.6: Graphical representation of the first circular trajectory used in the search phase (in orange), the workspace of the arm's wrist (in yellow), the line passing through the arm's base (in light blue), the distance (double grey arrow) from the arm's base which defines the line of the left and right limits (in blue).

rotating. Moving to the torque controllers for the arm, they are four. For the sake of this thesis, only the general ones are going to be described since are applicable also to other robots, while "Custom Arm EE" is dependent on the arm used in this work. The three remaining controllers, as said at the beginning of this chapter, PD impedance controller (equation 4.3). More precisely the first is in joint space and controls all the joints, the proportional and derivative errors are the following:

$$\mathbf{e}_p = q_{ad} - q_a \tag{4.8}$$

$$\mathbf{e}_d = -\dot{q}_a \tag{4.9}$$

where $q_a \in \mathbb{R}^{n_a}$ is the vector of the current joints' positions and $\dot{q}_a \in \mathbb{R}^{n_a}$ is the vector of the current joints' velocities.

Instead, the other two controllers are in cartesian space and they use the decoupled approach. Starting from the first, which is for the high inertia links, so joints 1, 2 and 4 (because the third is kept in a constant position), it has the following control laws:

$$\boldsymbol{\tau}_{a12d} = \mathbf{J}_{at12\text{wrist}}^{\top} \left[\mathbf{K}_p \, \mathbf{e}_p + \mathbf{K}_d \, \mathbf{e}_d\right] + \mathbf{h}_{a12} \tag{4.10}$$

$$\boldsymbol{\tau}_{a4d} = \mathbf{J}_{at4\text{wrist}}^{\top} \left[\mathbf{K}_p \, \mathbf{e}_p + \mathbf{K}_d \, \mathbf{e}_d\right] + \mathbf{h}_{a4} \tag{4.11}$$

where $\boldsymbol{\tau}_{a12d} \in \mathbb{R}^2$ is the torque vector only for the first two joints, $\mathbf{J}_{at12\text{wrist}} \in \mathbb{R}^{2\times3}$ is the translational part of the Jacobian matrix from base arm to wrist only for joints one and two, the last term $\mathbf{h}_{a12} \in \mathbb{R}^{2\times1}$ is the compensation vector for Coriolis, centrifugal and gravity only for the first two joints. The same definitions are valid for the second equation, the one regarding the fourth joint, with differences in dimensions as $\boldsymbol{\tau}_{a4d} \in \mathbb{R}$, $\mathbf{J}_{at4\text{wrist}} \in \mathbb{R}^{1\times3}$ and $\mathbf{h}_{a4} \in \mathbb{R}$. Regarding the proportional and derivative errors, in that case they are:

$$\mathbf{e}_p = {}_B\mathbf{p}_{B\text{Wrist}_d} - {}_B\mathbf{p}_{B\text{Wrist}} \tag{4.12}$$

$$\mathbf{e}_d = -{}_B\boldsymbol{v}_{B\text{Wrist}} \tag{4.13}$$

where ${}_B\mathbf{p}_{B\text{Wrist}_d} \in \mathbb{R}^3$ is the reference cartesian position for the wrist with respect to base frame and expressed in base, ${}_B\mathbf{p}_{B\text{Wrist}} \in \mathbb{R}^3$ is the current position of the wrist with respect to base and expressed in base frame, and ${}_B\boldsymbol{v}_{B\text{Wrist}} \in \mathbb{R}^3$ is the current translational cartesian velocity of the wrist with respect to base frame and expressed in base. The reason why in that case there are two control laws is because the joints are not concatenated since between the second and the fourth there is the three. The last thing that the controller does is to insert the computed torques in the associated indexes of a vector $\boldsymbol{\tau}_{a124} \in \mathbb{R}^{n_a}$ in order to after add them with the torques of joints of the low inertia links.

The last controller in this phase is the one for the joints 5, 6, and 7. As the one just presented it is in cartesian space but having end-effector has target instead of the wrist as before. The controller has the same logic as the previous one, but this time the joints are consecutive, and so there is only one control law:

$$\boldsymbol{\tau}_{a567d} = \mathbf{J}_{at567\text{EE}}^{\top} \left[\mathbf{K}_p \, \mathbf{e}_p + \mathbf{K}_d \, \mathbf{e}_d\right] + \mathbf{h}_{a567} \tag{4.14}$$

where $\boldsymbol{\tau}_{a567d} \in \mathbb{R}^3$ is the torque vector only for the last three joints, $\mathbf{J}_{at567\text{wrist}} \in \mathbb{R}^{3\times3}$ is the translational part of the Jacobian matrix from base arm to end-effector only for joints five, six and seven, the last term $\mathbf{h}_{a567} \in \mathbb{R}^{3\times1}$ is the compensation vector for Coriolis, centrifugal and gravity only for the last three joints. Instead for the errors, they are:

$$\mathbf{e}_p = {}_B\mathbf{p}_{BEEd} - {}_B\mathbf{p}_{BEE} \tag{4.15}$$

$$\mathbf{e}_d = -{}_B\boldsymbol{v}_{BEE} \tag{4.16}$$

where ${}_B\mathbf{p}_{BEEd} \in \mathbb{R}^3$ is the reference cartesian position for the end-effector with respect to base frame and expressed in base, ${}_B\mathbf{p}_{BEE} \in \mathbb{R}^3$ is the current position of the end-effector with respect to base and expressed in base frame, and ${}_B\boldsymbol{v}_{BEE}$ is the current translational cartesian velocity of the end-effector with respect to base frame and expressed in base. As before, these result torques are inserted in the associated indexes of a vector $\boldsymbol{\tau}_{a567} \in \mathbb{R}^{n_a}$ in order to after add them with the torques of joints of the high inertia links. The graphical representation of the control structure of the search is in Fig.(4.7).

4.3.2 Control for approaching the object

Once the object is detected, the approach phase starts and it can be divided in two stages, alignment to the object and approach of the object.

Starting from the alignment, to understand how much the base has to rotate, it is possible to define its target yaw orientation relative to the one before the start of the rotation. To do that the yaw difference between the segment which connect the object with the base and the x axis of the base is computed, as shown in Fig.(4.8).

During the alignment, the base creates disturbances for the arm due to the trot. These disturbances can bring the camera to lose the object, to avoid that a robust policy is needed. For that purpose Visual Servoing is adopted, since it allows to keep the object in the center of the image. In particular, that is done taking as features the coordinates of a pixel, and imposing as desired value the coordinates of the center pixel in the image. Remember that these pixel coordinates have to be

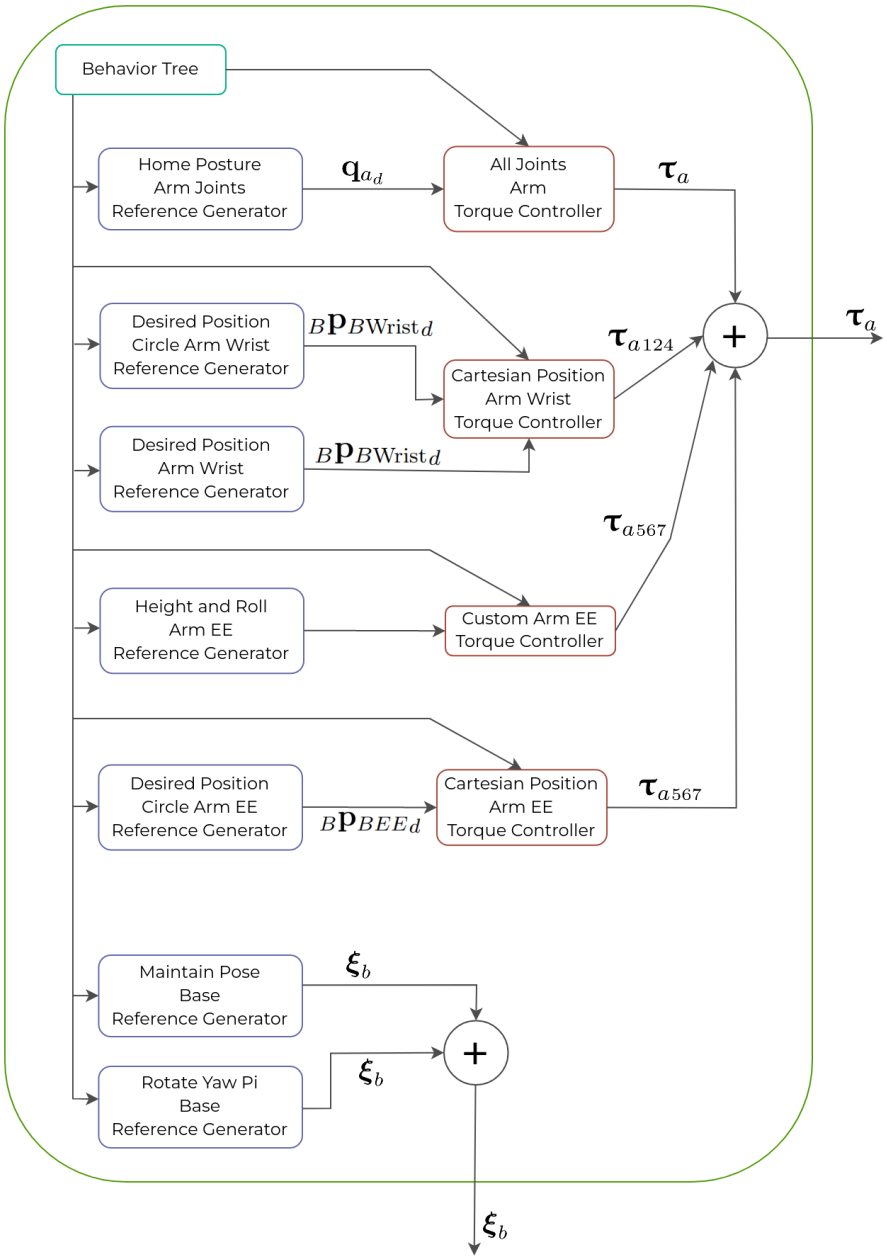


Figure 4.7: Graphical representation of the framework structure during the search phase.

expressed in the projection plane. Therefore the error vector is the following:

$$\mathbf{e}_{s'} = \begin{bmatrix} x - x^* \\ y - y^* \end{bmatrix} \tag{4.17}$$

where x and y are the current pixel coordinates and x^* and y^* are the desired ones,

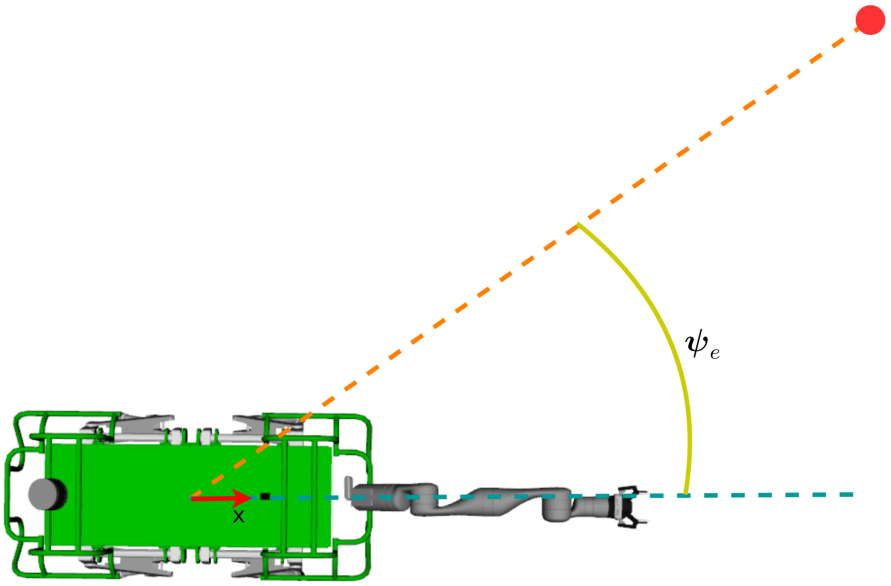


Figure 4.8: Graphical representation of the yaw error in order to align the base to the object.

all expressed in projection plane. Since the object detection neural network outputs the bounding box of the detected object, it is possible to use the coordinates of its center as the current features. Instead regarding the desired values, the center of the image expressed in projection plane has coordinates (0, 0), because it is the origin. So the errors become only the coordinates of the current center of the bounding box expressed in projection plane. Instead, the interaction matrix associated to these features is well known in the literature, and it is the equation 3.2.1. However, using only these features, the end-effector is free to change its roll orientation since all the rotations around the z axis of the camera are allowed. For the sake of that work it is preferable that the camera stays always oriented parallel to the base. To achieve that, it is possible to add a third feature which constrains such rotation of the camera. In order to define that feature is important to describe the problem in a formal way. Indeed, looking at the frame of the base and the one of the camera in Fig.(4.9), it is possible to note that when the camera is oriented parallel to the base, its y axis is minus the z axis of the base and as a consequence the x axis of the camera is always orthogonal to that axis.

From the vector theory is know that two vectors are orthogonal when their dot product is zero. Now, knowing that for Visual Servoing the error has to be expressed in camera frame, the following equation is derived:

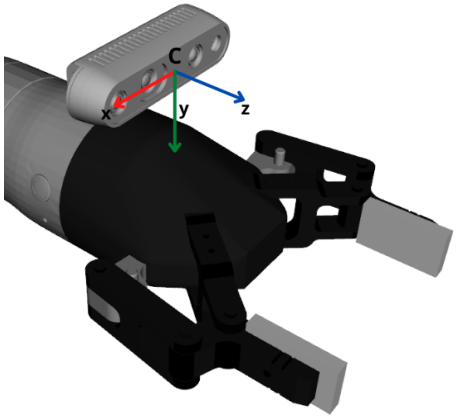


Figure 4.9: Graphical representation of the camera frame.

$$\mathbf{x}_C \cdot ({}^C\mathbf{R}_B \, \mathbf{z}_B)^\top = 0 \tag{4.18}$$

where $\mathbf{x}_C \in \mathbb{R}^3$ is the vector of the x axis of the camera frame, ${}^C\mathbf{R}_B \in \mathbb{R}^{3\times3}$ is the rotation matrix from base to camera and $\mathbf{z}_B \in \mathbb{R}^3$ is the vector of the z axis of the base frame. Performing the computations it is possible to obtain the following result:

$$({}^B\mathbf{R}_C)_{zx} = 0 \tag{4.19}$$

where $({}^B\mathbf{R}_C)_{zx} \in \mathbb{R}$ is the component of the rotation matrix which is at third row and first column. Regarding the interaction matrix for that feature s'', it can be derived knowing that the translations of the camera cannot change its orientation and the time derivative of a rotation matrix is the rotation matrix multiplied by the skew symmetric of the angular velocity ${}^B\dot{\mathbf{R}}_C = {}^B\mathbf{R}_C \, [\boldsymbol{\omega}_B]_\times$. Indeed, it is the following:

$$\mathbf{L}_{s''} = \begin{bmatrix} 0 & 0 & 0 & 0 & -({}^B\mathbf{R}_C)_{zz} & ({}^B\mathbf{R}_C)_{zy} \end{bmatrix} \tag{4.20}$$

Finally, stacking the new feature with the other two it is possible to obtain the final error vector and interaction matrix:

$$\mathbf{e}_s = \begin{bmatrix} x \\ y \\ ({}^B\mathbf{R}_C)_{zx} \end{bmatrix} \tag{4.21}$$

$$\mathbf{L}_s = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \\ 0 & 0 & 0 & 0 & -({}^B\mathbf{R}_C)_{zz} & ({}^B\mathbf{R}_C)_{zy} \end{bmatrix} \tag{4.22}$$

$\mathbf{s}$ = In order to reduce even more the possibility to lose the object, the decoupled control approach is applied. Actually, the just mentioned Visual Servoing is used only on the last joints of the arm, which are the ones moving the low inertia links allowing a more reactive behavior. Instead, the first joints, the ones moving the high inertia links, are used to guide the wrist in a desired cartesian position on the same circular trajectory used during the search phase. Concerning the desired point on the circle for the wrist, it is necessary to create a policy to choose it. To avoid redundant motions of the arm and to bring the wist in a position in which it can help the last joints to keep the object centered, the idea is to compute the intersection between the circle of the wrist's trajectory and the segment which connects the base and the target object. In that way the wrist will always reach the final alignment before the base giving more stability to the pipeline. However, when the object is on the side or behind the robot, that desired point on the circumference is after the left limit if it on the left of the robot and after the right limit otherwise. In these cases the wrist is brought at the limit on the circumference and when the desired point goes inside the limits the wrist will reach it. Some examples of desired points which are inside and outside the limits can be found in Fig.(4.10).

Once the base is aligned to the object, the approach stage starts. During that task, the manipulator uses the decoupled approach composed by Visual Servoing, using the previous defined features, for the last joints and a desired cartesian position for the wrist with the first joints. To be more precise the cartesian position for the wrist is the one it has when the arm is in home configuration, which is the same position in which the wrist is at the end of the alignment. Regarding the base, it approaches the object until the distance, on the plane xy of the base, between the end-effector and the object has reached a desired value. In particular, using the error between the desired distance and the current one, a translational velocity along x for the base is generated. However, using that reference the base is able only to move forward and if the object moves the base is not able to reach it. In order to solve that problem, another reference for the base is computed. This second reference is an angular velocity around the yaw axis for the base, and it is generated considering the error between the current yaw of the end-effector and the one that it has in the home posture (which is aligned with the heading of the base). That computation is

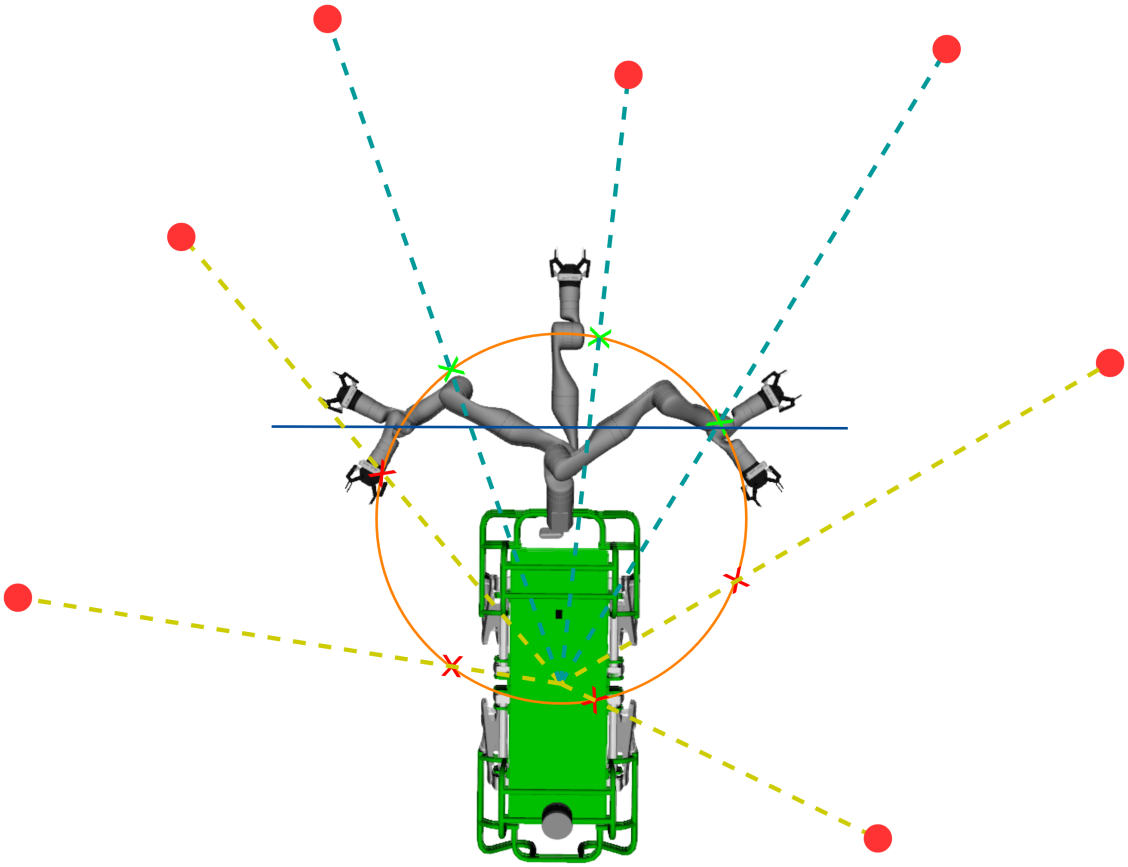


Figure 4.10: Graphical representation of the desired points on the circular trajectory for the wrist during the alignment of the base. The dotted lines are the segments from the base to the objects. Instead, the markers are the desired positions of the wrist on the circle, green ones are inside the limits, while the red ones are outside the limits.

possible since the end-effector follows the object with Visual Servoing. Using these two references, the base is able to approach the object even if it moves. A graphical representation of the two quantities used to compute the two references for the base can be found in Fig.(4.11).

Regarding the control structure, in the approach phase there are four arm reference generators and three base reference generators. Regarding the ones for the arm, the first four are the same used in the search phase. Instead, "Visual Servoing Arm EE", as the name suggest is the one which generates the references starting from the output of the Visual Servoing. In particular, as explained in section 3.2, the control law of Visual Servoing produces a twist for the camera. That twist is transformed in base frame and from that are subtracted the twist of the wrist and the twist of the

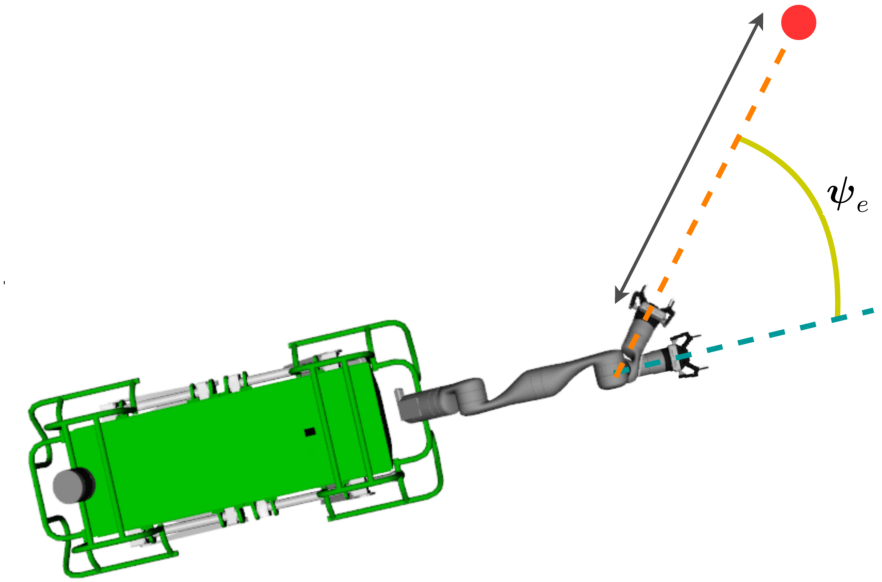


Figure 4.11: Graphical representation of the two quantities used to generate the references for the base during the approach stage. The grey double arrow denotes the distance between the end-effector and the object. Instead, the green line is the angle between the current yaw of the end-effector (dashed orange line) and the one in home posture (dashed light blue line).

base. The latter operation is done as a compensation for the motion of the wrist and of the base and it allows to use low λ gain in the Visual Servoing control law. After the just computed twist in base frame is pre-multiplied by the transpose of $\mathbf{J}_{a567\text{EE}} \in \mathbb{R}^{6\times3}$, which is the Jacobian matrix from base arm to end-effector only for joints five, six and seven. The result of this operation is $\dot{\mathbf{q}}_{a567d} \in \mathbb{R}^3$ which is the vector of desired velocities for the last joints. Finally integrating these joints velocities, it is possible to obtain $\mathbf{q}_{a567d} \in \mathbb{R}^3$ which is the vector of the desired positions for the last joints and also the output of the reference generator. Concerning the reference generators for the base, the first is also used in search phase. Instead, "Move Forward Distance EE Object Base" generates a twist, $\boldsymbol{\xi}_b \in \mathbb{R}^6$, which has only the x translational component which allows the robot to reach the desired distance from end-effector to the object. The other reference generator for the base, "Rotate Yaw As EE", produces a twist with only a z angular component, in order to keep aligned the base to the object using the yaw error of the end-effector as was explained before. Proceeding with the torque controllers for the robotic manipulator, in the approach phase they are three. The first two are the same already described in the previous section. Instead, the third arm controller is similar to the first, because it is a PD

impedance controller in joint space, but in this case it controls only the last joints. Indeed, the proportional and derivative errors are $\dot{\mathbf{q}}_{a567d} = \mathbf{J}^{\top}_{a567\text{EE}}\, \boldsymbol{\xi}_{tot}$:

$$\mathbf{e}_p = q_{a567d} - q_{a567} \tag{4.23}$$

$$\mathbf{e}_d = -\dot{q}_{a567} \tag{4.24}$$

where $q_{a567} \in \mathbb{R}^3$ is the vector of the current positions of joints five, six and seven and $\dot{q}_{a567} \in \mathbb{R}^3$ is the vector of the current velocities of joints five, six and seven. So the control law generates only torques for the last three joints, $\boldsymbol{\tau}_{a567d} \in \mathbb{R}^3$, which are inserted in the associated indexes of a vector $\boldsymbol{\tau}_{a567} \in \mathbb{R}^{n_a}$ in order to be added after with the torques of joints of the high inertia links. The graphical representation of the control structure of the search is in Fig.(4.12).

4.3.3 Control for grasping the object

Once the distance between the end-effector and the object reaches the desired value, the quadruped stops trotting and the grasp phase starts. In this phase the quadruped manipulator goes closer to the object and adjust its orientation to finally grab the target. Since this framework is developed for general objects, the grasp phase is structured in such a way that the robot can use another model for the object detection neural network in order to grasp a different object with respect to the one detected during the previous phases. The reason of the latter choice is that the target object to grasp can be far from the robot, therefore if the object is small it can be difficult to directly detect it. In these cases if the target is part of a bigger object, the latter can be used to bring the robot closer, hence it can be used for the search and approach phases. An example of that is the handle of a door, indeed it can be not detectable from the distance, while the door around it is visible also if the robot is far. Therefore, when the grasp phase starts the robot detects the (same or new) object, after that it computes the grasping pose with respect to the world frame and expressed in the world using an external module which takes as input the point cloud generated from the depth image of the RGBD camera. At that moment, the next motion performed by the robot is to bring the manipulator at the height of the grasping point. This is done to simplify the future grasping operation and to avoid the loss of the object by the camera since when the robot starts to go closer the field of view of the sensor gradually reduce. During this operation the decoupled control

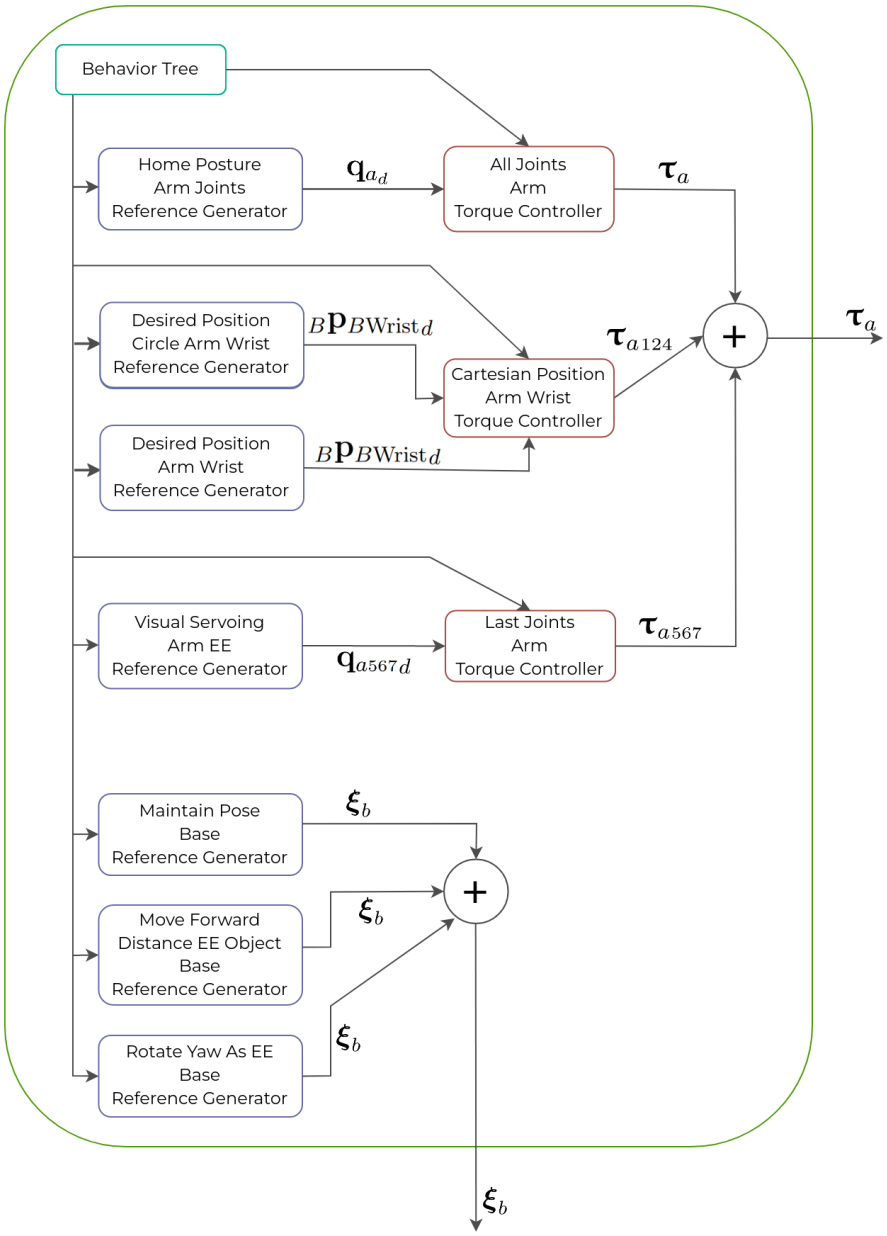


Figure 4.12: Graphical representation of the framework structure during the approach phase.

approach is adopted, using the visual servoing for the last joints to keep centering the object and the first joints to reach the desired height with the wrist. In particular the desired height component of cartesian point for the wrist is computed as the closest height to the one of the grasping position in order to maintain the wrist inside its workspace. In order to do so, it is useful to bring the arm as close as possible to the base, without overcoming a minimum distance. The motivation behind that is

the increase of the arm workspace in terms of height and also the increase, even in a small percentage, of the field of view of the camera. Instead, regarding the third component of the desired wrist position, it is set to the same as the grasping point in order to have the end-effector already aligned with it. In Fig.(4.13) an example of the final cartesian position of the wrist is shown, just notice that in this example the grasping position is too low to be achieved by the manipulator. In additon in the example the target object is below the base, but the policy is also applicable for objects which are higher than the base.

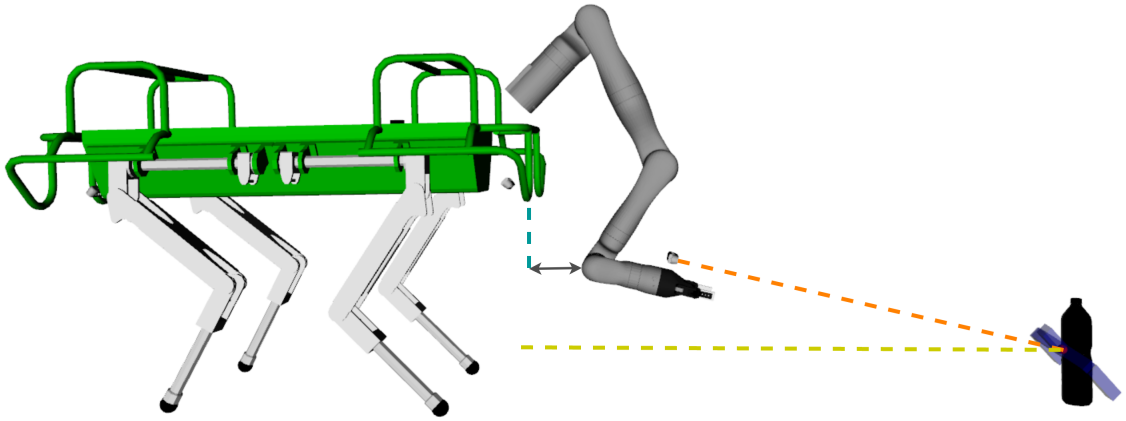


Figure 4.13: Graphical example of the height adjustment of the wrist during the grasp phase. The double grey arrow indicates the minimum distance from the wrist and the base, the orange dotted line is the segment connecting the camera to the grasping position, while the green dotted line is the height of the grasping position.

After that the robot moves forward until the grasping position is inside the workspace of the manipulator. During that motion the arm adopts again the decoupled approach, suing Visual Servoing for the last joints and the first joints to maintain the the cartesian position of the wrist. Once that motion has finished the grasping position is already in the workspace of the robot, but in order to simplify the grasping and to enlarge the workspace of the manipulator, the base performs a rotation around the pitch until the desired orientation is reached. During that motion the manipulator uses the same control strategy as the previous motion, the only difference is that the desired wrist position to adjust its height is computed again and expressed in world frame. That allows to maintain the same cartesian position even if the base frame is changing its orientation. In Fig.(4.14) an example of the robot after that motion is shown, a note regarding it is that the object is small and on the ground, that implies the end-effector to be such close to the object in order to have the grasping position

in the workspace of the robotic arm.

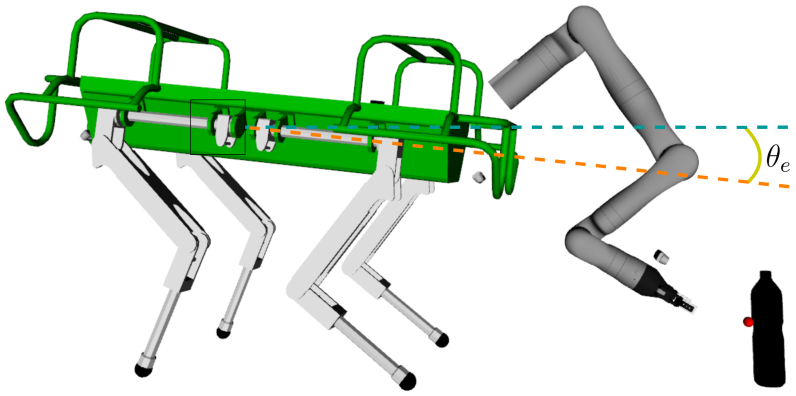


Figure 4.14: Graphical example of the pitch adjustment of the base during the grasp phase. The light blue dotted line is the initial pitch of the base, while the orange dotted line indicates the final pitch of the base.

Lastly the manipulator reach the grasp pose using all the joints and once in that desired pose it closes the gripper. An example of that can be seen in Fig.(4.15).

Regarding the control structure, in the grasp phase there are five arm reference generators and three base reference generators.

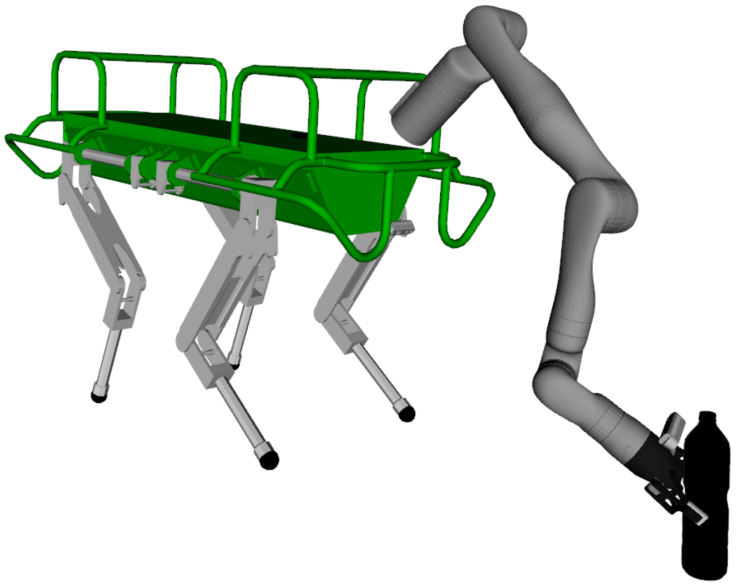


Figure 4.15: Example of the final result of the grasp phase.

Starting from the ones for the manipulators, three of them are the same used in the other two phases. Instead, "Adjust Height Arm Wrist" computes the desired

cartesian position of the wrist, ${}_B\mathbf{p}_{B\text{Wrist}_d} \in \mathbb{R}^3$, in order to bring it as close as possible to the height of the grasping position, using the method described above. The other new reference generator for the arm, "Grasping Pose Arm EE", it generates the desired position ${}_B\mathbf{p}_{BEE_d} \in \mathbb{R}^3$ and orientation expressed as the rotation matrix from base to desired end-effector frame ${}^B\mathbf{R}_{EE_d} \in \mathbb{R}^{3\times3}$ in order to reach the grasping pose. Concerning the reference generators for the base, the first is present also in the other two phases and has already been presented. Instead, the second "Move Forward Grasp in Workspace EE" generates a twist, $\boldsymbol{\xi}_b \in \mathbb{R}^6$, with only the x translational component which allows the grasping position to be in the workspace of the robotic manipulator. The other reference generator for the base, "Adjust Pitch To Grasp Base", produces a twist with only a y angular component, in order to enlarge the workspace of the manipulator in the direction of the grasping position. Continuing with the torque controllers for the robotic arm, the first four are present also in the previous phases. Instead, the last torque controller, "Cartesian Pose Arm EE", as the name suggest is an impedance controller in cartesian space which controls all the joints. That precise control law has already been presented in equation (4.3). Instead the proportional and derivative errors are the following:

$$\mathbf{e}_p = \begin{bmatrix} {}_B\mathbf{p}_{BEE_d} - {}_B\mathbf{p}_{BEE} \\ {}^B\mathbf{R}_{EE}\ a({}^{EE}\mathbf{R}_{EE_d}) \end{bmatrix} \tag{4.25}$$

$$\mathbf{e}_d = -\boldsymbol{\xi}_{EE} \tag{4.26}$$

where ${}_B\mathbf{p}_{BEE_d}$ and ${}_B\mathbf{p}_{BEE} \in \mathbb{R}^3$ are the desired and current cartesian positions of the end-effector, $a(\cdot) : \mathbb{R}^{3\times3} \to \mathbb{R}^3$ is a mapping function from rotation matrix to the associated angle axis representation, while ${}^B\mathbf{R}_{EE} \in \mathbb{R}^{3\times3}$ and ${}^B\mathbf{R}_{EEd} \in \mathbb{R}^{3\times3}$ are the rotation matrices from end-effector to base and from desired end-effector frame to end-effector. Instead $\boldsymbol{\xi}_{EE} \in \mathbb{R}^6$ is the current twist of the end-effector expressed in base frame. The graphical representation of the control structure of the grasp is in Fig.(4.16).

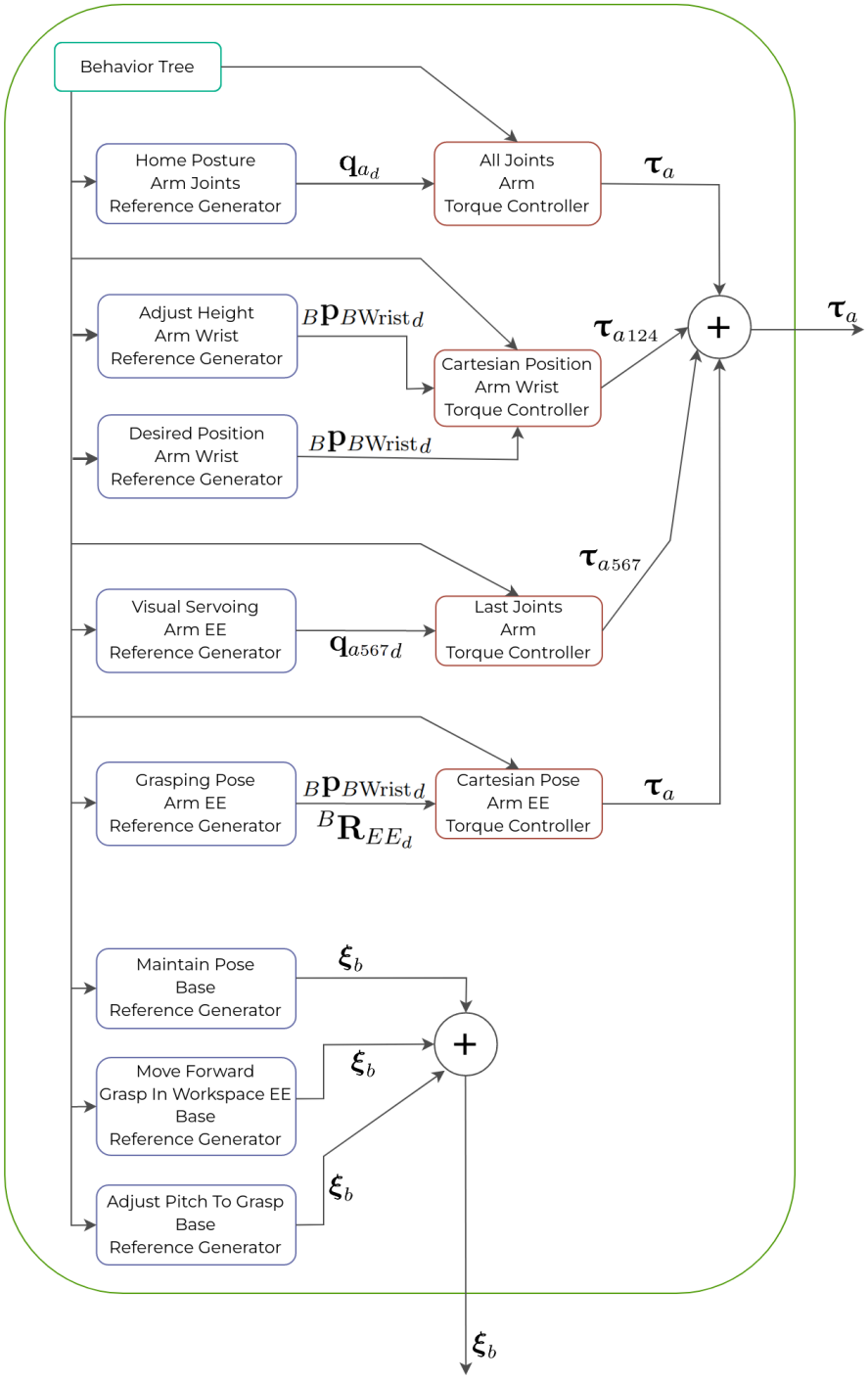


Figure 4.16: Graphical representation of the framework structure during the grasp phase.

Chapter 5

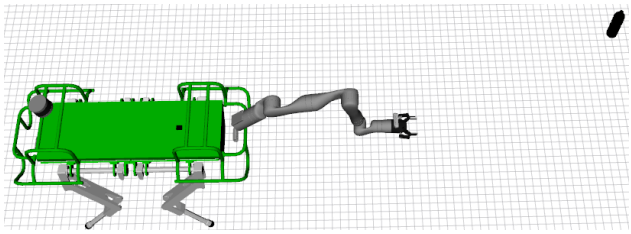
Simulation Results and Implementation Details

The developed framework has been tested and validated with simulations. The robot used in these simulations is IIT's HyQ [6], which is an hydraulic quadruped that weights 90 kg. In addition, it is equipped with a 7-DoF Kinova Gen3 [38] manipulator. Each part of the framework is tested, starting from the decoupled approach and ending with the grasping phase. This order is the same used for the sections of this chapter. All the simulations are perfomed using Gazebo and RViz as visualizer. The last section of this chapter 5.5 concludes providing implementation details of the framework.

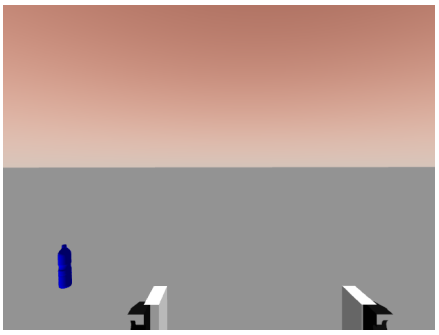
5.1 Decoupled Approach Simulations

In the previous chapter the decoupled approach has been described. To show that the proposed Visual Servoing approach works, two set of simulations are presented with different conditions: robot in stance (four legs on the ground) and robot trotting in place. During these simulations, the used reference generators are: "Desired Position Arm Wrist" and "Visual Servoing Arm EE". In other words, the first joints bring the wrist to a desired Cartesian position, while the last three joints are controlled by Visual Servoing. In that case the desired Cartesian position of the wrist is the home one in Fig.(4.4b). Regarding the simulations, the desired object is a bottle, since the pretrained models of YOLOv5 [37] are able to detect it with high accuracy. In particular, two tests are performed.
Regarding the target object, this is positioned on the ground on the left of the robot. In that way there is an error both for the x and y pixel coordinates expressed in the projection plane, since the object is on the bottom left of the image. The starting

conditions are shown in Fig.(5.1).



(a) External view of the starting condition. The arm is in home posture and the object on the left of the robot.



(b) Eye-in-Hand camera view. The target object is in the bottom left.

Figure 5.1: Starting condition of the decoupled approach simulations.

The results of the two simulations are in Fig.(5.2) and Fig.(5.3).

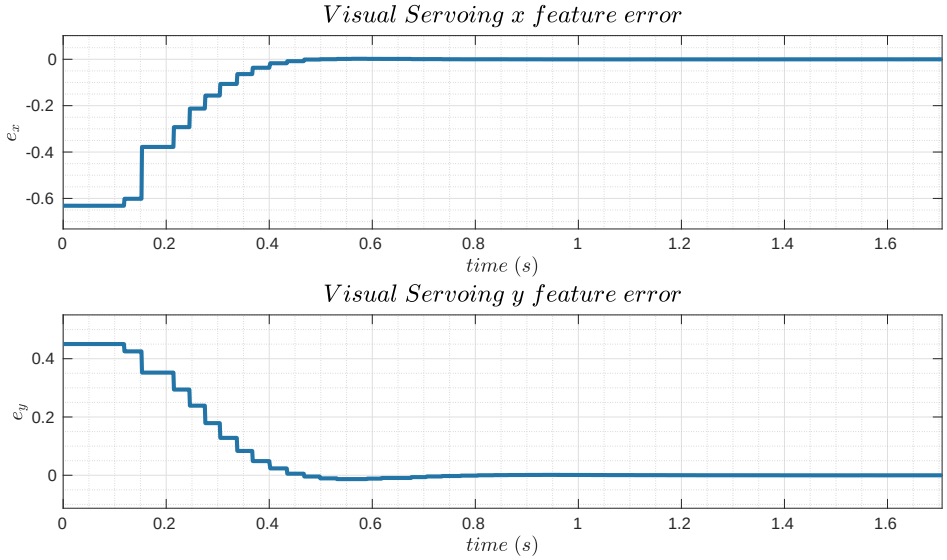


Figure 5.2: Results of decoupled approach with base steady.

Observing these plots, it is possible to note that when the base is steady the error crosses the zero earlier than when the base is trotting. In the latter case there are also oscillations around zero after that point is reached. The higher time needed to bring the error to zero and the oscillations, are due to the disturbances generated by the trot. However the amplitude of the oscillations is small. That means the object remains always close to the image center. In other words, our approach is able to don't lose the object even when the base is trotting.

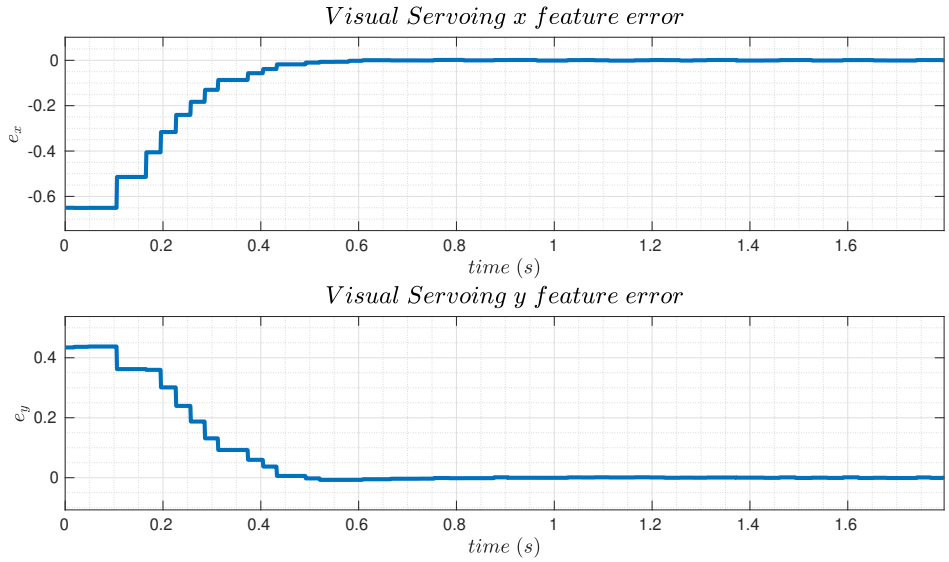


Figure 5.3: Results of decoupled approach with base trotting.

5.2 Search Phase Simulations

During the search phase, different motions are performed by the robot. In particular the first motion performed by the arm is a circular trajectory with the wrist. The other two motions are the circular trajectories with the arm's end-effector. Regarding the base, it performs a rotation of π around the yaw axis. In absence of target object, the overall sequence of these motions is the following:

1. Reach left limit on circle with the wrist
2. Move back and forward on the circle with the end-effector
3. Reach right limit on circle with the wrist
4. Move back and forward on the circle with the end-effector
5. Rotate the base around the yaw of π

In order to validate that phase a simulation with no objects is performed. That allows the quadruped manipulator to execute all the pipeline. The results are illustrated in Fig.(5.15) and Fig.(5.16). As it is possible to observe from these plots, wrist and end-effector perform circular trajectories, as previously explained in section 4.3.1. Regarding the base, to represent its heading it is used the base of the arm pointing

in the positive x axis of the base. That because it is fixed in the front of the base and aligned with the center of the base. Regarding the base, it is possible to observe that the π rotation around the yaw axis, is performed correctly. The sequence of simulation frames of that phase is depicted in Fig.(5.4).

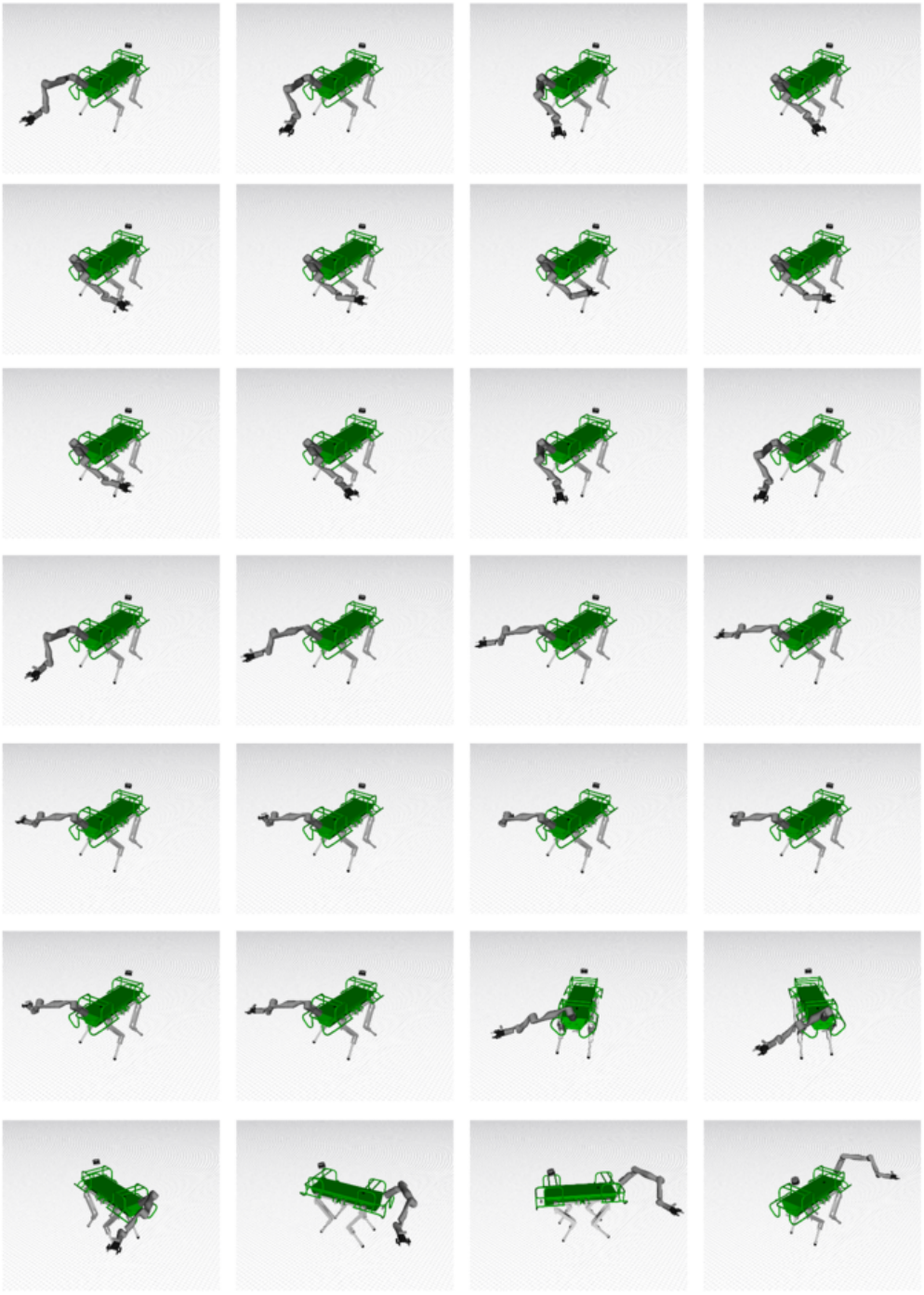


Figure 5.4: Sequence of the search phase in simulation.

5.3 Approach Phase Simulations

The approach phase is divided into two parts. The first one is the alignment with the object. Instead, the second part is about moving the robot closer to the object until a desired distance, on the xy plane, between end-effector and object has been reached. Starting from the alignment, it is possible to divide it into two motions. The first one is to move the wrist to the position on the circle which is intersected by the line connecting the base and the object. The other motion is to rotate the base in order to align its heading to the object. These motions have already been validated in the previous sections, 5.1 and 5.2. Therefore, to prove the alignment a sequence of simulation frames is used. In particular, two simulations are performed. The difference between the simulations is the position of the object. In particular, in the first simulation the object is inside the limits of the wrist circle. Hence, in that simulation the wrist can directly reach the desired position, as is shown in Fig.(5.5)).

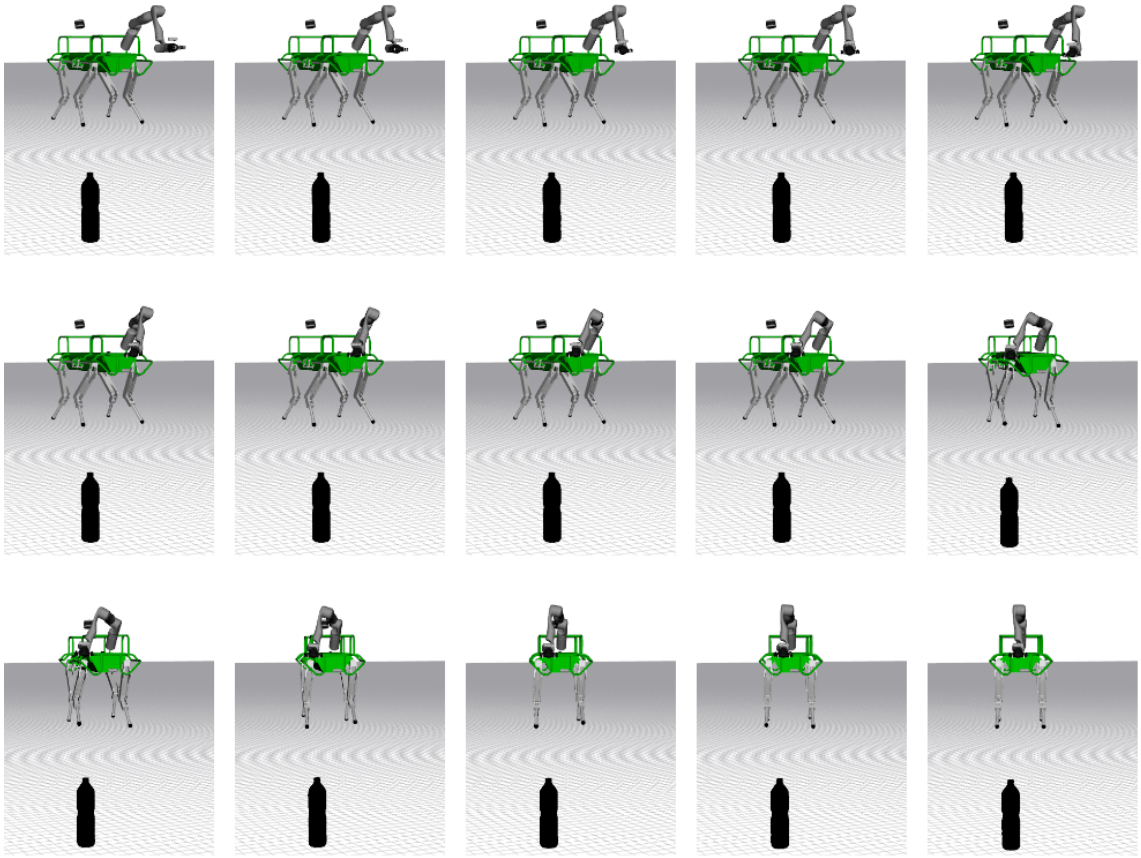


Figure 5.5: Sequence of the alignment in approach phase with object inside wrist limits.

Instead, in the second simulation, the object is outside the limits of the wrist circle. In that case the wrist goes first to the left limit, since the object is in that direction. After the base starts to rotate. During that rotation, once the desired point enters in the limits of the wrist, the latter reaches it. The sequence which describes this second test is in Fig.(5.6).

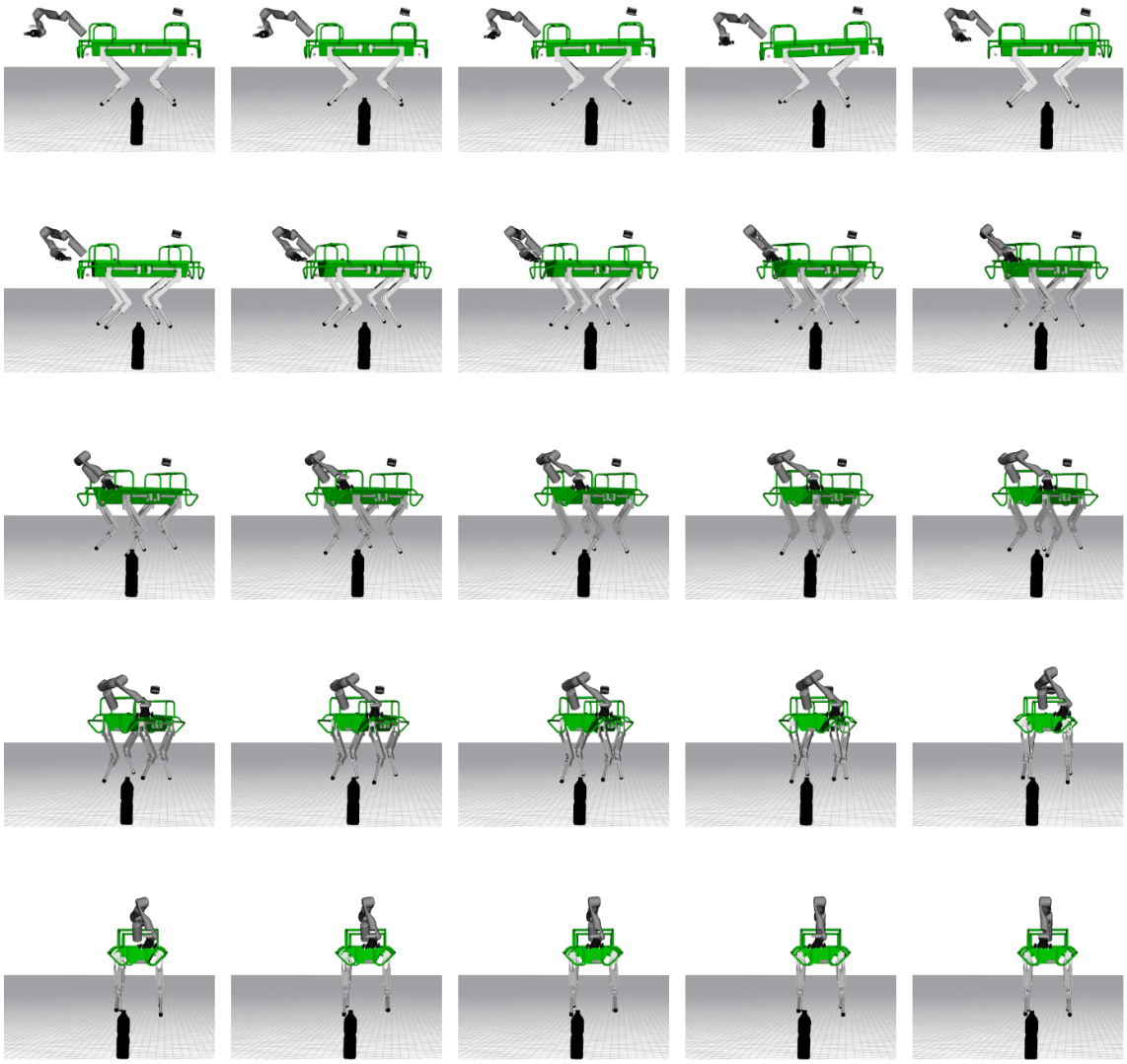


Figure 5.6: Sequence of the alignment in approach phase with object outside wrist limits.

Instead, in the second simulation the object moves two times, one time to the left and the other to the right. As explained in Section 4.3.2, to realign to the object the yaw error of the end-effector is used. The consequence of that is the behavior of the base which follows the end-effector. It is possible to observe that in Fig.(5.8).

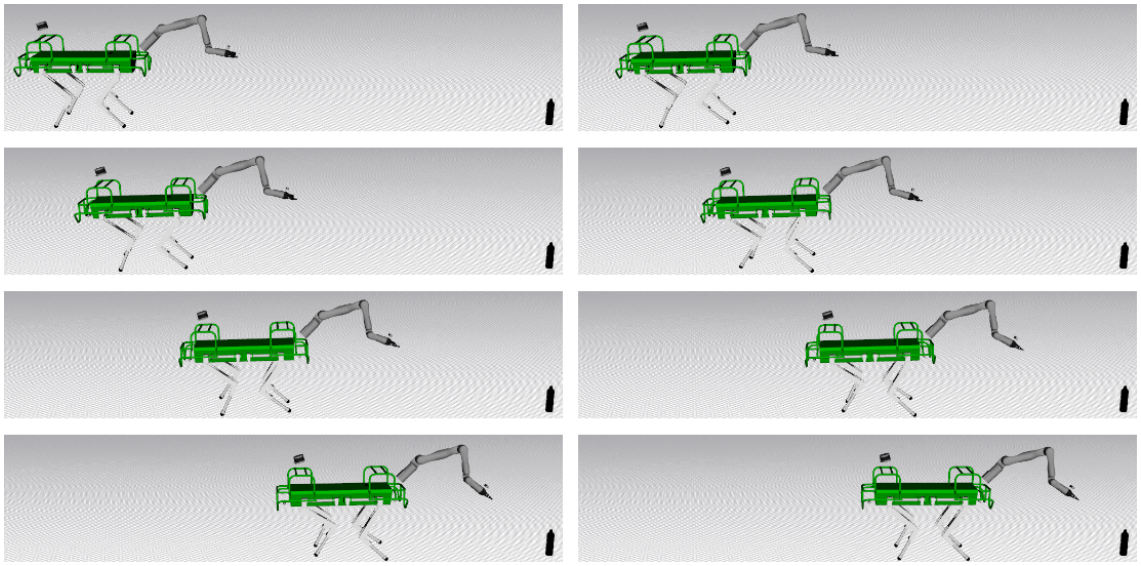


Figure 5.7: Sequence of the robot getting closer to steady object in approach phase.

Regarding the second part of the approach phase, it is also composed of two motions. The first is to bring the robot forward, keeping the wrist in home position and using visual servoing to maintain the object centered in the camera. The second one is to realign the base rotating around yaw if the object moves. Also for that part, sequence of simulation frames are used. The reason is that the new motion, moving forward the robot to reach a desired distance, does not need plots to be validated. Also in this case are used two different simulations. This time, the initial position of the object does not change. In both simulations, the desired distance between the end-effector and the object is set to 60 centimeters. More precisely, in the first simulation the object remains steady. That means the robot can go straight because it does not have to realign with the object. This behavior is shown in Fig.(5.7).

5.4 Grasp Phase Simulations

During the grasp phase the robot performs several motions. It is possible to divide these motions into four parts. The first one is to bring the wrist closer to the base and as close as possible to the height of the grasping position. The second part is to get closer to the object to get it into the workspace of the arm. The third one is to pitch the base to enlarge the workspace of the arm in the direction of the grasping position. Lastly, the fourth part is to grasp the object bringing the end-effector to the grasping pose. All these parts can be validated with sequences of simulation frames.

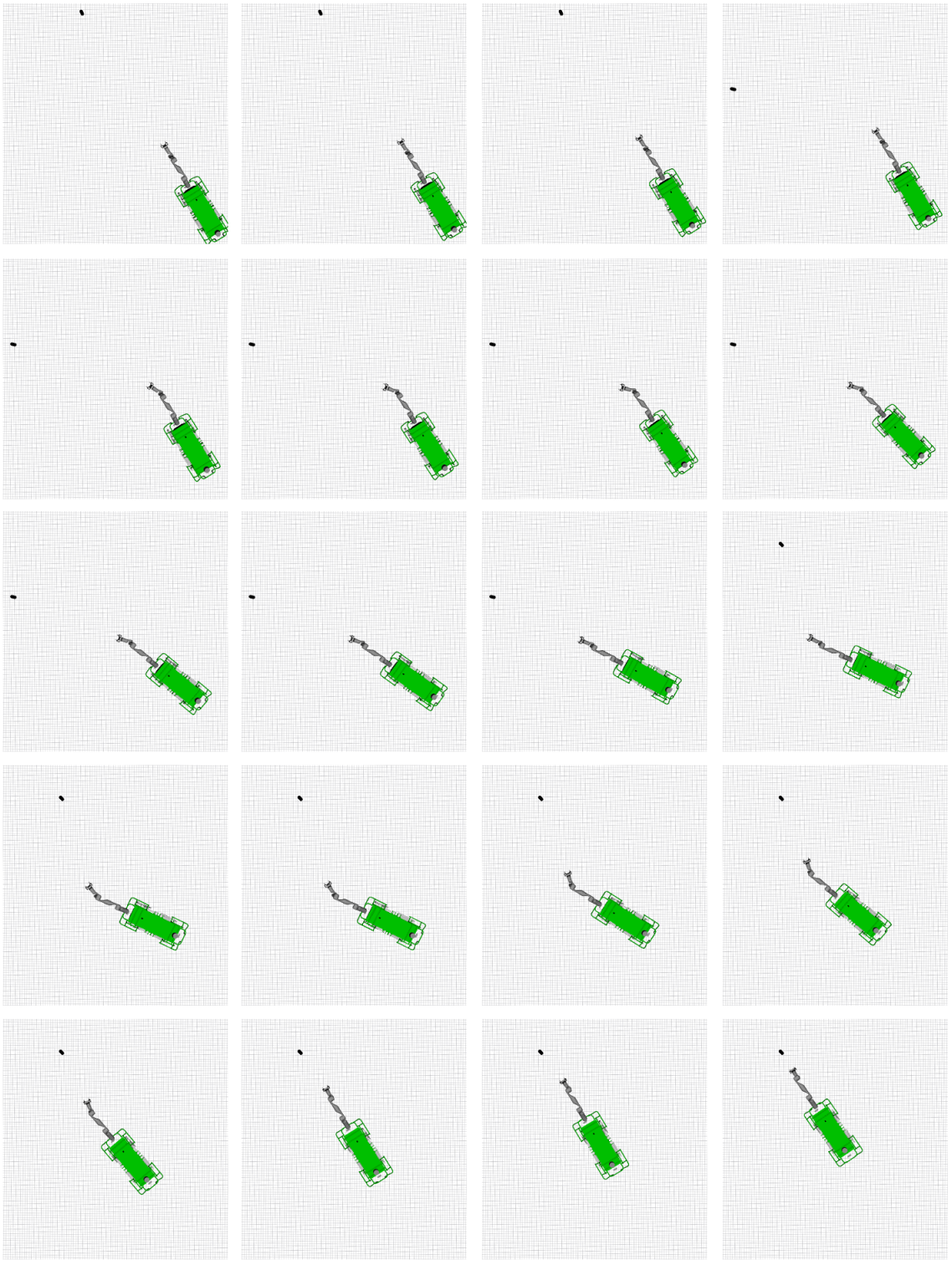


Figure 5.8: Sequence of the robot getting closer to changing position object in approach phase.

Regarding the height adjustment for the wrist, in this simulation the object is on the ground so the grasping position is below the base of the robot. The associated sequence is represented in Fig.(5.9).

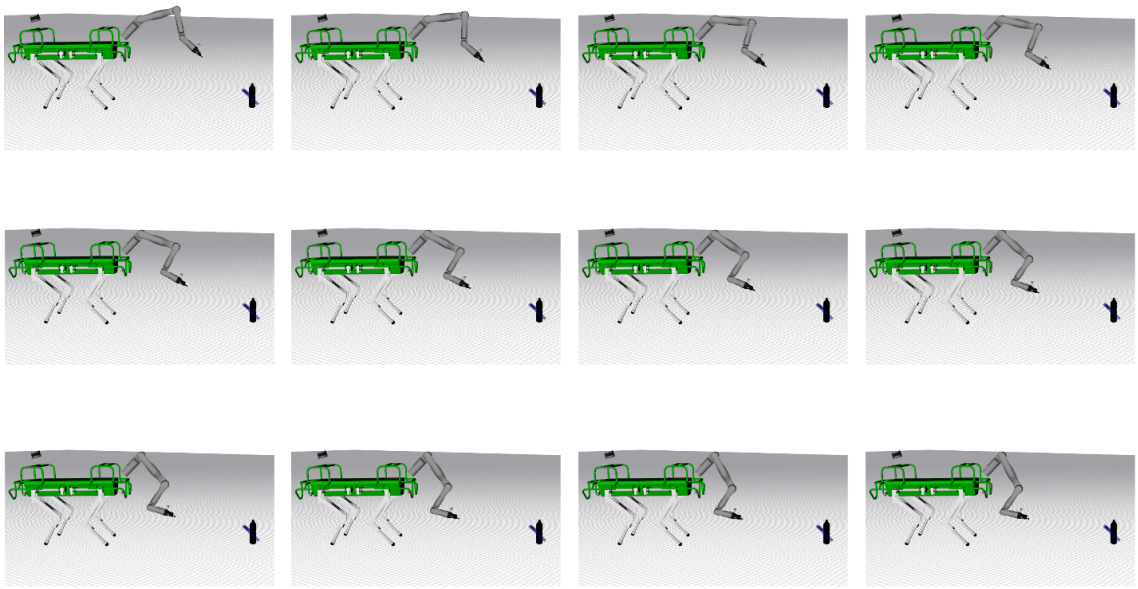


Figure 5.9: Sequence of wrist height adjustment in grasp phase.

Concerning the part which brings the robot closer to the object, it is similar to the one in the approach phase. One difference is the distance measured, since it is the one from the center of the workspace of the arm to the object. The other difference is the desired position for the wrist. Indeed the achieved height is maintained by the arm. The sequence for that part is in Fig.(5.10).
Before grasping the object, there is the third part of this phase. It is the adjustment of the pitch of the quadruped. Again, in this simulation the object is on the floor so the rotation around the pitch is positive. During that motion the wrist is moved to maintain the height. That part is illustrated in Fig.(5.11).
It is time to perform the grasp. The last part of the pipeline. That is not the scope of our framework, since it is an open problem in robotics. For that reason the grasping position is computed with an external module GPD [36]. The grasping is performed in open loop. The reason is the time required by the module to compute the grasp pose and also because the camera on the wrist loses the object when it is very close to it. Despite that, also that part is validated via a simulation. The frames of the latter are depicted in Fig.().

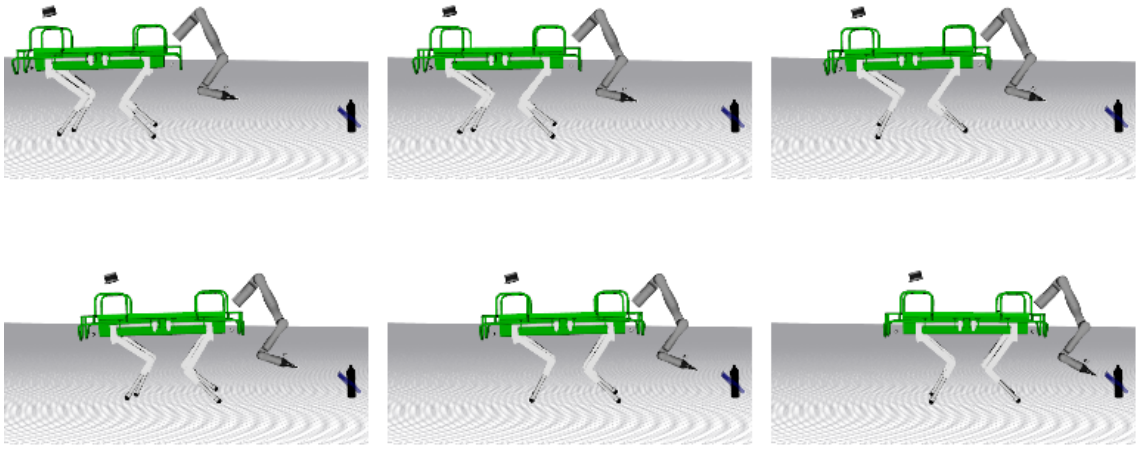


Figure 5.10: Sequence of robot moving to allow the object to be in the arm workspace in grasp phase.

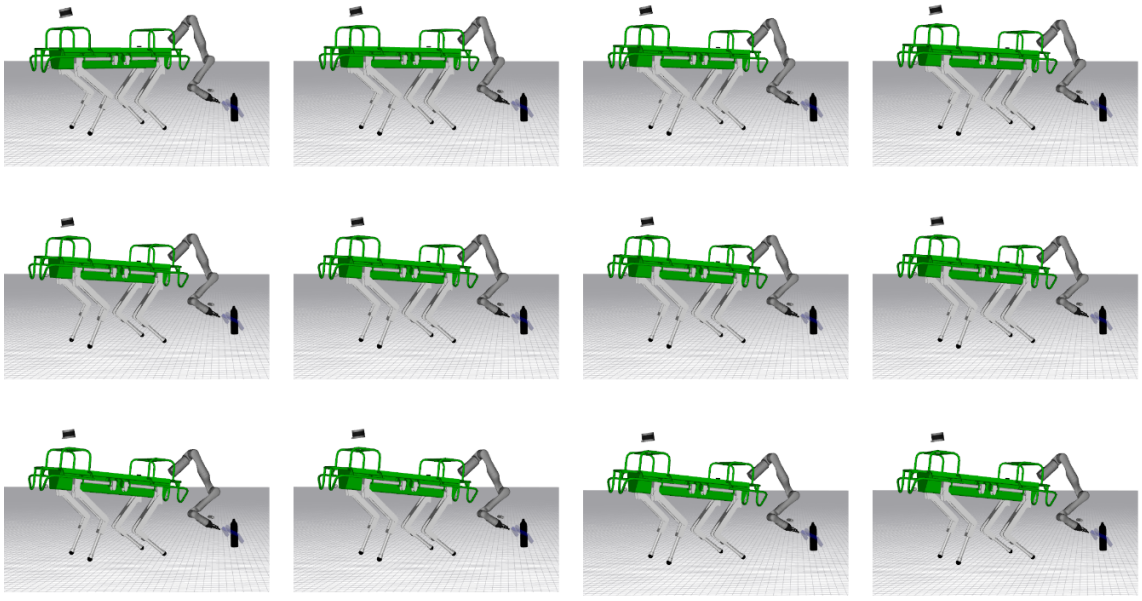


Figure 5.11: Sequence of base pitch adjustment in grasp phase.

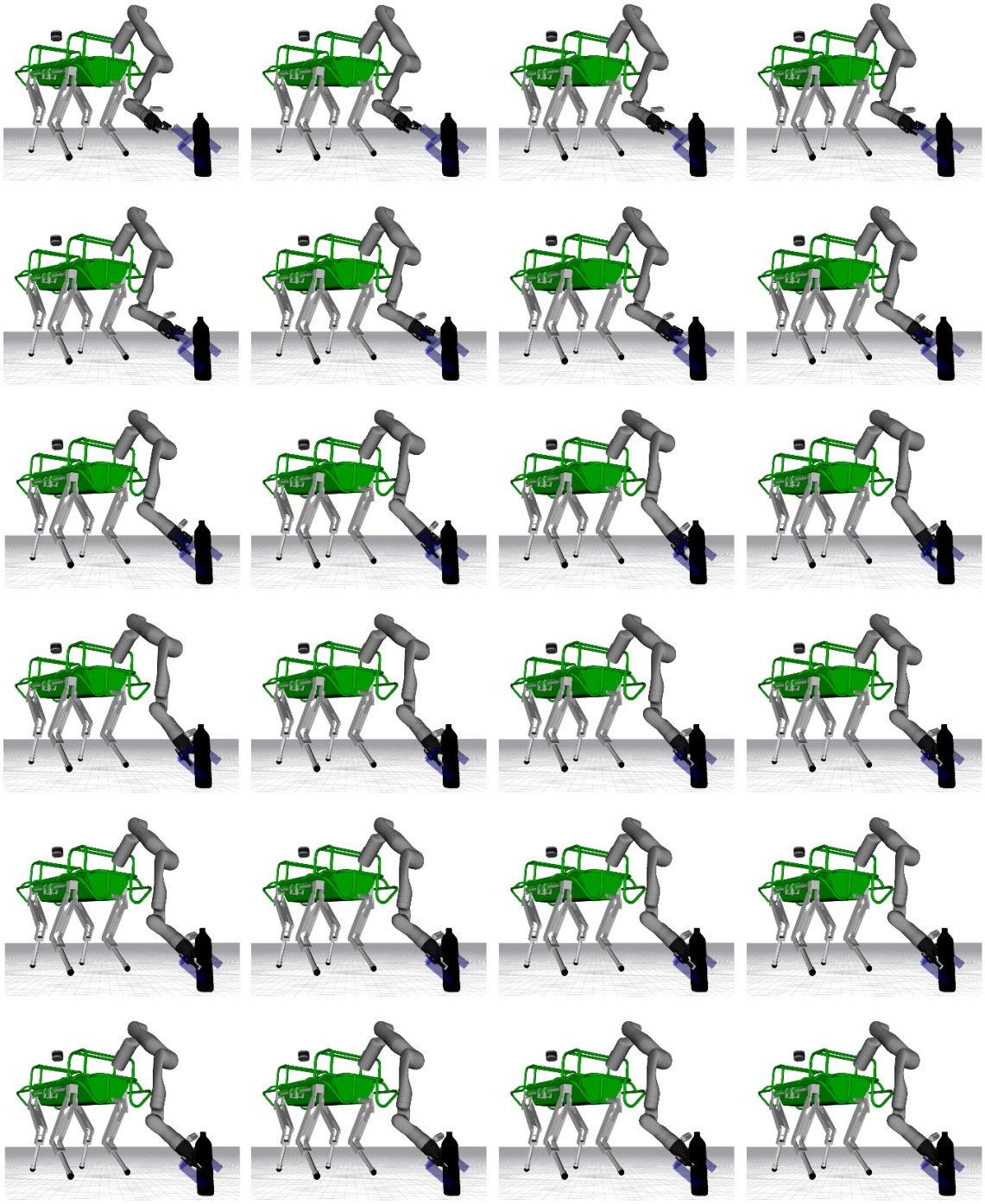


Figure 5.12: Sequence of robot grabbing the object in grasp phase.

5.5 Implementation Details

The developed framework is based on the Robot Operating System (ROS) and it is structured into two packages. The first one was named "acquire_and_detect" and it is written in Python. Its function is to extract images from the camera, to perform the inference with YOLOv5's object detection neural network [37] and to take the bounding box which has the name of the target object. In case there are several bounding boxes relative to the same type of object, the one with highest accuracy is selected. That package also uses "depth_image_proc" [41], as external dependency, to transform a depth image into a point cloud, which is sent then to GPD [36] for the computation of the grasping pose. The second main package of the framework is structured into three classes, all written in C++. The first class is called "SearchApproachGrasp", and contains all the arm controllers and the reference generators, apart from the visual servoing one. Indeed the latter it is implemented in the second class, called "VisualServoing", which computes the twist for the camera using the features from the bounding box of the target object. The third class is called "SAGBehaviorTree" and implements the Behavior Tree of the framework using the library BehaviorTree.CPP [4]. The framework is structured in that way since the two packages are meant to run in two different environment. In particular acquire_and_detect should be executed on a machine which has the possibility to use CUDA [42], such as an NVIDIA Jetson [43]. Instead, regarding the second main package, SearchApproachGrasp has to be included in the code which contains the control loop of the robot. This is done because SearchApproachGrasp offers three functions, to compute the torques for the arm, the twist for the base and the state of the gripper (open or close). These three functions are meant to be called at each iteration of the control loop. In details, each of the three functions execute internally another function in order to use the right control modules. These last three functions are decided by the Behavior Tree depending on the current state of the robot. The graphical representation of the just explained implementation structure can be found in Fig.(5.13). Instead, the Behavior Tree including all the three phases is represented in Fig(5.14).

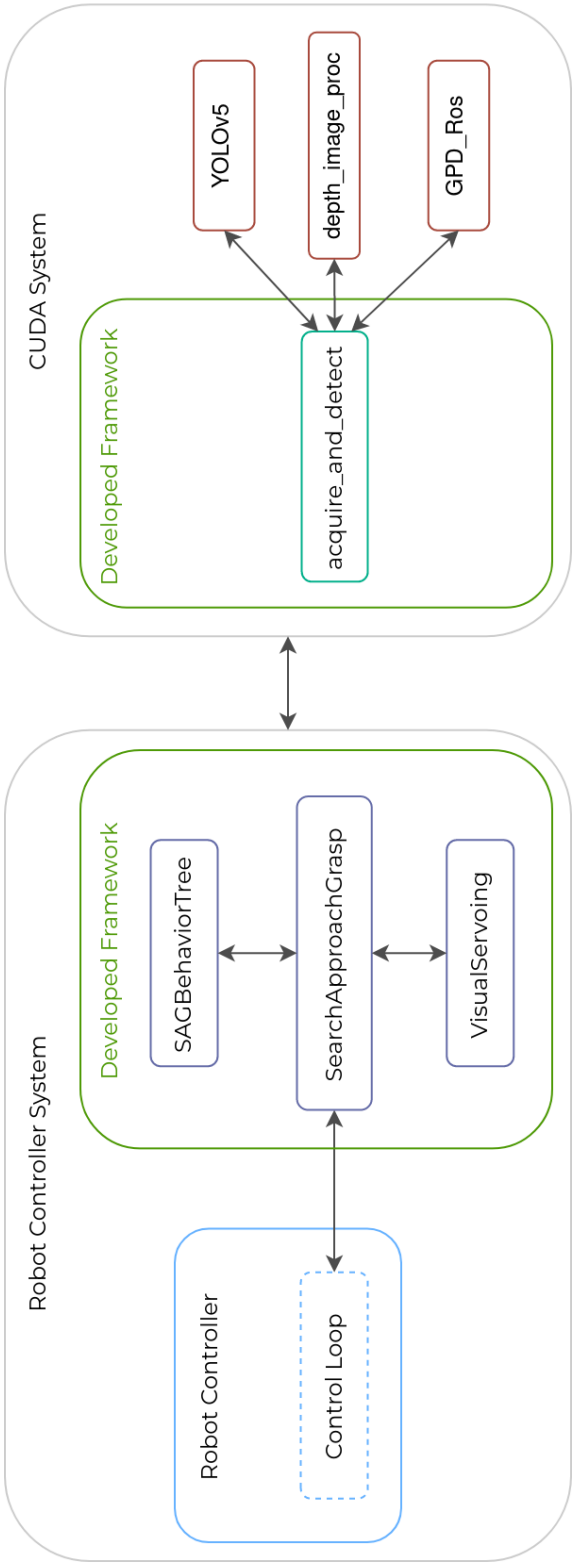


Figure 5.13: Graphical representation of the implementation structure.

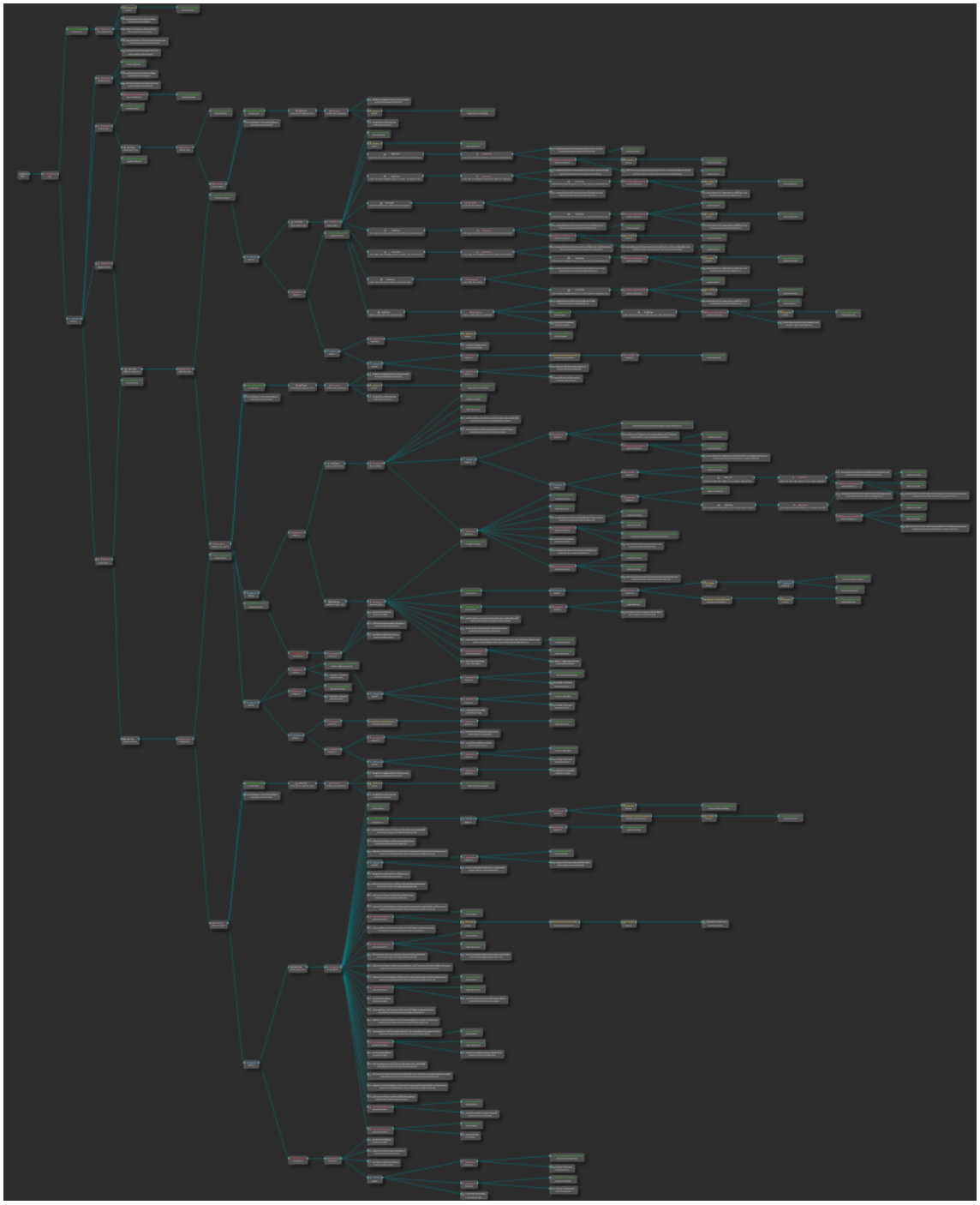


Figure 5.14: Behavior Tree of all the SAG pipeline.

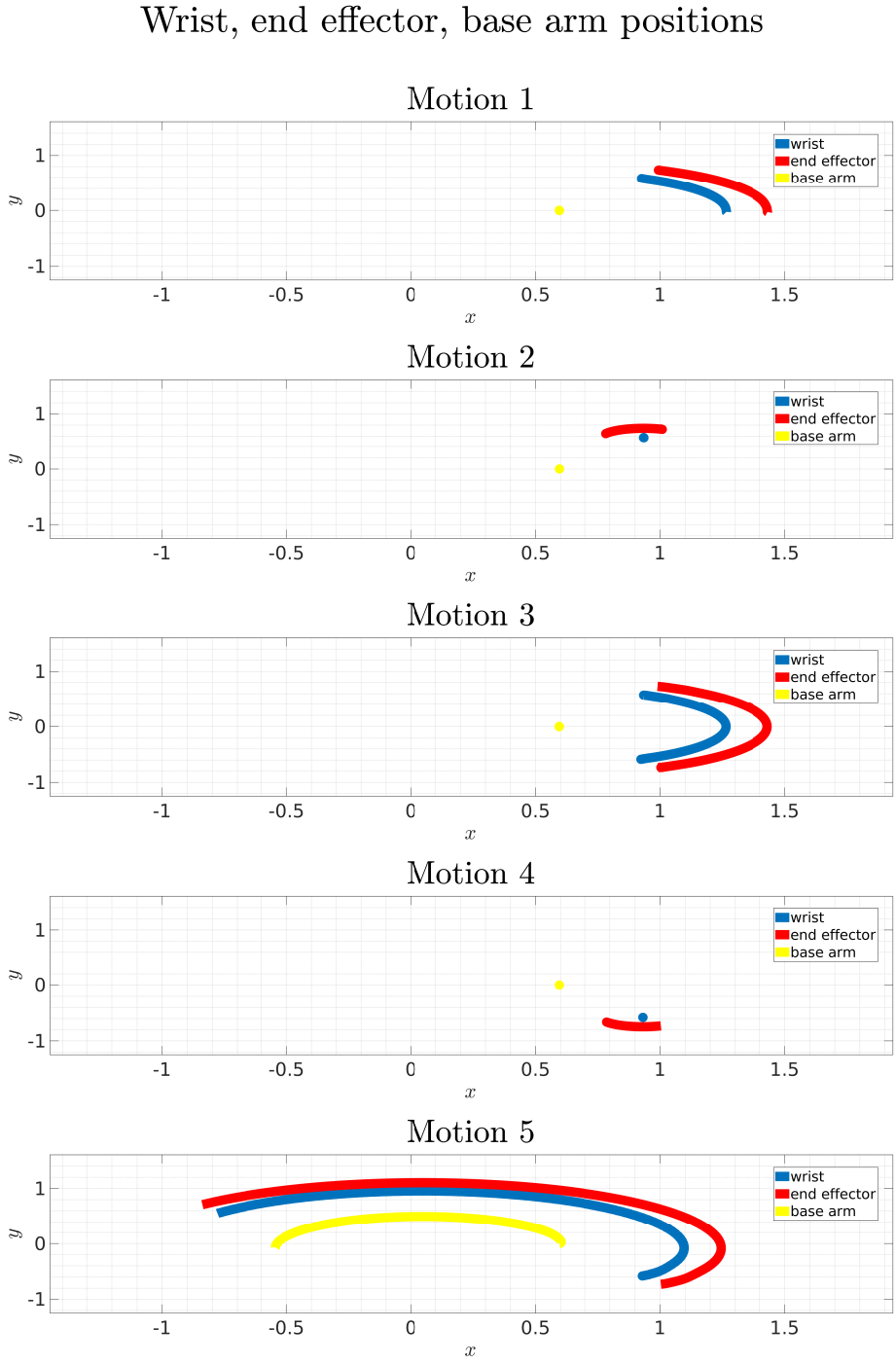


Figure 5.15: Graphical representation of wrist, end-effector and base arm positions during each motion.

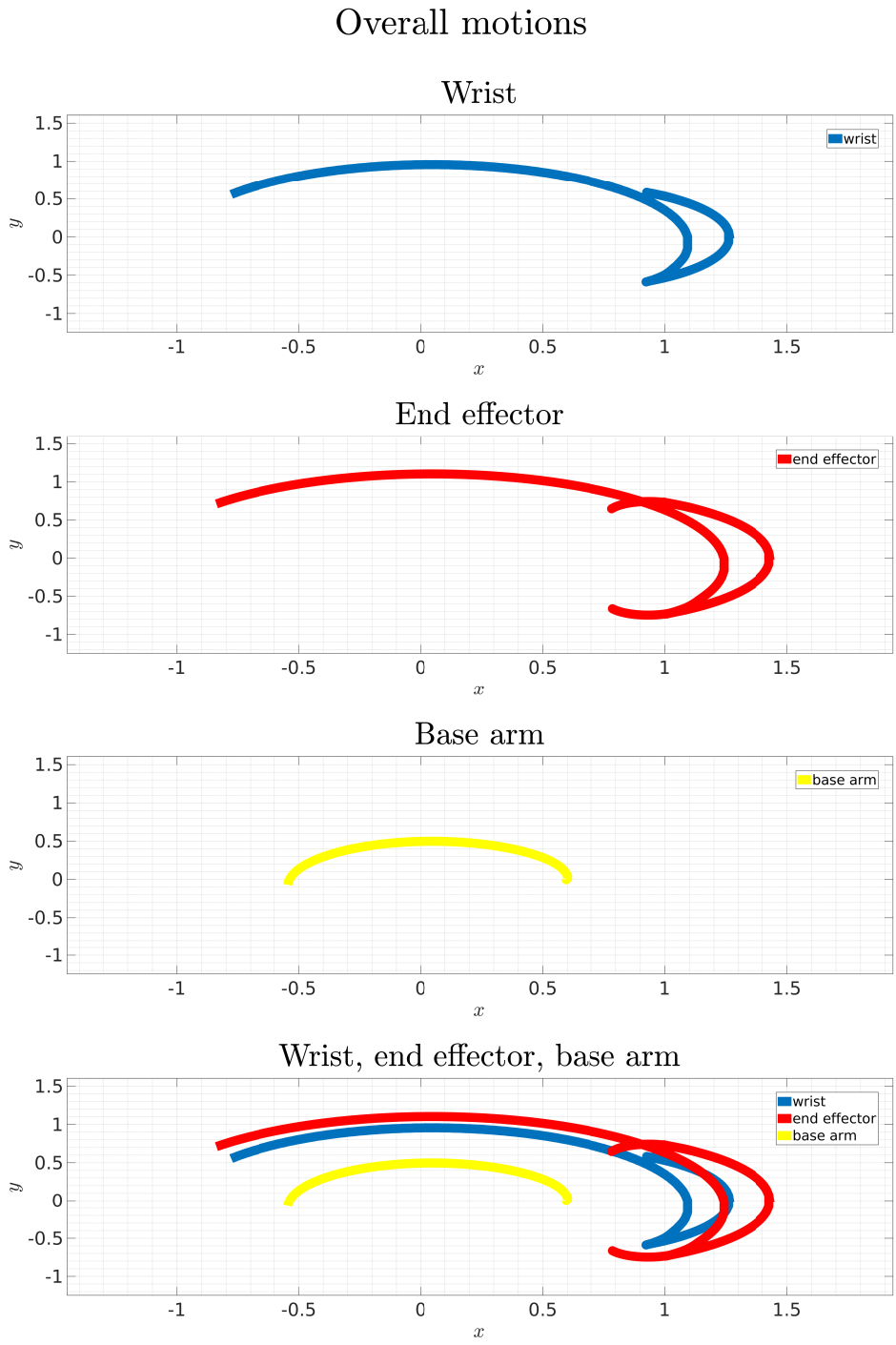


Figure 5.16: Graphical representation of wrist, end-effector and base arm positions during all the motions.

Chapter 6

Conclusion

This thesis presented a framework to perform the search, approach and grasp of an object using a quadruped manipulator. The two subsystems, namely base and robotic arm, are controlled in a decoupled fashion. Hence, the base is used to locomote the robot close to the object, when it is not reachable by the arm. A previously introduced locomotion stack is employed constituted by a trunk controller, that maps a desired wrench on the base to ground reaction forces, and a reference generator for the swing legs. For the arm, the higher inertia links are used to guide the arm's wrist to a desired position, and visual feedback is used to control the arm's end-effector using an RGBD camera. When the base is commanded to move, a dynamic gait is chosen, and visual feedback is shown to mitigate the disturbance of the base at the arm's end-effector. For the grasping pose, an off-the-shelf package is employed. Here, a straight line trajectory is used to drive the arm's end-effector from the object's proximity to the desired grasping pose. During this phase, given the limitation of the camera in terms of minimum depth, only proprioceptive information are processed. Managing the high number of actions for the goal of grasping an object, a Behavior Tree has been used. The developed framework has been validated performing simulations on the 90kg IIT's HyQ robot, equipped with a 7 DoFs Kinova Gen3 robotic arm. The framework is not limited to only quadrupedal based platforms, but it can be further extended to other type of mobile base and legged systems.

6.1 Future Works

Even if the framework is already validated with simulations, implementing it on a real robot would be a further validation of the adopted strategy. In addition to that, the pipeline could be tested with different types of objects having different

sizes and shapes. Regarding the grasping, a closed loop control strategy would benefit the overall performances, in terms of grasping accuracy and task repeatability. Additionally, another camera can be mounted on the robot for increasing the field of view when the camera at the arm's end-effector is occluded by the object. When the object exits the camera's view, the proposed strategy brings the arm to a default configuration, and start a new search phase. Future work could aim at improving this heuristics, considering past memory of the object location. Finally, to avoid collisions with the objects or with other entities interacting with the arm, an obstacle detection and avoidance module would benefit the safety of the overall system.

Bibliography

[1] S. Ge and G. Sartoretti, "Bio-inspired Visual Servoing for a Legged Robot NUS MSc Project (2019/2020)," p. 8.

[2] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," Apr. 2018, arXiv:1804.02767 [cs]. [Online]. Available: http://arxiv.org/abs/1804.02767

[3] R. Newbury, M. Gu, L. Chumbley, A. Mousavian, C. Eppner, J. Leitner, J. Bohg, A. Morales, T. Asfour, D. Kragic, D. Fox, and A. Cosgun, "Deep Learning Approaches to Grasp Synthesis: A Review," Jul. 2022, arXiv:2207.02556 [cs]. [Online]. Available: http://arxiv.org/abs/2207.02556

[4] "BehaviorTree.CPP." [Online]. Available: https://www.behaviortree.dev/

[5] ANYbotics, "ANYmal C Legged Robot – The Next Step in Robotic Industrial Inspection," 2019. [Online]. Available: https://www.youtube.com/watch?v=_ffgWvdZyvk

[6] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, "Design of HyQ – a hydraulically and electrically actuated quadruped robot," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 225, no. 6, pp. 831–849, Sep. 2011. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0959651811402275

[7] "Mini cheetah is the first four-legged robot to do a backflip." [Online]. Available: https://news.mit.edu/2019/mit-mini-cheetah-first-four-legged-robot-to-backflip-0304

[8] C. Semini, V. Barasuol, M. Focchi, C. Boelens, M. Emara, S. Casella, O. Villarreal, R. Orsolino, G. Fink, S. Fahmi, G. A. Medrano-Cerda,

D. Caldwell, D. Sangiah, J. Lesniewski, K. Fulton, M. Donadon, and M. Baker, "Brief introduction to the quadruped robot HyQReal," Oct. 2019, publisher: Zenodo. [Online]. Available: https://zenodo.org/record/4782613

[9] "Spot® - The Agile Mobile Robot." [Online]. Available: https://www.bostondynamics.com/products/spot

[10] P. Fankhauser, "Perceptive Locomotion for Legged Robots in Rough Terrain," Ph.D. dissertation, Jan. 2018.

[11] G. Heppner, T. Buettner, A. Roennau, and R. Dillmann, "Versatile - High Power Gripper for a Six Legged Walking Robot," Jul. 2014.

[12] C. D. Bellicoso, K. Krämer, M. Stäuble, D. Sako, F. Jenelten, M. Bjelonic, and M. Hutter, "Alma - articulated locomotion and manipulation for a torque-controllable robot," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8477–8483.

[13] Istituto Italiano di Tecnologia, "Robot Teleoperativo – IIT's new advanced robotic teleoperation system, in collaboration with INAIL," 2021. [Online]. Available: https://www.youtube.com/watch?v=66ZMUaBLjaM

[14] O. Villarreal, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini, "Mpc-based controller with terrain insight for dynamic legged locomotion," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 2436–2442.

[15] L. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *International Journal of Humanoid Robotics*, vol. 2, no. 04, pp. 505–518, 2005.

[16] C. D. Bellicoso, C. Gehring, J. Hwangbo, P. Fankhauser, and M. Hutter, "Perception-less terrain adaptation through whole body control and hierarchical optimization," in *IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 558–564.

[17] M. Risiglione, J.-P. Sleiman, M. V. Minniti, B. Çizmeci, D. Dresscher, and M. Hutter, "Passivity-based control for haptic teleoperation of a legged manipulator in presence of time-delays," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 5276–5281.

[18] B. U. Rehman, M. Focchi, J. Lee, H. Dallali, D. G. Caldwell, and C. Semini, "Towards a multi-legged mobile manipulator," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 3618–3624.

[19] Y. Ma, F. Farshidian, T. Miki, J. Lee, and M. Hutter, "Combining learning-based locomotion policy with model-based manipulation for legged mobile manipulators," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2377–2384, 2022.

[20] J.-R. Chiu, J.-P. Sleiman, M. Mittal, F. Farshidian, and M. Hutter, "A Collision-Free MPC for Whole-Body Dynamic Locomotion and Manipulation," Feb. 2022, arXiv:2202.12385 [cs]. [Online]. Available: http://arxiv.org/abs/2202.12385

[21] S. Zimmermann, R. Poranne, and S. Coros, "Go fetch! - dynamic grasps using boston dynamics spot with external robotic arm," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 4488–4494.

[22] B. Griffin, "Task-Focused Few-Shot Object Detection for Robot Manipulation," Jan. 2022, arXiv:2201.12437 [cs]. [Online]. Available: http://arxiv.org/abs/2201.12437

[23] M. Arduengo, C. Torras, and L. Sentis, "Robust and adaptive door operation with a mobile robot," *Intelligent Service Robotics*, May 2021. [Online]. Available: http://dx.doi.org/10.1007/s11370-021-00366-7

[24] C. Dario Bellicoso, C. Gehring, J. Hwangbo, P. Fankhauser, and M. Hutter, "Perception-less terrain adaptation through whole body control and hierarchical optimization," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, 2016, pp. 558–564.

[25] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, "Anymal - a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 38–44.

[26] "Development of Human Support Robot as the research platform of a domestic mobile manipulator | ROBOMECH Journal | Full Text." [Online]. Available: https://robomechjournal.springeropen.com/articles/10.1186/s40648-019-0132-3

[27] A. Bicchi and V. Kumar, "Robotic grasping and contact: a review," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 1, Apr. 2000, pp. 348–353 vol.1, iSSN: 1050-4729.

[28] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis—a survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014.

[29] M. Mateas and A. Stern, "A behavior language for story-based believable agents," *IEEE Intelligent Systems*, vol. 17, no. 4, pp. 39–47, Jul. 2002. [Online]. Available: http://ieeexplore.ieee.org/document/1024751/

[30] D. I. B. March 11 and 2005, "GDC 2005 Proceeding: Handling Complexity in the *Halo 2* AI," Mar. 2005, section: programming. [Online]. Available: https://www.gamedeveloper.com/programming/gdc-2005-proceeding-handling-complexity-in-the-i-halo-2-i-ai

[31] G. Florez-Puga, M. Gomez-Martin, P. Gomez-Martin, B. Diaz-Agudo, and P. Gonzalez-Calero, "Query-Enabled Behavior Trees," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, no. 4, pp. 298–308, Dec. 2009. [Online]. Available: http://ieeexplore.ieee.org/document/5325892/

[32] P. Ogren, "Increasing Modularity of UAV Control Systems using Computer Game Behavior Trees," in *AIAA Guidance, Navigation, and Control Conference*. Minneapolis, Minnesota: American Institute of Aeronautics and Astronautics, Aug. 2012. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2012-4458

[33] J. A. Bagnell, F. Cavalcanti, L. Cui, T. Galluzzo, M. Hebert, M. Kazemi, M. Klingensmith, J. Libby, T. Y. Liu, N. Pollard, M. Pivtoraiko, J.-S. Valois, and R. Zhu, "An integrated system for autonomous robotics manipulation," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vilamoura-Algarve, Portugal: IEEE, Oct. 2012, pp. 2955–2962. [Online]. Available: http://ieeexplore.ieee.org/document/6385888/

[34] M. Mittal, D. Hoeller, F. Farshidian, M. Hutter, and A. Garg, "Articulated Object Interaction in Unknown Scenes with Whole-Body Mobile Manipulation," Mar. 2022, arXiv:2103.10534 [cs]. [Online]. Available: http://arxiv.org/abs/2103.10534

[35] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *IEEE Robotics Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[36] A. t. Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp Pose Detection in Point Clouds," Jun. 2017, arXiv:1706.09911 [cs]. [Online]. Available: http://arxiv.org/abs/1706.09911

[37] "ultralytics/yolov5," Oct. 2022, original-date: 2020-05-18T03:45:11Z. [Online]. Available: https://github.com/ultralytics/yolov5

[38] Kinova gen3 7-dof manipulator arm. [Online]. Available: https://www.kinovarobotics.com/product/gen3-robots#Product__resources

[39] V. Barasuol, J. Buchli, C. Semini, M. Frigerio, E. R. De Pieri, and D. G. Caldwell, "A reactive controller framework for quadrupedal locomotion on challenging terrain," 5 2013, pp. 2554–2561.

[40] M. Focchi, A. D. Prete, I. Havoutis, R. Featherstone, D. G. Caldwell, and C. Semini, "High-slope terrain locomotion for torque-controlled quadruped robots," *Autonomous Robots*, vol. 41, pp. 259–272, 2017.

[41] "depth_image_proc - ROS Wiki." [Online]. Available: http://wiki.ros.org/depth_image_proc

[42] R. S. Dehal, C. Munjal, A. A. Ansari, and A. S. Kushwaha, "Gpu computing revolution: Cuda," in *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, 2018, pp. 197–201.

[43] "Jetson Modules, Support, Ecosystem, and Lineup," Oct. 2020. [Online]. Available: https://developer.nvidia.com/embedded/jetson-modules