Chapter 4

Control architecture for legged and climbing robots

"We work hard today to make something happen tomorrow"

Carlitos Páez

As detailed in Section 2 of this dissertation, in the literature, it is possible to find references concerning control architectures for legged robots. In most of them, the number and arrangement of their components is previously defined, and it cannot be changed, limiting in this way the range of applications and target environments. In this chapter, a control architecture for a legged robot with a variable number of legs is described. The architecture is baptized with the name MoCLORA, which comes from Modular Climbing-and-Legged Robotic Organism Architecture. Although the control architecture is generic, a specific implementation is presented for ROMERIN with some peculiar components. Furthermore, the virtualization and simulation of the physical robot are presented.

4.1 MoCLORA

The Modular Climbing-and-Legged Robotic Organism Architecture arises to control a robotic organism as defined previously. MoCLORA is implemented in C++ and uses ROS2 communication tools to share information between architecture components and devices. C++ with ROS2 has been used due to its wide diffusion at the robotic level. MoCLORA is designed for a general robotic organism composed of leg-shaped robots. In order to generate the architecture, some requirements and design guidelines are established:

- It seeks to imitate the behavior of animals, specifically those of legged insects.
- Modules support an intentional movement expressed by the body.
- The number of modules can be variable.

- The components of the robot can be replaced by virtual components (digital twin).

In summary, the control architecture focuses on controlling body position and velocity, to mimic the animal world, where individuals care about body movement, without thinking about control of one leg alone. The proposed architecture serves as the basis for the control of L&C robots with any morphology, and more components can be included in the general framework presented to improve the performance of the organism and to specialize the system for a specific robot.

The control architecture is illustrated in Figure 4.1 and has all the components required to be used in the control of a legged robot with a variable number of legs, both in simulation and with a physical machine.

In the upper part of the architecture, the Graphical User Interface (GUI) is found. This component communicates only with the CC, which is the core of the control architecture and is divided into three layers: HAL, Executive Level, and Scheduler. In the lower part, external devices are found. These devices, where the robotic organism is found, can be physical or simulated.

4.1.1 External physical and simulated devices

In the lower part of the architecture, N physical modules are available. They have the MCU which is responsible for low-level control and communication with external devices. In this generic architecture, a module with communication capabilities of any robot is valid.

Similarly to the physical modules, the simulation environment (which may be integrated within the Computer of Control, CC, or externally) includes the description of the world in which the robotic organism is described. The **Module Plugin** is a virtual realization of the firmware of a physical module. That is, it implements the same interface and behavior as the physical modules, so that the control of a simulated or virtual module is exactly the same. Similarly, the **Body plugin** implements the same behavior as the body components (pantilt, camera, IMU, ultrasonic sensor, etc.), which are wire or wireless connected to the CC.

4.1.2 HAL

The HAL, Hardware Abstraction Layer [Murray IV et al., 2005], isolates the modules from the controllers of the modules, that is, it provides a tool to hide complexity when systems become too difficult to work with efficiently. The module controllers send messages to the corresponding system and receive the relevant information from them, in such a way that the controller does not know whether it is moving a real module or a dummy module. The **Module Communicator** has the task of correctly route messages to make this information transmission work and transparent. In summary, the HAL level produces the result in such a way that higher levels do not need to take into account if messages are sent to modules of the robotic organism or modules of the digital twin.

Messages could arrive from a higher level, in which case they specify a command to be handled by a module, through ROS2 topics with the topic identifier *"/moduleName/-controlToModule"*. These messages are redirected to the appropriate module based on its communication information, typically the IP address and port. In the opposite direction,

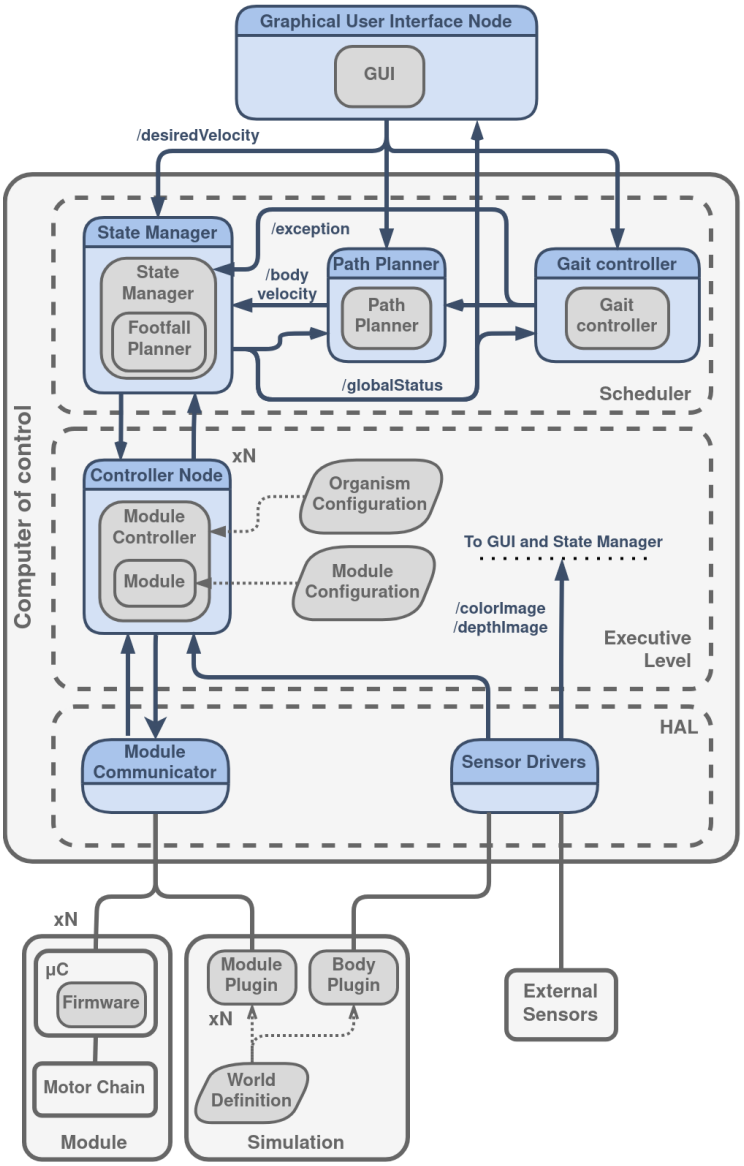


Figure 4.1: Generic reactive control architecture for legged robots. Rounded rectangles represent hardware devices, while dashed ones represent architecture levels. The dark gray boxes represent C++ objects. Parallelograms are configuration files. The blue containers are ROS2 nodes that use the objects to execute a routine. Gray lines represent inter-device communication, blue ones ROS2 topics for messages exchange (from publisher to subscriber), and dashed ones represent the use of a resource.

messages can arrive on any of the channels provided to inform about the current status of the corresponding module. They are redirected through ROS2 topics with the topic identifier *"/moduleName/moduleToControl"*.

In this layer, different **Sensor Drivers** can be located with the objective of isolating the sensors connection and protocols from higher layers. Any sensor driver can be found, but it is recommended to have at least an IMU for gravity vector detection and a camera for the environment visualization.

4.1.3 Executive Level

The Executive Level contains the components that control the modules at a higher level. It is made up of N **Controller Nodes**, created dynamically according to the configuration of the organism. Each one contains a **Module Controller**, which is responsible for computing the configuration of the modules, given the objective position of the body. It includes the **Module** object, which directly controls the components of a module, that is, actuators, sensors, and suction cup. It includes the computation of gravity compensation (Chapter 5), which can be closely related to the forward and inverse kinematics and dynamic of a module (FK, IK, FD and ID, respectively).

Thus, the **Module Controller** uses the tools provided by the **Module** object to control it according to the body motion requirements. **Module Controller** reads from the configuration file **Organism Configuration**, which establishes the number of available modules, their name, position with respect to the center of the body, IP address and port, and body details (maximum allowed speed, weight, and inertial data). On the other hand, the **Module** object reads from the **Module Configuration**, which details the kinematic and dynamic parameters of a specific module, its joints limits, the maximum velocity and acceleration of the TCP, the inverse kinematic calculation parameters, and the conversion parameters between normalized joints and actuator encoders (for example, for ROMERIN the joint position is given by the Dynamixel servomotors in an integer value that has to be transformed to degrees or radians).

Each **Controller Node** publishes the module status (called local status) to the higher level for its treatment in higher architecture levels through the topic *"/moduleName/status"*:

Local status:
- Name
- Module identifier
- TCP position
- Module COM
- Module mass [1]
- Estimated body position
- Gravity compensation torques
- Module status (Table 4.1)

Meanwhile, the **Controller Node** receives from the the **State Manager Node** the following commands to be executed:

[1]The module mass changes when the suction cup is attached to a surface because the mass of the gripping system is considered part of the environment and, therefore, is ignored.

Parameter	Description
Battery	Battery voltage (V)
Motor status [6]	Status of the module motors (good, overload, temperature error, etc.)
Motors position [6]	Actuators position (rad)
Turbine power	Current power throttle (%)
Suction cup force	Force generated by the negative pressure in the suction cup (N)
Swing phase completed	Flag to indicate that the swing phase has been successfully completed

Table 4.1: Module status structure.

Module command:
- Desired body velocity
- Gravity vector
- Estimated body position (generated with all the estimations)
- Reaction force position
- Reaction force value
- Swing phase turn
  - Target position of the swing phase

4.1.4 Scheduler

The higher level, named Scheduler, is responsible for generating a trajectory based on the user's commands and consequently deciding the most appropriate sequence of body movements and establishing the gait.

To do so, the **Path Planner** generates a speed profile as a consequence of the command received from the user. Given this profile, the **State Manager Node** sends commands to the lower level. Internally, the **State Manager** computes the desired body position to comply with the path, the estimation of the robot state, the COM of the entire robotic organism, and the next position where a module should step. One of the main duties of the **State Manager** is to take into account the static and gripping stability of the commanded positions of the robot organism based on the modules and body positions, direction of gravity and reaction forces computed in the suction cups.

Within the **State Manager**, the **Footfall Planner** selects the best position where the leg in the swing phase has to step in order to ensure safety and comfort. It can optimize a function, or generate a predefined sequence of movements as a Central Pattern Generator (CPG).

The **Gait Controller** is responsible for generating the swing/stance phases. It constantly checks the global status of the robot to detect uncomfortable positions according to the leg position, torque, and suction cup status (more details are explained in Section 6.4). When this situation is detected, the **Gait Controller** throws an exception to stop the motion and plan a new trajectoy or move one of the conflicting legs.

4.1.5 Graphical User Interface

A graphical user interface (GUI) is a digital interface in which the user interacts with graphical components such as icons, buttons, and menus. In a GUI, the visuals displayed in the user interface convey information relevant to the user, as well as actions that they can take. The GUI typically controls the robotic organism at a higher level, detects malfunctions, moves the robot, operates over the environment, and visualizes the robot status and surroundings.

4.2 Implementation of MoCLORA in ROMERIN

The generic control architecture presented previously has been specifically implemented for ROMERIN as illustrated in Figure 4.2. In this section, the main differences with the generic architecture are included.

4.2.1 External physical and simulated devices

In the case of ROMERIN, the robot modules described in Chapter 3 are found. Although there are several forms of communication implemented, UDP has been selected for convenience. Similarly, the simulated modules (**Module Plugins**) communicates with the CC via UDP, whereas the **Body Plugin** uses sockets to communicate. It implements the behavior of the pantilt, RGBd camera, and IMU. More information about the simulated environment is found in Section 4.3.

The pantilt is found as an external device connected via serial connection (USB) to the CC. It includes its motors and the U2D2 device that controls them. The RealSense D435i camera is also connected via USB with the CC, while the IMU is connected via I2C.

4.2.2 HAL

In this layer, the main component drivers are also implemented. The **IMU Driver** reads the data that the IMU sends via I2C. These data are published in the *"/gravityAtBody"* topic for its use in higher levels.

Simultaneously, the **Realsense Node**, reads the color and depth images to be published, as well as the embedded IMU data, which is sent to the **Pantilt Node** to be transformed to the body reference frame. To do so, it uses the **Realsense Driver**, which implements a serial connection to communicate with the RealSense D435i.

The **Pantilt Node** includes the **Pantilt Driver**, which manages the pantilt position with closed-loop control. It controls the pantilt actuators with the U2D2 device. By reading the IMU data from the Realsense camera, this node transforms the vector to the body frame and publishes it to the *"/gravityAtBody"* topic.

In the case of ROMERIN, there are two available sources of the gravity vector, the RealSense D435i camera and the IMU. To avoid conflicts, there are several strategies, such as a Kalman filter or a mean measure. However, it has been considered more practical to introduce a parameter established by the user to select one of the available sources. In both cases, the nature of the measurement is analogous, since it is carried out by means

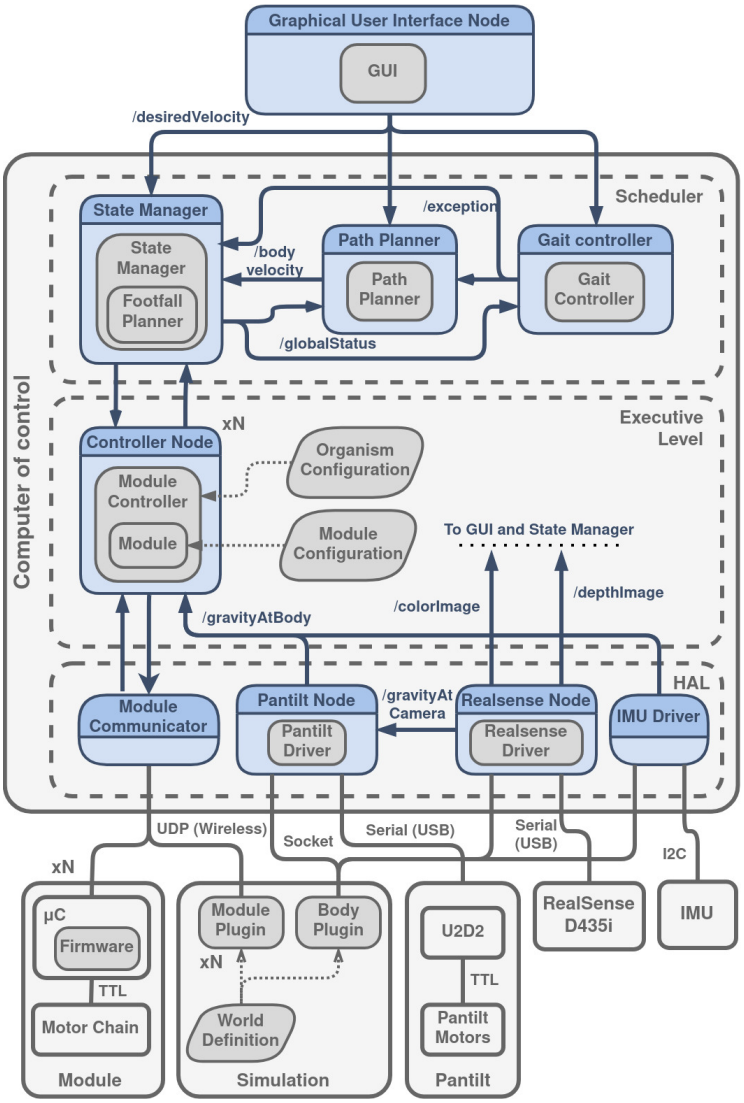


Figure 4.2: Implementation of the MoCLORA architecture in the ROMERIN robot. Rounded rectangles represent hardware devices, while dashed ones represent architecture levels. The dark gray boxes represent C++ objects. Parallelograms are configuration files. The blue containers are ROS2 nodes that use the objects to execute a routine. Gray lines represent inter-device communication, blue ones ROS2 topics for messages exchange (from publisher to subscriber), and dashed ones represent the use of a resource.

of accelerometers. Thus, according to the user decision, the **Pantilt Node** or the **IMU Driver** does publish into the *"/gravityAtBody"* topic.

4.2.3 Executive Level

In the **Module** objecte of ROMERIN has been implemented the management of module trajectories for both linear TCP (Tool Center Point) trajectories and time-optimal joint trajectories. In both trajectories, the TCP is defined in the suction cup frame (as shown previously in Figure 3.19 and similarly in Figure 4.3). Due to the configuration of the

wrist, where the last two motors move the last two joints together, the **Module** object performs the conversion between the positions and velocities of the two motors and the positions and velocities of the last two joints.

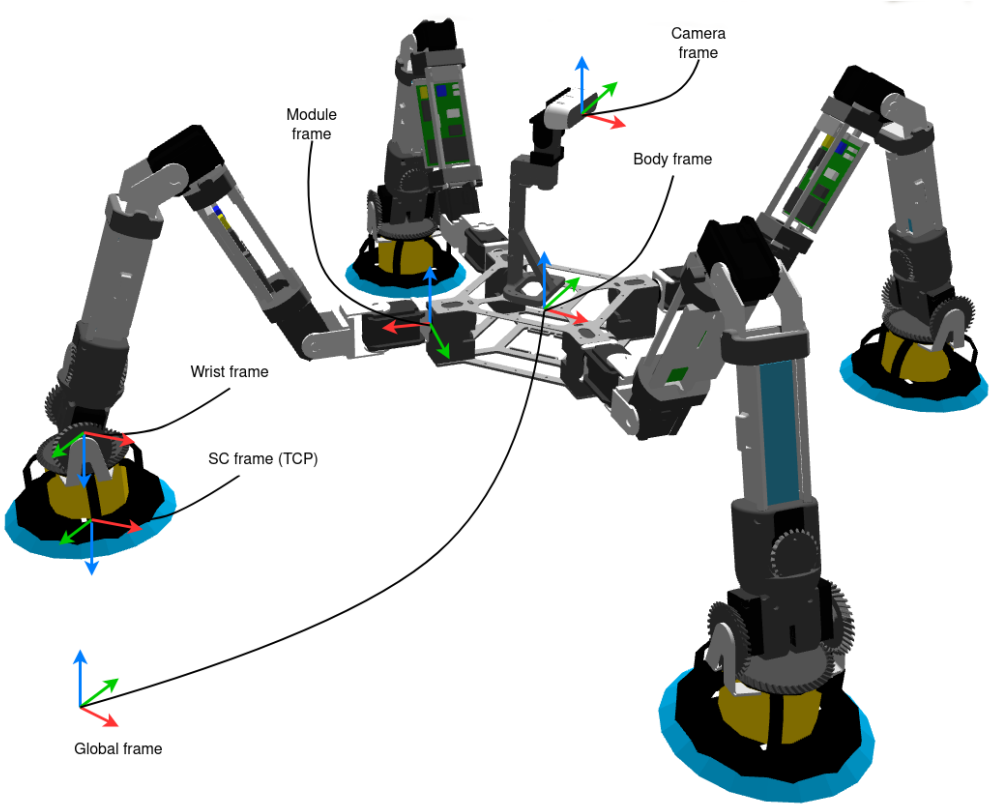


Figure 4.3: Organism frames.

In the following, some clarifications are made regarding the different movement modes that can be carried out by the modules during both the swing and stance phases. To understand them, Figure 4.3 shows the reference frames that we use in the implementation. The following movements are allowed and implemented in the **Module** object, and are used depending on the module situation:

1. **Module Movement in Joint Space**. Since the module has 6DOF, a joint path is defined by $q(t) \in \mathbb{R}^6$. This type of movement is used when a module is not in contact with a surface, that is, it is a non-support module (during the swing phase). In this case, a vector of joint position, velocity, and acceleration is generated to follow a trajectory. This trajectory might be entirely specified or created from waypoints. The duration of the motion is calculated based on the current state of the joints, their desired state, and the maximum velocity and acceleration of the joint defined in *Module configuration*. This type of movement makes use of the ROMERIN module kinematics presented in Section 3.8.2 and its simplification by transforming the second and third joints (computation with 6DOF).

2. **Module Movement in Cartesian Space**. Non-support module joints can also be controlled with motion commands in Cartesian space. In this case, the position, velocity, and acceleration of the TCP are generated to follow a trajectory (SC frame).

The features are similar to the previous motion, but, in this case, the maximum TCP velocity and acceleration are defined. The IK solver makes use of the track-IK [Beeson and Ames, 2015] library, which is a concurrent inverse Jacobian and non-linear optimization solver. This type of movement uses the same kinematics as the module movement in the joint space.

3. **Body Movement in Cartesian Space**. This motion determines the position and velocity of the body in Cartesian space, by means of the definition of a point as an interesting point. The desired body position is followed with the help of the support modules (those that are in the stance phase). For the IK, there are two options; the first is to use the leg motions in Cartesian space, where the desired position, velocity, and acceleration are obtained for the TCP frame according to the base trajectory (the same kinematics as the previous movements). The second option neglects the wrist joints and, in this way, the desired position, velocity, and acceleration are calculated with respect to the wrist frame. To do so, geometric methods are used to calculate the required values of the first joints of the kinematic chain (by means of the simplified 3DOF kinematics that was explained in Section 3.8.2).

   Due to the hyperstaticity of the organism, the system is over-constraint, and small errors in the estimation of the TCP position (due to the mechanical gaps and mismatches) lead to motor overload. For this reason, during trajectory execution, the torque transmission of the wrist motors is deactivated and the wrist remains in free movement, thus avoiding the appearance of torques and allowing free orientation against the coupling surface. That is why the second case is strongly recommended, especially by means of the torque-based control explained in Chapter 5.

4.2.4 Scheduler

For the specific case of ROMERIN, in this dissertation it is proposed a **Path Planner** that generates a speed profile that minimizes power consumption in actuators and maximizes robot safety (behavior defined in Section 6.3).

Similarly, the proposed **Footfall Planner** selects the best position where the leg in the swing phase must step to ensure safety and comfort, as explained in Section 6.5. It is a sampling-based footfall planner that considers all the possible footsteps, removes those that are not valid, and selects the best footfall point according to a cost function that optimizes the actuators effort, adhesion safety, and robot velocity.

The proposed **Gait Controller**, which is further explained in Section 6.4, is presented as a non-periodic, non-symmetric and non-regular bio-inspired reactive controller. It continuously checks the risk of detachment of the suction cups, the kinematic comfort, and the torque requirements.

The proposed state estimator for ROMERIN is explained in more detail in Section 5.7.

4.2.5 Graphical User Interface

ROMERIN counts with two different graphical user interfaces (created with Qt Creator, with the aim of achieve a cross-platform application):

1. The first serves to control the available modules independently (Figure 4.4). It

is designed as a low-level GUI to control the actuators and receive feedback from them (errors, position, velocity, current, voltage, and temperature). It allows one to read the suction cup sensors and act over the turbine (power throttle %). It also includes the battery voltage and current consumed. Lastly, it communicates with the modules through wireless (BT, and Wi-Fi) and wired (CAN bus) means, and allows to modify the communication information, such as module name, IP address, port, Wi-Fi router data, etc.

This GUI serves to configure the modules and test their components. Therefore, it does not appear in the control architecture diagram.

2. The second serves to control the robotic organism at a higher level (Figure 4.5). This GUI does not have direct access to the module motion. Instead, the user can command body velocity, move the pantilt to visualize the environment with the camera, observe the modules status, record information of the robot status, and launch different visualizers. This GUI communicates with the robot via ROS2 topics, so it is enough to connect the computer where the GUI is executed to the same network used by the CC.

   This GUI is included in the upper part of the control architecture. More information about the implementation details can be found in [de las Heras, 2023].

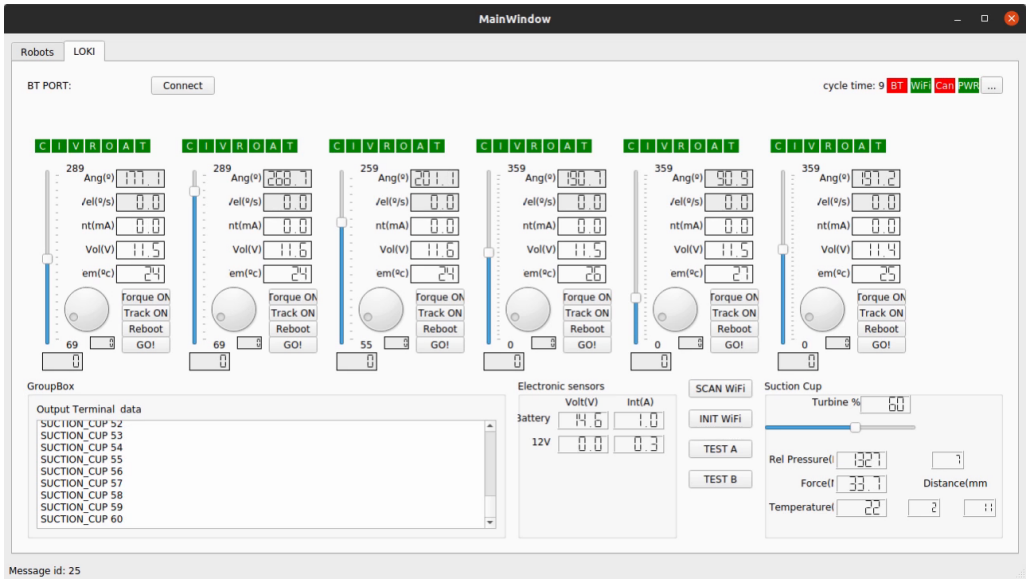


Figure 4.4: GUI to control modules independently.

4.3 Simulation environment

The development of reliable simulation systems is crucial; therefore, there is a high interest in the simulation of robots and their monitoring in virtual environments. Robot platforms are expensive due to their materials, sensors, and actuators. A simulation environment can avoid system damage while testing algorithms, reducing maintenance and testing time, wasted material, and costs. Furthermore, it is possible to vary the environmental



Figure 4.5: Current GUI [de las Heras, 2023].

conditions to perform tests in different situations. In general, a reliable virtual model of the system is required when working on large projects. Due to the simulation of complex systems, the testing time is considerably reduced, and in our case, it is possible to test the behavior of organisms with different arrangements and number of modules. In fact, it is not the only advantage. Learning has been demonstrated to have a time reduction benefit when using simulators in robotics [Liu et al., 2013]. Thus, future ROMERIN developers will have a faster and more intuitive tool for the development and testing of algorithms.

The use of simulators in robotics research is widespread, underpinning the majority of recent advances in the field. Physics simulators enable the vast majority of robotics research. It is common to test and prove theoretical methods initially or solely in a simulator, as robots themselves are often expensive, fragile, and scarce [Collins et al., 2021]. As the result of a large study on robotic simulators, in Figure 4.6 the popularity of robotics simulators is visually portrayed.

Even though there are many simulators, not everyone includes the components that ROMERIN requires, such as a RGBd camera, force sensors, batteries, etc. Another important aspect to take into account while selecting a simulator is the availability of documentation. The most broadcasted simulators count with a large documentation and community. This is the case of Gazebo, MuJoCo, CoppeliaSim, Webots, CARLA, and PyBullet. Among them, the most used robotic simulator according to [Collins et al., 2021] is Gazebo. It provides a 3D physics engine and rendering to support nearly enough of all robot types. It provides users with the ability to use a range of sensor plugins, support from a really large user/developer community, and easy resolution to most common issues.

In this thesis, we use the Gazebo simulator to simulate the ROMERIN organism in three modalities:

a) A simulation environment to test and validate algorithms for robotic organisms with

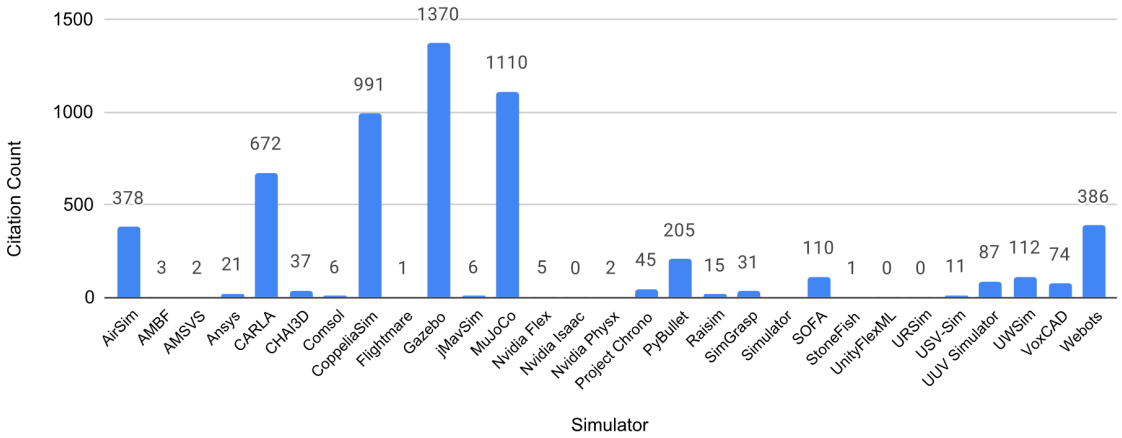


Figure 4.6: Citation count from 2016 to 2020 for reviewed simulators according to Google Scholar. Source: [Collins et al., 2021].

different topologies.

b) A digital twin of the physical robotic organism.

c) A digital mirror of the physical robotic organism to check malfunctions.

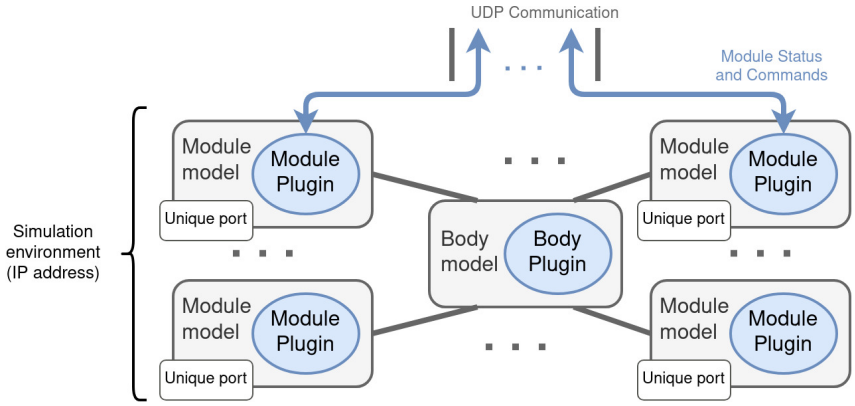


Figure 4.7: Structure of all simulation modalities of a robotic organism.

All modalities have the same structure (Figure 4.7) and change the way they communicate with the CC. First of all, based on the robot design of the hardware engineer, the 3D models of the module components are created. Then, a model is constructed per set of components that are connected to another set by a joint. Figure 4.8 represents the four typical 3D models in the process of simulating a link:

1. The first model (the one on the left) is the set of components that make up a link.

2. In the second one, the base reference frame is moved to the COM of the model.

3. The third one is a simplification of the second one to reduce computational cost during simulation. It is used by the simulator for visualization.

4. The last one is a simple envelope of the second model, so that the simulator uses it to detect collisions between objects, considerably reducing the computational cost and increasing the simulation frequency.

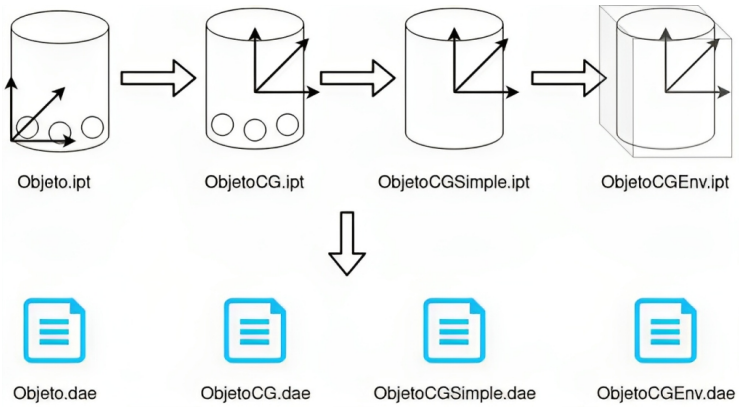


Figure 4.8: Creation of models for simulation.

Those 3D models are connected within a descriptive file named *module model* (Figure 4.7), which may be placed as many times as desired, accompanied by a plugin that specifies the behavior of the module (called *module plugin*). Each *module model* has an intrinsic definition of the kinematics and dynamics of a single module, as well as all its components and features. The plugin defines a behavior similar to that of the real modules. This is, as specified in Table 3.5, sends periodic messages about motors, suction cup, and analog information. In the same way, it receives the motor and suction cup commands to be executed and act in the simulated environment.

The communication between the simulated environment and the CC is done via UDP. Thus, similarly to what happens with the physical leg modules, it is required to specify the IP addresses and ports of the leg modules to communicate. Unlike communication with the physical modules (where each module has a unique IP address), each *module model* has a unique port. This is because they are all simulated in the same computer, and, therefore, the IP address is the same.

Similarly to the *module model*, the 3D models of the body components are encapsulated within the *body model*, whose *body plugin* defines its behavior, that is, specifies the performance of the body battery, IMU, camera, and pantilt.

Simulated actuators can be simulated with different controllers, as well as the Dynamixel servomotors. To make it as realistic as possible, it has been implemented a position control mode (with the corresponding PID controller) and a current control model (torque-based control). In the second case, the actuators are only controlled by current, and it is a great tool to validate torque-based controllers such as the one presented in Chapter 5.

A proper simulation requires from realistic environments. Figure 4.9 shows some of the environments created for testing. Among them, a test-bench, bridges, tunnels, buildings, and wind generator are found.

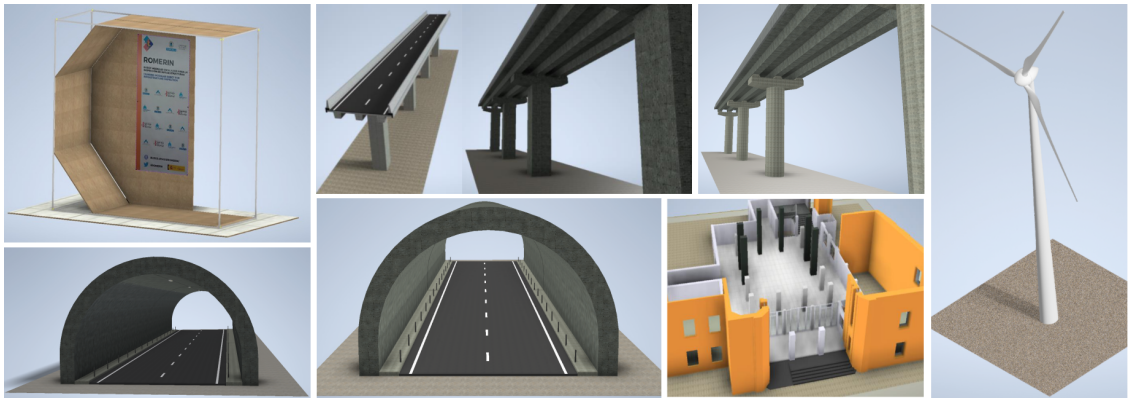


Figure 4.9: Virtual environments for testing [de las Heras, 2023].

4.3.1 Simulation for testing and validation

As denoted previously, a simulation environment serves to reproduce the expected behavior of a large number of variable organisms. Second, it serves to test and validate algorithms and to prove theoretical methods. In this thesis, the use of this environment has served as a great tool to test over many types of organism arrangements the response of the impedance control, the path and footfall planning that are presented in future chapters. Some configuration examples are shown in Figure 4.10:

a) A four-legged organism identical to the physically built body, that is, legs are mounted at the corners of a square with an angle of 45° (Figure 4.10(a)).

b) A six-legged organism with the previous body where the new legs are added to the sides and in opposite orientation (Figure 4.10(b)).

c) A seven-legged organism, adding a leg to the back of the previous organism (Figure 4.10(c)).

d) A six-legged organism, where the lateral legs are moved far from the body (Figure 4.10(d)).

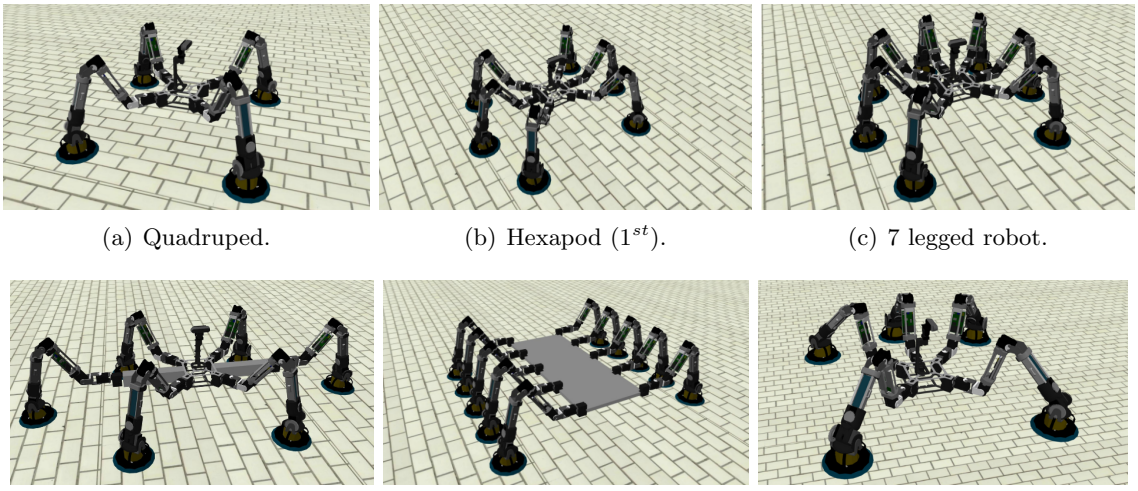
e) A ten-legged organism. In this case, the body is a large aluminum plate, where five legs have been added to each side of the plate (Figure 4.10(e)).

f) A five-legged organism, where one of the lateral legs of the six-legged organism is removed (Figure 4.10(f)).

Figure 4.11 indicates how the simulation environment is integrated within the low part of the control architecture explained in Chapter 4. As observed, all the physical devices are ignored and the control relapses over the simulation area. The data through the *Module Communicator* is redirected to the module plugins, the simulated pantilt is controlled and the data from the simulated RGBd camera and IMU are obtained.

4.3.2 Digital twin (DT)

A digital twin refers to the virtual model of any physical entity, both of which are interconnected and exchange data in real time [Singh et al., 2021]. Through its use, we can



(a) Quadruped. (b) Hexapod (1^{st}). (c) 7 legged robot.

(d) Hexapod (2^{nd}). (e) 10 legged robot. (f) 5 legged robot.

Figure 4.10: Examples of organism arrangements.

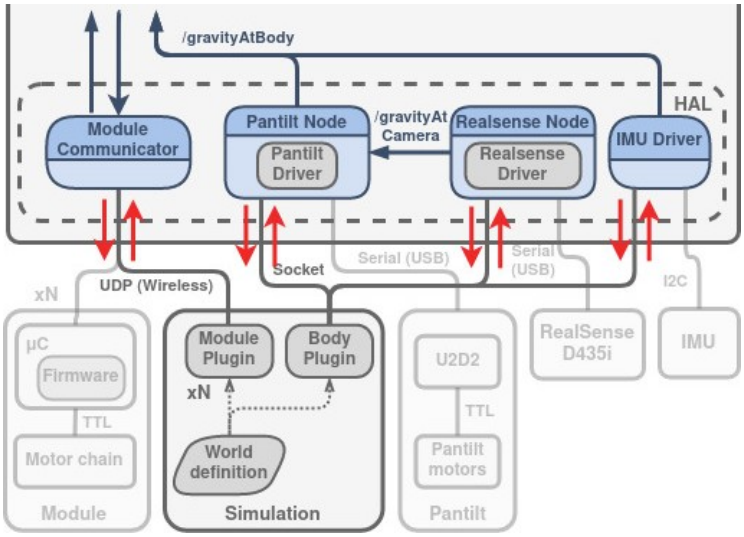


Figure 4.11: Simulation environment structure.

detect malfunctions in the real robot when the same command is sent to both and the behavior is different. For instance, it may be used for the following analyses:

a) checking that actuators have enough torque capacities,

b) checking that robot assembly is correct,

c) estimating gaps and inaccuracies,

d) comparing global behavior (battery duration, motion, etc.),

e) including terms and malfunctions that were not expected,

f) detecting motors overload,

g) checking reliability of the suction cup adhesion,

h) detecting if the robot is going to fall.

As an example, Figure 4.12 shows the comparison between the behavior of the physical organism and the digital twin while the CC is commanding a circular trajectory.

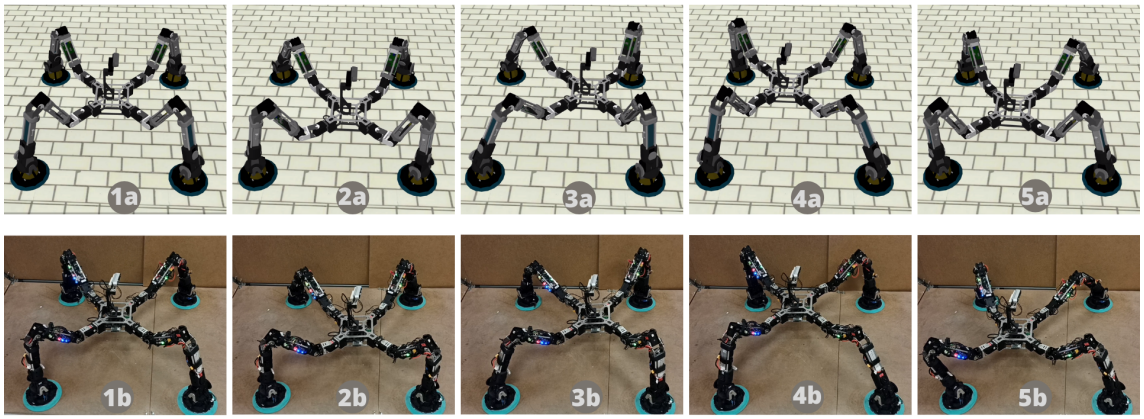


Figure 4.12: Results for the digital twin (up) and physical (down) organisms while carrying out a circular trajectory. Pictures 1 show the initial configuration, and the rest show catches of the trajectory.

Similarly to the simulation for testing and validation, Figure 4.13 indicates how the digital twin is integrated within the lower part of the control architecture explained in Chapter 4. As observed, all the physical devices and simulation components are used simultaneously. Thus, we can control both systems independently.

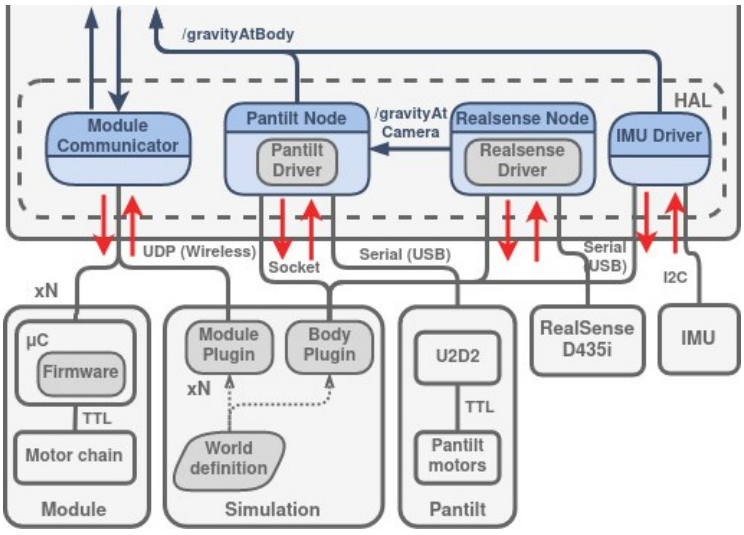


Figure 4.13: Digital twin structure.

4.3.3 Digital mirror (DM)

A digital mirror refers to the virtual model of any physical entity that shows the behavior that the real system is having [2]. It includes, among others, the following advantages and uses:

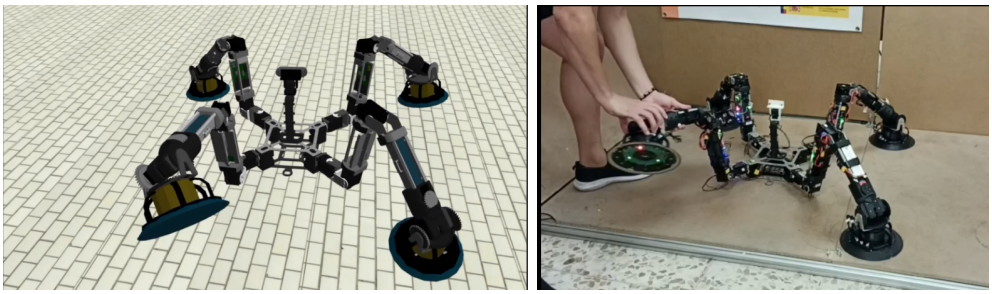
a) Immersive visualization experience in the graphical user interface (GUI),

b) malfunctioning checking (wrong calibration, pieces breakage detection, etc.),

c) remote visualization improvement,

d) each and every response of the organism is visually communicated to the user easily with a low computational cost.

As an example, in Figure 4.14, the second proposed use is illustrated.



(a) Digital mirror side. (b) Reality side.

Figure 4.14: Digital mirror example of use (calibration test).

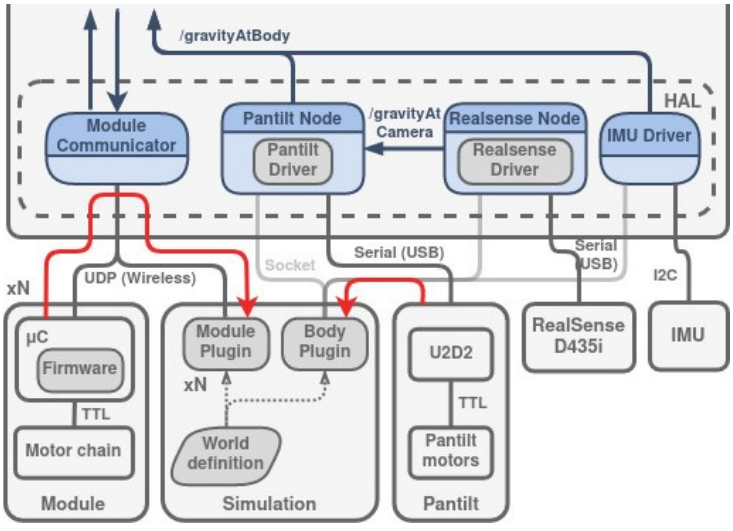


Figure 4.15: Digital mirror structure.

Figure 4.15 indicates how the digital mirror is integrated within the low part of the control architecture explained in Chapter 4. As observed, physical modules and pantilt data flows to the simulated environment for its replication and, consequently, its visualization.

[2]Video available at `https://youtu.be/5-jJpLtUR-I`, accessed on 10 April 2023.

4.4 Summary

In this chapter, a control architecture for legged robots with a variable number of legs is described. The architecture named MoCLORA (Modular Climbing-and-Legged Robotic Organism Architecture) arises from the need of body position and velocity control of legged robots with different numbers of legs and arrangements, and has been extended specifically for the ROMERIN robot.

In addition, the structure to simulate in Gazebo a legged-and-climbing robotic organism, and the information flow between components, are described. Three modalities have been included: a simulated environment for algorithm testing, a digital twin of ROMERIN, and a digital mirror that mimics the physical organism behavior.

The knowledge acquired in this chapter has been partially published in [Prados et al., 2022a] and [Prados et al., 2022b].

Chapter 5

Torque-based control of hyper-static multilimbed systems

"Our virtues and our failures are inseparable, like force and matter. When they separate, man is no more."

Nikola Tesla

Legged robots belong to the class of floating base or free-floating systems, that is, to systems that are not rigidly connected to their environment like a robotic arm [Hutter, 2013]. Instead, legged robots can move anywhere in space using the environment as a support. Floating bases are characterized by the disposal of an unactuated base, while interacting with the surrounding through multiple and continuously changing contact points to stand still or propel them forward.

In this chapter, the torque-based control of hyperstatic multilimbed systems is described. First of all, an overview of the control of a multibody robotic system is included, and the proposed approach of statics modeling is justified. A review of existing force distribution problem (FDP) solver methods is included and the approach used is explained further.

Later in this chapter, a description of the proposed gravity compensation is included. The method is first introduced for a planar biped robot with 2DOF per leg for better understanding since it is less complex than ROMERIN. After that, the gravity compensation method for a modular legged-and-climbing robot is described. This method is compared with the classical method, and the advantages of the implemented method are highlighted. Lastly, the robot state estimator and the body trajectory tracking control are described.

In this chapter, it is important to know the reference frames and component variables defined in Section 3.8.1.

5.1 Introduction

The control of multibody robotic systems can be done by two types of controllers. Those based on inverse kinematics ("inverse kinematics controllers", IKC) and those based on inverse dynamics ("inverse dynamics controllers", IDC) [Li et al., 2022]. The first type generates velocity commands for the joints, while the second one produces torque commands. Although IKCs are still used to control legged robots [Tanaka et al., 2022, Qiaoling et al., 2019], it is more recommended to approach the problem with IDCs and torque-based controllers [Kim et al., 2019, Bellicoso et al., 2016, Herzog et al., 2016]. To control each of the modules (legs) of a legged robot, two situations can be distinguished depending on the status of the particular module to control:

- When the module is attached to the environment. In this case, the action of gravity is reflected in the reaction forces that appear, and the module must be controlled by an IDC to avoid an overload of the actuators. Therefore, the dynamic equation of a manipulator can be applied (Equation 5.1).
- When the module is not attached and is free. In this case, no reaction forces appear in the module, and, therefore, it can be controlled with an IKC, or by means of an IDC (neglecting the last term of Equation 5.1).

The dynamic model is defined by Equation 5.1, where $M(q)$ is the inertia matrix, $C(q, \dot{q})$ that of centrifugal and Coriolis forces, $g(q)$ the gravitational component, and the last term denotes the force and moment vector exerted by the end effector on the environment.

$$\tau = M(q) \cdot \ddot{q} + C(q, \dot{q}) \cdot \dot{q} + g(q) + J(q)^T \cdot F_{ext} \tag{5.1}$$

One of the problems studied for multi-legged robots is how to determine the best sequence to lift and place the feet [Ding et al., 2010]. Non-climbing multi-legged robot locomotion can be classified into dynamic locomotion (running and hopping), and static locomotion as walking, where the body is constantly stable. For them (non-climbing robots), the vertical projection of the center of gravity of the robot has to be within the supporting polygon that links the positions of all supporting feet.

The dynamic gait is a faster and more difficult style, and so several models describe dynamic gait to improve robot speed. The spring-mass model consisting of a massless spring attached to a point mass describes the interdependency of mechanical parameters that characterize the running and jumping of humans as a function of speed [Full and Koditschek, 1999]. Robots with dynamic gaits, such as [Wang et al., 2015] or [Provancher et al., 2010], have been proven to be feasible with a pendulum mechanism or springs to climb walls with a fast adhesion or gripper system.

For a system such as the one proposed in this thesis, dynamic gait is discarded because of the attaching/detaching process of the suction cup and because of safety reasons (related as well to the adhesion system). In fact, for the first of the situations proposed at the beginning of this section, each module of the robot has very low speed and acceleration. Thus, the inertial and centrifugal components of the dynamic model can be neglected with respect to the gravitational and external forces components. This problem simplification concludes in the static model of the system (Equation 5.3), where only the forces and

torques produced by the action of gravity and the reaction forces appear. The result of the equation is related to the torques that each joint has to apply to compensate for body weight. This equation is the result of considering a fixed base at the first joint of each module (as obtained in the module kinematics of Section 3.8).

$$\tau = M(q) \cdot \cancelto{0}{\ddot{q}} + C(q, \dot{q}) \cdot \cancelto{0}{\dot{q}} + g(q) + J(q)^T \cdot F_{ext} \tag{5.2}$$

$$\tau = g(q) + J(q)^T \cdot F_{ext} \tag{5.3}$$

This general robot has $N \cdot J$ controllable DOFs, of which $N \cdot 3$ are forced to be passive when the leg module is attached (wrist), while only six DOFs related to body position and orientation can be controlled ($\mathbb{R}^3 \times SO(3)$). The system is hyperstatic if there are more DOFs to act than DOFs to control. As this difference increases, the system becomes more complex and more inaccuracies are penalized in the estimation of the torques to be applied. The passivity of the wrist is made by deactivating the power of the actuators. This passivity reduces hyperstaticity and allows for computing the force distribution problem (FDP) according to [Hernando et al., 2019].

5.2 Force distribution problem (FDP)

The force distribution problem is one of the challenges for legged robots. Although many legged robots, such as [Tanaka et al., 2022], use force/torque sensors to skip the FDP and reduce software complexity, it leads to increased costs, so it is recommended to solve the FDP. According to the characteristics of multi-chain systems of legged robots, it is explicitly remarked that the chains located in the support phase are tightly attached to each other [Zhuang et al., 2014], and it is particularly relevant for climbing robots. This fact has been highlighted in this dissertation as a hyperstatic problem.

During the stance phase, the reaction forces that appear in the robot have to meet physical constraints:

1. For a walking legged robot located in the z-plane (opposite to the gravity vector), the normal contact forces of the support feet are positive:

$$F_{i,z} \geq 0 \tag{5.4}$$

   This means that if the legged robot is walking on a slope (moving on x-direction), positive tangential forces are strictly required during the stance phase:

$$F_{i,x} \geq 0 \tag{5.5}$$

   On the other hand, for climbing legged robots, the normal contact forces of the support can be as negative as desired, as long as it is ensured that the torque of the actuators does not exceed the limits (point 4), and the adhesion system is not at risk of detaching from the surface.

2. The resultant normal force of the stance phase is equal to the force produced by the weight of the legged robot. That is, the sum of the reaction forces compensates the gravity forces, and the sum of the momentum is zero:

$$\sum \vec{F}_i + M \cdot \vec{g} = 0 \tag{5.6}$$

$$\sum \vec{M}_i = 0 \tag{5.7}$$

When estimating forces by solving the FDP, the values may differ slightly due to the set thresholds for the convergence of numerical methods. Similarly, for climbing robots, there should also be a moment equilibrium.

3. For legged robots, the support feet must not slip:

$$\sqrt{F_{i,x}^2 + F_{i,y}^2} \leq \mu F_{i,z} \tag{5.8}$$

where $F_{i,x}$ and $F_{i,y}$ are the foot forces of leg i of the support phase in the x and y directions, respectively, and μ is considered the relevant friction coefficient of the ground. For climbing robots, the normal force must be lower than a limit value, which is generally denoted as the grip force F_{sc}:

$$F_{i,z} < F_{sc} \tag{5.9}$$

4. Finally, the torques in each actuator have to be lower than their torque limits:

$$-\boldsymbol{\tau}_{i,max} \leq \mathbf{J}_i^T \cdot {}^i\mathbf{F}_i \leq \boldsymbol{\tau}_{i,max} \tag{5.10}$$

where $\boldsymbol{\tau}_{i,max} \in \mathbb{R}^{J\times 1}$ is the maximum articulated torque vector of the leg i; $\mathbf{J}_i^T \in \mathbb{R}^{J\times 3}$ is the Jacobian of the leg i; and ${}^i\mathbf{F}_i$ is the reaction forces related to Σ_i.

5.2.1 FDP methods

The solution of FDP for legged robots is a complex mathematical programming problem [Zhuang et al., 2014]. The main solving methodologies are as follows:

- **Linear-Programming (LP) Method**. The LP method is considered the most traditional and fundamental programming algorithm for optimizing the foot force distribution. The problem is formulated as a linear-programming problem, by replacing friction cone with a piecewise linear pyramid so as to express friction constraints by linear inequalities. Examples of the method are found in [Orin and Oh, 1981,Klein and Kittivatcharapong, 1990, Cheng and Orin, 1991, Villard et al., 1994, Zribi et al., 1999].

  This method has many flaws, such as the large number of calculations required, the discontinuous solutions that easily occur, and the difficulty in application to real-time control. When friction constraints and maximum articulated torque constraints are viewed as objective functions that stop slippage and keep articulated torques within specific limits, higher energy consumption can be obtained for the robot system.

- **Compact-Dual Linear-Programming (CDLP) Method**. CDLP uses the dual principle. Although it requires a relatively smaller number of calculations than the LP method, it is also unable to completely overcome solution discontinuities, and only achieves temporary solution continuity. The number of free variables determines the size of the problem in the CDLP formulation, and the generality of the CDLP functions determines the number of linear objective functions in real-time solutions of the force distribution. To minimize the power consumption of a legged robot, all articulated torques can be viewed as variables in the CDLP formulation. However, the CDLP method is computationally expensive and then more difficult to integrate real-time applications even though the number of variables is reduced. Examples of the CDLP method are found in [Cheng and Orin, 1990, Kwon and Lee, 1998].

- **Quadratic-Programming (QP) Method**. The computing time of the QP method does not depend on the initial hypotheses, and the QP method can overcome the solution discontinuity problem [Chen et al., 1998, Nahon and Angeles, 1991b]. The QP method is superior to the LP method, not only in calculation speed but also in quality of the solution obtained [Nahon and Angeles, 1991a]. However, the QP method does have flaws during fast movements. Nevertheless, the QP method has excellent advantages in maintaining real-time stability for lower-mobility legged robots.

- **Analytical Method**. Based on the Analytical Method, the mathematical programming problem can be transformed into an analytical problem. The main transformation processes consist of listing equilibrium relations of the forces for the support feet, incoming equality constraints, eliminating constrained conditions of the inequalities, etc. However, the application of the Analytical method is partially restricted because some inequality constraints are sacrificed in optimization of the foot force distribution.

  Although the Analytical method has shortcomings, it has still typically been chosen to perform the force distribution calculations. Optimizing the force distribution is highly complex for heavy-duty legged robots. Apart from the above four strategies, some researchers have also developed other algorithms to optimize the foot force distribution.

5.2.2 Proposed method

The alternative presented in this section attacks the problem for a hyperstatic system, as explained in [Hernando et al., 2019]. In principle, to solve a problem like this one, it would be necessary to consider the elastic characteristics of the robot limbs and motors, which would make the system too complex to be included in the control loop of a robot. Therefore a simplified mathematical model is sought, faithful enough to be able to:

a) decide which leg to release, and where to place it,

b) inform about the convenience of moving the center of gravity of the robot in one direction or another,

c) compute the torque in each actuator to compensate the force and moment vector exerted by the end effector on the environment, and

d) guarantee the safety of the robot by ensuring that it will not become detached from its surroundings.

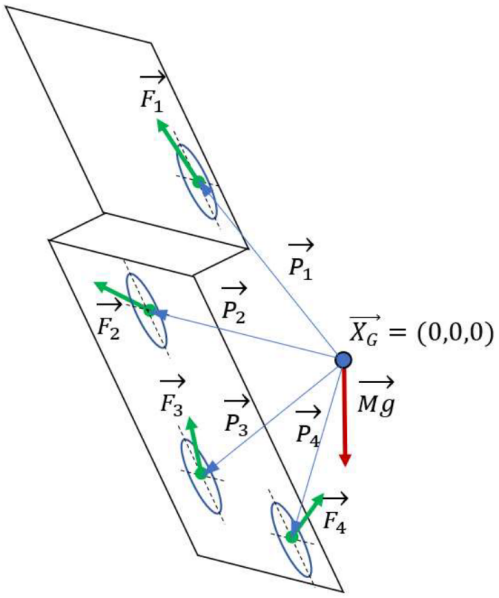


Figure 5.1: Representation of the forces produced at the wrist points when the robot is attached to the environment [Hernando et al., 2019].

To obtain the static model of the reaction forces in the suction cups, a simplified dynamic model is presented given the hyperstatic nature of the problem. The simplifying hypotheses that have been assumed are the following:

a) Robot legs are considered rigid, non-deformable.

b) In the suction cups, due to the universal joint that holds them, only reactive forces will be produced, not torques.

c) In order to solve the hyperstaticity, it will be assumed that the contact points are elastic with a constant K

With these assumptions, the reaction forces are located at the wrist points. Thus, an approximation of the load distribution at each wrist point can be obtained in a relatively simple way. In the absence of external forces, the center of gravity of the robot is located at the point $\vec{X_G}$ initially zero (Figure 5.1).

As a consequence of gravity and reactions at the wrist points, the center of gravity moves and rotates according to ${}^0T_R = ({}^0R_R, {}^0\vec{R}_R)$, where ${}^0R_R \in SO(3)$ y ${}^0\vec{R}_R \in \mathbb{R}^3$ are the orientation matrix and the position vector of the new position of the center of gravity with respect to the initial (Figure 5.2).

With these premises, for the system to be stable, the equations of static equilibrium must be fulfilled:

$$\sum \vec{F_i} + M \cdot \vec{g} = 0 \;\; ; \;\; \sum \vec{\tau_i} = 0 \tag{5.11}$$

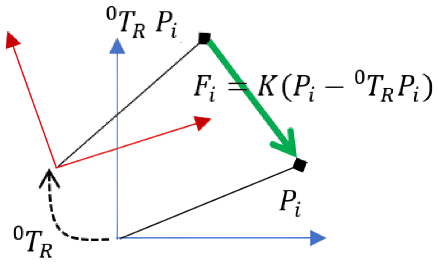


Figure 5.2: Reaction force computation as an elastic force due to the COM displacement.

According to these hypotheses, the reactive force $\vec{F_i}$ in each wrist point would therefore be obtained from the following:

$$\vec{F_i} = K\left(\vec{P_i} - {}^0T_R\vec{P_i}\right) = K\left(\vec{P_i} - {}^0R_R\vec{P_i} - {}^0P_R\right) \tag{5.12}$$

$$\vec{F_i} = K\left(\left(I - {}^0R_R\right)\vec{P_i} - {}^0\vec{P}_R\right) \tag{5.13}$$

On the other hand, applying the equation of equilibrium of moments:

$$\sum \vec{\tau_i} = \sum \vec{F_i} \times \left({}^0R_R\vec{P_i} + {}^0\vec{P}_R\right) + M\vec{g} \times {}^0\vec{P}_R = 0 \tag{5.14}$$

$$\sum \vec{F_i} \times \left({}^0R_R\vec{P_i}\right) + \left(\sum \vec{F_i} + M \cdot \vec{g}\right) \times {}^0\vec{P}_R = 0 \tag{5.15}$$

That by the equation of forces can be simplified in

$$\sum \vec{F_i} \times \left({}^0R_R\vec{P_i}\right) = 0 \tag{5.16}$$

$$\sum K\left(\left(I - {}^0R_R\right)\vec{P_i} - {}^0\vec{P}_R\right) \times \left({}^0R_R\vec{P_i}\right) = 0 \tag{5.17}$$

$$\sum K\left(\vec{P_i} - {}^0R_R\right) \times \left({}^0R_R\vec{P_i}\right) = 0 \tag{5.18}$$

Therefore, parameterizing ${}^0T_R$, by means of the spatial location vector of the displacement and rotation of the center of gravity $\vec{x} = (x, y, z, \alpha, \beta, \gamma)$, and as consequence:

$${}^0\vec{P}_R = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \quad {}^0R_R = RotX(\alpha)RotY(\beta)RotZ(\gamma) \tag{5.19}$$

$${}^0R_R = \begin{pmatrix} c_\beta c_\gamma & -c_\beta s_\gamma & s_\beta \\ s_\alpha s_\beta c_\gamma + c_\alpha s_\gamma & -s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & -s_\alpha c_\beta \\ -c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma & c_\alpha s_\beta s_\gamma + s_\alpha c_\gamma & c_\alpha c_\beta \end{pmatrix} \tag{5.20}$$

As a result, we have a system of six non-linear equations with six unknown variables that we must solve. Newton's method is one of the most popular numerical methods used to solve for the equation $f(x) = 0$. We know from Linear Algebra that we can take systems of equations and express those systems in the form of matrices and vectors. With this in mind, we can express the nonlinear system as a matrix with a corresponding vector. Thus, applying the Newton's method, the following equation is derived:

$$x^{(k)} = x^{(k+1)} - J(x^{(k+1)})^{-1} \cdot F(x^{(k+1)}) \tag{5.21}$$

where $F : \mathbb{R}^6 \to \mathbb{R}^6$ is the function that maps $(x, y, z, \alpha, \beta, \gamma)$ and $(\sum F_x, \sum F_y, \sum F_z, \sum \tau_x, \sum \tau_y, \sum \tau_z)$, and $J(x)$ is the Jacobian matrix of F.

5.3 Simplified example - Gravity compensation of a planar biped robot with 2 DOF per leg

With the idea of analyzing the compensation strategy, within the framework of the thesis, a two-dimensional model of the problem was developed to verify the algorithms developed. Thus, a planar biped climbing robot of 2DOF per leg is shown in Figure 5.3. In this robot, it is desired to control the body position, where both legs are housed. To simplify the example, links are considered cylinders without thickness, and L_{ij} is defined as the length of the link k of the leg i, m_j the mass of the link k, and L_B the length of the body. In this case, the robot has 4 active DOF (firsts and seconds joints) and 2 passive DOF (wrists), whereas the value to be controlled has only 3DOF (position and orientation).

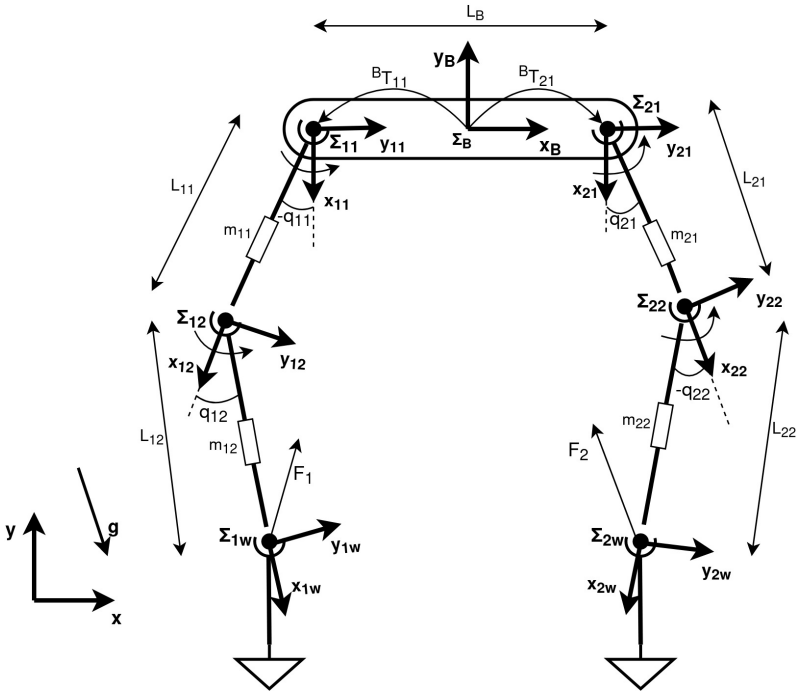


Figure 5.3: A two-legged-robot with 2 DOF per leg

Because the problem is approached as a modular organism, the kinematic model of each leg is computed separately. Thus, being the frame Σ_i the base of the chain, the forward kinematic model of the TCP, which in this case is the wrist point Σ_{iw}, is computed as:

$$\begin{pmatrix} x_w \\ y_w \end{pmatrix} = \begin{pmatrix} L_1 \cdot c_1 + L_2 \cdot c_{12} \\ L_1 \cdot s_1 + L_2 \cdot s_{12} \end{pmatrix} \tag{5.22}$$

While the inverse kinematic is given by:

$$q_2 = \pm \arccos\left(\frac{x_w^2 + y_w^2 - L_1^2 - L_2^2}{2 \cdot L_1 \cdot L_2}\right) \tag{5.23}$$

$$q_1 = \arctan 2\,(y_w, x_w) \pm \arctan 2\,(L_2 \cdot s_2, L_1 + L2 \cdot c_2) \tag{5.24}$$

The calculation of q_2 should be considered in two ways, the positive and negative value of the second term. However, to avoid that the elbow goes under the body, only the positive value of the second term is considered for the leg of the left side, whereas the negative value is considered for the leg of the right side.

The Jacobian matrix is obtained by derived from Equation 5.22, as indicated in Equation 5.25.

$$\begin{pmatrix} \dot{x}_w \\ \dot{y}_w \end{pmatrix} = \underbrace{\begin{pmatrix} -L_1 \cdot s_1 - L_2 \cdot s_{12} & -L_2 \cdot s_{12} \\ L_1 \cdot c_1 + L_2 \cdot c_{12} & L_2 \cdot c_{12} \end{pmatrix}}_{J(q)} \cdot \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \end{pmatrix} \tag{5.25}$$

The transformation matrices are given by:

$${}^{B}T_1 = \begin{pmatrix} 0 & 1 & -\frac{L_B}{2} \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5.26}$$

$${}^{B}T_2 = \begin{pmatrix} 0 & 1 & \frac{L_B}{2} \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5.27}$$

Before calculating the gravity compensation by the different methods, it is required to compute the COM of the robotic organism to solve the FDP. A model-based computation can be applied since the joint position is known, as well as the mass and the dimensions of the links and body.

To do so, each leg calculates its own COM with respect to Σ_i. To calculate it, the mass of the gripping system is neglected because it is considered part of the environment. Then, for each leg:

$$\begin{pmatrix} {}^{i}x_{i,cg} \\ {}^{i}y_{i,cg} \end{pmatrix} = \frac{1}{m_1 + m_2} \begin{pmatrix} \frac{L_1}{2} \cdot c_1 \cdot m_1 + (L_1 \cdot c_1 + \frac{L_2}{2} \cdot c_{12}) \cdot m_2 \\ \frac{L_1}{2} \cdot s_1 \cdot m_1 + (L_1 \cdot s_1 + \frac{L_2}{2} \cdot s_{12}) \cdot m_2 \end{pmatrix} \tag{5.28}$$

Consequently, COMs are transformed into the body frame Σ_B, whereas the resultant

orientation is not taken into account:

$$\begin{pmatrix} {}^{B}x_{i,cg} \\ {}^{B}y_{i,cg} \\ 1 \end{pmatrix} = {}^{B}T_i \begin{pmatrix} {}^{i}x_{i,cg} \\ {}^{i}y_{i,cg} \\ 1 \end{pmatrix} \tag{5.29}$$

Then, the COM of the whole system is referred to Σ_B:

$$\begin{pmatrix} {}^{B}x_{cg} \\ {}^{B}y_{cg} \end{pmatrix} = \frac{1}{N \cdot (m_1 + m_2) + m_B} \begin{pmatrix} \sum_{i=1}^{N} \left[(m_1 + m_2) \cdot {}^{B}x_{i,cg} + m_B \cdot 0 \right] \\ \sum_{i=1}^{N} \left[(m_1 + m_2) \cdot {}^{B}y_{i,cg} + m_B \cdot 0 \right] \end{pmatrix} \tag{5.30}$$

As denoted in Section 5.2.2, the reaction forces refer to the COM. Therefore, a transformation of Equation 5.31 is needed to determine the position of the WPs related to the center of gravity, as indicated in Figure 5.4. In this simplified example, the body COM and the body frame match, but it may be different.

$${}^{cg}T_{iw} = {}^{B}T_{cg}{}^{-1} \cdot {}^{B}T_i \cdot {}^{i}T_{iw} \tag{5.31}$$

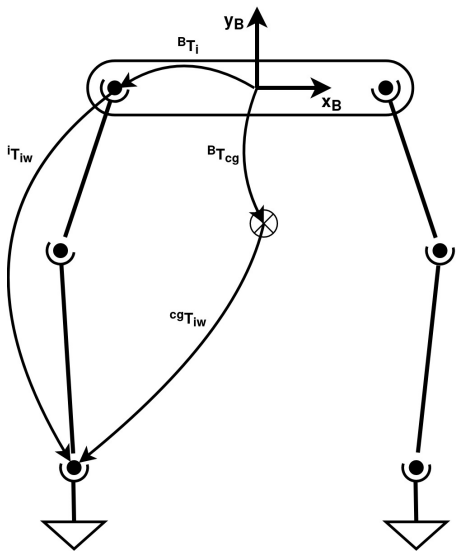


Figure 5.4: Transformation of contact points from body leg i reference frame to COM reference frame

Once the position of the WPs and the direction of the gravity vector are known, the reaction forces can be computed as indicated in Section 5.2.2.

5.3.1 Method 1 - Classical method

In the world of static forces the transpose of the robot Jacobian relates the forces exerted by the end-effector to the forces exerted by each joint. Thus, Equation 5.32 indicates how to calculate the required torques whenever an external force is applied to the end-effector.

$$\tau_{ext} = J(q)^T \cdot F_{ext} \tag{5.32}$$

There are several analytical ways to obtain dynamic equations, as well as the static one: Euler-Lagrange [Paul, 1981], recursive Lagrange [Hollerbach, 1980], recursive Newton-Euler [Luh et al., 1980], Kane's equations [Kane and Levinson, 1983], the D'Alembert principle [Lee et al., 1983] and some alternate algorithms that are based on velocity constraint matrices [Saha, 1999] or divide-and-conquer approach [Featherstone, 1999]. To compute the gravitational component of the static model, Lagrange's equations of motion of a robot manipulator are implemented:

$$\mathcal{L}(q, \dot{q}) = \kappa(q, \dot{q}) - \upsilon(q) \tag{5.33}$$

$$\frac{d}{dt}\left[\frac{\partial \mathcal{L}(q, \dot{q})}{\partial \dot{q}}\right] - \frac{\partial \mathcal{L}(q, \dot{q})}{\partial q} = \tau \tag{5.34}$$

As denoted previously, the kinetic energy can be neglected compared to the static forces ($\kappa(\boldsymbol{q}, \dot{\boldsymbol{q}}) \simeq 0$), while the potential energy is computed as

$$\begin{aligned} U(\boldsymbol{q}) = m_1 \cdot \left(\frac{L_1}{2} \cdot c_1 \cdot {}^i g_x + \frac{L_1}{2} \cdot s_1 \cdot {}^i g_y\right) + m_2 \\ \cdot \left(L_1 \cdot c_1 + \frac{L_2}{2} \cdot c_{12} \cdot {}^i g_x + L1 \cdot s_1 + \frac{L_2}{2} \cdot s_{12} \cdot {}^i g_y\right), \end{aligned} \tag{5.35}$$

where

$${}^i g = {}^B T_i^{-1} \cdot {}^B g \tag{5.36}$$

Thus, Lagrange equation and the corresponding torques to each joint are:

$$\begin{aligned} \tau_1 = m_1 \cdot \left(-\frac{L_1}{2} \cdot s_1 \cdot {}^i g_x + \frac{L_1}{2} \cdot c_1 \cdot {}^i g_y\right) + m_2 \\ \cdot \left(-L_1 \cdot s_1 - \frac{L_2}{2} \cdot s_{12} \cdot {}^i g_x + L1 \cdot c_1 + \frac{L_2}{2} \cdot c_{12} \cdot {}^i g_y\right) \end{aligned} \tag{5.37}$$

$$\tau_2 = m_2 \cdot \left(-\frac{L_2}{2} \cdot s_{12} \cdot {}^i g_x + \frac{L_2}{2} \cdot c_{12} \cdot {}^i g_y\right) \tag{5.38}$$

5.3.2 Method 2 - Proposed method

In this dissertation, a method to compute the torque that each actuator has to apply to compensate gravitation forces is proposed without the use of the Jacobian matrix for the forces exerted by the end-effector, and without the use of other analytical calculations for the gravitational components.

To summarize the forces that appear in the robot, Figure 5.5 shows them for the simplified example. Weight forces appear at each part center of mass (links and body) in the same direction as gravity acts. In the same way, the reaction forces previously calculated appear at the wrist points. On the other hand, Figure 5.6 indicates the distances from the joint position and the point where forces are applied.

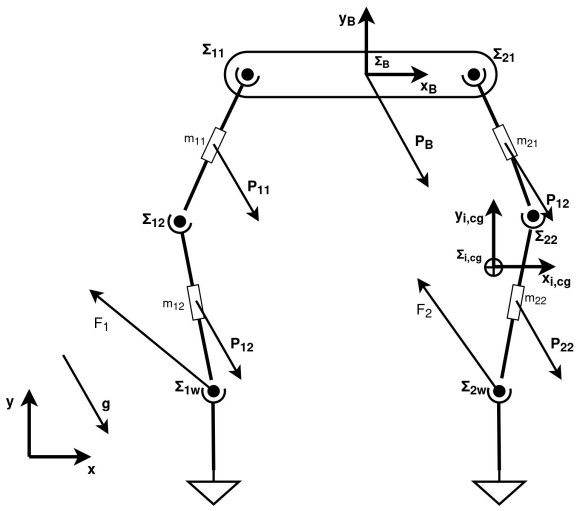
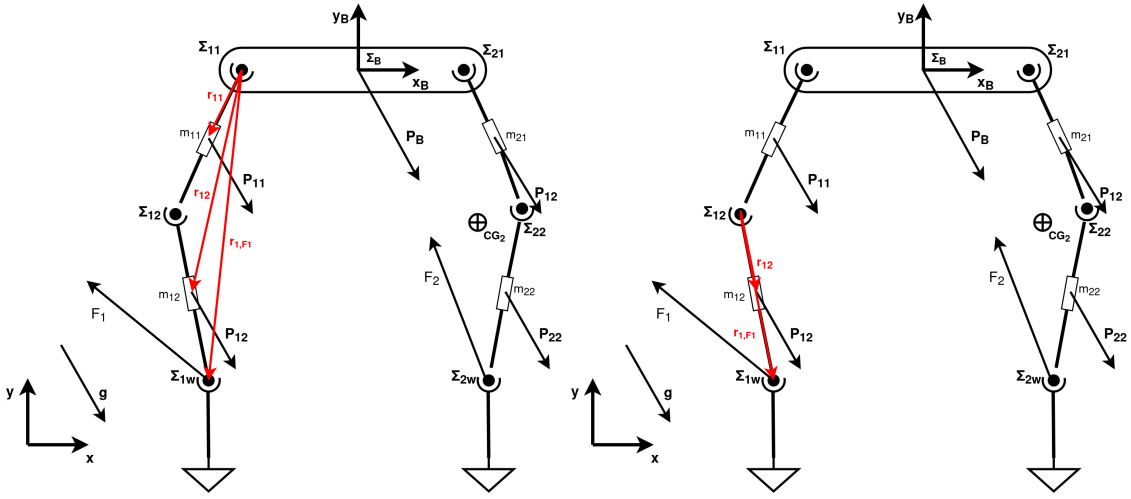


Figure 5.5: Forces that appear at the organism



(a) At joint 1.

(b) At joint 2.

Figure 5.6: Forces and distances vector to calculate the required torque of both joints of module 1.

In this proposed method, the Newton-Euler method is used. Basically, the Newton–Euler equations describe the combined translations and rotational dynamics of a rigid body. They are used as the basis for more complicated "multi-body" formulations (screw theory) that describe the dynamics of systems of rigid bodies connected by joints and other constraints.

As included in the formulation, the torques required to compensate the gravity forces are calculated for the leg joints. Equation 5.39 calculates the momentum produced on an object based on the amount of force applied, $\vec{F}$, to the object at a distance, $\vec{r}$, from the pivot point from where the force is applied to the pivot point of the object. In the formula, $\vec{u_q}$ represents the unitary vector that projects the moment on the joint axis.

$$\vec{\tau} = \vec{u_q} \cdot \left(\vec{r} \times \vec{F} \right) \tag{5.39}$$

In a similar way to the resolution of bending moment diagrams, a hyperstatic system can be approached by analyzing the forces that appear on it at certain points of interest (in this case, at the joints of the organism). By making a section at each of the joints, an analysis of the forces remaining on one of the sides is performed. The simplest way is to apply the forces appearing in the module itself, i.e. the weights of the links farthest from the body and the reaction force appearing in the given module.

5.3.3 Validation of the proposed method

Matlab Simscape is a tool to quickly create models of physical systems within the Simulink environment. It is possible to create models of physical components based on physical connections that integrate directly with block diagrams and other modeling paradigms. Thus, the biped robot shown in Figure 5.7 has been created for testing and algorithm validation.

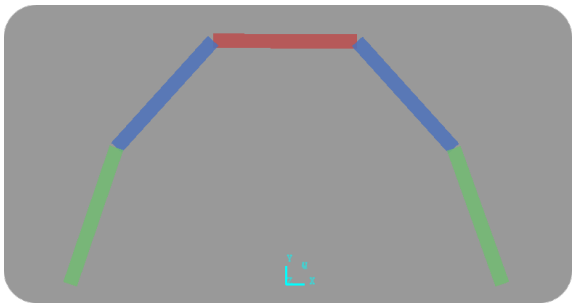
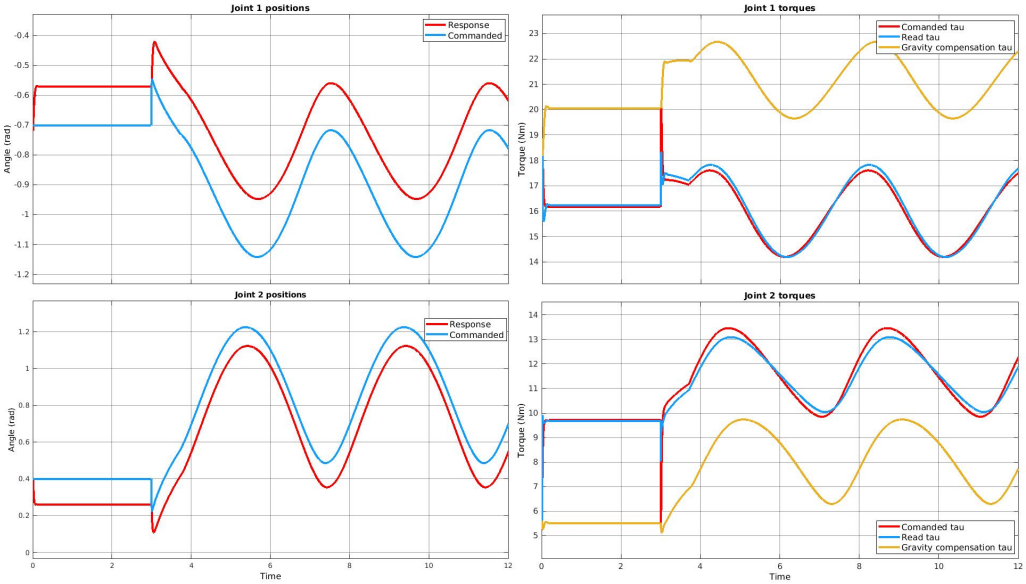


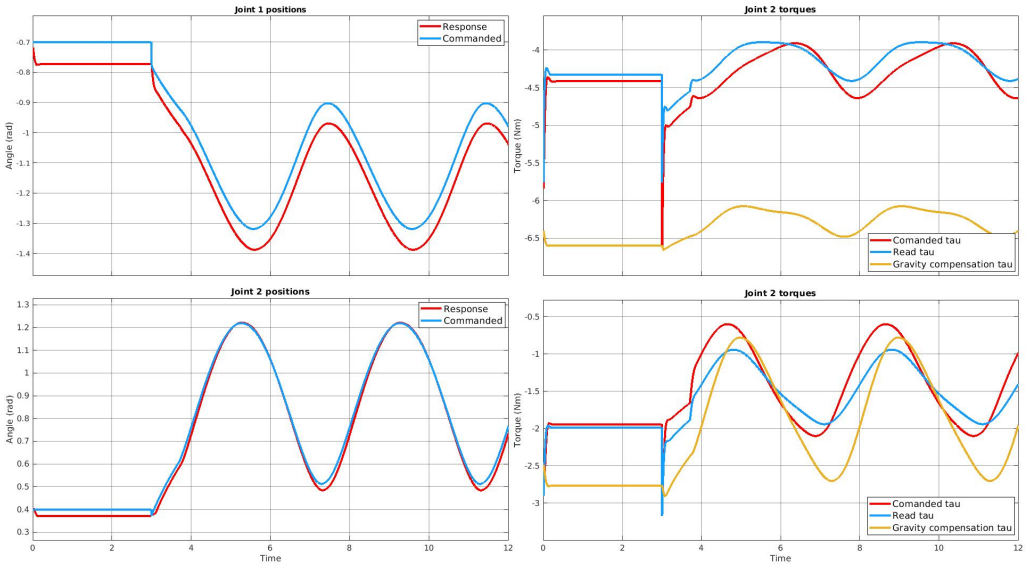
Figure 5.7: Representation of the simplified example (biped robot with 2DOF per leg).

To check gravity compensation with position control, Figure 5.8 shows the behavior of the biped robot in simulation with $K_p = 30$ when a circular trajectory is commanded and the robot is located in a wall (90°, shown in Figure 5.8(a)) and a ceiling (180°, shown in Figure 5.8(b)). Given the robot described previously, it has been simulated with dimensions of $L_1 = L_2 = L_B = 0.1m$ and weights of $m_1 = m_2 = 1.0kg$ and $m_B = 5.0kg$.

As observed, due to the high height of the body, the required torque to compensate gravity forces is much higher in the case of the wall. This is because the distance between the vector of the links weight and the joint axis is higher in the case of locating the robot on a wall. This experiment has been carried out for algorithm validation and the values are



(a) Wall.



(b) Ceiling.

Figure 5.8: Results of the biped robot with 2DOF per leg doing a circular trajectory.

indicative since the masses and dimensions have been set without any physical meaning. However, this experiment serves to identify the obvious problem that the bipedal robot would be more comfortable from a torque point of view the closer its body is to the wall. In fact, the torque to compensate the gravitational forces is three times less in the case of the ceiling for the first joint and four times less for the second one.

5.4 Gravity compensation of ROMERIN

The planar model of a legged robot allows establishing the bases and operating scheme of the gravity compensation method for the ROMERIN robot. In the real system, the robot has $(N \cdot 6)$ active DOF, being N the number of modules, whereas it is desired to control the body position and orientation $(\mathbb{R}^3 \times SO(3))$. For a robotic organism with 4 modules, there are 24 active DOFs to control 6DOFs $(\mathbb{R}^3 \times SO(3))$. This can be simplified by passivizing the wrist joints, although the level of hyperstaticity is still high with 12 active DOF.

The process of solving the static robot model is essentially the same as in the simplified case. However, there are some minor modifications to be taken into account. For each non-supporting leg (swing phase), the COM of the leg i is calculated as follows:

$$ {}^{i}\Sigma_{i,cg} = \frac{1}{m_i} \cdot \sum_{k=1}^{J} \left({}^{i}x_{ik} \cdot m_{ik} \right) \tag{5.40} $$

However, for the supporting legs, the calculation is different (note that $J-3$ is used to remove the wrist joints):

$$ {}^{i}\Sigma_{i,cg} = \frac{1}{\sum_{k=1}^{J-3} m_{ik}} \cdot \sum_{k=1}^{J-3} \left({}^{i}x_{ik} \cdot m_{ik} \right) \tag{5.41} $$

As an example with the ROMERIN robot, the calculation of the COM of the non-supporting leg i is computed as:

$$ \begin{pmatrix} {}^{i}x_{i,cg} \\ {}^{i}y_{i,cg} \\ {}^{i}z_{i,cg} \end{pmatrix} = \frac{1}{\sum_{k=1}^{3} m_{ik}} \cdot \begin{pmatrix} \frac{L_1 \cdot c_1}{2} \cdot m_1 + \left(L_1 \cdot c_1 + \frac{L_c \cdot c_1 \cdot c_2}{2} \right) \cdot m_2 + \left(L_1 \cdot c_1 + L_c \cdot c_1 \cdot c_2 + \frac{L_d \cdot c_1 \cdot c_{3-2}}{2} \right) \cdot m_3 \\ \frac{L_1 \cdot s_1}{2} \cdot m_1 + \left(L_1 \cdot s_1 + \frac{L_c \cdot s_1 \cdot c_2}{2} \right) \cdot m_2 + \left(L_1 \cdot s_1 + L_c \cdot s_1 \cdot c_2 + \frac{L_d \cdot s_1 \cdot c_{3-2}}{2} \right) \cdot m_3 \\ \frac{L_c \cdot s_2}{2} \cdot m_2 + \left(L_c \cdot s_2 - \frac{L_d \cdot s_{3-2}}{2} \right) \cdot m_3 \end{pmatrix} \tag{5.42} $$

For both cases, supporting and non-supporting legs, the position of the leg COM is transformed to Σ_B as:

$$ \begin{pmatrix} {}^{B}x_{i,cg} \\ {}^{B}y_{i,cg} \\ {}^{B}z_{i,cg} \\ 1 \end{pmatrix} = {}^{B}T_i \begin{pmatrix} {}^{i}x_{i,cg} \\ {}^{i}y_{i,cg} \\ {}^{i}z_{i,cg} \\ 1 \end{pmatrix} \tag{5.43} $$

where ${}^{B}T_i$ denotes the homogeneous transformation matrix from Σ_B to Σ_i. Being ${}^{B}p_{i,cg} = \left({}^{B}x_{i,cg}, {}^{B}y_{i,cg}, {}^{B}z_{i,cg}\right)$ the position of the leg COM referred to Σ_B, and ${}^{B}p_{cg} = \left({}^{B}x_{cg}, {}^{B}y_{cg}, {}^{B}z_{cg}\right)$ the position of the COM of the whole system referred to Σ_B, ${}^{B}p_{cg}$ is given by:

$$ {}^{B}p_{cg} = \frac{1}{M'} \sum_{i=1}^{N} \left[m'_i \cdot {}^{B}p_{i,cg} \right] \tag{5.44} $$

where M' represents the robot mass that is considered for the computation of the reaction forces, $m'_i = m_i$ for non-supporting legs, and $m'_i = m_{i1} + m_{i2} + m_{i3}$ for supporting legs. Being N_s and N_n, the number of supporting and non-supporting legs, respectively, the value of M' is given by:

$$ M' = m_B + N_s \cdot (m_{i1} + m_{i2} + m_{i3}) + N_n \cdot m_i \tag{5.45} $$

The position of the WPs (where the reaction force appears) and the total COM of the robot are required to compute the reaction forces with the implemented estimator. To do so, Equation 5.46 denotes the transformation between the center of gravity and the WP pose as follows:

$$ {}^{cg}T_w = {}^{B}T_{cg}{}^{-1} \cdot {}^{B}T_i \cdot {}^{i}T_w(q) \tag{5.46} $$

where ${}^{i}T_w(q)$ can be obtained with the forward kinematics is the transformation matrix between Σ_B and Σ_{cg}.

Later, the reaction forces are calculated as before, and the same process is followed. As an example, Figure 5.9 represents the forces that appear in a quadruped robot in 3D when the gravity force is acting vertically and pointing down.

Similarly, in Figure 5.10 two examples are illustrated of the reaction forces that appear in the WPs when the robot is located at different positions.

5.5 Comparison of methods

The methods presented in Sections 5.3.1 and 5.3.2, respectively, give the same results under the same conditions. The first method has been widely used for legged robots, and it is one of the unique methods found in the literature. A great example is [Vidoni and Gasparetto, 2011], where the authors use the Jacobian transpose to compute the torques required by external forces, which are computed thanks to the combination of the QP method with the reduction in the problem size proposed in [Chen et al., 1999] to solve the FDP.

Table 5.1 indicates the computation time for each method for one leg (including the standard deviation) and for a modular robot of four and six legs. Although the growth of

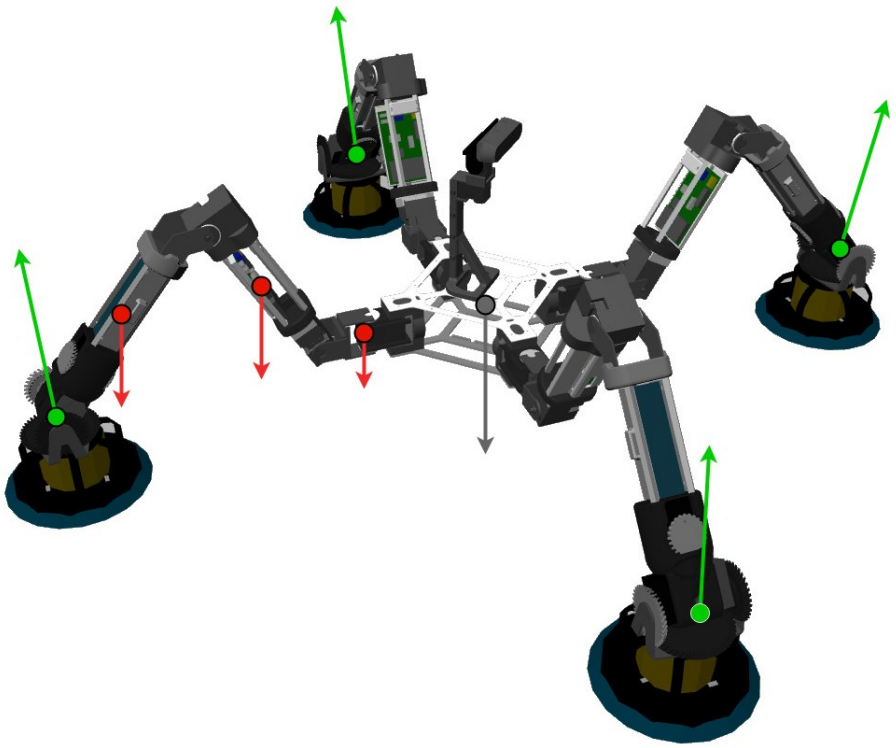


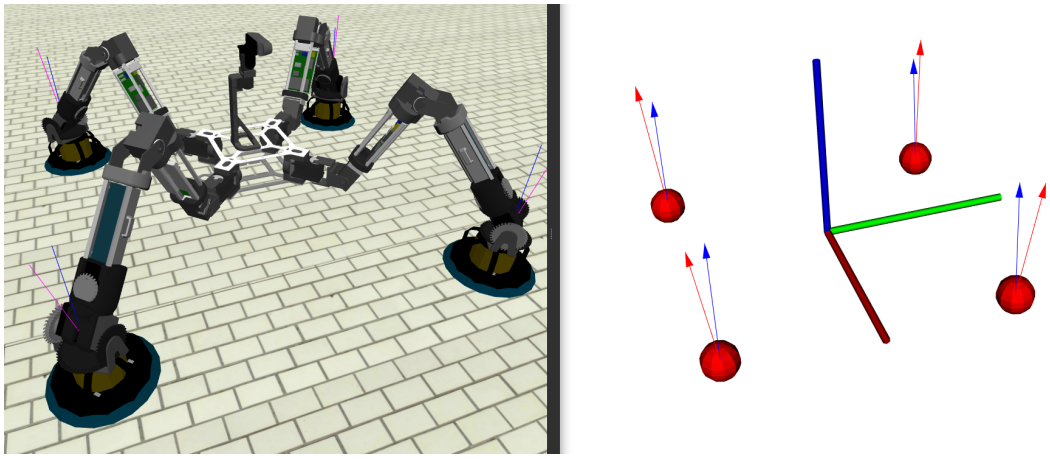
Figure 5.9: Forces that appear in a 4-legged robot. Green arrows are reaction forces, gray one the weight of the body, and red ones the weight of the links of the module that is calculating the torques to apply on the joints. Spheres represent the points where the reaction forces apply and where the COM of each link and body are located.

the computation time is linear for both cases, it is possible to observe that as the number of legs increases, the computation time for the first method increases considerably. It is particularly important for a multi-legged system where the logic runs on a single computer and where the control rate and computational cost are critical. This importance increases for systems such as ROMERIN, where the number of legs can be considerably increased to adapt to specific real-time applications. In addition, it is especially important in sampling-based algorithms for motion planning, where the computation is performed many times.

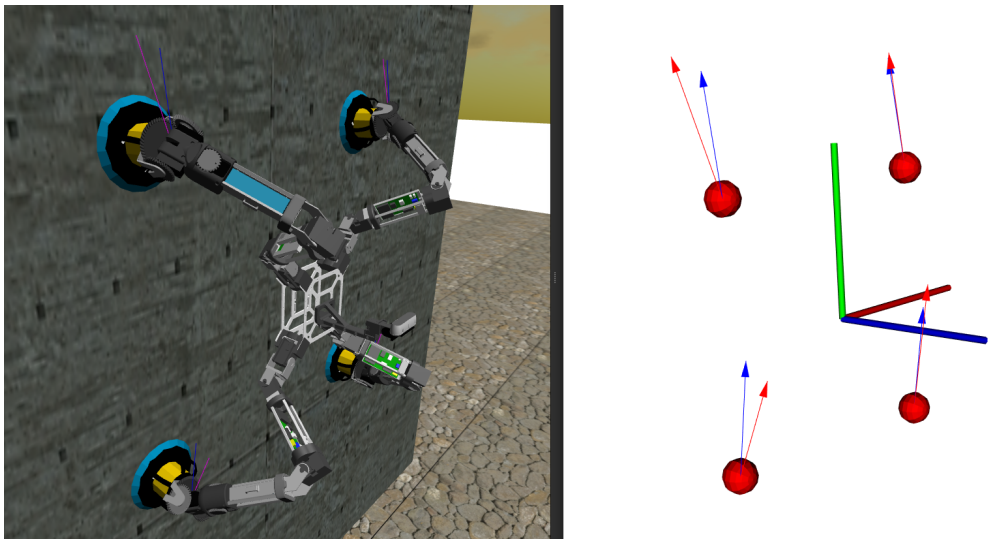
Method	$\mu(t_c)(us)$	$\sigma(t_c)(us)$	$\mu(t_c)(us)$ **4 Legs**	$\mu(t_c)(us)$ **6 Legs**
Classical method	145	165	580	870
Proposed method	77	95	308	462

Table 5.1: Comparison of computational time (t_c) of presented methods.

As denoted in the table, the average computational cost of the proposed method is half that of the classical method. This is due to the use of the Jacobian matrix, which increases the computation requirements. To emphasize the importance of reducing computational time, in the case of using a sampling-based algorithm for motion planning in an hexapod robot, the method proposed in this thesis saves 0.41 s of computation with 1000 samples.



(a) Organism located in the ground.



(b) Organism located in a wall.

Figure 5.10: Reaction forces that appear in the organism in different situations.

5.6 Impedance Control of Leg Modules

There are two main alternatives to control the position of the robot while sending torque commands [Kelly et al., 2005]. The first case is the proportional control plus velocity feedback, whereas the second case is the PD control. Both cases represent closed-loop controllers, where the proportional and derivative constants help the modules push the body of the robot.

The proportional control plus velocity feedback with gravity compensation is a closed-loop controller that may be used for robot manipulators. The conceptual application of this control strategy is common in the angular position control of DC motors. In this application, the controller is also known as a proportional control with tachometric feedback. The equation of proportional control plus velocity feedback is given by Equation 5.47, where K_p and $K_d \in \mathbb{R}^{(J-3)\times(J-3)}$ are diagonal and symmetric positive definite matrices that refer to the position gain and the velocity gain (or derivative), respectively,

$\tau_g \in \mathbb{R}^{(J-3)}$ refers to Equation 5.3, $q_d \in \mathbb{R}^{(J-3)}$ is the desired joints position and $q \in \mathbb{R}^{(J-3)}$ and $\dot{q} \in \mathbb{R}^{(J-3)}$ are the current position and velocity respectively.

$$\tau = \tau_g + K_p \cdot (q_d - q) - K_d \cdot \dot{q} \tag{5.47}$$

On the other hand, the proportional Derivative (PD) control with gravity compensation is an immediate extension of proportional control plus velocity feedback. As its name suggests, the control law is not only composed of a proportional term of the position error as in the case of proportional control, but also of another term that is proportional to the derivative of the position, i.e., to its velocity error:

$$\tau = \tau_g + K_p \cdot (q_d - q) + K_d \cdot (\dot{q}_d - \dot{q}) \tag{5.48}$$

Proportional control plus velocity feedback may be understood as a PD control where the desired velocity is zero to guarantee stability and reduce rude movements.

Figure 5.11 shows the control diagram that is integrated in each module, which can be denoted as an impedance control. A body position is commanded Σ_B^{ref} and compared with the estimated body position $\hat{\Sigma}_B$. The transformation ${}^B T_{iw}^{-1}(q)$ serves to obtain the desired WP position, whereas the inverse kinematics is used to determine the desired joints position q_d. A PD controller is applied to obtain the torque based on the joint position error. On the other hand, based on the position estimation and the gravity compensation proposed in this section, another torque component is generated. The sum of both torques is applied to the joints.

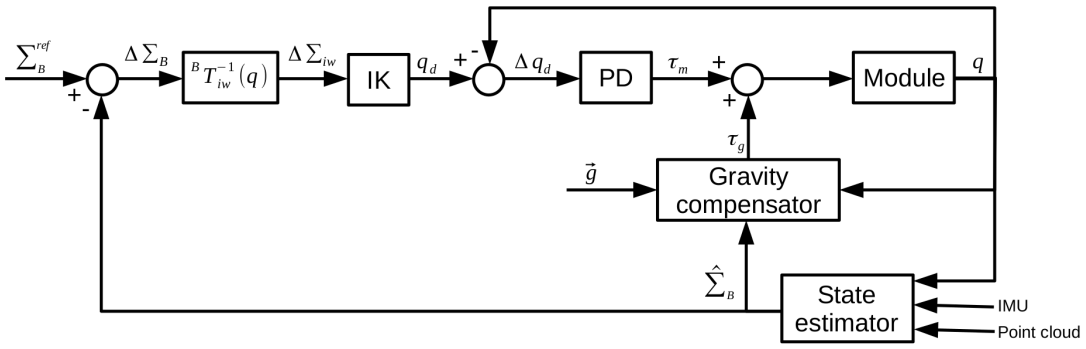


Figure 5.11: Control scheme of a module

5.6.1 Body trajectory tracking

Following the objective of biomimetics, in this section we describe the body trajectory tracking control without dynamic compensation used in the ROMERIN robot. In the animal world, the body position is usually not controlled directly. On the contrary, animals "establish" their body velocity, and the legs are those that accompany the movement.

Unlike path-following, which focuses on following a predefined path without time constraints, trajectory tracking does not guarantee that the path is faithfully followed. Instead, it involves time as a constraint by correcting the position error. Trajectory tracking

can be implemented with a body velocity control without dynamic compensation, which is described as:

$$\dot{x} = \dot{x}_d + K_B \cdot \tilde{x} \tag{5.49}$$

where $\dot{x}_d \in \mathbb{R}^6$ is the desired body linear and angular velocity, $K_B \in \mathbb{R}^{6\times6}$ denotes the constant that attempts to correct the position error $\tilde{x} \in \mathbb{R}^6$.

5.7 State estimator

Body position estimation is obtained through the FK of the supporting legs. Because the initial state of the robot is known, the estimated body position by one of the modules is straightforwardly computed as:

$$\hat{\Sigma}_{Bi} = {}^{B}\Sigma_i \cdot {}^{i}\Sigma_{iS,t=0} \cdot {}^{i}\Sigma^{-1}_{iS,t=k} \cdot {}^{B}\Sigma^{-1}_i \tag{5.50}$$

where $\hat{\Sigma}_{Bi}$ is the estimation of the position of the body frame by the module i, ${}^{B}\Sigma_i$ the transformation from the body frame to the first joint frame of the module i, ${}^{i}\Sigma_{iS,t=0}$ its FK in the initial state, and ${}^{i}\Sigma_{iS,t=k}$ its FK at time k. Since there is the same number of valid solutions as legs, currently the robot odometry is implemented as the average of them as:

$$\hat{\Sigma}_B = \frac{1}{N}\sum_{i=1}^{N} \hat{\Sigma}_{Bi} \tag{5.51}$$

On the other hand, the mean robot orientation odometry can be computed as:

$$Q = [a_1 q_1, a_2 q_2, ..., a_N q_N] \tag{5.52}$$

where a_i is the weight of the i^{th} quaternion (q_i) averaged, included as a column vector. In this case, given the modularity of the robot, a_i is evenly distributed. Q is therefore a $4 \times N$ matrix.

The normalized eigenvector corresponding to the largest eigenvalue of QQ^T is the weighted average. Since QQ^T is self-adjointed and at least positive, semi-definite, fast, and robust methods of solving that eigen problem are available. Furthermore, this is scalable. Computing the matrix–matrix product is the only step that grows with the number of elements that are averaged.

Robot odometry, which is a simple approach to estimate robot position, can be included in a more complex state estimator, such as the one implemented in [Sesmero et al., 2021].

5.8 Summary

In this chapter, a generalizable, low computational cost, simple, and fast gravity compensation method is presented for climbing robots with a variable number of legs. It is based on the static problem, which is a reduction of the dynamic model of the robot that takes

advantage of the low velocity of climbing robots. To solve it, a method that computes the torque to be applied by each actuator to compensate for the gravitational forces is presented without using the Jacobian matrix for the forces exerted by the end-effector and without using analytical methods for the gravitational components of the model. The proposed method is compared with the most popular method and it is concluded that the proposed one is twice as fast. Using it, a torque-based PD controller is presented for the position and velocity control of the leg modules, which results in a body velocity control without dynamic compensation. The knowledge acquired in this chapter was first introduced in [Prados et al., 2022b] and partially published in [Prados et al., 2023].

Chapter 6

MPC-based path and footfall planning

"A man who dares to waste one hour of time has not discovered the value of life."

Charles Darwin

Climbing robots are deployed in dangerous situations, where power consumption must be optimized to guarantee completion of the task. Possible solutions to reduce power consumption are found in weight reduction, the calculation of the best path, or the optimization of walking and climbing patterns.

Trajectory planning, dynamics, and control form the basic framework for the analysis of legged robots walking, running, and climbing [Shah et al., 2012]. For climbing robots, it is especially noted that during practical applications there are high requirements of energy supply. In this area, some authors focus on optimizing the trajectory based on energy consumption [Li et al., 2018], while others focus on improving motion patterns to reduce consumption and increase travel distance.

A widely used concept to determine the behavior of legged systems is the virtual model control introduced by [Pratt et al., 2001]. It is a motion control framework that uses virtual component simulations to test different alternatives and generate the desired joint torques. Virtual model control is a static method in which the joint torque is adjusted and the reaction is created as if external forces were applied.

In this chapter, relevant definitions of legged locomotion are included in Section 6.1. An analysis about how insects behave to walk and climb under certain disturbances, and how they modify their usual way of doing it. Later, in Section 6.3 a path planner that minimizes energy consumption and optimizes robot safety is proposed. To do so, it fronts the FDP by selecting those configurations in which the robot is most comfortable. Furthermore, the gait controller responsible for determining the walking and climbing sequence is presented in Section 6.4. It is presented as a reactive, non-predefined, and bio-inspired control that selects the most convenient leg to move. In Section 6.5 it is proposed a footfall planner as an optimizer of the best position to step, whereas in Section 6.5.3 the implemented

collision detector between modules is described. Lastly, in Section 6.6, the process of preparing for swing phase and the trajectory itself are presented.

6.1 Previous definitions

There are several relevant definitions of locomotion planning that define how efficient locomotion is in legged and climbing robots [Ding et al., 2010]:

a) Swing/stance phase. A supporting leg is one that rests on the ground/wall and stays stuck to it. A supporting leg is considered to be in the stance phase, whereas a non-supporting leg is in the swing phase.

b) Gait period/cycle time, T. It is a complete cycle of a leg that includes swing and stance phases.

c) Duty factor β. It is the time fraction of the stance phase of a leg with respect to the cycle time T. $\beta = \frac{T_{si}}{T_i}$ where T_{si} denotes the stance phase time of the leg i; T_i denotes the cycle time of the leg i.

d) Stroke length. It is the distance that the body moves through the stance phase of a leg.

e) Stride length. It is the distance the COM translates during one complete locomotion cycle.

f) Kinematics margin. It is defined as the distance from the current foothold of leg i to the boundary of the reachable area of leg i, measured in the opposite direction of body motion.

g) Periodic gait. A gait is periodic if similar states of the same leg during successive strokes occur at the same interval for all legs, that interval being the cycle time.

h) Symmetric gait. A gait is symmetric if the motion of legs of any right-left pair is exactly half a cycle out of phase.

i) Regular gait. A gait is said to be regular if all legs have the same duty factor.

j) Body height. Body height is the distance of the COM of the body from the support surface along the vertical axis of the body.

k) Protraction / retraction of the leg. It is the forward and backward motion, respectively, of the swing leg before touchdown.

l) Lateral offset. It is the shortest distance between the vertical projection of the hip on the ground and the corresponding track.

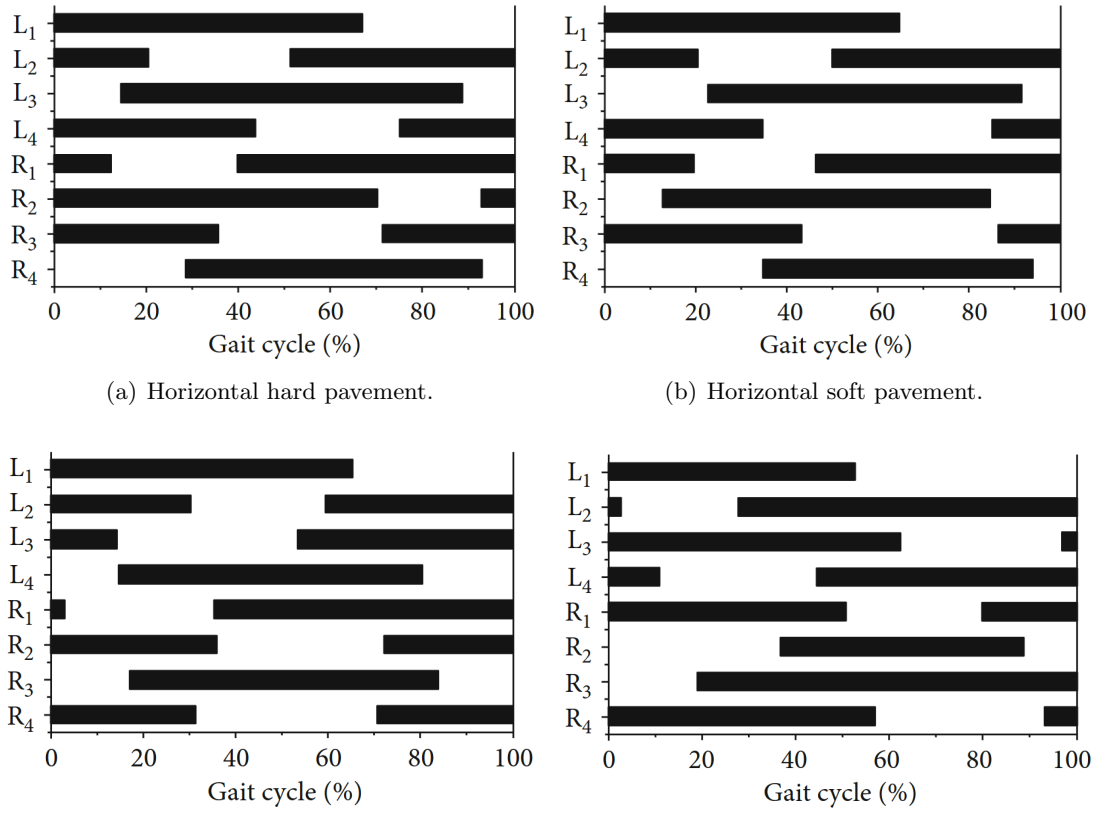
6.2 Locomotion analysis

Legged robots often mimic legged animals, such as humans or insects, as an example of biomimicry. Biomimetics or biomimicry is the emulation of models, systems, and elements

of nature for the purpose of solving complex human problems. Biorobots based on the physiology and methods of locomotion of animals. To do so, it is required to analyze the animal's gait and leg structures. Many bio-inspired robots imitate legs structures but fail in the search for reactive gaits by defining pre-defined gaits within a Central Pattern Generator (CPG) [Bai et al., 2019, Zhang et al., 2023].

This dissertation works with a bio-inspired robot that attempts to imitate insects. Animals decide which leg to move without thinking, according to the comfort they feel in the joints and limbs. In robotics, this solution can be approached with a gait controller that decides according to current state of the robot.

As an example in the animal world, Figure 6.1 shows an analysis of the driving modes under different ground conditions of a spider when moving forward (*Grammostola rosea*). In that study carried out at [Hao et al., 2019], a complete gait cycle was defined as starting when the first leg on the left side of the spider touched the ground and ending the next time that this leg touched the ground. L and R indicate left and right, and the numbers start from the first pair of feet in the anterior-posterior sequence. During an entire cycle, all eight legs went through the stance phase and swing phase for horizontal/sloped hard/soft pavements. As observed in the different cases, the spider adapts the motion sequence to the surface conditions and the inclination of the pavement.



(a) Horizontal hard pavement.

(b) Horizontal soft pavement.

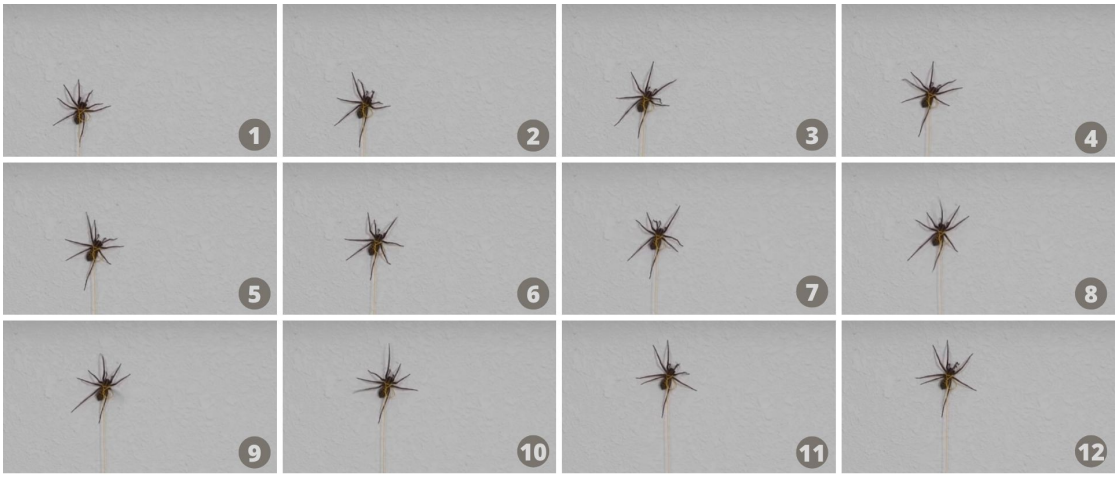
(c) Sloped soft pavement type 1.

(d) Sloped soft pavement type 2.

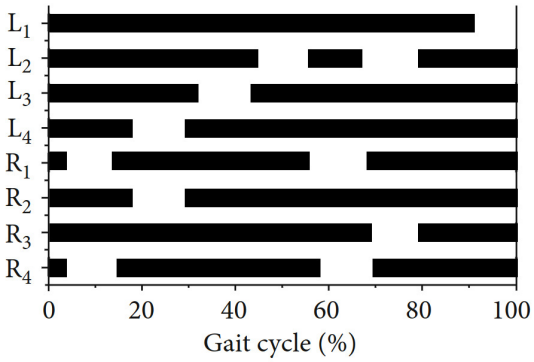
Figure 6.1: Example of a spider locomotion analysis [Hao et al., 2019]. Black boxes represent stance/support phases.

Another example is illustrated in Figure 6.2, where another spider is climbing a vertical

wall. The spider has been tied up with a payload of approximately twice its weight. As observed in Figure 6.2(b), the spider has to adapt its behavior to the conditions of a vertical wall and high payload, and thus does not follow a regular, symmetric and periodic gait pattern. Instead, it feels which leg is under discomfort and steps a given leg into a place where it will be comfortable in a short future. Furthermore, it is observed that the duty factor is higher in the second case compared to the one presented in Figure 6.1. Again, the spider adapts itself to maintain more legs in stance phase because of the adverse and unfavorable conditions of higher payload and vertical motion. As well as before, a complete gait cycle was defined as starting when the first leg on the spider's left side touched the ground and ending the next time that this leg touched the ground.



(a) Screenshots of the spider climbing [1].



(b) Gait cycle. Black boxes represent stance/support phases.

Figure 6.2: Spider locomotion analysis while climbing a vertical wall with a payload of two times the spider weight.

Other examples can be found in the literature. For example, in [Wang et al., 2011] the authors point to interesting details of the gait cycle of the spider *Ornithoctonus huwena*. The body position is moved up and down to raise or lower the COM of the animal, as well as the velocity of the spider increases or decreases depending on the supporting legs. As most approaches focus on pre-defined walking patterns, in [Barrio et al., 2020] authors

[1]Pictures taken from `https://www.youtube.com/watch?v=0-MLh1GsOeg&t=181s`

focus on numerical techniques to detect patterns in biological CPGs. However, it is difficult to predict how a multi-legged animal behaves after losing one leg with a pre-defined gait. Furthermore, a non-general gait for different insects can be only applied to a specific type, and not for all legged animals.

In order to create a general bio-inspired gait, in this dissertation a generic **path planner** is presented first. It optimizes the robot trajectory by looking for the most comfortable and convenient robot configurations and positions. Second, a generic **gait controller** is presented. It reactively defines the best swing/stance phase sequence as the animals do, feeling the comfort of the legs. Lastly, a generic **footfall planner** is included. It selects the best position where the leg on the swing phase has to step by ensuring safety and comfort, and optimizing the distance traveled and energy consumption.

6.3 Path planner: A model predictive control approach

The force distribution problem is one of the unique challenges for legged robots, especially heavy and medium-heavy ones. Based on the reasonable gaits that put the center of gravity within the stability margin, effectively optimizing the foot force distribution is a more difficult problem to solve, especially while maintaining the stability of the robot [Zhuang et al., 2014]. In fact, the multi-chain system created by the supporting legs of a legged robot is tightly associated, creating hyperstatic systems.

In addition, safety analysis is an important consideration when designing climbing robots. There are two dangerous situations that might occur, slipping and falling of the robot. These two circumstances can occur if the suction cup slips or detaches, respectively. Analytically, it is common for the pressure to be produced by the suction cup to be calculated taking into account the force and momentum that appear on it [Bahr et al., 1996]; however, in our case, since the wrist is disabled during coupling to reduce the hyperstaticity of the system during the attachment of the suction cup, no torques appear on the suction cup. In this way, only forces have to be taken into account to detect if the suction cup is at risk of slipping or detaching.

When a robot is moving upon such non-horizontal surfaces, gravity causes deformation of elements of the robot. This deformation, which could be particularly strong in legged robots due to a larger number of joints and less rigid parts, can lead to a deviation of the robot from the given trajectory of movement. In [Akinfiev and Armada, 2002], the authors present a mathematical calculation to estimate the positional loss due to this effect while climbing. This computation can be used to correct the odometry position estimation to improve the result of a more complex state estimator.

Generation of the body trajectory is a fundamental part in ensuring the safety of the robot. A path planner must ensure two main premises in a legged-and-climbing robot. First of all, the gripping system (in this case suction cups) is required to adhere if the system decides to do so. Second, the load that a single motor supports should be as low as possible to avoid overloads.

We have studied in Chapter 5 how to compute the required torque of each joint to maintain the weight of the entire system by compensating gravitatory forces, regardless of the number of legs and their disposal. It is possible to use those tools to optimize the torques generated by the motors.

In addition, the inverse kinematics (IK) problem for this robot is defined as solving

for joint angles from known body and end-effector positions (suction cups contact points). Each module is treated as a serial arm, and the IK for each one is solved independently. The proposed MPC body path planner generates virtual positions (relative to the current position) discreetly in the direction of movement that the user establishes and selects those that optimize the path according to a criterion that is explained later in this dissertation. Thus, a Monte Carlo algorithm [2] composed of four steps is proposed: 1) body position points generation, 2) analysis of the robot configuration under the generated body positions, 3) determine how appropriate the configuration is according to the defined criteria, and 4) path generation by body poses connection. In the following, the four steps are explained in detail.

6.3.1 Step 1 - Sampling

The user or a higher agent in the architecture, via joystick, GUI, or command line, establishes a velocity for the robot body. The algorithm then generates P discrete points λ_p in the direction of the commanded velocity. They are located in their reference frame $\Sigma_{\lambda p}$, with $p \in [1, ..., P]$. Each point represents a position and orientation (translated and rotated) regarding the current position of the robot body, that is, a virtual configuration of the entire robotic organism (Figure 6.3). To generate virtual configurations related to the current body position, it is possible to use two main distributions:

- Random distribution: In this case, the maximum displacement that the robot body can move due to the kinematics of the legs is taken into account, given in Section 3.8.2. For example, Figure 6.4(a) shows two points of view of a random distribution, when the user establishes a positive velocity in the direction X. As we observe in both pictures, there is a random deviation from the line that the robot is expected to follow.

- Grid distribution: In this case, a grid of depth x width x height (DxWxH) is generated according to the configuration established by the user. For example, Figure 6.4(b) shows two points of view of a grid distribution, when the user establishes a positive velocity in the direction X.

6.3.2 Step 2 - Sample analysis

Each relative position of the robot body can be analyzed considering the risk of detachment of the gripping system together with the overload level of the motors. First, the estimation of gravity at $\Sigma_{\lambda p}$ is done by Equation 6.1, where ${}^{\lambda p}\vec{g}$ represents the estimated gravity in the virtual position λ_p, ${}^{B}R_{\lambda p}$ the rotation matrix from the origin point to the generated point λ_p (which is probably not with the same orientation than the original one), and ${}^{B}\vec{g}$ the current gravity in the body.

$$ {}^{\lambda p}\vec{g} = {}^{B}R_{\lambda p} \cdot {}^{B}\vec{g} \tag{6.1} $$

[2] A Monte Carlo algorithm is a non-deterministic randomized algorithm that provides approximate solutions by analyzing a set of samples.

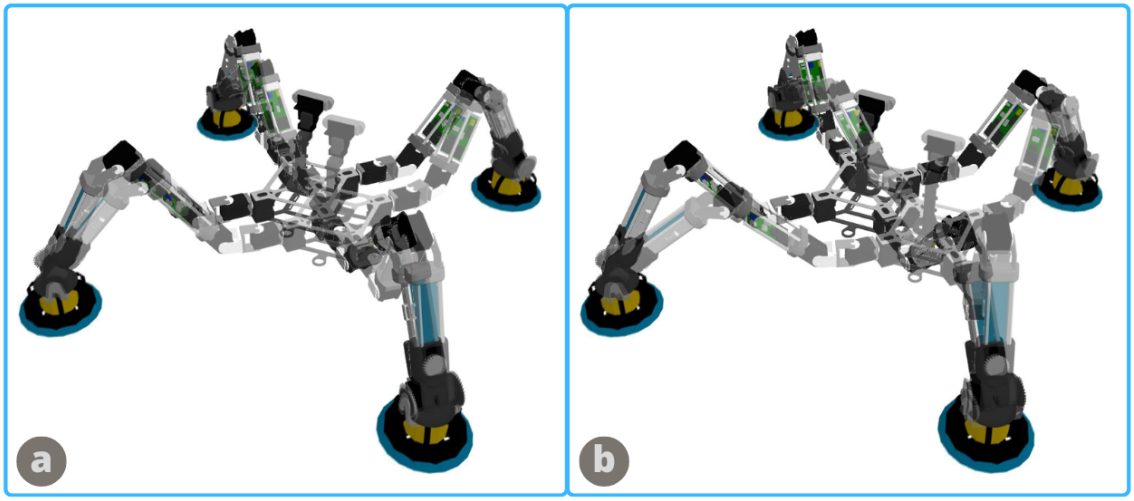
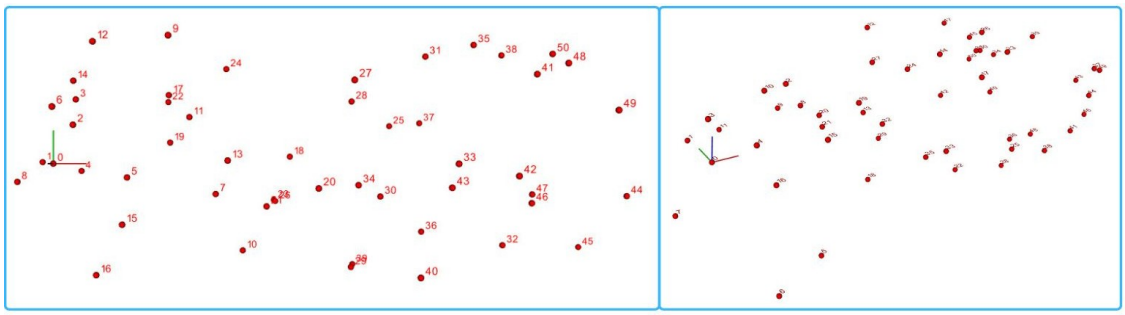
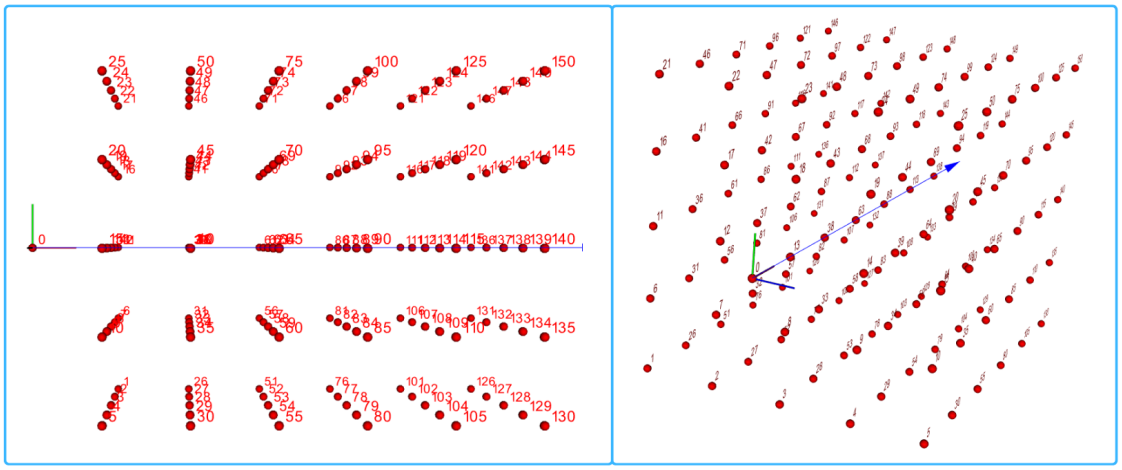


Figure 6.3: Virtual configuration of the organism in a given position. a) and b) represent two examples of randomly generated robot positions.



(a) Example of a random sampling with 50 points in two points of view.



(b) Example of a grid sampling with 6x5x5 points in two points of view.

Figure 6.4: Different alternatives to generate body position points in the direction that the user has established, $v = (0.01, 0, 0)\frac{m}{s}$, which is a preferential movement in the X axis.

Later, it is necessary to estimate all the TCP positions ${}^{i}T_{iS}$, wrist points ${}^{i}T_{iw}$, and

joint configurations at the point λ_p (as if the robot's body were located at that position). The TCP positions and wrist points in the virtual situation where the body is located in ${}^{i}T_{iw}$ can be computed taking into account the current TCP position of the module with the forward kinematic (as indicated in Equation 6.2. Once the TCP position is calculated for the point λ_p, the joint configuration is computed with the inverse kinematic. Thus, the TCP position of the module i at the point λ_p is given by

$$^{\lambda p}\left[{}^{i}T_{iS}\right] = {}^{B}T_{i}^{-1} \cdot {}^{B}T_{\lambda p}^{-1} \cdot {}^{B}T_{i} \cdot {}^{i}T_{iS} \tag{6.2}$$

The COM of the modules is computed in an analogous way to the one of the entire organism. After that, it is possible to compute reaction forces and torques to compensate the gravitational forces of the weight of the components of the robotic organism, as explained in Section 5. Once the required gravity compensation is computed, it is only required to calculate the risk of detachment of the suction cup, r_{sc}. The result of the risk of detachment of the suction cup of module i, $r_{sc,i}$, is given by the percentage that is calculated from the division between the critical force that is required to be attached and the force exerted by the suction cup turbine.

To evaluate the risk of detaching the entire organism, it is proposed that the value of r_{sc} is given by the combination of risk of detachment of the different modules $r_{sc,i}$ as

$$r_{sc} = \prod_{i=1}^{N}(1 - r_{sc,i}) \tag{6.3}$$

Figure 6.5 represents the magnitudes used to calculate the risk of suction cup detachment. In the picture, R represents the radius of the suction cup, ${}^{iw}M$ is the momentum in the WP and ${}^{iw}F_i$ is the force that the suction cup is suffering at the WP. With these parameters, the risk for each of the suction cups, $r_{sc,i}$, is obtained according to the following expression:

$$r_{sc,i} = \frac{F_{c,i}}{F_{sc,i}} \tag{6.4}$$

where $F_{c,i}$ is the minimal force that the suction cup has to generate to avoid detachment, and $F_{sc,i}$ is the force that the turbine is generating. $F_{c,i}$ is obtained based on the maximum between the forces that try to separate the suction cup from the gripping surface ($F_{fall,i}$) and those that try to displace it ($F_{slide,i}$). This is, the maximum between the critical force of detachment and sliding, respectively, whose value is given as

$$F_{fall,i} = \frac{{}^{iw}M}{R} + {}^{iw}F_{i,z} \tag{6.5}$$

$$F_{slide,i} = \frac{\sqrt{{}^{iw}F_{i,x}{}^{2} + {}^{iw}F_{i,y}{}^{2}}}{\mu} \tag{6.6}$$

where μ is the friction factor between the suction cup and the environment material. In the case of ROMERIN, ${}^{iw}M$ arises from the moment generated as a consequence of the gripping force with the surface since the moment produced from the structure of the robot is zero when the wrist is passivized. For simplicity, this component can be neglected, since its value is much lower than that produced by the force component ${}^{iw}F_{i,z}$.

The value of the force that the suction cup is suffering, ${}^{iw}F_i$, can be obtained directly by the reaction forces as:

$$ {}^{iw}F_i = ({}^{B}R_i \cdot {}^{i}R_{iw}) \cdot \begin{bmatrix} F_{i,x} \\ F_{i,y} \\ F_{i,z} \end{bmatrix} \tag{6.7} $$

where ${}^{B}R_i$ is the rotation matrix between Σ_B and Σ_i, and ${}^{i}R_{iw}$ is the rotation matrix between Σ_i and Σ_{iw}.

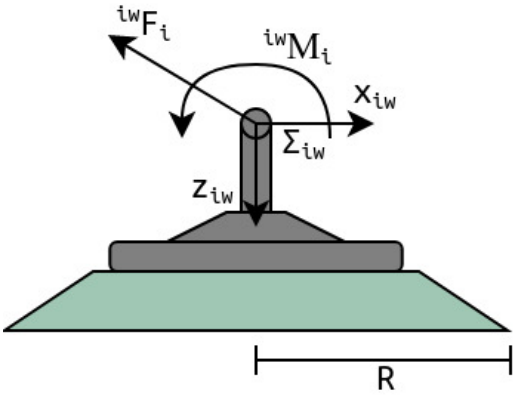


Figure 6.5: Schematic of the suction cup with the main force and moment.

It is important to note that the value of the risk of detachment $r_{sc,i}$ is in the range $[0, \infty]$, being $r_{sc,i} = 0$ the best situation with no risk of detaching, and $r_{sc,i} = 1$ the critical situation where the suction cup generates exactly the same negative pressure as required. In this way, a value equal to or greater than 1 represents that the suction cup is going to detach. Due to the inaccuracies of suction cup manufacturing and safety, a value greater than 0.7 is considered at risk, and a value greater than 0.8 is considered at high risk of detachment. The value of r_{sc} will then be much lower than $r_{sc} = N$ in a risk-free situation.

6.3.3 Step 3 - Sample weighting

In order to weight the risk (robot falling and actuator overload) of each point, different cost functions have been proposed to be minimized by the path planner during the trajectory computation. Given the complexity and importance of the different parameters, a set of cost functions has been empirically generated. Subsequently, their behavior has been studied and based on the results, the strategy that best reflects risk situations has been selected. A summary of the proposed cost functions is as follows.

1. **Cost function 1** is reflected in Equation 6.8 as the weighted sum of the parameters (suction cups detaching risk and actuators overload separately by motors). They are included as a percentage of risk.

2. **Cost function 2** is reflected in Equation 6.9 as the multiplication of the parameters, which represent a percentage of risk.

3. **Cost function 3** is reflected in Equation 6.10 as the multiplication of parameters, which represent the difference between the worst case and the current case (the higher the value, the better the situation). The cost function is inverted to be minimized during optimization.

4. **Cost function 4** is reflected in Equation 6.11 as the multiplication of parameters, which represent the percentage of the margin to get the worst situation (the higher the value, the better the situation). It is multiplied by a non-linear cost factor that penalizes those risky situations. The cost function is inverted to be minimized during optimization.

5. **Cost function 5** is reflected in Equation 6.14 as the weighted sum of the parameters. It is a combination of the third and fourth functions that normalizes the parameters separately to increase the difference between points.

6. **Cost function 6** is reflected in Equation 6.18 as the weighted sum of the parameters. It is an extension of the fifth function that penalizes positions when motors with the same ID in different modules differ their value (in order to distribute the load between modules). In addition, the cost factor is converted to a continuous function.

6.3.3.1 Cost function 1

The first proposed cost function is denoted as

$$C_p = \prod_{i=1}^{N} \left[W_{t,0} \cdot \frac{\tau_{i0}}{\tau_{i0,max}} + W_{t,1} \cdot \frac{\tau_{i1}}{\tau_{i1,max}} + W_{t,2} \cdot \frac{\tau_{i2}}{\tau_{i2,max}} + W_{sc} \cdot r_{sc} \right] \tag{6.8}$$

where C_p is the cost value of the sampling point λ_p, $W_{t,j}$ and W_{sc} is the weight (importance) that the user establishes to avoid torque overload in the joint j and suction cup detachment, respectively, τ_j the torque in the joint j, $\tau_{j,max}$ the maximum torque of the joint j, and r_{sc} the safety factor of suction cups. The cost function penalizes the percentage of torque that each motor applies with respect to the stall torque, and the percentage risk of detachment of each suction cup r_{sc}. As shown in the equation, the function computes the sum of all components.

This function has been noticed to be inefficient due to a wrong penalization of risk situations (detailed later). The main reason is because it is strongly related to the value of the weights and because there is a very small difference between the cost value of all the sampling points.

6.3.3.2 Cost function 2

Similarly to the first cost function, the second one is defined as

$$C_p = \prod_{i=1}^{N} \left(\frac{\tau_{i0}}{\tau_{i0,max}} \right) \cdot \left(\frac{\tau_{i1}}{\tau_{i1,max}} \right) \cdot \left(\frac{\tau_{i2}}{\tau_{i2,max}} \right) \cdot (r_{sc}) \tag{6.9}$$

This function also penalizes the percentage of torque that each motor applies with respect to the stall torque, and the percentage risk of detachment of each suction cup r_{sc}. However, in this second case the function computes the multiplication of them. In this case, the weights are removed because they affect all the points' cost equally.

This second function has also been noticed to be inefficient because of a wrong penalization of risk situations (detailed later). The main reason is that there is a very small difference between the cost values of all the sampling points.

6.3.3.3 Cost function 3

Alternatively, the third cost function is reflected in Equation 6.10. It is proposed to improve the penalization of those cases that are more contentious.

$$C_p^{-1} = \prod_{i=1}^{N} (\tau_{i0,max} - \tau_{i0}) \cdot (\tau_{i1,max} - \tau_{i1}) \cdot (\tau_{i2,max} - \tau_{i2}) \cdot (r_{sc,max} - r_{sc}) \tag{6.10}$$

6.3.3.4 Cost function 4

Similarly to the third function, in order to penalize those situations where a small set of joints is suffering the most, the third function is converted to nonlinear in the fourth one as

$$C_p^{-1} = \prod_{i=1}^{N} f_{cost} \cdot \frac{\tau_{0,max} - \tau_0}{\tau_{0,max}} \cdot \frac{\tau_{1,max} - \tau_1}{\tau_{1,max}} \cdot \frac{\tau_{2,max} - \tau_2}{\tau_{2,max}} \cdot \frac{r_{sc,max} - r_{sc}}{r_{sc,max}} \tag{6.11}$$

where the cost factor f_{cost} tries to penalize those worse cases, and whose value is given by the multiplication of the cost factors of each actuator risk, $f_{cost,ij}$, as

$$f_{cost} = f_{cost,i0} \cdot f_{cost,i1} \cdot f_{cost,i2} \tag{6.12}$$

The value of $f_{cost,ij}$ is defined by a non-linear piecewise function (values can vary) as

$$f_{cost,ij} = \begin{cases} 0.1 & if \quad \dfrac{\tau_{ij}}{\tau_{ij,max}} > 0.75 \\ 0.25 & if \quad 0.75 \geq \dfrac{\tau_{ij}}{\tau_{ij,max}} > 0.5 \\ 0.75 & if \quad 0.5 \geq \dfrac{\tau_{ij}}{\tau_{ij,max}} > 0.25 \\ 1.0 & if \quad \dfrac{\tau_{ij}}{\tau_{ij,max}} \leq 0.25 \end{cases} \tag{6.13}$$

6.3.3.5 Cost function 5

This alternative is proposed in Equation 6.14 as a function that allows the user to manually select the most important terms in the cost function. Being $\boldsymbol{C}$ the vector composed of P items denoted as $C_p \in \boldsymbol{C}$. Each item represents the cost value of the sampling point λ_p (given by the analysis carried out for each point in Step 2). The cost value of each sampling point C_p is given as

$$C_p = W_t \cdot C_{p,t} + W_{sc} \cdot C_{p,sc}, \tag{6.14}$$

where $C_{p,t}$ and $C_{p,sc}$ are the cost values related to the torque and suction cup safety parameters, respectively. The sets of P cost values create two cost vectors $\boldsymbol{C_t}$ and $\boldsymbol{C_{sc}}$, in such a way that $C_{p,t} \in \boldsymbol{C_t}$ and $C_{p,sc} \in \boldsymbol{C_{sc}}$.

The vectors $\boldsymbol{C_t}$ and $\boldsymbol{C_{sc}}$ are the normalized version of the vectors $\boldsymbol{\alpha_t}$ and $\boldsymbol{\alpha_{sc}}$, to evenly assess the different parameters (or unevenly with the weights W_t and W_{sc}):

$$\boldsymbol{C_t} = \frac{\boldsymbol{\alpha_t}}{||\boldsymbol{\alpha_t}||}, \quad \boldsymbol{C_{sc}} = \frac{\boldsymbol{\alpha_{sc}}}{||\boldsymbol{\alpha_{sc}}||} \tag{6.15}$$

The vectors $\boldsymbol{\alpha_t}$ and $\boldsymbol{\alpha_{sc}}$ are composed of components P, in such a way that $\alpha_{p,t} \in \boldsymbol{\alpha_t}$, and $\alpha_{p,sc} \in \boldsymbol{\alpha_{sc}}$. The values of $\alpha_{p,t}$ and $\alpha_{p,sc}$ are then given as

$$\alpha_{p,t} = \prod_{i=1}^{N} \prod_{j=0}^{J-3} f_{cost,ij} \cdot (\tau_{ij,max} - \tau_{ij}) \tag{6.16}$$

$$\alpha_{p,sc} = r_{sc,max} - r_{sc} \tag{6.17}$$

6.3.3.6 Cost function evaluation

Once possible cost functions have been proposed that allow the planner to obtain the best path, an evaluation of them in different cases is carried out. Figure 6.6 shows the result of the proposed cost functions on a set of samples in the different cases. These four cases are considered representative of the most important situations that we want to evaluate in order to avoid them:

- **Case 1.** The weight is distributed among modules. This case is the most desired.
- **Case 2.** One leg supports the weight of the organism. This case is undesirable because it may lead to overloading of one leg.
- **Case 3.** One joint has a lot of load. This case is also undesirable because it would produce a failure in one of the legs (due to the overload of a motor).
- **Case 4.** The weight of the organism falls mainly at the second joint of all the modules.

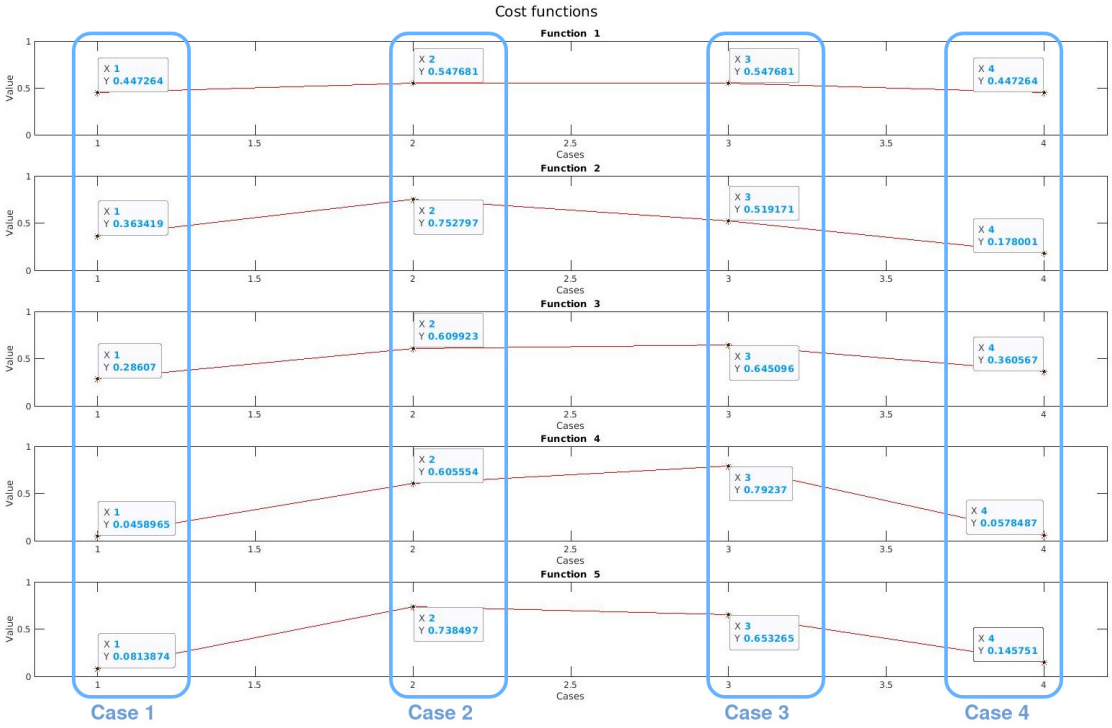


Figure 6.6: Result of the different cost functions over a set of samples (values for the different cases are normalized for each cost function).

It is observed how the last three cost function proposals penalize more strongly the cases that have been classified as worst. From this analysis, although there are few experiments, it is possible to deduce the following:

- Cost function 1 do not penalized correctly risk situations. In fact, the evaluation is more or less the same in all cases. As a consequence, when planning is applied, the resulting trajectory will be directly a straight line, since the cost function will not have a relevant influence.
- Cost function 2 penalizes better the second and third cases, but it penalizes more the best case against the fourth case.
- Cost function 3 penalizes more those dangerous cases.
- Cost function 4 improves the previous one by increasing the difference between the hazard cases and the desired cases.
- Cost function 5 makes a good differentiation between cases and allows the user to give more or less importance to the torque and suction cup safety.

6.3.3.7 Implemented cost function

Given the experience of the previous cost functions, Equation 6.18 is proposed as the definitive cost function. It is considered an extension of the cost function 5. It includes a component that penalizes situations when motors with the same ID in different modules

differ in value greatly (the third component of the equation). Furthermore, the cost factor is not a piecewise function in this case, using an exponential formula. The cost function is then given by

$$C_p = W_t \cdot C_{p,t} + W_{sc} \cdot C_{p,sc} + W_{std} \cdot C_{p,std} \tag{6.18}$$

As well as in the previous cost function, $C_{p,t}$ and $C_{p,sc}$ are the cost values related to the torque and suction cup safety parameters, respectively. However, in this case, $C_{p,std}$ is included as the standard deviation cost value to penalize situations when actuators with the same ID in different modules differ in their value greatly. The sets of P cost values create three cost vectors of P size $\boldsymbol{C_t}$, $\boldsymbol{C_{sc}}$ and $\boldsymbol{C_{sc}}$, in such a way that $C_{p,t} \in \boldsymbol{C_t}$, $C_{p,sc} \in \boldsymbol{C_{sc}}$ and $C_{p,std} \in \boldsymbol{C_{std}}$.

Similarly to the process of the fifth cost function in Equation 6.15, the vectors $\boldsymbol{C_t}$, $\boldsymbol{C_{sc}}$ and $\boldsymbol{C_{std}}$ are the normalized version of the vectors $\boldsymbol{\alpha_t}$, $\boldsymbol{\alpha_{sc}}$ and $\boldsymbol{\alpha_{std}}$, to accurately evaluate the different parameters (or unevenly with the weights W_t, W_{sc} and W_{std}). These vectors are also composed of P components, in such a way that $\alpha_{p,t} \in \boldsymbol{\alpha_t}$, $\alpha_{p,sc} \in \boldsymbol{\alpha_{sc}}$ and $\alpha_{p,std} \in \boldsymbol{\alpha_{std}}$. The cost value of each point is given as

$$\alpha_{p,t} = \prod_{i=1}^{N} \prod_{j=0}^{J-3} e^{-K \cdot \frac{\tau_{ij}}{\tau_{ij,max}}} \cdot \left(\frac{\tau_{ij,max} - \tau_{ij}}{\tau_{ij,max}} \right) \tag{6.19}$$

$$\alpha_{p,sc} = r_{sc,max} - r_{sc} \tag{6.20}$$

$$\alpha_{p,std} = \sum_{j=1}^{J} \sigma_j = \sum_{j=1}^{J} \sqrt{\frac{1}{N} \sum_{i=1}^{N} (|\tau_{ij}| - \bar{\mu}_j)^2} \tag{6.21}$$

where K may vary its value to penalize more those high torque values, and where μ_j is given as

$$\bar{\mu}_j = \frac{1}{N} \sum_{i=1}^{N} |\tau_{ij}| \tag{6.22}$$

However, before normalizing the vectors, the bias is removed for all vectors $\boldsymbol{\alpha_t}$, $\boldsymbol{\alpha_{sc}}$ and $\boldsymbol{\alpha_{std}}$. Since the cost value is desirable to be as small as possible, for the first two vectors $\boldsymbol{\alpha_t}$ and $\boldsymbol{\alpha_{sc}}$ the cost value $\alpha_{p,t}$ and $\alpha_{p,sc}$ is subtracted from the maximum value of the vectors. This is because the larger $\alpha_{p,t}$ and $\alpha_{p,sc}$ are, the better the situation. For the remaining vector $\boldsymbol{\alpha_{std}}$, the minimum value of the vector is subtracted from the cost value $\alpha_{p,std}$. In this case the lower $\alpha_{p,std}$ is, the better the situation.

$$\alpha_{p,t} = max(\boldsymbol{\alpha_t}) - \alpha_{p,t} \tag{6.23}$$

$$\alpha_{p,sc} = max(\boldsymbol{\alpha_{sc}}) - \alpha_{p,sc} \tag{6.24}$$

$$\alpha_{p,std} = \alpha_{p,std} - min(\boldsymbol{\alpha_{std}}) \tag{6.25}$$

To understand it with an example, for an ideal point where $\alpha_{p,t}$ and $\alpha_{p,sc}$ are the largest of their respective vectors and $\alpha_{p,std}$ the smallest of $\boldsymbol{\alpha_{std}}$, the values would be reduced to zero and the final cost value C_p would be zero.

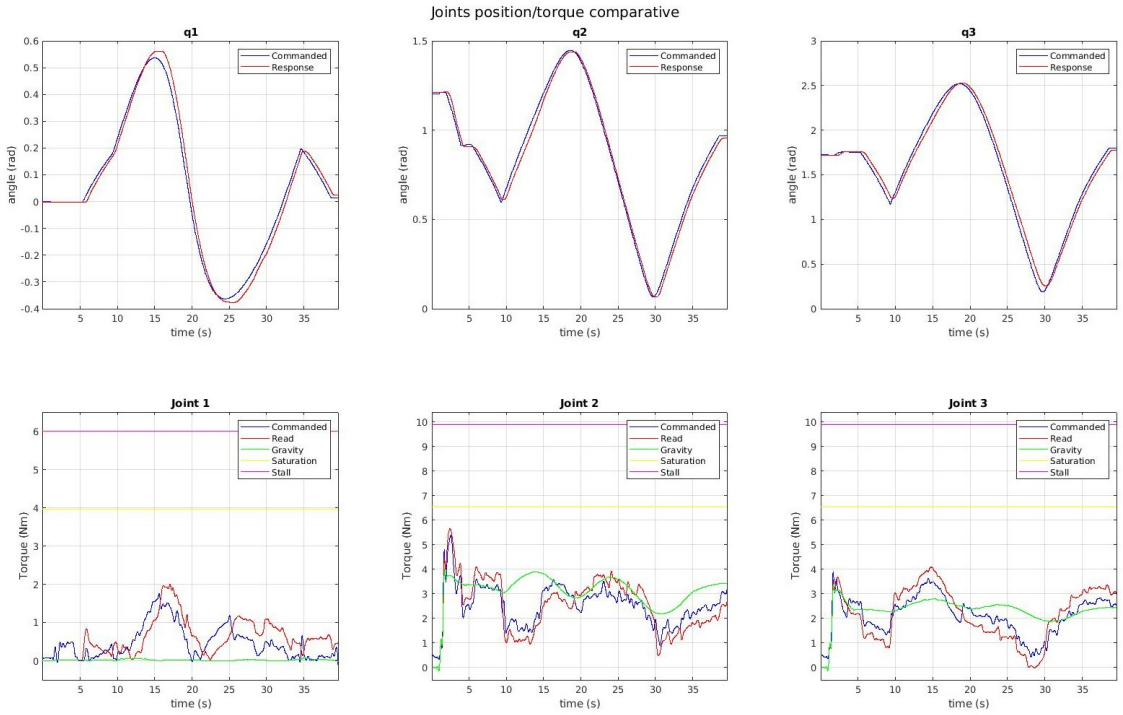


Figure 6.7: Position and torque comparative of one of the modules during a real trajectory.

In order to give more veracity to the results, a body trajectory with all the modules attached to the environment has been recorded. It is represented in Figure 6.7, where it is possible to observe the trajectory of the joints (joints 1-3). It is also shown the commanded torque to follow the trajectory, the torque suffered by the joint (simulated), the required torque to compensate gravity (Chapter 5), the established saturation torque to avoid overloads, and the stall torque. To analyze the most critical situations, the most representative local maximum is obtained in the computed torque to compensate gravity (Figure 6.8). To do this, given the noise level of the signal, it has been decided to identify the maximums of the curve resulting from an approximation by a cubic spline.

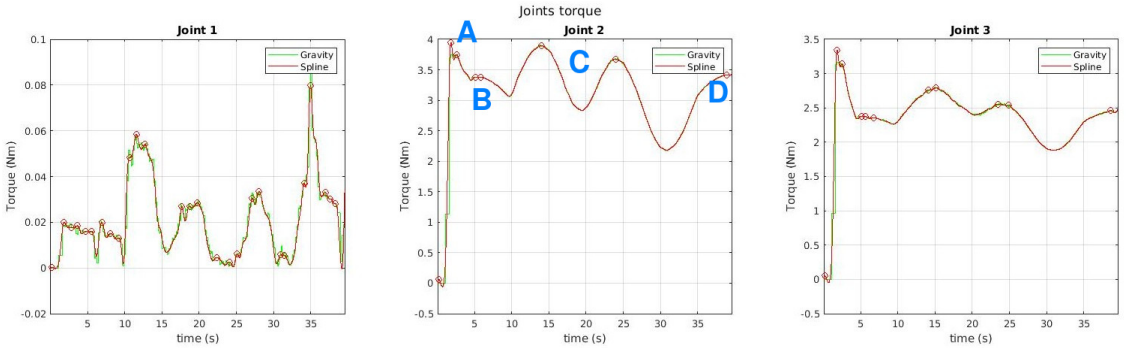


Figure 6.8: Local maximum detected in the gravity compensation torques of the second joint (extracted of Figure 7.9).

Since the torques of the first motor of the chain are very low and the peaks of the second and third motors are located in a similar place, we have selected those maximum torques of motor 2. For each case, we have tested the costs functions and are represented in Figure 6.9. The peaks are split into four blocks:

- A: High torques distributed between all modules. The standard deviation between the load that modules support is very low, but torques are high, so cost is medium high.
- B: Torques are lower than in block A and they are distributed between all modules, so cost is medium low.
- C: High torques with a high variance between the torques that each module is supporting. The cost function is the highest.
- D: Medium torque with low variance. The cost function is medium low.

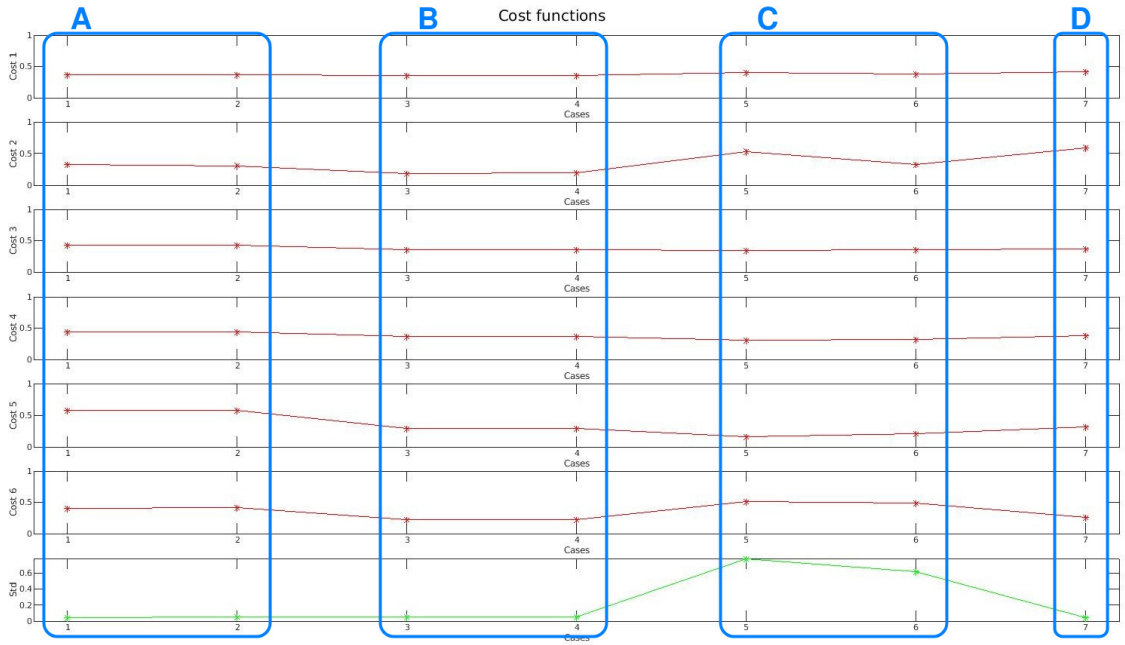


Figure 6.9: Cost values of the six proposed cost functions. The last subplot includes the standard deviation between the torques to compensate gravity of motors with the same ID of different modules. Cases A-D are related to peaks detected in second joint of Figure 6.8.

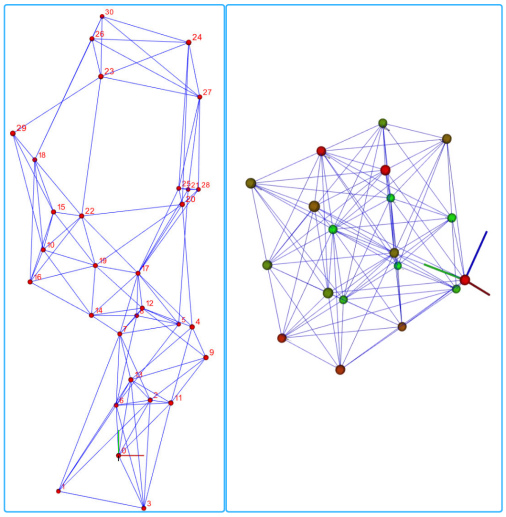
In summary, Equation 6.18 is proposed as the heuristic cost function to be minimized by the path planner during the trajectory computations. It includes a component that penalizes configurations when joints with the same ID in different modules differ in value greatly. Being $\boldsymbol{C}$ the vector composed by P items denoted as $C_p \in \boldsymbol{C}$, which represent the cost function of the path planner that is desired to be as low as possible. Being $\boldsymbol{C'_t}$, $\boldsymbol{C'_{sc}}$, and $\boldsymbol{C'_{std}}$ the cost function vectors related to the torque, suction cup safety, and standard deviation parameters, with $C_{p,t} \in \boldsymbol{C_t}$, $C_{p,sc} \in \boldsymbol{C_{sc}}$, and $C_{p,std} \in \boldsymbol{C_{std}}$. And being $\boldsymbol{C_t}$, $\boldsymbol{C_{sc}}$, and $\boldsymbol{C_{std}}$ the normalized values of $\boldsymbol{C'_t}$, $\boldsymbol{C'_{sc}}$, and $\boldsymbol{C'_t}$ to evenly evaluate the different

parameters (or unevenly with the weights W_t, W_{sc}, and W_{std}), and K the constant term that vary to penalize more those high torque values. Finally, the components $\boldsymbol{C''_{p,t}}$, $\boldsymbol{C''_{p,sc}}$, and $\boldsymbol{C''_{p,std}}$ are included to remove the bias.

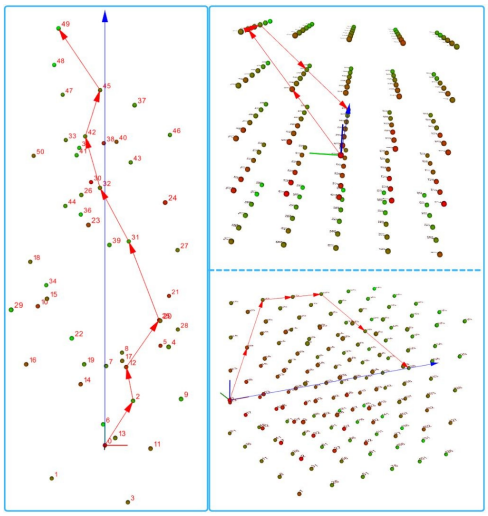
6.3.4 Step 4 - Graph generation and path finding

In order to find the best path, a weighted and directed k-nearest neighbor graph (k-NNG) is proposed. The k-NNG is a graph in which two vertices p and q are connected by an edge, if the distance between p and q is among the k-th smallest distances from p to the others.

The graph is composed of vertices that represent body positions, and edges that connect the neighborhood, as exemplified in Figure 6.10(a). Each vertex (virtual position) has an associated cost value according to the cost function explained in Section 6.3.3.7. The weight of the edges is assigned with the cost value of the vertex that the edge points (directed graph). To avoid safety problems and to reduce computational cost, vertices with a higher risk weight of security breach are removed from the graph.



(a) Example of two generated graphs. The number of items has been reduced because of representation purposes. The example of the random sampling has 30 vertices and 5 neighbours. The example of the grid sampling has a 2x3x3 structure.



(b) Computed trajectory for a commanded velocity of $v = (0, 0.01, 0)\frac{m}{s}$.

Figure 6.10: Example of graphs, and computed trajectory for random and grid samplings.

Dijkstra algorithm is applied to find the safer path from the origin to the target point (Figure 6.10(b)). Before applying this algorithm, a compensation is carried out to avoid that the algorithm looks for the shortest euclidean path without taking into account the edges weight. To do so, each edge weight is multiplied by the distance between the two associated vertices.

The destination point is located on the direction line in such a way that the final position of the robot best responds to the commanded trajectory by the user. After finishing a trajectory, a new trajectory is generated from the position where the robot has

stopped (which may be different from the target point because a recollocation has been required). If the stopped point is different from the target point, the new target point continues as the previous target point of the new trajectory in order to keep the desired direction of movement that the user has established.

6.3.5 Coordination of elements

As described in Chapter 4, the path planner is located in the higher level (Scheduler). To coordinate the graph generation with the robot status, the flowchart illustrated in Figure 6.11 is proposed. If any module is in the swing phase (determine by the **gait controller** explained in future sections), it waits for all modules to be in the stance phase. Otherwise, under non-exception received from the **gait controller**, it computes a trajectory and sends the trajectory commands (time, position, and velocity steps) to be executed by the robot (being coordinated by the **state manager**). In case of receiving an exception, it extracts the last module of the exception list and searches for a feasible body position to lift that module. It creates a trajectory to move the body to that position and sends the trajectory commands. Later on, it waits for the module to finish the swing phase and, if no more exceptions have been detected, it starts again.

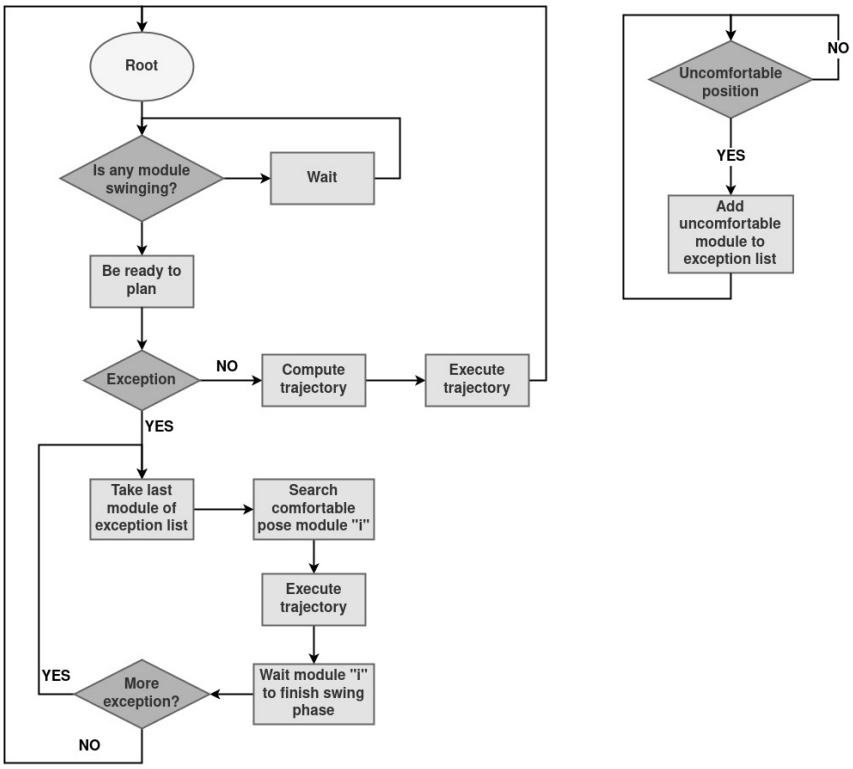


Figure 6.11: Flowchart of the path planner.

6.4 Gait Controller

In this chapter, a reactive, non-periodic, non-symmetric, and non-regular bio-inspired gait controller is presented. Its behavior is defined as a system that constantly checks if the robot is safe. Thus, the gait controller continuously checks the risk of suction cup detachment using Equation 6.4. As indicated previously, two limits to $r_{sc,i}$ are established, the critical percentage and the limit percentage. The **critical percentage** indicates a real risk of detachment of the suction cup, and the robot is asked to stop motion waiting for a solution from the user. The **limit percentage** indicates that the suction cup is entering a complicated and committed situation. In this second case, the gait controller sends an exception to both the State Manager and the Path Planner (including the exception to the uncomfortable module list in Figure 6.11). The values of critical and limit percentages are adjustable parameters and, therefore, they might vary according to robot dimensions, suction cup performance, and weight/vacuum ratio. In ROMERIN, they have been experimentally established in 75% and 50%, respectively.

Additionally, the gait controller continuously checks that the modules are comfortable from a positional point of view and that the torque to compensate the gravitational forces is in a range. The following conditions are presented to determine whether a leg module is in comfortable circumstances:

- The leg module is not too shrunk or stretched. Otherwise, the situation would be undesired because the stability may be compromised, the weight of the organism might not be properly distributed, the range of motion is smaller, there could be excessive mechanical stress on some joints, the maneuverability may be worse, and gait may become inefficient and slow.

  To avoid a leg moving in consecutive occasions and to improve the gait, two aggregated conditions are also included: a) if the module has been recently moved (meaning to be in swing phase), and b) if the leg were more comfortable thanks to the movement of the robot's body. Thus, it is considered comfortable if the following condition is fulfilled

$$\phi_{x,min} - \kappa_1 \cdot \frac{\rho_i}{N} + \kappa_2 \cdot {}^i\vec{v} \; < \; {}^i x_{iS} \; < \; \phi_{x,max} + \kappa_1 \cdot \frac{\rho_i}{N} + \kappa_2 \cdot {}^i\vec{v} \tag{6.26}$$

  where $\phi_{x,min}$ and $\phi_{x,max}$ are the base low and high range limits in X direction of Σ_i, κ_1 is the maximum distance given by the first aggregated condition, ρ_i is the position in a list that include the swing phase past sequence (a module that has recently moved is in the last position, whereas a module that has moved a long time ago is in the first positions), κ_2 is the maximum distance given by the second aggregated condition, and ${}^i\vec{v}$ is the robot body commanded velocity in the module base frame Σ_i, which is calculated as

$${}^i\vec{v} = {}^B R_i^{-1} \cdot {}^B\vec{v} \ . \tag{6.27}$$

- The leg module is not too crooked. Similarly to the previous point, the opposite sit-

uation is undesired for the same reasons. Additionally, the first aggregated condition is added, that is, if the module has been recently moved (meaning to be in swing phase). Therefore, it is considered to be under comfort if the following condition is fulfilled

$$-\phi_y - \kappa_1 \cdot \frac{\rho_i}{N} \; < \; {}^i y_{iS} \; < \; \phi_y + \kappa_1 \cdot \frac{\rho_i}{N} \tag{6.28}$$

where ϕ_y is the base lateral range limit in Y direction of Σ_i.

- The leg module is not too high or low. Apart from the problem of the situation explained in previous points, if the leg module is too high, that means that the organism height is also high. With a high height of the body of the organism, the required torque to compensate gravity forces is much higher while climbing an inclined or vertical wall.

  In this case, the first aggregated condition has also been added. Therefore, it is considered to be comfortable if the following condition is fulfilled

$$\phi_{z,min} - \kappa_1 \cdot \frac{\rho_i}{N} \; < \; {}^i z_{iS} \; < \; \phi_{z,max} + \kappa_1 \cdot \frac{\rho_i}{N} \tag{6.29}$$

  where $\phi_{z,min}$ and $\phi_{z,max}$ are the base low and high range limit in Z direction of Σ_i.

- The joints are between a reasonable and comfortable range of positions. In the opposite situation, there could be problems related to the range of motion of the body, mechanical stress, maneuverability problems, and singularities.

  In this case, the first aggregated condition has been added. Thus, it is considered to be under comfort if the following condition is met

$$\phi_{qj,min} - \kappa_{qj} \cdot \frac{\rho_i}{N} \; < \; q_j \; < \; \phi_{qj,max} + \kappa_{qj} \cdot \frac{\rho_i}{N} \tag{6.30}$$

  where $\phi_{qj,min}$ and $\phi_{qj,max}$ are the base low and high range limits of the joint j, and κ_{qj} is the maximum angle given by the first aggregated condition.

- The joints are within reasonable torques to compensate the gravity forces. In the opposite situation, the most problematic issue is that actuator overload could occur. This situation must be avoided because the loss of one actuator for a certain time can seriously compromise the organism behavior.

  Thus, it is considered to be under comfort if the following condition is fulfilled

$$-\phi_{j,tau} \cdot \tau_{j,max} \; < \; \tau_{g,j} \; < \; \phi_{j,tau} \cdot \tau_{j,max} \tag{6.31}$$

  where $\phi_{j,tau}$ is the percentage of the stall torque to throw an exception in the joint j, and $\tau_{j,max}$ is its stall torque.

In all the proposed situations, if one of them is not fulfilled, the gait controller sends an exception to both the **State Manager Node** and **Path Planner Node**, as indicated in the control architecture scheme. The exception is used to move the leg to a new position by

6.5 Footfall planner: A model predictive control approach

A crucial part of legged robots is to decide where to step in the environment, that is, generate footfall points for their locomotion. Providing a properly tuned heuristic cost function, footfall planners based on graph-search can quickly converge to a feasible solution when there is not sufficient time to plan the entire contact sequence [Griffin et al., 2019]. In this section, different alternatives for determining where the leg in the swing phase should step are analyzed. According to the advantages and drawbacks of the proposed methods, the most convenient has been selected and explained in detail later in this section. Similar to the path planner, this alternative is also based on a Monte Carlo algorithm, which is a non-deterministic randomized algorithm that provides approximate solutions by analyzing a set of samples.

6.5.1 Evaluation of alternatives

Several strategies have been defined for generating the next point where a leg in the swing phase should step. In such a way that, after testing and evaluating them, the one with the best performance has been chosen for different situations. The alternatives are as follows:

1. **Based on the future configuration of the organism**. In this alternative, a virtual state of the organism is created in a future position according to the robot motion. This alternative is illustrated in Figure 6.12(a), where the future module position after the swing phase is computed with the virtual state created. The future module position is denoted as ${}^{i}T_{iS}{}'$ and computed as

$$ {}^{i}T_{iS}{}' = {}^{B}T_{i}{}^{-1} \cdot {}^{B}T_{B'} \cdot {}^{B}T_{i} \cdot {}^{i}T_{iS,0} \tag{6.32} $$

   where ${}^{B}T_{B'}$ is the homogeneous matrix from the current body position, $\sum_B$, to the virtual one, $\sum_B'$; and ${}^{i}T_{iS,0}$ is a comfortable position of the module obtained experimentally.

   It has been checked that this methodology works well with rotatory motion, but fails with translations since it is very sensitive to user parameters.

2. **Based on the velocity vector projected to the module**. In this alternative (illustrated in Figure 6.12(b)) the future module position after the swing phase, ${}^{i}T_{iS}{}'$, is given in Equation 6.33. This alternative focuses only on the movement of the legs. To do so, the position of the leg after the swing phase is computed as the displacement of the TCP position of the module according to the body velocity referred to the reference frame of the module Σ_i.

$$ {}^{i}T_{iS}{}' = {}^{i}T_{iS,0} \cdot {}^{iS,0}T_{iS'} \tag{6.33} $$

   where ${}^{iS,0}T_{iS'}$ is given by projecting the body velocity vector in the module TCP.

   This methodology has been denoted to work well with translations, but fails with rotatory motions.

3. **Combination of the two previous methods**. This methodology gathers the two advantages of the previous points. To do so, the computation of the TCP future position is computed taking into account the rotation in the first method and the translation with the second one. Both methods give a target TCP position where the module would step. Given the result of the first alternative:

$$[x_{tcp,1}, y_{tcp,1}, z_{tcp,1}, \Psi_{tcp,1}, \theta_{tcp,1}, \varphi_{tcp,1}] \tag{6.34}$$

and the result of the second one:

$$[x_{tcp,2}, y_{tcp,2}, z_{tcp,2}, \Psi_{tcp,2}, \theta_{tcp,2}, \varphi_{tcp,2}]\,. \tag{6.35}$$

The combination of both results is given by pondering them according to the commanded linear and angular velocity. If the linear velocity is higher than the angular velocity, the second option has more influence. Otherwise, the first method is more influential. The proposed weighting, ζ, which is dimensionless, is given by the following expression

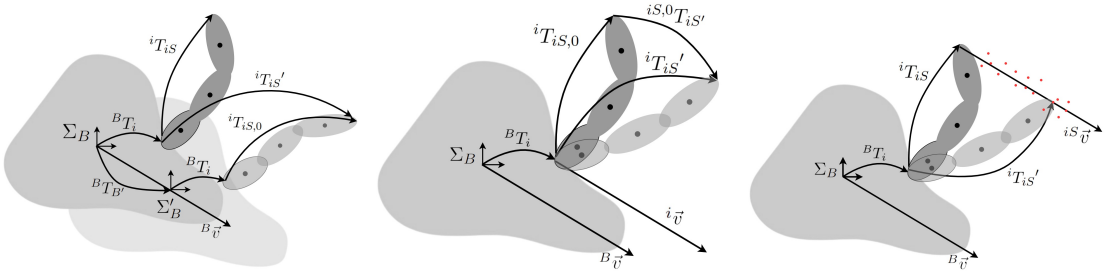
$$\zeta = \frac{\frac{||^B\vec{v}||}{v_{max}}}{\frac{||^B\vec{v}||}{v_{max}} + \frac{||^B\vec{\omega}||}{\omega_{max}}} \tag{6.36}$$

where ${}^B\vec{v}$ is the commanded linear body velocity, ${}^B\vec{\omega}$ the angular, and v_{max} and ω_{max} the maximum linear and angular body velocities established by the user. Since the height at which the module would step and the TCP orientation is given by a predefined comfortable position, the positions x_{tcp} and y_{tcp} are given by

$$x_{tcp} = \zeta \cdot x_{tcp,2} + (1-\zeta) \cdot x_{tcp,1} \tag{6.37}$$
$$y_{tcp} = \zeta \cdot y_{tcp,2} + (1-\zeta) \cdot y_{tcp,1} \tag{6.38}$$

4. **Based on a footprint generator (footfall planner)**. As illustrated in Figure 6.12(c), this methodology implements a Monte Carlo algorithm (sampling-based) to estimate the best position to step. It is more complex and is explained further in Chapter 6.5.

The different strategies have been tested with the digital twin of ROMERIN. As denoted previously, the first alternative works well with rotatory motions, and the second one with translations. The third option works well with both rotatory and translation motions, but it does not optimize the position where the leg will step. Therefore, the last option, which uses a footfall planner, is considered the best option, since it optimizes the robot velocity and safety. Since the implementation of this alternative is not trivial, the following subsections have been dedicated to its explanation.



(a) Based on the future configuration of the organism.

(b) Based on the velocity vector projected to the module.

(c) Based on a footprint generator.

Figure 6.12: Different alternatives to decide where a leg should step.

6.5.2 Footfall planner steps

The proposed footfall planner is designed to walk/climb in a regular plane. An example of a similar to the proposed MPC footfall planner is found in [Shim et al., 2023], where the authors determine which is the best footstep by using a topology-based approach that enables MPC to simultaneously plan full-body motions, torque commands, footstep placements, and contact surfaces in real time. Thus, similarly to the third method presented in [Webster et al., 2022], we propose a sampling-based footfall planner that considers all possible footsteps, removes those that are not valid, and selects the best footfall point according to a cost function.

The footfall planner follows a 5 steps process: plane detection, contact point projection, random sampling, sample filtering, and sample weighting. To reduce complexity, the following used frames and positions are referred to the body reference frame $\sum_B$ by default, since it serves as the origin point for the whole process.

6.5.2.1 Step 1 - Plane detection

To detect the plane in which the modules are attached, the TCP position of each module is used. Two options are proposed for computing the average plane defined by the points located in each TCP position (referred to the body reference frame), as indicated in Figure 6.13.

1. By using the singular value decomposition (SVD), which is a matrix factorization found in a large number of engineering problems.
2. By means of the least square problem. This solution is less numerically stable but is faster than the first option, and so it has been selected in this dissertation.

To compute the average plane defined by points located at each TCP position, first of all, a plane is defined in its common form:

$$P : a \cdot x + b \cdot y + c \cdot z + d = 0. \tag{6.39}$$

The solution is found by solving the least square problem in a maximum of four iterations. Each iteration assumes that a, b, c or d are equal to 1. For example, supposing

that d is unitary

$$\underbrace{\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ & \dots & \\ x_N & y_N & z_N \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a \\ b \\ c \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} -1 \\ -1 \\ \dots \\ -1 \end{bmatrix}}_{B}, \tag{6.40}$$

we find a solution only if $d \neq 0$ (otherwise the values of the solution are infinite), as

$$x = (A^T \cdot A)^{-1} \cdot A^T \cdot B, \tag{6.41}$$

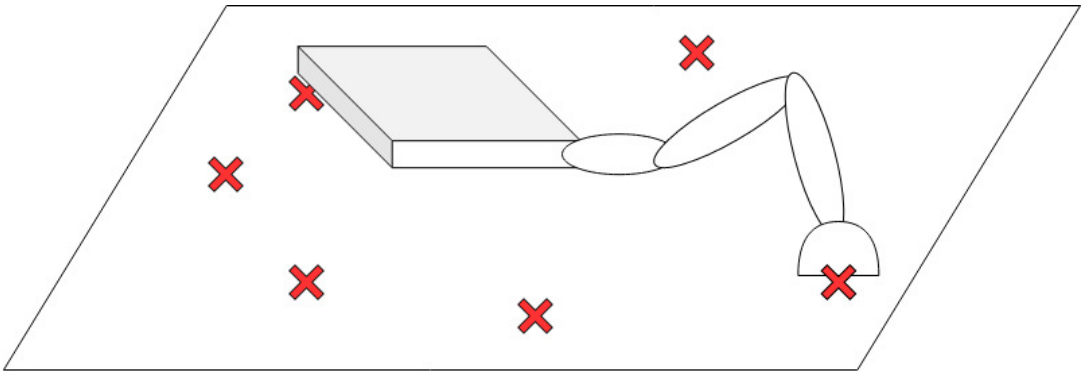


Figure 6.13: Plane detection based on the contact points.

6.5.2.2 Step 2 - Contact point projection

Later, the contact points are projected onto the plane. Given the plane P and the contact point Σ_{iS} located at the position (x_{iS}, y_{iS}, z_{iS}) (which is most likely out of the plane), the projection of Σ_{iS} on the plane is denoted as Σ'_{iS}, which is located in $(x'_{iS}, y'_{iS}, z'_{iS})$.

The contact points are illustrated in Figure 6.14. Furthermore, in that figure it is possible to observe, among others, the body velocity vector commanded by the user ${}^{B}\vec{v}$, the velocity vector with respect to the frames Σ_{iS} and Σ'_{iS} (denoted as ${}^{iS}\vec{v}$ and ${}^{iS'}\vec{v}$ respectively), the possible footfall points $\Sigma_{\lambda p}$, and their projection to plane P as $\Sigma'_{\lambda p}$.

To carry out the projection, one has to compute the line between Σ_{iS} and Σ'_{iS} points, which is defined as

$$\begin{bmatrix} x'_{iS} = a \cdot h + x_{iS} \\ y'_{iS} = b \cdot h + y_{iS} \\ z'_{iS} = c \cdot h + z_{iS} \end{bmatrix}. \tag{6.42}$$

Then, substituting into the plane P equation, it is obtained

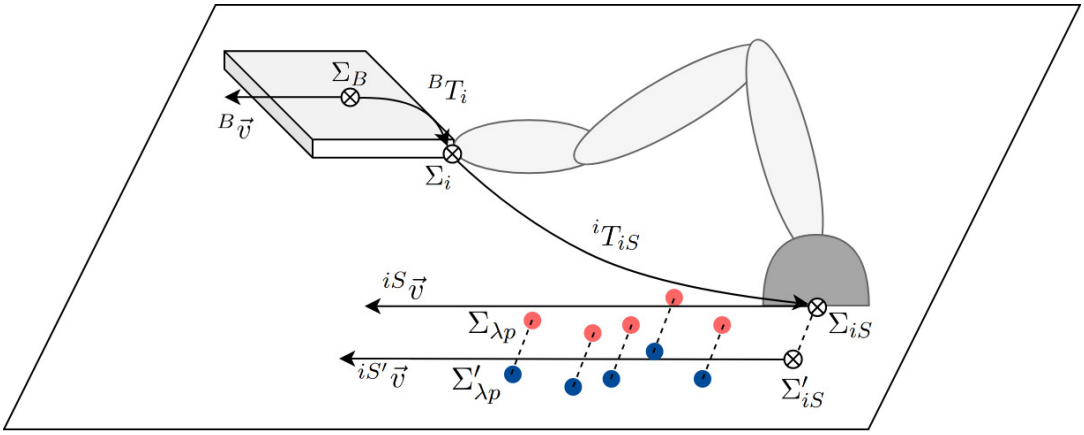


Figure 6.14: Relevant frames on the footfall planner.

$$h = \frac{-(a \cdot {}^B x_{iS} + b \cdot {}^B y_{iS} + c \cdot {}^B z_{iS} + d)}{a^2 + b^2 + c^2}, \tag{6.43}$$

and therefore, it is possible to compute x'_{iS}, y'_{iS}, and z'_{iS} with Equation 6.42.

6.5.2.3 Step 3 - Random sampling

Random sampling is generated in a way similar to the path planner proposed in Section 6.3. To generate a set of P random points λ_p located at positions $\Sigma_{\lambda p}$, with $p \in [1, ..., P]$, and correlate them with the contact plane to obtain P random points λ'_p located at positions $\Sigma'_{\lambda p}$, two options are proposed:

1. The first consists in projecting the velocity vector (commanded by the user, ${}^B\vec{v}$) into the plane (Figure 6.15). To do so, a generic point is created that is contained in the line defined by the vector ${}^B\vec{v}$ and projected to the plane P. The union between Σ'_{iS} and that point defines the velocity vector ${}^{iS'}\vec{v}$. Later, a random sampling is carried out similarly to the one of the path planning to obtain a set of points λ'_p.

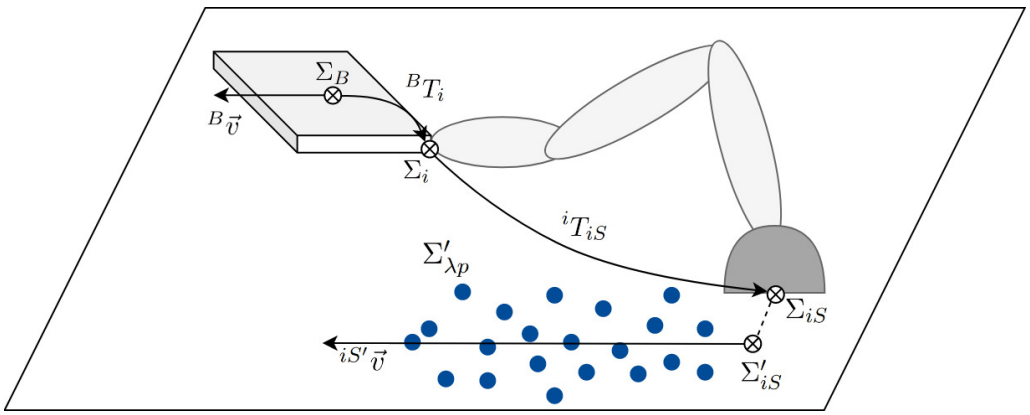


Figure 6.15: First option of the random sampling for the footfall generation.

2. The second option (shown in Figure 6.16) consists of generating the random sampling λ_p, and then projecting the points into the plane to obtain a set of points λ'_p. This option has been selected because it is simpler to implement.

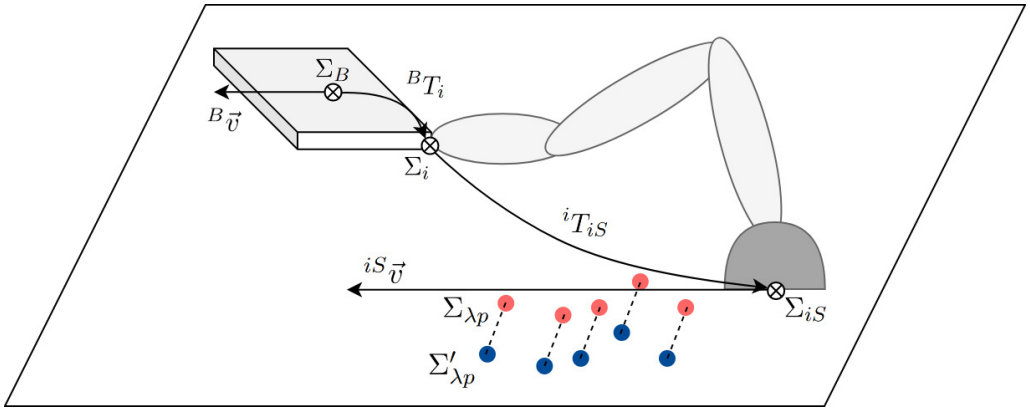


Figure 6.16: Second option of the random sampling for the footfall generation.

6.5.2.4 Step 4 - Sample filtering

To reduce computational time in future steps, the kinematic comfort of the position after the step is also taken into account, as well as the future kinematic comfort of the module to swing. This last component has been demonstrated to be required since a possible footfall can be valid at the moment after the step but is kinematically uncomfortable or unrealizable after a body movement in the motion direction. The following points are considered to decide whether or not a step point is considered in the optimization:

1. **Check module comfort immediately after stepping**. To do so, for each point randomly generated, the future position of the module to move is calculated as

$$ {}^i T_{\lambda p} = {}^B T_i^{-1} \cdot {}^B T_{\lambda p} \tag{6.44} $$

   Once the leg position is calculated after finishing the swing phase, the logic explained in Section 6.4 is applied, where κ_1 is set to zero to not take into account the first aggregated condition.

2. **Check module comfort after a while of stepping**. Similarly to the previous point, a virtual image of the robot is done after a displacement of the robot body after finishing the swing phase in the direction of motion. Being ${}^i T'_{\lambda p}$ the position of the TCP after the body displacement (after the swing phase), and ${}^B T_{B'}$ the body displacement in the direction of motion established by the user. That is, the comfort is checked for the following TCP position of the module:

$$ {}^i T'_{\lambda p} = {}^B T_i^{-1} \cdot {}^B T_{B'} \cdot {}^{B'} T_i \cdot {}^i T_{\lambda p} \tag{6.45} $$

where ${}^{B'}T_i = {}^{B}T_i$ for a rigid body. The TCP position is computed several times for different values of ${}^{B}T_{B'}$. That is, by establishing a predefined maximum displacement, an interpolation is performed and it is checked for each intermediate point that the comfort condition is met. Thus, the logic explained in Section 6.4 is applied to check the comfort, where κ_1 is set to zero to not take into account the first aggregated condition.

3. **Check if the swing phase is feasible**. To do so, it checks if the motion between Σ_{iS} and $\Sigma_{\lambda p}$ is possible from a kinematic point of view by computing the swing phase trajectory (the different alternatives are explained in Section 6.6.2).

4. **Check collisions after stepping and during the swing phase**. To do so, the low-weight collision detector explained in Section 6.5.3 is used.

6.5.2.5 Step 5 - Sample weighting

Given the experience of the path planner, Equation 6.46 is proposed as the heuristic cost function of the footfall planner to assess which point is the best. Oppositely to the cost function of the path planner, where Dijkstra is later applied, the cost function proposed in the footfall planner is desired to be as maximum as possible, so the best alternative is one with greater value.

$$C_p = W_t \cdot C_{p,t} + W_{sc} \cdot C_{p,sc} + W_d \cdot C_{p,d}, \tag{6.46}$$

Being $\boldsymbol{C}$ the vector composed of P items denoted as $C_p \in \boldsymbol{C}$, with C_p the cost value for point λ_p. Being $\boldsymbol{C_t}$, $\boldsymbol{C_{sc}}$ and $\boldsymbol{C_d}$ the cost function vectors related to the torque, suction cup safety, and distance parameters, with the cost values $C_{p,t} \in \boldsymbol{C_t}$, $C_{p,sc} \in \boldsymbol{C_{sc}}$, and $C_{p,d} \in \boldsymbol{C_d}$. These vectors are the normalized version of $\boldsymbol{\alpha_t}$, $\boldsymbol{\alpha_{sc}}$ and $\boldsymbol{\alpha_d}$ to evenly evaluate the different parameters (or unevenly with the weights W_t, W_{sc} and W_d).

$$\alpha_{p,t} = \prod_{i=1}^{J-3} f_{cost,ij} \cdot \frac{\tau_{ij,max} - \tau_{ij}}{\tau_{ij,max}} \tag{6.47}$$

$$\alpha_{p,sc} = r_{sc,max} - r_{sc} \tag{6.48}$$

$$\alpha_{p,d} = ||\vec{proj}|| \tag{6.49}$$

$$\tag{6.50}$$

The value of $\alpha_{p,d}$ is only positive with positive distances in the direction of movement in order to avoid giving a high weight to those points located in opposite directions of motion. To do so, the projection of vector of ${}^{iS'}T_{\lambda p'}$ over ${}^{iS'}\vec{v}$ (illustrated in Figure 6.17 as a green arrow, and computed in Equation 6.51) reveals whether footfall goes in a positive or negative direction compared to ${}^{iS'}\vec{v}$.

$$\vec{proj} = \frac{{}^{iS'}\vec{v}_{\lambda p'} \,.\, {}^{iS'}\vec{v}}{{}^{iS'}\vec{v} \,.\, {}^{iS'}\vec{v}} \; {}^{iS'}\vec{v}, \tag{6.51}$$

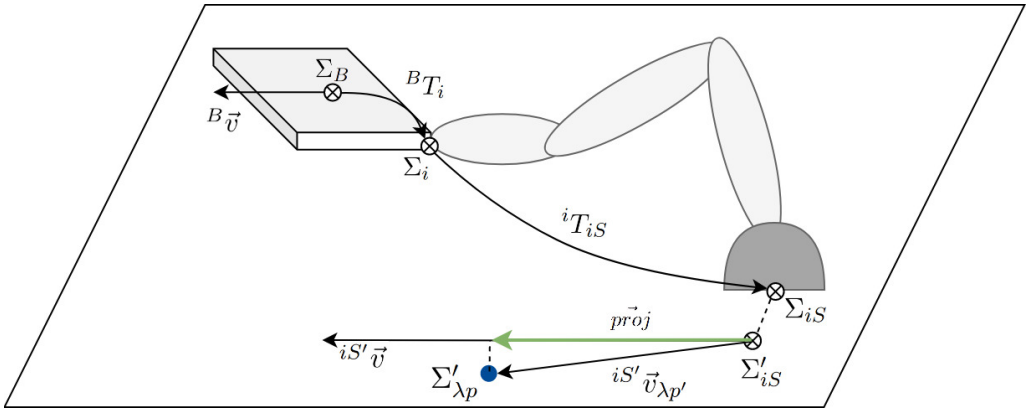


Figure 6.17: Projection of the vector between the current contact point to the footfall point.

6.5.3 Low-weight collision detector for modular robots

There are different options to detect collision between modules for a generic robotic organism (Figure 6.18). In [Cai et al., 2014], the authors use the Axis Aligned Bounding Boxes (AABB) method to detect collisions between two cuboid bounding boxes, whereas in [Wang et al., 2021] the authors use the same technique to prevent collisions in mobile construction machinery. AABB is the quickest algorithm to determine whether the two entities overlap or not. This consists of wrapping entities in a non-rotated (thus axis-aligned) box and checking the positions of these boxes in the 3D coordinate space to see if they overlap (Figure 6.19(a)). The overlapping area between two non-rotated boxes can be checked with logical comparisons alone, whereas rotated boxes require additional trigonometric operations, which are slower to calculate. Oriented Bounding Box (OBB) collision algorithm is considered as the standard in rigid body collision detection algorithms. In [Ding, 2013] the authors argue that it has the best performance compared to others presented in Figure 6.19 and its construction complexity is how to select the best direction of the bounding box. In contrast, in [Wang et al., 2021] the authors state that it is the most complex in processing since it needs to rotate with objects, which requires additional computation effort.

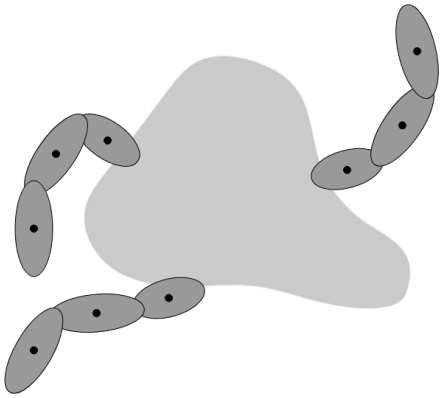
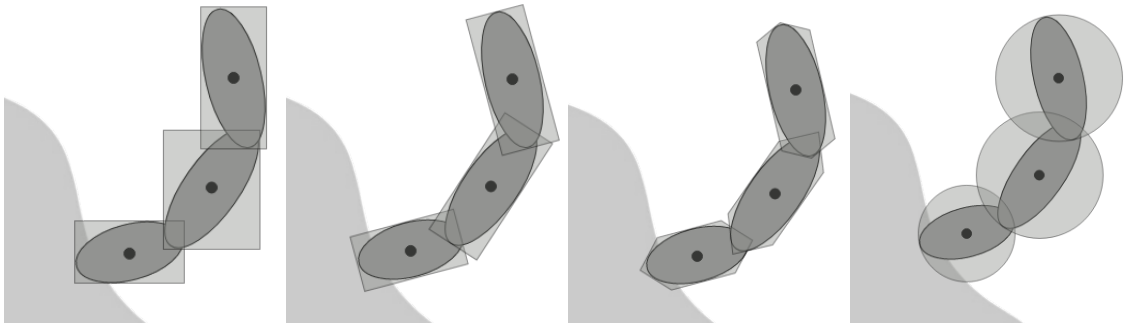


Figure 6.18: Simplified collision model.

Other alternatives [Pazouki et al., 2012] propose a simplified way in which the di-

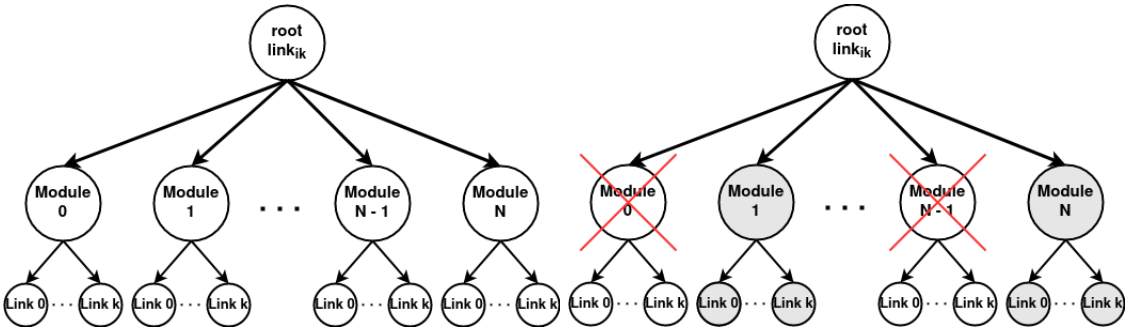
mensions of each link of each module are represented as a three-dimensional ellipsoid defined by their eigenvalues $(\lambda_1, \lambda_2, \lambda_3)$ and eigenvectors $(\vec{v}_1, \vec{v}_2, \vec{v}_3)$. Others propose the simplification of a large number of geometric objects with bounding volumes, such as the use of a "discrete orientation polytope" ("k-DOP"), a convex polytope whose facets are determined by halfspaces whose outward normals come from a small fixed set of k orientations [Klosowski et al., 1998] (Figure 6.19(c)). Others use spheres to check collision between objects (Figure 6.19(d)), such as the works presented in [Palmer and Grimsdale, 1995] or [Jiménez and Segura, 2008].



(a) AABB method. (b) OBB method. (c) k-DOP (k=6). (d) Bounding sphere.

Figure 6.19: Bounding boxes options for object representation.

For the case of ROMERIN, the use of OBB bounding boxes is proposed. To do so, the position and orientation of each link respecting $\sum_B$ is obtained with the FK. To reduce the computational cost, a generic collision detector tree is proposed and shown in Figure 6.20, where all links are initially available. Branches where collision is not possible from a positioning point of view are removed from the tree. For example, if modules are located in opposite directions, and the body is between them, then they cannot collide. Modules' self-collision is neither checked, and it is the responsibility of module controllers.



(a) Generic collision detector tree for collision checking. (b) Branches removed.

Figure 6.20: Decision tree for collision detection.

The FCL library [Pan et al., 2012] is used for OBB collision detection. Even though that library allows the use of meshes, due to the links' shape, OBB boxes are used to reduce the computational cost. Collision is checked between groups of links (gathering by modules).

6.6 Swing phase

In this section, two parts are included. The first describes the process between the exception reception from the gait controller and the swing phase. That is, the behavior of the system to ensure that the organism can endure and continue to function even when the leg changes to the swing phase. The second part includes several alternatives for the trajectory of the leg in the swing phase and the main advantages and disadvantages of them.

6.6.1 Preparation for the swing phase

Before a supporting leg transits to non-supporting, the organism has to be sure of being stable and able to lift the leg without risk. To do so, the path planner presented in Section 6.3 is used to move the body to a feasible position to lift the leg. To find the comfortable position, the **State Manager Node** sends information about a virtual situation in which the leg to move is lifted. With respect to the presented path planner, the preparation for the swing phase has the following differences:

- The random/normal sampling does not take into account the commanded body velocity. Instead, it generates points in the surroundings according to parameters established by the user.
- More torque is permissible since no motion is required, and the actuators are able to support more weight if they are still.
- The target vertex in the graph is set to a) the minimum weighted vertex, or b) the closest vertex with feasible conditions.

6.6.2 Swing phase trajectory generation

A non-supporting leg that is in the swing phase is involved in the process of moving from the previous supporting point position to a desired position computed by another agent. In this dissertation, two alternatives to mark movements in a step are presented:

1. By passing through several waypoints. In this case, the non-supporting leg TCP passes through defined waypoints ($P_1 - P_4$) as indicated in the dashed line of Figure 6.21.

   This alternative is easy to implement, but the motion is not smooth or elegant. Besides, the motion velocity has to be reduced (at least when the TCP is close to the waypoints) to avoid discontinuous jumps in speed and direction.

2. By using a Bézier curve. It is a parametric curve used broadly in computer graphics. In this case, a set of discrete control points defined a smooth and continuous curve employing a formula. This curve approximates a real-world shape that has no mathematical representation or whose representation is unknown or too complicated. A cubic Bézier curve is implemented using four points, as shown in Figure 6.21. Usually, robot TCP does not pass through P_1 and P_2, as they only provide directional information. The distance between P_1 and P_2 determines "how far" and "how fast"

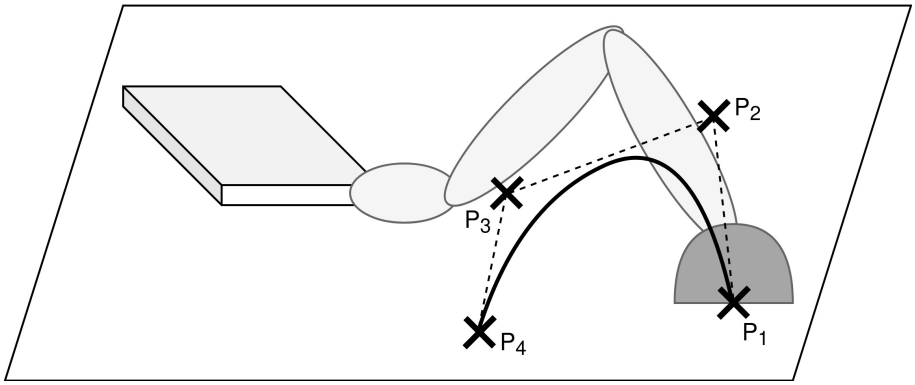


Figure 6.21: Swing phase trajectory.

the curve moves towards P_1 before turning towards P_2. The formula that defines the cubic Bézier curve is given by

$$B(t) = (1-t)^3 \cdot P_0 + 3 \cdot (1-t)^2 \cdot t \cdot P_1 + 3 \cdot (1-t) \cdot t^2 \cdot P_2 + t^3 \cdot P_3 \ , \ 0 \leq t \leq 1. \tag{6.52}$$

Where t is computed in each iteration in whole value is given iteratively by

$$t = iteration \cdot \frac{t_s}{swing_duration} \ , \tag{6.53}$$

with t_s as the sampling time.

This option appears as the solution to most of the problems of the previous one. The trajectory is smoother and more elegant, and the TCP velocity does not change drastically at any moment by following a curve trajectory. However, this option still has the problem that the leg may be stretched at some points of the trajectory or in a large part of it. This causes higher torques in the joints, not only in those of the leg that is in the swing phase but also in other legs. This problem is especially important in the leg that is in the swing phase, which has been noticed to have problems with some movements due to the low torque of the actuators.

3. By folding the leg to reduce momentum. To do so, the positions of q_2 and q_3 are set first to maximum. Second, q_1 moves to its target position (computing the inverse kinematics of the point P_3 or P_4 of Figure 6.21). Lastly, the leg is moved to P_3 and P_4 sequentially.

   This option appears as the solution to the problem with the torque of the previous alternative. Moving q_2 and q_3 to their maximum values, the leg module remains folded and the torque requested in q_1 is drastically reduced. This alternative is also easy to implement, but the motion is not smooth or elegant. Similarly to the first alternative, special care must be taken about speed and discontinuities.

All alternatives use the same discrete points $P_1 - P_4$, which are computed as:

1. P_1: Current TCP position.
2. P_2: It is created by displacing P_1 in the opposite direction of the surface.
3. P_3: It is created by displacing P_4 in the opposite direction of the surface.
4. P_4: Footfall point generated by another agent.

6.7 Summary

In this chapter, it is presented:

a) locomotion definitions for L&C robots,

b) a locomotion analysis of insects to justify the bio-inspiration of our approach,

c) a path planner for energy consumption and robot safety optimization,

d) a non-predefined, and bio-inspired gait controller that selects the most convenient leg to lift and move,

e) a footfall planner that selects the position to step after the swing phase with a low-weight collision detector to avoid collisions during swing phases between leg modules and with the body, and

f) the preparation and trajectory planning of the swing phase.

The knowledge acquired in this chapter is pending to be published.

Chapter 7

Results

"Science without religion is lame, religion without science is blind."

Albert Einstein

The research presented in this thesis in the areas of hardware and control development finally culminates in a series of experiments with the modular robot ROMERIN and its digital twin. By combining the individual building blocks presented beforehand, the robot can perform reactive gaits, from static walking to climbing on varying inclinations.

7.1 Experimental setup

7.1.1 Software configuration - ROS2 implementation

MoCLORA architecture has been fully integrated in ROS2 and tested in versions Dashing [1] and Foxy [2]. Each single function has been included in a single ROS2 package in order to reduce compilation time and complexity, and increase modularity. Thus, the classification shown in Figure 7.1 is proposed, where the main packages are included in clusters that define their objective. These clusters are algorithms (such as package to generate graphs, compute the estimation of the robot reaction force, or compute the gravity compensation torques), applications (such as collision detection, path planner, or graphical user interface), communication tools (such as UDP or Bluetooth protocol classes), devices (such as classes to control Dynamixel motors, the imu, the pantilt, and the RealSense camera), robot (such as the control of one leg as indicated in the Executive level of the control architecture of Chapter 4), simulation (such as plugins for the legs, camera, or environment conditions modification), and utilities (such as useful types definitions, and math, plotting, or vision utilities).

Among the most relevant packages, in Figure 7.2 the main dependencies between them are illustrated. Arrows point from the package that needs the dependency to the dependency. This dependence graph does not mean that packages without direct connections

[1] `https://docs.ros.org/en/dashing/index.html`
[2] `https://docs.ros.org/en/foxy/index.html`

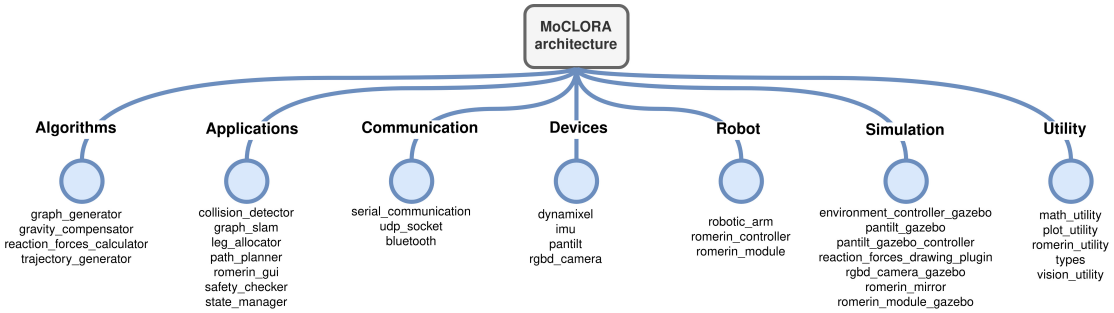


Figure 7.1: ROS2 node structure.

do not communicate with others. In fact, they can communicate using ROS2 topics or services.

As an example, Figure 7.3 shows the ROS2 communication topics between different nodes when running the control for a quadruped robot in simulation. The nodes that appear are closely related to the control architecture components. There, it is possible to find the nodes of the Scheduler layer: State Manager Node, Path Planner node, and Safety Checker Node related to the Gait Controller. Also, the four Controller Nodes of the Executive Level for the legs with names THOR, FRIGG, ODIN, and LOKI. Lastly, the Module Communicator of the HAL is located. Since the system is running the simulation environment, other nodes that publish data related to the camera, IMU, and pantilt data are found.

7.1.2 Hardware configuration

In order to comply with the torque-based control, the first three Dynamixel servomotors of the module's chain have been set to current-based position control mode. In addition, an embedded microcontroller in the servomotor has been configured to return the status packet immediately after receiving an instruction packet in order to increase the control rate in the modules MCU.

In addition, to improve the control rate, the reading of the suction cup sensors has been considerably improved. The control rate has been improved from 10Hz (100ms/loop) to 100Hz(10ms/loop) without penalizing its behavior.

7.2 Validation of the ROMERIN concept with an inverse kinematic controller (IKC)

MoCLORA has been tested with both the digital twin and the real robot. First, simple body movements[3], and secondly, body trajectories such as circles or squares have been commanded[4]. Figure 7.4 shows a sequence of a circular motion performed in both the simulated and physical robotic systems. The trajectory is generated by the *Trajectory Generator* node of the Scheduler layer, which sends a velocity profile to the *State Manager Node*.

[3]Videos available at `https://youtu.be/w02W8tUM64E` and `https://youtu.be/9Q0KD2YcXns`
[4]Videos available at `https://youtu.be/myqL09stIFk` and `https://youtu.be/6PVj6xO7kCk`

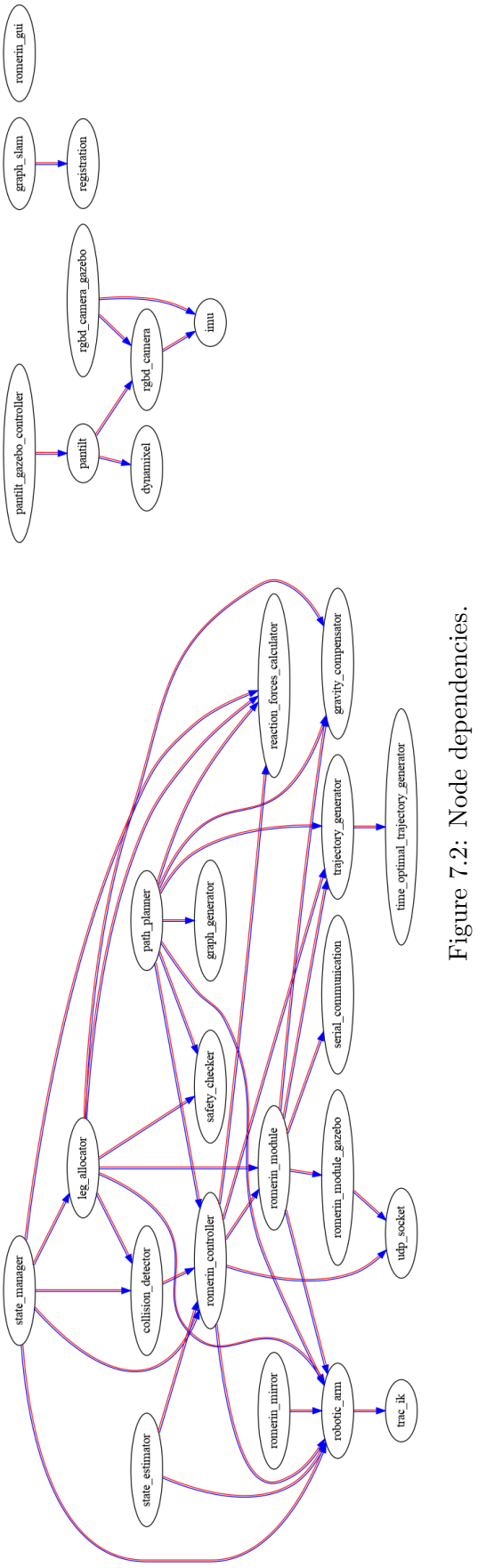


Figure 7.2: Node dependencies.

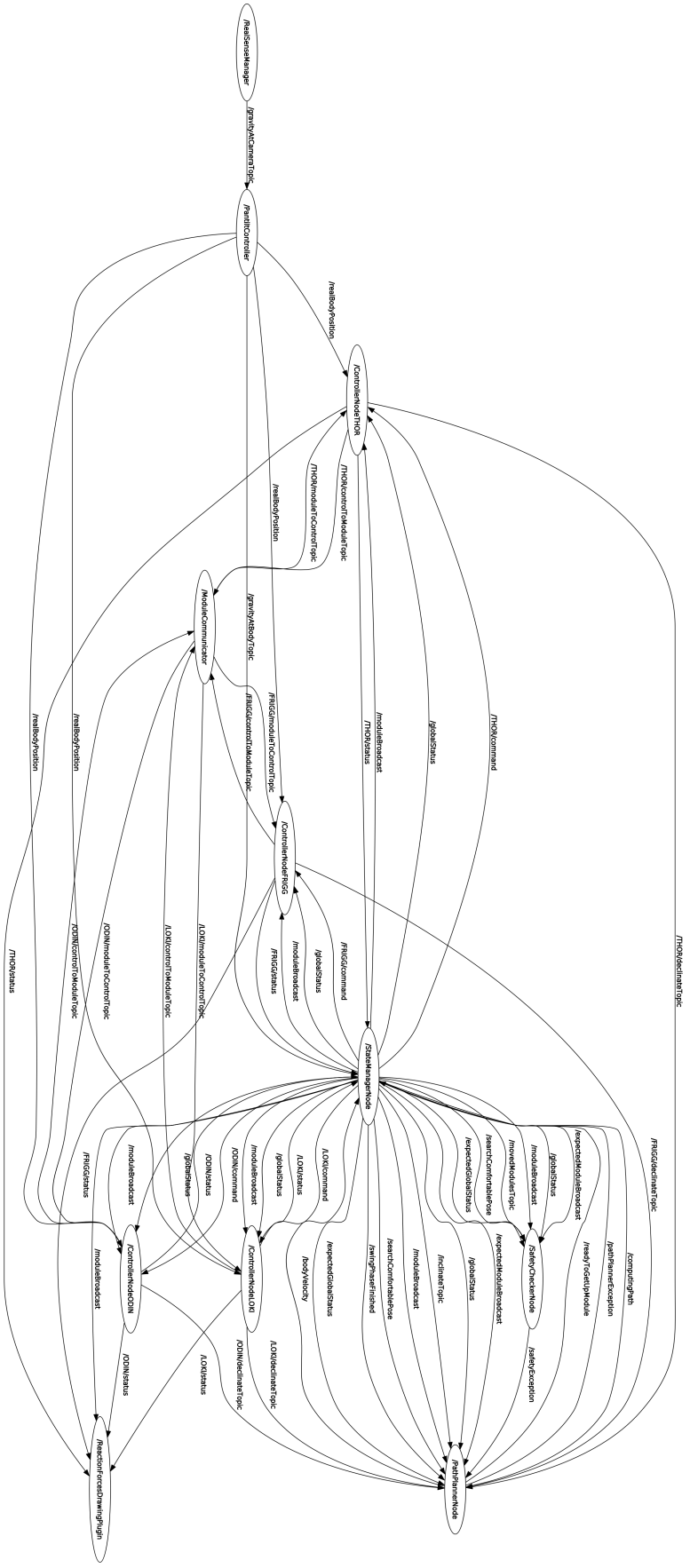


Figure 7.3: ROS2 node graph.

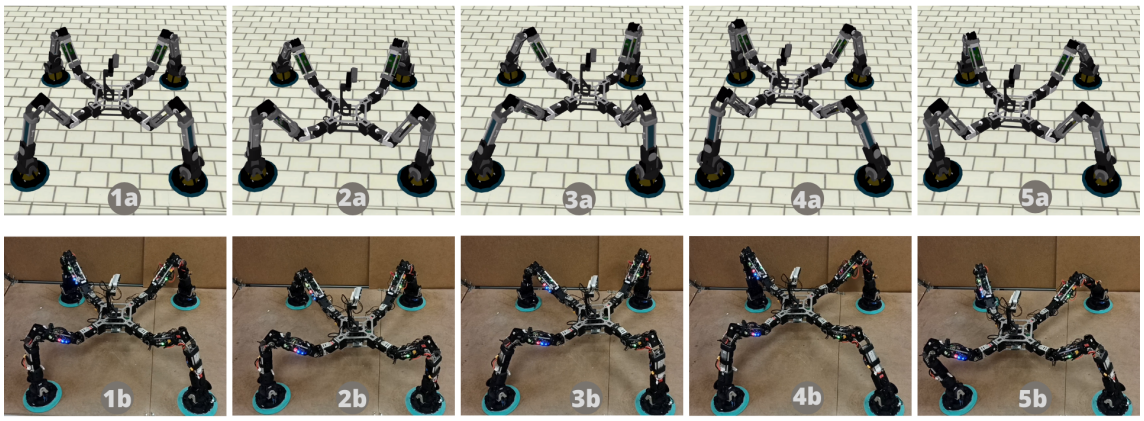
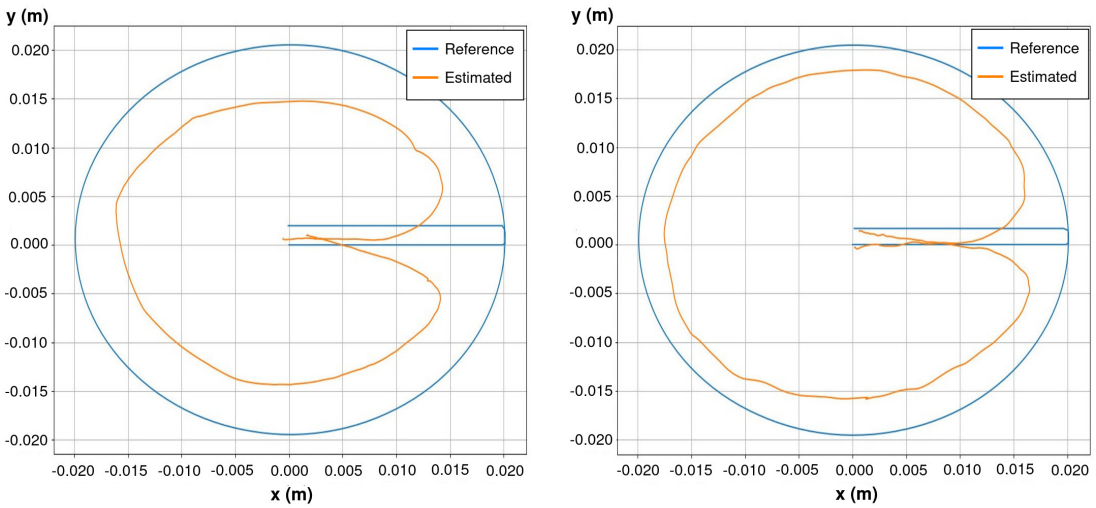


Figure 7.4: Results for the digital twin (up) and physical (down) organisms while carrying out a circular trajectory. Picture 1 shows the initial configuration, and the rest pictures show the catches of the trajectory.

Square and circle trajectories have been used for testing hardware, firmware, and software infrastructure. Figure 7.5 shows the commanded and estimated body trajectory for both the digital twin and the physical organism. It is observed to have higher accuracy in the digital twin. This fact is attributed to the looseness of the connecting pieces, which causes the motors to not reach the desired position due to the hiperstaticity of the system while adhering to the environment. In that figure, estimated values are obtained through the FK of the legs, that is, only robot odometry is used.



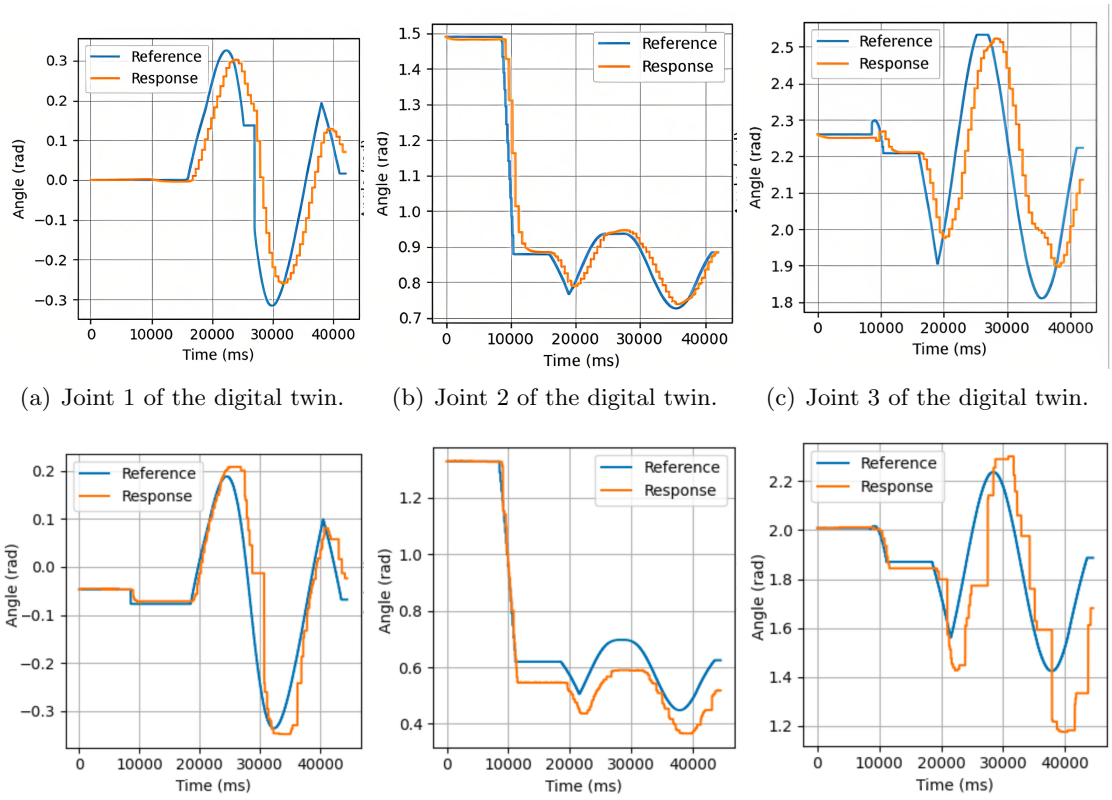
(a) Body trajectory of the physical organism.

(b) Body trajectory of the digital twin.

Figure 7.5: Body commanded and estimated position of both digital twin and physical organism.

The corresponding commanded and achieved joint trajectories for motors 1 to 3 of a module are shown in Figure 7.6 for both cases. As observed in the results, the commanded joints positions are analogous. As indicated previously, during the control of the body by means of the support modules, the wrist motors torque transmission is deactivated (only

from a torque point of view, the encoder reading, communication with the motor, and other features is available), and consequently its movement is free, that is why the position of those joints are not shown. In Figure 7.6 is also possible to observe that the motors response do not get the reference signal completely. This is deduced from the fact that the control of each motor is carried out with a PD controller, which prevents the motors from applying an excessive force when the reference is not reached. This fact would be common for L&C robots like ours, which are hyperstatic systems while adhered to the environment, and due to the looseness of links, the reference would not be reached at any time without putting the integrity of the robot at risk. For the same reason, the integrative component is removed from the controller, which would cause the motor to overload.



(a) Joint 1 of the digital twin. (b) Joint 2 of the digital twin. (c) Joint 3 of the digital twin.

(d) Joint 1 of the physical organism. (e) Joint 2 of the physical organism. (f) Joint 3 of the physical organism.

Figure 7.6: Digital twin (a-c) and real joints trajectories (d-f) of a single module while performing a circular movement (trajectory shown on Figure 7.5).

We have used MoCLORA in different organism configurations for the proposed trajectories[5]. More specifically, the execution of various types of trajectories with the arrangement of modules shown in Figure 4.10 has been tested on the simulator.

From the simulation of body trajectories with different arrangements and the study of their joint trajectories, the good behavior of the control architecture is concluded for different numbers of modules located in different positions of the body. The implementation of such arrangements in physical robotic organisms requires the construction of more modules and the design of new bodies that can house them (including, of course,

[5]Video available at `https://youtu.be/nOxRKEuagnw`

the necessary elements proposed in Section 3.7). The only difference with respect to the control of the quadruped organism proposed in this dissertation is the redefinition of the *Organism Configuration* component of the control architecture shown in Figure 4.2, while all other components are kept intact.

On the other hand, Figure 7.7 shows the angular and distance errors of a module during the circular trajectory denoted previously. In the same picture, the module's manipulability is plotted. As observed experimentally, the angular and positional errors fluctuate in the opposite direction to the manipulability. That is, maximum estimation errors correspond to the moments when the module is more shrunken or stretched, and the manipulability decreases, preventing the body from moving comfortably. Manipulability is computed with the track-IK solver [Beeson and Ames, 2015] as indicated in (7.1):

$$w = c_s \cdot det(J \cdot J^T)^{0.5} \tag{7.1}$$

where c_s is the scalability constant for visualization, and J is the Jacobian matrix of the last six joints of a module. Regarding the angular and positional error with different arrangements, there is no significant variation with respect to the test carried out with the 4-legged organism. The mean error is about 0.072 ± 0.011 rad and 2.8 ± 0.5 cm.

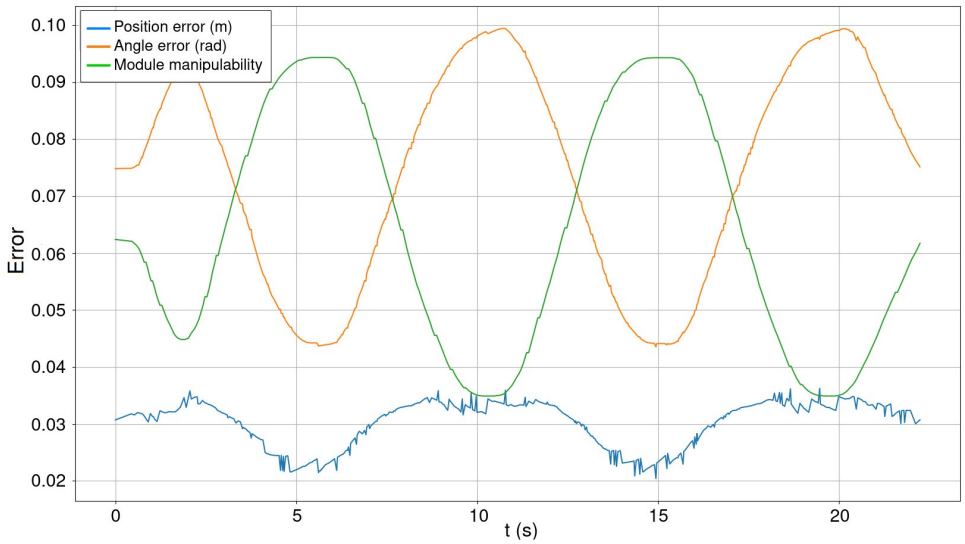


Figure 7.7: Module manipulability over the angular and distance error of the estimation that a module does during a circular trajectory (Figure 7.5). Manipulability line has been scaled for representation purposes.

The robot payload increases with the number of modules and varies with the distribution of the modules. In the case of the developed body, the payload when it is standing, as shown in the videos, is around 2.4 kg (including the weight of the developed body). Although the payload depends on the modules arrangement, it is estimated that each module is capable of holding 0.6 kg with a distributed load between modules.

7.3 Torque-based control experiments

Controlling a climbing robot, which is hyperstatic by nature, with IKC presents several problems, such as reduced accuracy, actuator overload, and pieces breaking. Thus, the torque-based control presented has been tested and compared with the IKC.

ROMERIN capabilities and performance were evaluated and demonstrated in discrete and continuous environments and ground tests listed in Table 7.1. ROMERIN can climb vertical walls while walking on the ground and the ceiling [6].

Experiment	**Video start**	**Environment**	**Objective**
Torque-based control (Sec. 7.3.1)	0:10	Suction cups over different planes	Verify FDP and impedance control with physical platform
Gravity compensation (Sec. 7.3.2)	0:36	Ground, different planes, wall, and ceiling	Verify gravity compensation and impedance control
ROMERIN gait (Sec. 7.4)	1:40	Ground, wall, and ceiling	Verify impedance control with gait

Table 7.1: List of experiments and objectives

7.3.1 PD control of ROMERIN



Figure 7.8: Impedance control experiments by using a predefined manual circular trajectory.

ROMERIN was operated using a predefined manual circular trajectory to verify the combination of FDP solver and impedance control. It has been tested with two different IMU data sources, the IMU embedded in the body and the one embedded in the RealSense D435i. In this second case, the acceleration information is transformed to Σ_B as

$$^{B}g = {}^{B}R_C \cdot {}^{C}g \tag{7.2}$$

where ${}^{B}R_C$ denotes the rotation matrix from Σ_B to Σ_C. Figure 7.8 shows the behavior of a PD control with gravity compensation, where the constants that are given

[6] Video available at `https://youtu.be/nTHPNJAwBs4`, accessed on 26 July 2023.

experimentally are

$$K_p = \begin{pmatrix} 40 \\ 60 \\ 25 \end{pmatrix} \; ; \; K_d = \begin{pmatrix} 8 \\ 10 \\ 5 \end{pmatrix} \; ; \; K_B = \begin{pmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0.15 \\ 0.15 \\ 0.15 \end{pmatrix} \tag{7.3}$$

and the maximum allowed torque has been set to the 65% of the actuator stall torque to avoid overloads. The position and the resultant torques are shown in Figure 7.9, while the cartesian position of the same leg Σ_{iS} related to Σ_i is illustrated in Figure 7.10. As observed, the leg follows the commands received, where the maximum error is found in the Z axis, which corresponds to the gravity component. The mean error in body position has been reduced from 0.2 ± 0.05 rad and 6.0 ± 1.5 cm of the ROMERIN IKC [Prados et al., 2022a], to 0.1 ± 0.02 rad and 3.0 ± 0.6 cm with the proposed torque-based controller of this dissertation.

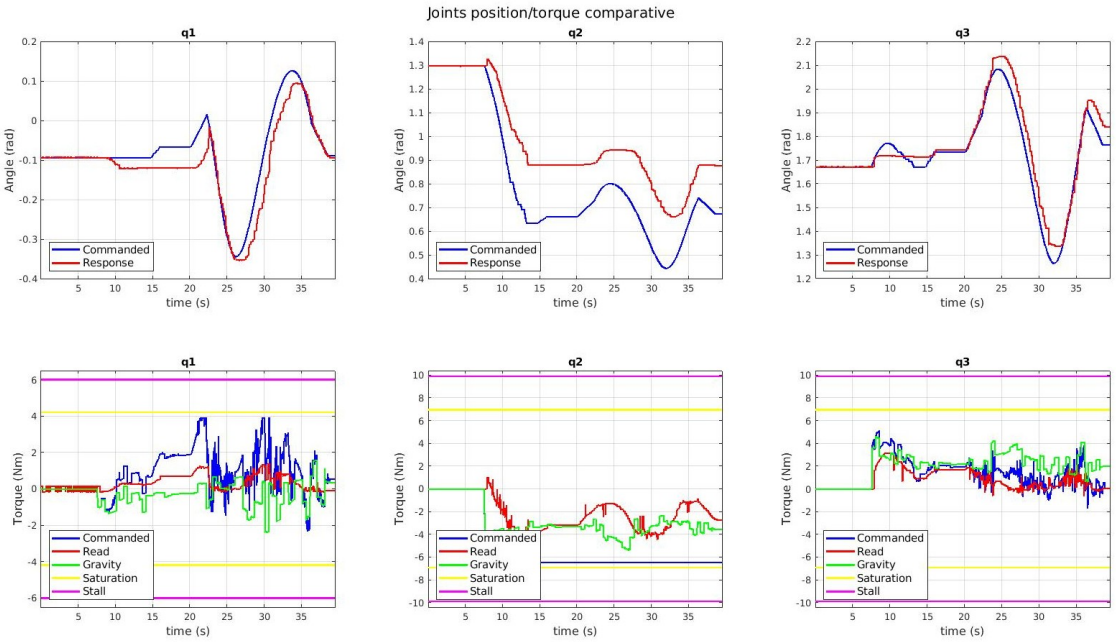


Figure 7.9: Position and torques of q_1-q_3.

7.3.2 Gravity compensation

The same predefined manual circular trajectory to verify the combination of FDP solver and impedance control has been tested in different situations, such as on the ground, wall, and ceiling. In this case, the PD component has been removed from time to time, keeping gravity compensation only. To verify the performance of gravity compensation, the ROMERIN digital twin [Prados et al., 2022a], which can be configured to only accept torque commands, has been used. In this way, thanks to its reliability, it is possible to

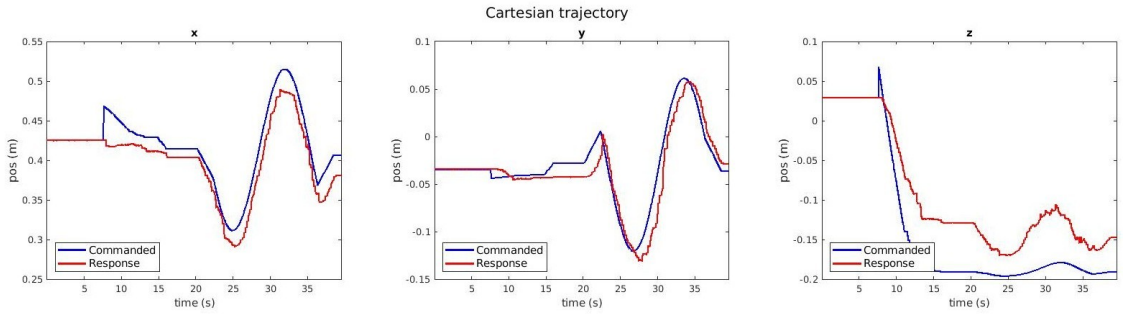


Figure 7.10: Cartesian position of one of the modules.

verify the experiments from a theoretical point of view. It has been demonstrated that when the PD component is removed, the robot stays still in different situations. Figure 7.11 shows the example where the robot is in the ceiling. As observed, the joints follow the commands faithfully and remain constant when the PD component is removed (blue boxes), keeping the gravity compensation only. For this experiment, constant values of Equation 7.3 are set.

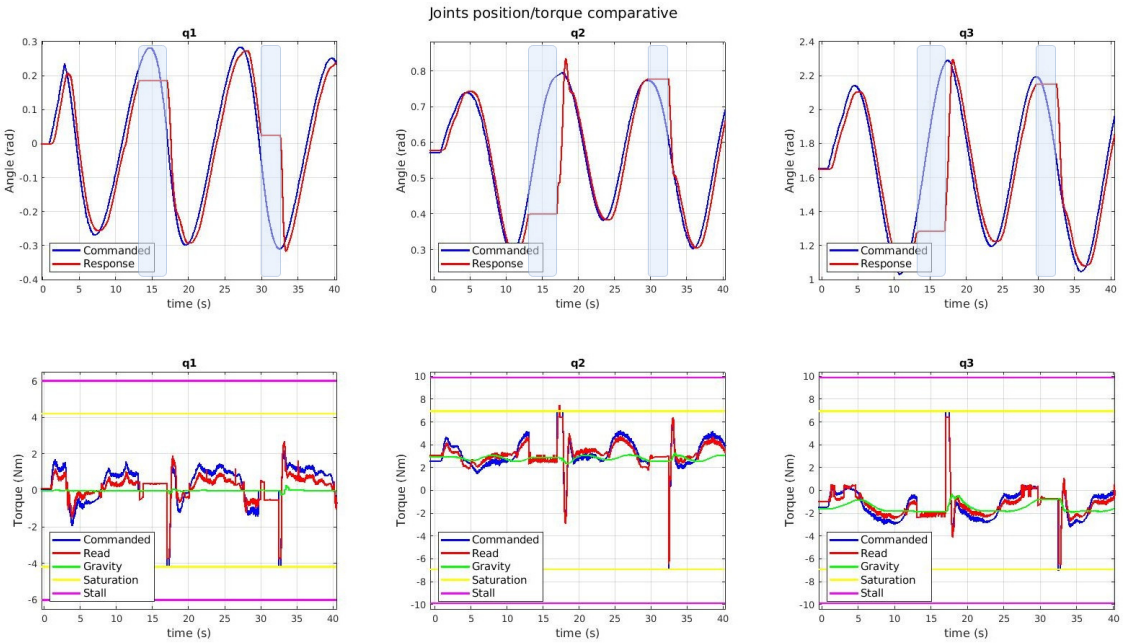


Figure 7.11: Position and torques of q1-q3 for a constant predefined manual circular trajectory with the robot in the ceiling. Blue boxes indicates when the PD component is removed.

7.3.3 PD control of other multilimbed robots

In this dissertation, a generalizable, low computational cost, simple, and fast gravity compensation method is presented for legged robots with a variable number of legs. This thesis focuses on climbing robots, which are slow given their constraints. This means that the algorithm is based on the static problem, which is a reduction of the dynamic model,

and can be used in all those cases where velocity is low enough to guarantee that the dynamic components can be neglected.

The access of robots to all types of natural environments is still a challenge to solve. This is the case of [Godon et al., 2023], where the authors discuss the problem of locomotion in deformable grounds. They point out the importance of solving the problem to facilitate new robotic applications such as search and rescue in wet forests, muddy fields, avalanches, and mudslides; agricultural vehicles operating on wet soils; exploration or excavation of materials with minimal environmental impact; etc. In the case of manned vehicles and wheeled robots, they deform the environment and can fail if the medium is too deep and weak because the actuators cannot generate sufficient tangential forces.

To solve this problem, they propose the use of legged robots with a deformable big foot for static walking without sinking. In particular, they equip the robot GO1 of Unitree with a custom-designed foot to reduce sinking, which makes the embedded force/torque (F/T) sensor unusable.

This doctoral thesis has participated in the described project by implementing the reaction force estimator (FDP solver), the gravity compensation algorithm, and the PD control (Section 5). As a result, the GO1 robot is capable of performing static gaits to step in muddy environments and studying its features [Godon et al., 2022]. First of all, the FDP solver has been validated both with the simulation and with the F/T sensors of the robot (without using the equipped foot). As observed in Figure 7.12, the reaction forces estimator is very accurate and can replace the F/T sensor.

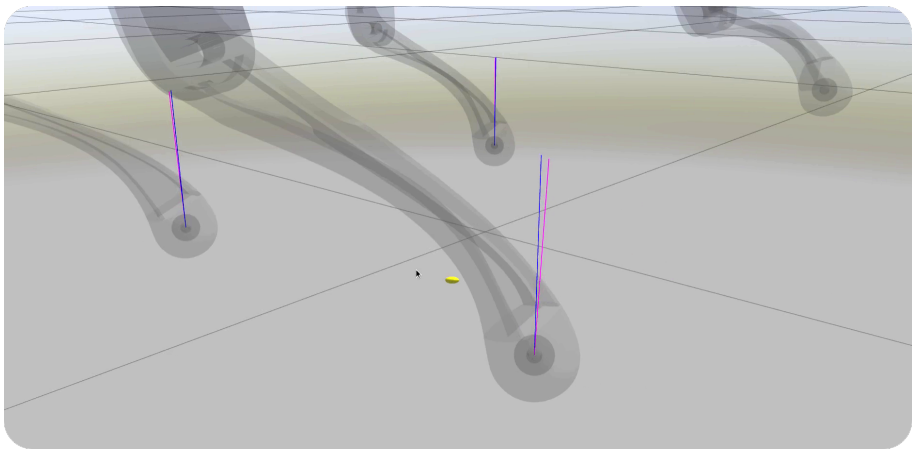


Figure 7.12: Reaction forces given by the force/torque sensor in the simulator (purple lines) and estimated forces by the FDP solver (blue lines) when the robot has one leg in swing phase.

As an example of the system while operating (static gait), Figure 7.13 shows the position, velocity, and torque register of one foreleg and one rear leg with:

$$K_p = \begin{pmatrix} 14 \\ 36 \\ 60 \end{pmatrix} \quad ; \quad K_d = \begin{pmatrix} 3.3 \\ 8.8 \\ 16.5 \end{pmatrix} \tag{7.4}$$

As observed in the plots, the position follows the commanded trajectory, whereas the

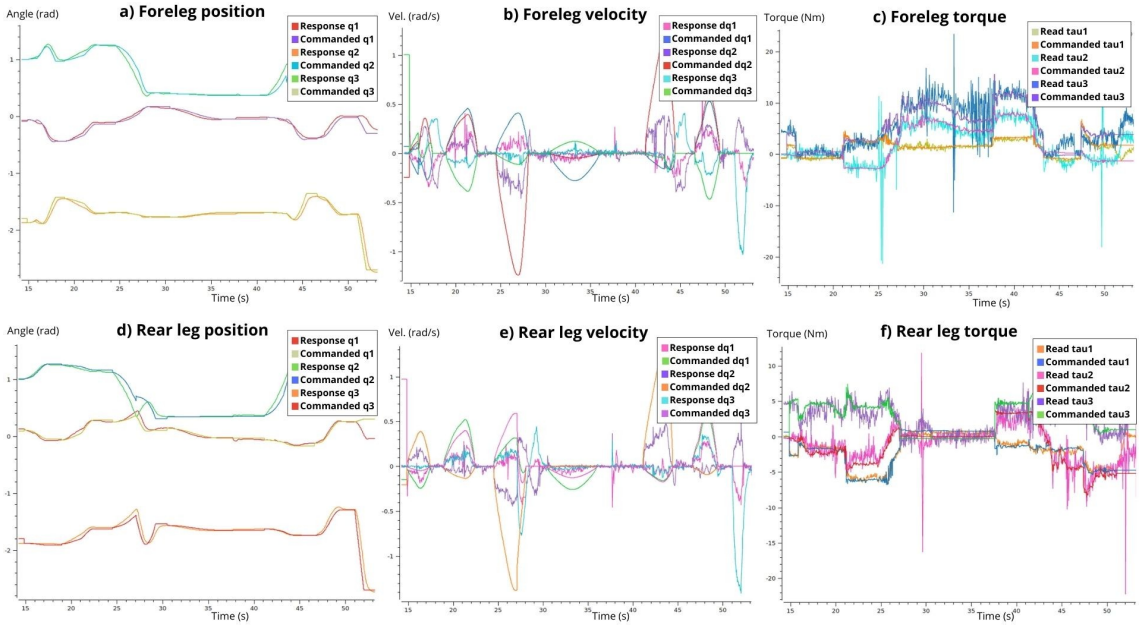


Figure 7.13: Results of a static gait with the robot GO1 of Unitree.

read torque from the motors is very similar to the commanded torque (which is the torque to compensate gravitational forces). Other two important aspects are:

a) Detect when a leg can be lifted (from stance to swing phase) without risking static stability. To do so, two different options are proposed:

- By checking if the reaction forces in the others after changing the leg to swing phase are enough to keep the robot stable. To do so, a virtual state is created supposing that the leg with a lower reaction force is lifted. In this state, the reaction forces are computed, and the leg can be lifted if the following conditions are all satisfied:

$$\underbrace{\left(\frac{\vec{F}_i \cdot \vec{g}}{\vec{g} \cdot \vec{g}} . \vec{g}\right)}_{\text{Projection of } \vec{F}_i \text{ over } \vec{g}} \cdot \vec{g} < 0 \tag{7.5}$$

$$|\vec{F}_i| > \phi_F \tag{7.6}$$

  where Equation 7.5 determines if both vectors $\vec{F}_i$ and $\vec{g}$ have the same directions, and therefore the reaction force is not valid, ϕ_F is the safety threshold to guarantee stability.

- The second option does not create a virtual state and checks for all legs if the reaction force in one of them is negative or less than a threshold. So in this case, the same conditions have to be satisfied with quite different threshold values.

b) Detect if a leg is touching the ground. To do so, a basic logic is proposed to identify external contacts from joint torque measurements. Based on the static model and its

reliability, a leg is considered to be in contact with the ground when, with very low velocity, the root-mean-square error between the gravitational compensation torque and the read torque in the actuators of a leg is:

$$RMSE\left(\tau_g, \tau\right) > \phi_\tau \tag{7.7}$$

This assumption can be made thanks to the gravity compensation calculator, since it reduces the PD gains considerably and makes an estimation of the torques that the robot has to apply.

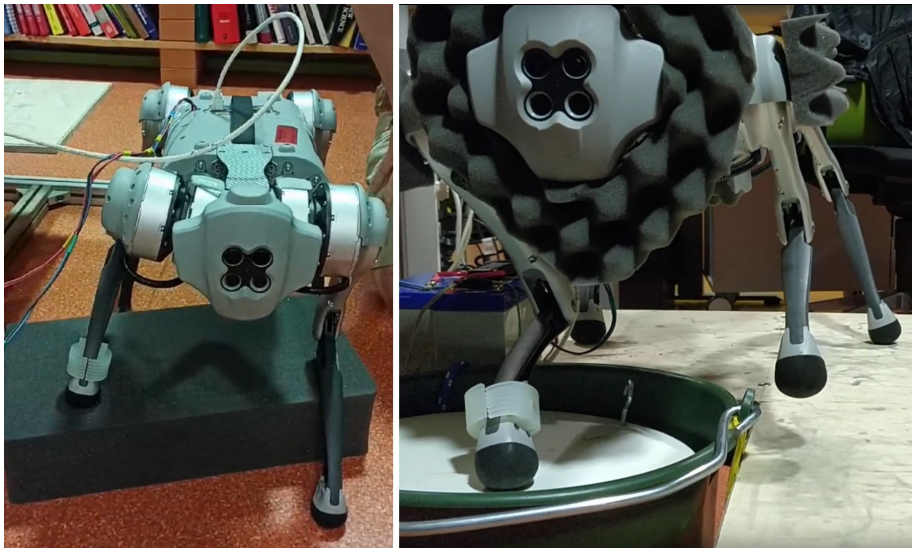


Figure 7.14: Pictures of the experiments with the robot GO1 of Unitree.

7.4 Locomotion experiments

In this section, locomotion experiments are carried out. First, the path and footfall planners are analyzed, followed by an analysis of the locomotion of different arrangements of robotic organisms.

7.4.1 Path and footfall planner experiments

An illustrative example of the MPC-based path planner is shown in Figure 7.15, where it is possible to observe the replanification that the planner makes after raising an exception. These discontinuous steps correspond to the swing phase of a module. Since the body position after the step is not exactly the same as before, it is possible to observe jumps on the final step of a planification and the initial step of the following.

Regarding the MPC-based footfall planner, an illustrative example is shown in Figure 7.16, where the generated points are observed in different colors according to their suit-

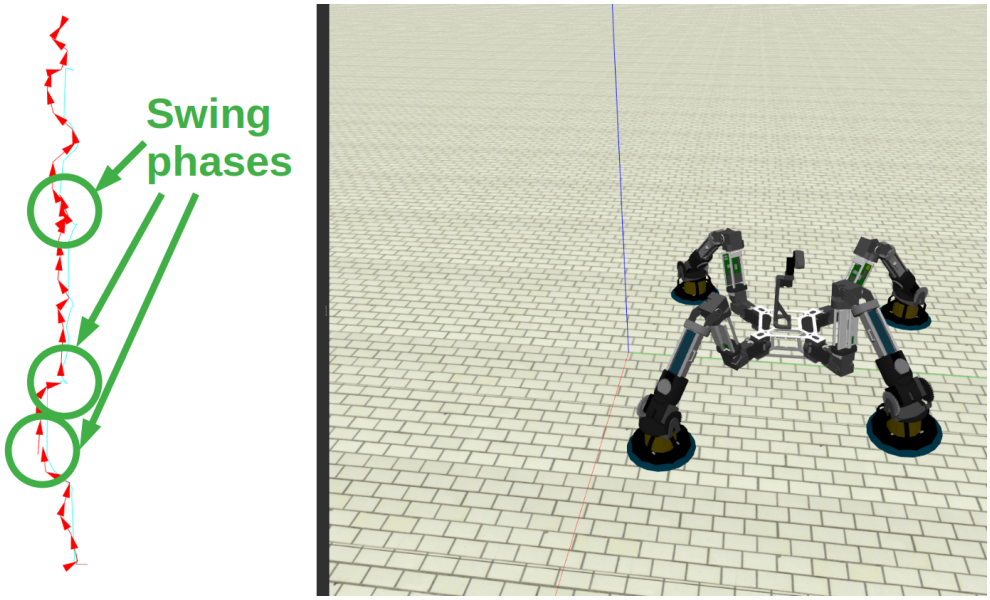


Figure 7.15: Replanification and swing phases.

ability. The visualization has been created with the Point Cloud Library (PCL), and has the components included in the caption of the figure.

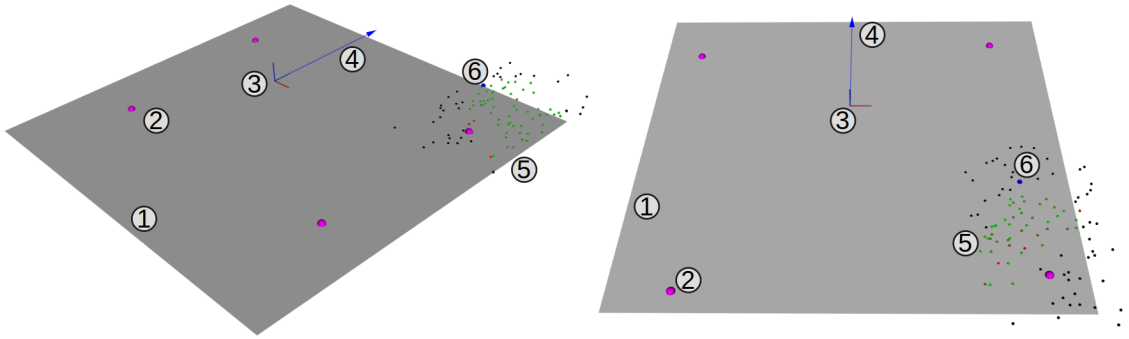


Figure 7.16: Footfall planner examples. Legend: (1) Detected plane; (2) contact points; (3) body COM; (4) body velocity vector; (5) random sampling with color gradient, being black non-suitable, red low suitability, and green high suitability; and (6) best point according to the cost function.

Although the behavior of both model-based path and footfall planners is demonstrated in the next subsection, how it operates is also shown in a video separately [7].

7.4.2 Locomotion analysis

The gait has been tested in the ground, inclined walls, vertical walls, overhangs, and ceilings with the objective of verifying all together the torque-based control method, the path and footfall planning, the gait controller, and the swing phase generation. First, various instances of the digital twin and other simulated robot topologies are provided as

[7] Video available at `https://youtu.be/A9UFr_7qf7k`

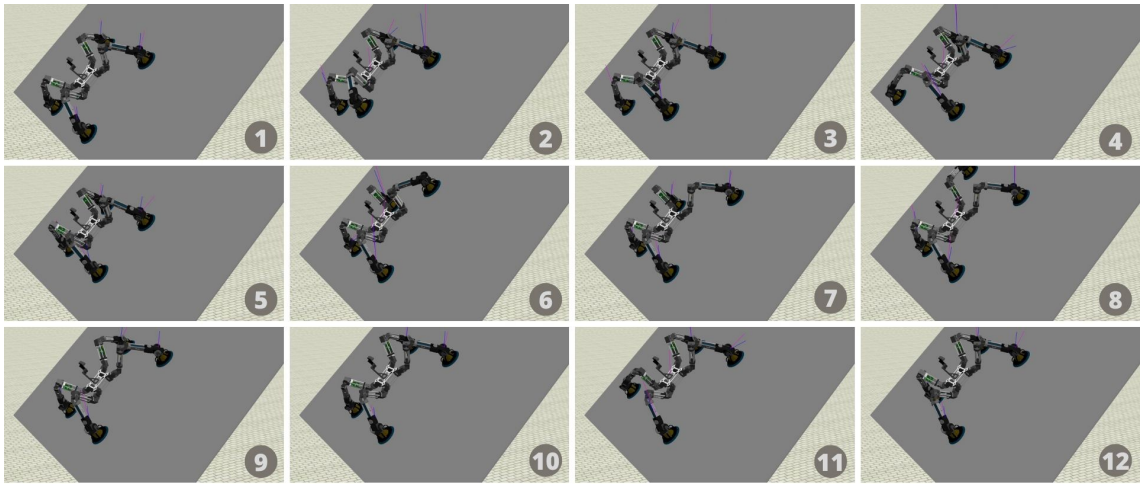


Figure 7.17: ROMERIN (quadruped version) gait over an inclined wall (45°).

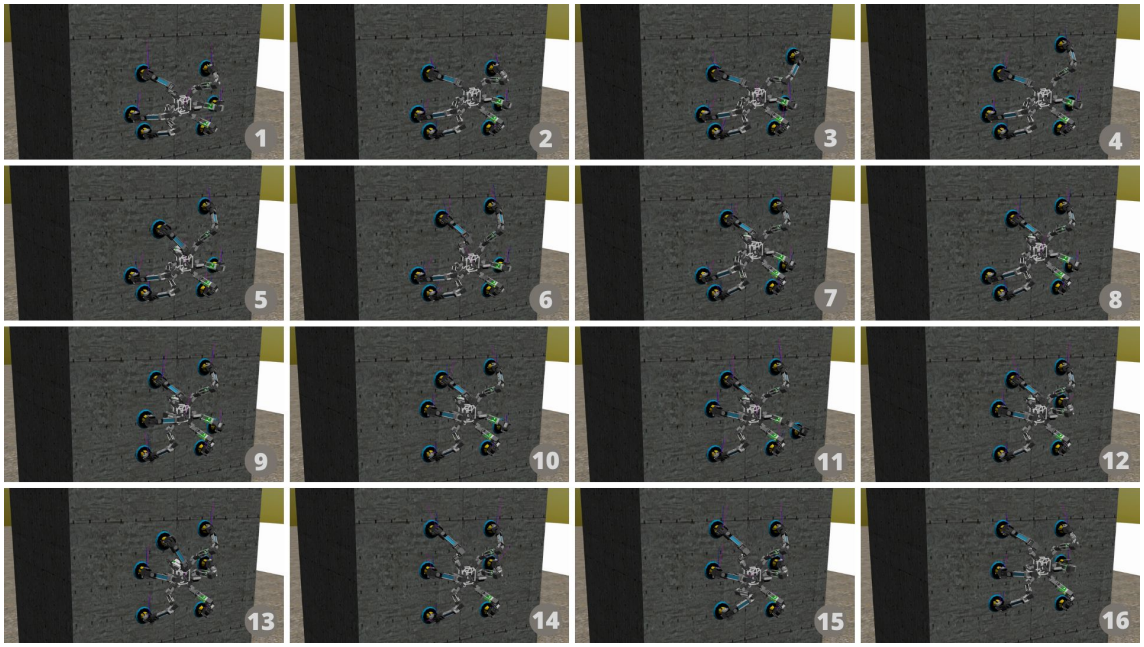


Figure 7.18: ROMERIN (hexapod version) gait on the vertical wall of a bridge (90°).

illustrative examples. As previously highlighted in this dissertation, the use of simulators serves a pivotal purpose by safeguarding the system from potential damage during algorithm testing, ultimately leading to a reduction in maintenance and testing time, material wastage, and associated costs. Later, a couple of examples with the ROMERIN robot are illustrated.

As an example with the simulated environment, Figure 7.17 shows screenshots of the behavior of a quadruped robot on an inclined wall of 45°. Similarly, Figure 7.18 shows the behavior of the ROMERIN hexapod configuration on a vertical wall.

Regarding the physical platform ROMERIN, Figure 7.19 shows the behavior of the quadruped and built robotic organism detailed in Chapter 3 on a wall of 80° of inclination

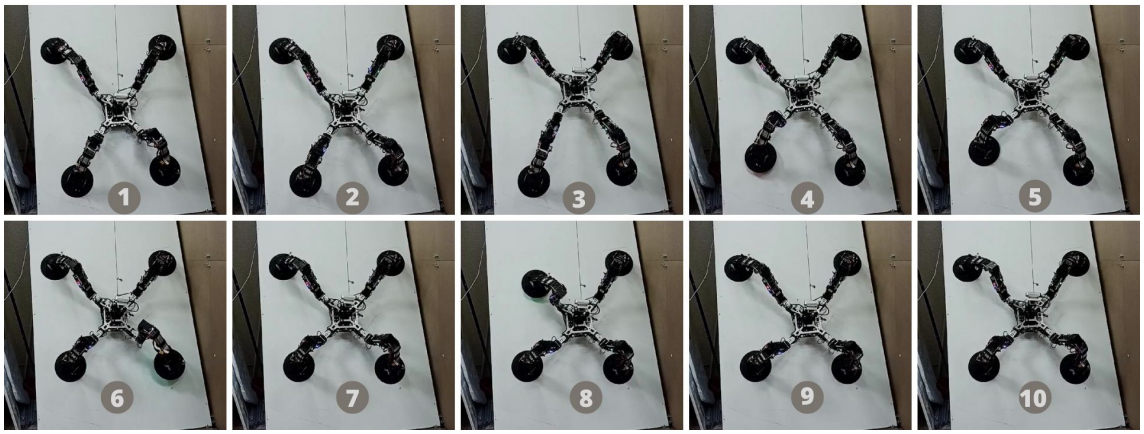


Figure 7.19: The quadruped robot ROMERIN climbing on a wall (80° of inclination).

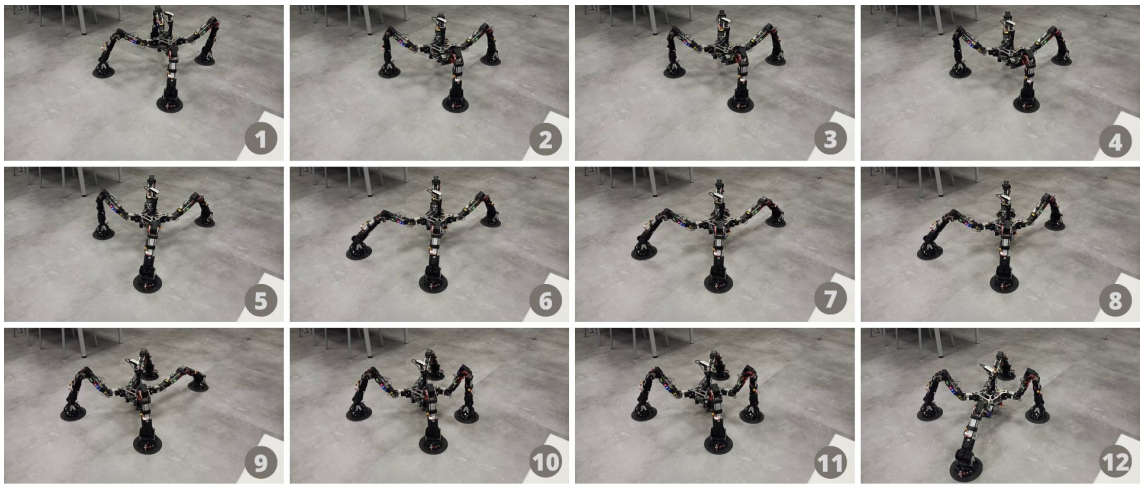


Figure 7.20: The quadruped robot ROMERIN walking on the horizontal floor.

located in the ETSIDI school [8]. Similarly, Figure 7.20 shows the behavior of ROMERIN walking on the horizontal floor [9].

Figure 7.21 shows a sequence of the simulated quadrupled robot rotating on the horizontal floor [10]. This picture serves as a good example to illustrate the coordination of elements of the path and footfall planners (Section 6.3.5). First of all, the robot rotates according to the user's command. Before lifting a leg, it is verified that the robot is going to be in a stable situation and within the limits established by the Safety Checker. If the robot will be in a conflict situation, the path planner defines the trajectory to follow to obtain the best possible position. After finding a feasible position, the module indicated by the Safety Checker changes from stance to swing phase and moves to a position established by the footfall planner.

As denoted previously, these experiments serve to test torque-based control and MPC-based path and footfall planning. To analyze locomotion, the gait parameters of a symmetric quadruped robot configuration are included for different inclinations in Table 7.2.

[8] Video available at `https://www.youtube.com/shorts/_7qXBQ_rdeQ`
[9] Video available at `https://youtu.be/cU3i-Q7CyUA`
[10] Video available at `https://youtu.be/zHRZ_ANPpSo`

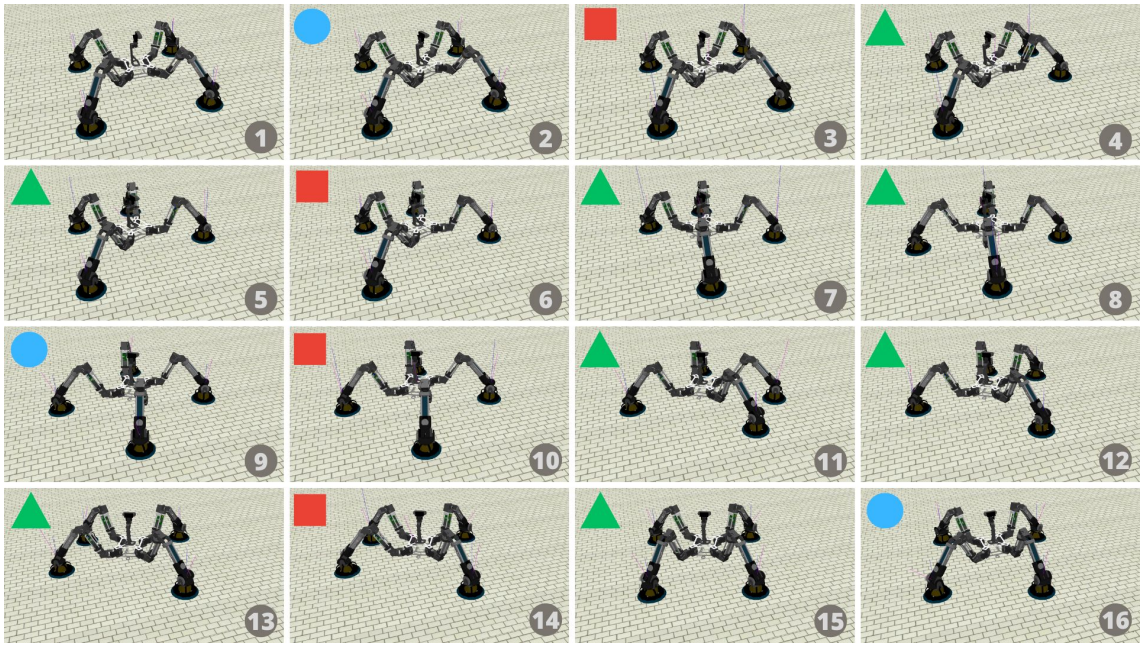


Figure 7.21: ROMERIN (quadruped version) rotating on the horizontal floor. Blue circles represent rotation, red squares represent robot relocation while searching a comfortable position before the swing phase, and green triangles represent swing phases.

In the analysis, the stoke and stride length have the same value since during the swing phase the body keeps its desired position constantly. Similarly, in Table 7.3 an analysis is included for a hexapod configuration.

Parameters	**Horizontal**	**Inclined**	**Vertical**	**Overhang**	**Ceiling**
Average velocity (cm/s)	0.81	0.73	0.68	0.64	0.76
Step distance (cm)	$21,86 \pm 4.12$	18.15 ± 6.56	21.87 ± 3.45	22.81 ± 5.06	23.18 ± 3.17
Swing time (s)	3.05 ± 0.03	3.03 ± 0.03	3.17 ± 0.68	4.09 ± 1.40	3.86 ± 0.55
Cycle time (s)	25.71 ± 3.63	21.29 ± 4.61	33.38 ± 7.96	29.32 ± 5.99	29.06 ± 9.26
Duty factor (%)	87.85 ± 2.23	84.75 ± 5.10	89.56 ± 1.89	87.04 ± 4.70	85.44 ± 5.15
Stroke and stride length (cm)	21.08 ± 3.56	16.92 ± 4.41	19.93 ± 3.21	20.77 ± 3.76	20.74 ± 8.31
Body height (cm)	10.18 ± 0.13	10.92 ± 0.02	10.05 ± 0.03	10.73 ± 0.01	10.77 ± 0.02
Lateral offset (cm)	27.15 ± 0.24	27.98 ± 0.17	28.78 ± 0.20	27.98 ± 0.20	25.17 ± 0.34

Table 7.2: Gait parameters of a symmetric quadruped robot configuration under different inclinations. The maximum body speed is set to $8cm/s$.

Parameters	**Horizontal**	**Inclined**	**Vertical**	**Overhang**	**Ceiling**
Average velocity (cm/s)	0.49	0.62	0.39	0.58	0.62
Step distance (cm)	17.05 ± 5.72	23.57 ± 5.87	15.41 ± 5.30	24.08 ± 4.32	26.60 ± 6.78
Swing time (s)	3.14 ± 0.22	3.41 ± 0.30	3.05 ± 0.05	3.52 ± 0.54	4.15 ± 1.96
Cycle time (s)	29.22 ± 11.85	34.93 ± 9.85	29.67 ± 8.98	42.37 ± 5.05	49.98 ± 19.58
Duty factor (%)	88.52 ± 4.40	89.53 ± 3.61	88.91 ± 3.18	91.43 ± 1.05	90.48 ± 3.83
Stroke and stride length (cm)	13.71 ± 5.04	20.07 ± 5.92	13.27 ± 2.31	23.48 ± 3.15	24.46 ± 5.93
Body height (cm)	11.92 ± 0.02	13.13 ± 0.04	9.60 ± 0.03	11.34 ± 0.02	10.77 ± 0.04
Lateral offset (cm)	26.08 ± 0.21	27.91 ± 0.32	27.80 ± 0.21	24.66 ± 0.39	24.91 ± 0.36

Table 7.3: Gait parameters of an hexapod robot configuration under different inclinations. The maximum body speed is set to $8cm/s$.

7.5 Summary

This chapter demonstrates the static locomotion capabilities of ROMERIN using the hierarchical task-space torque-based control framework presented in Chapter 5 and included in the MoCLORA architecture presented in Chapter 4. ROMERIN robot presented in Chapter 3 has been used for the experiments, as well as its digital twin and other simulated configurations presented in Chapter 4.3.

Starting with the control architecture implementation setup, the improvement of the low-level control loop of the actuators was highlighted. The control architecture and the robot platform has been validated with an inverse kinematic controller (IKC), and the problems to use this type of controller have been highlighted.

The torque-based control presented in Chapter 5 has been evaluated and demonstrated in discrete and continuous environments and in ground tests. The proposed method has been evaluated with both hardware and a simulated version of the ROMERIN robot. Furthermore, our controller has been compared with the usual kinematic inverse controllers, demonstrating that the mean angular and linear error is significantly reduced, as well as the power requirements of the actuators. To do so, first, the position control of the 2DOF per leg biped robot has been included to validate the control approach. Second, the control of the robot platform ROMERIN and the applicability of the gravity compensation method have been demonstrated. Lastly, the same approach has been implemented in another multilimbed robot, the Unitree GO1, to validate torque-based control.

To demonstrate the accuracy of the reaction force estimator, a static gait has been included with the GO1 robot using the reaction force estimator to determine robot steps. It has been used to detect if a leg can be lifted without compromising the stability and to detect if a leg is touching the ground without using force/torque sensors in the foot.

To point out the behavior of the model-based path and footfall planner, a locomotion analysis with the ROMERIN robot, its digital twin, and other robot configurations have been included for horizontal, inclined, vertical, overhang, and ceiling inclinations.

Chapter 8

Conclusions and main contributions

"It is strange that only extraordinary men make the discoveries, which later appear so easy and simple"

Georg C. Lichtenberg

This dissertation started with the vision of developing a multi-legged climbing robot that achieves locomotion capabilities in different inclinations. Inspired by nature and driven by the continuous progress in system engineering, a legged climbing system is built and controlled. It is capable of performing reactive gaits, adaptable to different morphologies, and optimizes its own power consumption and safety. It was not the mission to focus on reproducing single theoretical findings, but to come up with a robotic system that combines versatility, efficiency, and robustness in one single device. Closing the gap between the develop machines and the natural counterparts requires solutions that are advanced with respect to all of these properties.

8.1 Discussion of contributions

This thesis presents the following original contributions:

1. **Validation of a manually configured and homogeneous legged and climbing robotic organism with power-sharing capabilities.**

   This thesis lays the foundations for control for a modular versatile legged and climbing system, called ROMERIN, which is expected to inspect a large number of civil infrastructures with difficult access. The robot is capable of adapting its characteristics to move on the ground, in inclined walls, vertical walls, overhangs, and even in ceilings. It has the capability of carrying navigation and inspection equipment such as 3D cameras or LIDAR, and NDT equipment such as ultrasonic sensors and penetration georadars to perform inspection tasks. A custom-made suction cup, equipped

with its own turbine to create the vacuum needed to ensure robot adhesion, has been optimized and installed in each leg.

The thesis validates the robot concept for climbing and inspection applications. The ROMERIN robot, which does not have an analog system in the literature, has been used and controlled for the validation of the approach.

2. **Design of a control architecture specialized for the control of climbing multi-limbed robots.**

   This thesis presents a novel modular architecture that coordinates the modules of a legged and climbing robotic organism with any morphology, at the same time allowing exchanges between the physical robot and its digital twin. It includes the basic low-level layers to control and coordinate all the elements for locomotion and the high-level components to control the robot motion in the environment. The control architecture allows the user to add new higher-level components to improve the behavior of the organism. The proposed control architecture gathers the advantages and applicable concepts and components of other architectures found in the state of the art.

3. **Creation of a simulated environment for fast-testing of variable robots with different numbers of legs.**

   This thesis includes a simulated environment with different modalities for the simulation, testing, monitoring, and illustration of a robotic organism with a variable number of legs. It presents a tool for algorithm testing, a digital twin of the ROMERIN robot, and a digital mirror that mimics the behavior of the physical organism. The architecture proposed in this thesis allows versatility not only in terms of the arrangement of the leg modules, but also in terms of the totally transparent combination of real and virtual instances.

4. **Generalizable, low computational cost, simple and fast method of gravity compensation for climbing robots with a variable number of legs.**

   The proposed gravity compensation method is based on the static problem, which is a reduction in the dynamic model of the robot that takes advantage of the low velocity of the climbing robots. The presented method has been shown to **be twice as fast** compared to the most popular one, which is based on the Lagrange equation and the use of the robot Jacobian. This fact is particularly important for a multi-legged system where the logic runs on a single computer and where the control rate and computational cost are critical. This importance increases for systems such as ROMERIN, where the number of legs can be considerably increased to adapt to specific real-time applications. In addition, it is especially important in sampling-based algorithms for motion planning, where computation is performed many times.

5. **Model-based path and footfall planner for a modular legged and climbing robot with a non-periodic, non-symmetric, and non-regular bio-inspired gait controller.**

   In this thesis, model-based path and footfall planners are presented for a modular legged and climbing robot. Both planners minimize the actuators' energy consumption and optimize the robot adhesion safety. To do so, the FDP is solved and used by selecting those configurations in which the robot is most comfortable from a kinematic and effort point of view.

This doctoral thesis proposes a sampling-based method (Monte Carlo algorithm) for the generation of trajectories and the selection of support points. To do so, a set of cost functions are proposed that penalize those configurations where the actuators are overloaded and the suction cups are at risk of detachment. In addition, the cost functions attempt to distribute the weight between the analog actuators of the modules and maximize the distance traveled when the organism takes a step. After computing the cost functions for the samples, the path with the lowest cost is chosen with a graph exploration algorithm. Similarly, the support point after the step is chosen according to the cost function of the footfall planner.

A gait controller is also included as a non-periodic, non-symmetric, and non-regular bio-inspired method that selects the most convenient leg to move by ensuring comfort, safety, and robot capabilities.

8.2 Future work

- **Improvement of the hardware (1) - Mechanical design**. Although the initial leg design has been considerably improved through several iterations, flaws have been detected during experiments. The main origin of the problems has been detected in the components made up of plastic pieces, which make the legs have too large clearances. Links composed of several pieces are also a source of inaccuracies. In addition, under normal operation, it is common for plastic pieces to break or several pieces to dismount due to inner torques.

  Another important hardware flaw is detected in the union between the suction cup and the leg modules. Even though the idea of the concurrent actuators in the wrist is correct and compact, the mounted gears use to loose steps frequently, producing a loss of real information from the actuators encoder. Furthermore, due to this initial design, it has been detected that applying low forces radially to the suction cup, it is possible to deform the position of the suction cup up to 15° without actuator motion, which estimates the position of the TCP very inaccurate during operation. Furthermore, the small components of the suction cup are weak and easy to break or dismount during operation. This is because many of the unions are done mechanically or using screws that do not have a fastening nut.

  The first version of the leg modules has shown a large deformation because the first motor uses to support a strong radial force. Under normal operation, a gap of up to 10 cm of estimation has been detected between the forward kinematic and real TCP position of the leg module. This radial force produces a deformation that, together with the flaws of the robot mechanics, makes the inaccuracies too large.

  In summary, plastic pieces and the set of mechanically linked pieces are strongly recommended to be replaced by aluminum-designed components. Aluminum has low density, excellent corrosion resistance, and can be easily cast, machined, and formed. By using this material instead of plastic, a smaller and more robust system can be achieved, significantly improving the robot capabilities, control accuracy, and state estimation, and reducing the actuator torque requirements. Furthermore, a deeper mechanical design is required by analyzing where the main forces are located and by locating bearings and other elements that constrain relative motion to only the desired one.

- **Improvement of the hardware (2) - Actuators substitution**. A Dynamixel servomotor is a smart actuator system developed to be the exclusive connecting joints on a robot or mechanical structure. They are designed to be modular and daisy chained on any robot or mechanical design for powerful and flexible robotic movements. However, the computation of the static model reveals that the actuator features are under the robot requirements, especially during the swing phase of any of the legs. Although it can be compensated with an improved mechanical design, the higher torque requirements should be satisfied in future versions.

  The current trend in legged robotics is to use very powerful actuators for dynamic locomotion. For example, StarlETH uses 200W actuators of 3.24Nm and 1:100 of reduction for a 23kg quadruped robot, whereas our better actuator is 58.8W and 9.9Nm (which is the best actuator of Dynamixel). Another example is Go1 from Unitree, which uses 23.7Nm actuators. Since in ROMERIN only static locomotion is required, the requirements would be lower than in other legged robots. However, it is strongly recommended to increase the actuator power.

- **Improvement of the state estimator**. The state estimator explained in Section 5.7 is the first approach to estimating the robot position. It serves as a first approximation to determine the odometry of the robot. However, a more reliable estimator should be used, especially for longer travel distances. Other works with legged robots opt, among others, for Kalman filters [Bloesch et al., 2013], or factor graph optimization methods [Wisth et al., 2019].

- **Adhesion points detection**. It would consist of detecting feasible contact points for the adhesion of the robot suction cups. These points have to meet some requirements. For instance, the roughness of the material has to be low to guarantee proper vacuum generation. The surface should also be flat, to be able to place the suction cup without producing pressure losses, and larger than the area of the suction cup. The detected areas will be evaluated with the footfall planner defined in this dissertation.

- **Graphical user interface improvement**. It is recommended to improve the GUI to add new functionalities, as well as the integration of a gamepad for robot manipulation. A crucial part that is not yet implemented is to allow the user to move and rotate the robot with the gamepad.

8.3 Related publications

Throughout these years of research, the following publications related to the thesis have been produced.

8.3.1 Journal Articles

"Behavior-Based Control Architecture for Legged-and-Climber Robots"

Authors: Hernando, M., Alonso, M., Prados, C., & Gambao, E.

Journal: Applied Sciences

Reference: 2021, 11.20, p. 9547

ISSN/EISSN: -/2076-3417
Impact factor: 2.838 (2021)
Quartil: Q2
Area: 39 out of 92 in Engineering, multidisciplinary
DOI: 10.3390/app11209547
URL: https://doi.org/10.3390/app11209547.

"ROMERIN: A new concept of a modular autonomous climbing robot"
Authors: Hernando, M., Gambao, E., Prados, C., Brito, D., & Brunete, A.
Journal: International Journal of Advanced Robotic Systems
Reference: 2022, 19.5, p. 17298806221123416
ISSN/EISSN: 1729-8806/1729-8814
Impact factor: 2.3 (2022)
Quartil: Q4
Area: 24 out of 30 in Robotics
DOI: 10.1177/17298806221123416
URL: https://doi.org/10.1177/17298806221123416.

"ROMERIN: Organismo robótico escalador basado en patas modulares con ventosas activas"
Authors: Prados, C., Hernando, M., Gambao, E., & Brunete, A.
Journal: Revista Iberoamericana de Automática e Informática industrial
Reference: 2022, 20.2, p. 175-186
ISSN/EISSN: 1697-7912/1697-7920
Impact factor: 1.5 (2022)
Quartil: Q4
Area: 55 out of 65 in Automation & Control systems; 28 out of 30 in Robotics
DOI: 10.4995/riai.2022.18749
URL: https://doi.org/10.4995/riai.2022.18749.

"MoCLORA - An Architecture for Legged-and-Climbing Bio-Inspired Modular Robotic Organism"
Authors: Prados, C., Hernando, M., Gambao, E., & Brunete, A.
Journal: Biomimetics
Reference: 2023, 8.1, p. 11
ISSN/EISSN: -/2313-7673
Impact factor: 4.5 (2022)
Quartil: Q1
Area: 22 out of 90 in Engineering, multidisciplinary
DOI: 10.3390/biomimetics8010011
URL: https://doi.org/10.3390/biomimetics8010011.

"Torque-based control of a bio-inspired modular climbing robot"
Authors: Prados, C., Hernando, M., Gambao, E., & Brunete, A.
Journal: Machines
Reference: 2023, 11.7, p. 757
ISSN/EISSN: -/2075-1702
Impact factor: 2.6 (2022)
Quartil: Q2
Area: 62 out of 135 in Engineering, mechanical
DOI: 10.3390/machines11070757
URL: `https://doi.org/10.3390/machines11070757`.

In Preparation **"Path and footfall planning for a modular legged and climbing robot - A model predictive control approach"**
Authors: Prados, C., Hernando, M., & Gambao, E.
Journal: Journal of Field Robotics
ISSN/EISSN: 1556-49595/1556-4967
Impact factor: 8.3 (2022)
Quartil: Q1
Area: 5 out of 30 in Robotics

In Preparation **"Whole body control of a quadruped robot on yielding grounds"**
Authors: Godon, S., Prados, C., Chemori, A., Kruusmaa, M. & Ristolainen, A.
Journal: IEEE Transactions on Robotics
ISSN/EISSN: 1552-3098/1941-0468
Impact factor: 7.8 (2022)
Quartil: Q1
Area: 7 out of 30 in Robotics

8.3.2 Conference Proceedings

"ROMERIN: Organismo robótico escalador basado en patas modulares con ventosas activas"
Authors: Prados, C., Hernando, M, Gambao, E, & Brunete, A.
Conference: Jornadas de Robótica, Educación y Bioingeniería
Place and date: Málaga (Spain), 18-20 May, 2022
Role: Author and speaker

"ROMERIN: Organismo robótico escalador basado en patas modulares con ventosas activas"
Authors: Prados, C., Hernando, M, Gambao, E, & Brunete, A.
Conference: I Jornada de Investigación de la ETSIDI

Place and date: Madrid (Spain), 21 November, 2022
Role: Author and speaker

Bibliography

[Ahmadzadeh et al., 2016] Ahmadzadeh, H., Masehian, E., and Asadpour, M. (2016). Modular Robotic Systems: Characteristics and Applications. *Journal of Intelligent and Robotic Systems*, 81(3):317–357. https://doi.org/10.1007/s10846-015-0237-8.

[Akinfiev and Armada, 2002] Akinfiev, T. and Armada, M. (2002). The influence of gravity on trajectory planning for climbing robots with non-rigid legs. *Journal of Intelligent and Robotic Systems*, 35:309–326.

[Alami et al., 1998] Alami, R., Chatila, R., Fleury, S., Ghallab, M., and Ingrand, F. (1998). An architecture for autonomy. *The International Journal of Robotics Research*, 17(4):315–337. https://doi.org/10.1177%2F027836499801700402.

[Alkalla et al., 2017] Alkalla, M. G., Fanni, M. A., Mohamed, A. M., and Hashimoto, S. (2017). Tele-operated propeller-type climbing robot for inspection of petrochemical vessels. *Industrial Robot: An International Journal*, 44(2):166–177. https://doi.org/10.1108%2Fir-07-2016-0182.

[Andrikopoulos et al., 2019] Andrikopoulos, G., Papadimitriou, A., Brusell, A., and Nikolakopoulos, G. (2019). On model-based adhesion control of a vortex climbing robot. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1460–1465. IEEE. https://doi.org/10.1109%2Firos40897.2019.8968069.

[Arikawa and Hirose, 1996] Arikawa, K. and Hirose, S. (1996). Development of quadruped walking robot titan-viii. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS'96*, volume 1, pages 208–214. IEEE. https://doi.org/10.1109%2Firos.1996.570670.

[Azayev and Zimmerman, 2020] Azayev, T. and Zimmerman, K. (2020). Blind hexapod locomotion in complex terrain with gait adaptation using deep reinforcement learning and classification. *Journal of Intelligent & Robotic Systems*, 99(3-4):659–671. https://doi.org/10.1007%2Fs10846-020-01162-8.

[Baghani et al., 2005] Baghani, A., Ahmadabadi, M., and Harati, A. (2005). Kinematics modeling of a wheel-based pole climbing robot (UT-PCR). In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation.* IEEE. https://doi.org/10.1109%2Frobot.2005.1570423.

[Bahr et al., 1996] Bahr, B., Li, Y., and Najafi, M. (1996). Design and suction cup analysis of a wall climbing robot. *Computers & electrical engineering*, 22(3):193–209. https://doi.org/10.1016/0045-7906(95)00039-9.

[Bai et al., 2019] Bai, L., Hu, H., Chen, X., Sun, Y., Ma, C., and Zhong, Y. (2019). Cpg-based gait generation of the curved-leg hexapod robot with smooth gait transition. *Sensors*, 19(17):3705. https://doi.org/10.3390%2Fs19173705.

[Balaguer et al., 1998] Balaguer, C., Pastor, J., Giménez, A., Padrón, V., and Abderrahim, M. (1998). Roma: Multifunctional autonomous self-supported climbing robot for inspection applications. *IFAC Proceedings Volumes*, 31(3):563–568. https://doi.org/10.1016/s1474-6670(17)44145-0.

[Bandyopadhyay et al., 2018] Bandyopadhyay, T., Steindl, R., Talbot, F., Kottege, N., Dungavell, R., Wood, B., Barker, J., Hoehn, K., and Elfes, A. (2018). Magneto: A Versatile Multi-Limbed Inspection Robot. *IEEE International Conference on Intelligent Robots and Systems*, pages 2253–2260. https://doi.org/10.1109/IROS.2018.8593891.

[Bares et al., 1989] Bares, J., Hebert, M., Kanade, T., Krotkov, E., Mitchell, T., Simmons, R., and Whittaker, W. (1989). Ambler: An autonomous rover for planetary exploration. *Computer*, 22(6):18–26. https://doi.org/10.1109%2F2.30717.

[Barrio et al., 2020] Barrio, R., Lozano, Á., Rodríguez, M., and Serrano, S. (2020). Numerical detection of patterns in cpgs: Gait patterns in insect movement. *Communications in Nonlinear Science and Numerical Simulation*, 82:105047. https://doi.org/10.1016%2Fj.cnsns.2019.105047.

[Beeson and Ames, 2015] Beeson, P. and Ames, B. (2015). TRAC-IK: An open-source library for improved solving of generic inverse kinematics. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE. https://doi.org/10.1109%2Fhumanoids.2015.7363472.

[Bellicoso et al., 2016] Bellicoso, C. D., Gehring, C., Hwangbo, J., Fankhauser, P., and Hutter, M. (2016). Perception-less terrain adaptation through whole body control and hierarchical optimization. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 558–564. IEEE. https://doi.org/10.1109%2Fhumanoids.2016.7803330.

[Bendris and Cayero Becerra, 2022] Bendris, B. and Cayero Becerra, J. (2022). Design and experimental evaluation of an aerial solution for visual inspection of tunnel-like infrastructures. *Remote Sensing*, 14(1):195.

[Bensalem et al., 2009] Bensalem, S., Gallien, M., Ingrand, F., Kahloul, I., and Nguyen, T.-H. (2009). Toward a more dependable software architecture for autonomous robots. *IEEE Robotics and Automation Magazine*, 16(1):67–77.

[Bevly et al., 2000] Bevly, D. M., Farritor, S., and Dubowsky, S. (2000). Action module planning and its application to an experimental climbing robot. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 4, pages 4009–4014. IEEE. https://doi.org/10.1109/robot.2000.845356.

[Bisht et al., 2022] Bisht, R. S., Pathak, P. M., and Panigrahi, S. K. (2022). Design and development of a glass façade cleaning robot. *Mechanism and Machine Theory*, 168:104585. https://doi.org/10.1016/j.mechmachtheory.2021.104585.

[Bloesch et al., 2013] Bloesch, M., Hutter, M., Hoepflinger, M. A., Leutenegger, S., Gehring, C., Remy, C. D., and Siegwart, R. (2013). State estimation for legged robots-consistent fusion of leg kinematics and imu. *Robotics*, 17:17–24. https://doi.org/10.7551%2Fmitpress%2F9816.003.0008.

[Bonasso et al., 1996] Bonasso, R. P., Kortenkamp, D., Miller, D. P., and Slack, M. (1996). Experiences with an architecture for intelligent, reactive agents. In *Intelligent Agents II Agent Theories, Architectures, and Languages: IJCAI'95 Workshop (ATAL) Montréal, Canada, August 19–20, 1995 Proceedings 2*, pages 187–202. Springer. https://doi.org/10.1007%2F3540608052_67.

[Borrelly and Simon, 1990] Borrelly, J.-J. and Simon, D. (1990). Propositions d'architecture de contrôleur ouvert pour la robotique. *Rapports de recherche- INRIA*.

[Brakel et al., 2022] Brakel, P., Bohez, S., Hasenclever, L., Heess, N., and Bousmalis, K. (2022). Learning coordinated terrain-adaptive locomotion by imitating a centroidal dynamics planner. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10335–10342. IEEE. https://doi.org/10.1109%2Firos47612.2022.9981648.

[Brito, 2020] Brito, D. (2020). Romerin: Diseño mecatrónico de un módulo robótico con capacidad de compartir energía. http://138.100.100.141:8080/share.cgi?ssid=0Mf8VTX. Bachelor Thesis. Accessed: 2023-04-28.

[Brockmann et al., 2011] Brockmann, W., Maehle, E., Grosspietsch, K.-E., Rosemann, N., and Jakimovski, B. (2011). Orca: An organic robot control architecture. In *Organic Computing—A Paradigm Shift for Complex Systems*, pages 385–398. Springer. https://doi.org/10.1007%2F978-3-0348-0130-0_25.

[Brunete et al., 2005] Brunete, A., Hernando, M., and Gambao, E. (2005). Modular multiconfigurable architecture for low diameter pipe inspection microrobots. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 490–495. IEEE. https://doi.org/10.1109%2Frobot.2005.1570166.

[Brunete et al., 2017] Brunete, A., Ranganath, A., Segovia, S., de Frutos, J., Hernando, M., and Gambao, E. (2017). Current trends in reconfigurable modular robots design. *International Journal of Advanced Robotic Systems*, 14(3). https://doi.org/10.1177/1729881417710457.

[Brusell et al., 2016] Brusell, A., Andrikopoulos, G., and Nikolakopoulos, G. (2016). A survey on pneumatic wall-climbing robots for inspection. In *2016 24th Mediterranean Conference on Control and Automation (MED)*, pages 220–225. https://doi.org/10.1109/med.2016.7535885.

[Buettner et al., 2019] Buettner, T., Heppner, G., Roennau, A., and Dillmann, R. (2019). Nimble limbs - intelligent attachable legs to create walking robots from variously shaped objects. In *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE. https://doi.org/10.1109%2Faim.2019.8868845.

[Buettner et al., 2022] Buettner, T., Schwertfeger, O., Roennau, A., and Dillmann, R. (2022). A review of current approaches to configuration detection in modular legged robots. *Robotics in Natural Settings: CLAWAR 2022*, pages 167–180. https://doi.org/10.1007%2F978-3-031-15226-9_18.

[Buettner et al., 2018] Buettner, T., Wilke, D., Roennau, A., Heppner, G., and Dillmann, R. (2018). A scalable, modular leg design for multi-legged stair climbing robots. In *Robotics Transforming the Future*. CLAWAR Association Ltd. https://doi.org/10.13180%2Fclawar.2018.10-12.09.34.

[Cai et al., 2014] Cai, P., Indhumathi, C., Cai, Y., Zheng, J., Gong, Y., Lim, T. S., and Wong, P. (2014). Collision detection using axis aligned bounding boxes. In *Simulations, Serious Games and Their Applications*, pages 1–14. Springer. https://doi.org/10.1007%2F978-981-4560-32-0_1.

[Castano et al., 2002] Castano, A., Behar, A., and Will, P. M. (2002). The conro modules for reconfigurable robots. *IEEE/ASME transactions on mechatronics*, 7(4):403–409. https://doi.org/10.1109%2Ftmech.2002.806233.

[Chen et al., 2016] Chen, C.-A., Collins, T., and Shen, W.-M. (2016). A near-optimal dynamic power sharing scheme for self-reconfigurable modular robots. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5183–5188. https://doi.org/10.1109/ICRA.2016.7487724.

[Chen et al., 1998] Chen, J.-S., Cheng, F.-T., Yang, K.-T., Kung, F.-C., and Sun, Y.-Y. (1998). Solving the optimal force distribution problem in vehicles. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, volume 1, pages 471–476. IEEE. https://doi.org/10.1109%2Frobot.1998.677019.

[Chen et al., 1999] Chen, X., Watanabe, K., Kiguchi, K., and Izumi, K. (1999). Optimal force distribution for the legs of a quadruped robot. *Machine Intelligence and Robotic Control*, 1(2):87–93.

[Cheng and Orin, 1990] Cheng, F.-T. and Orin, D. E. (1990). Efficient algorithm for optimal force distribution-the compact-dual lp method. *IEEE Transactions on Robotics and Automation*, 6(2):178–187.

[Cheng and Orin, 1991] Cheng, F.-T. and Orin, D. E. (1991). Optimal force distribution in multiple-chain robotic systems. *IEEE Transactions on systems, man, and cybernetics*, 21(1):13–24. https://doi.org/10.1109%2F21.101132.

[Chong et al., 2018] Chong, B., Aydin, Y. O., Gong, C., Sartoretti, G., Wu, Y., Rieser, J. M., Xing, H., Rankin, J. W., Michel, K., Nicieza, A. G., et al. (2018). Coordination of back bending and leg movements for quadrupedal locomotion. In *Robotics: Science and Systems*, volume 20. Pittsburgh, PA. https://doi.org/10.15607%2Frss.2018.xiv.020.

[Clodic et al., 2006] Clodic, A., Fleury, S., Alami, R., Chatila, R., Bailly, G., Brethes, L., Cottret, M., Danes, P., Dollat, X., Elisei, F., et al. (2006). Rackham: An interactive robot-guide. In *ROMAN 2006-the 15th IEEE International Symposium on Robot and Human Interactive Communication*, pages 502–509. IEEE. https://doi.org/10.1109%2Froman.2006.314378.

[Collins et al., 2021] Collins, J., Chand, S., Vanderkop, A., and Howard, D. (2021). A review of physics simulators for robotic applications. *IEEE Access*, 9:51416–51431. https://doi.org/10.1109%2Faccess.2021.3068769.

[Cong et al., 2020] Cong, Z., Honglei, A., Wu, C., Lang, L., Wei, Q., and Hongxu, M. (2020). Contact force estimation method of legged-robot and its application in impedance control. *IEEE Access*, 8:161175–161187. https://doi.org/10.1109%2Faccess.2020.3021080.

[CosteManiere et al., 1992] CosteManiere, E., Espiau, B., and Simon, D. (1992). Reactive objects in a task level open controller. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 2732–2733. IEEE Computer Society. https://doi.org/10.1109%2Frobot.1992.219993.

[Das et al., 1999] Das, H., Bao, X., Bar-Cohen, Y., Bonitz, R., Lindemann, R. A., Maimone, M., Nesnas, I. A., and Voorhees, C. J. (1999). Robot manipulator technologies for planetary exploration. In *Smart Structures and Materials 1999: Smart Structures and Integrated Systems*, volume 3668, pages 175–182. International Society for Optics and Photonics.

[de las Heras, 2023] de las Heras, F. (2023). Modelado y simulación de infraestructuras en gazebo para la validación de estrategias de escalado del robot romerin. https://138.100.100.141/share.cgi?ssid=0XRSsFe. Bachelor Thesis. Accessed: 2023-06-28.

[De Santos et al., 2007] De Santos, P. G., Cobano, J. A., Garcia, E., Estremera, J., and Armada, M. (2007). A six-legged robot-based system for humanitarian demining missions. *Mechatronics*, 17(8):417–430. https://doi.org/10.1016%2Fj.mechatronics.2007.04.014.

[Dethe and Jaju, 2014] Dethe, R. D. and Jaju, S. (2014). Developments in wall climbing robots: a review. *International journal of engineering research and general science*, 2(3):33–42.

[Ding et al., 2010] Ding, X., Wang, Z., Rovetta, A., and Zhu, J. (2010). Locomotion analysis of hexapod robot. *Climbing and walking robots*, 28:291–310. https://doi.org/10.5772%2F8822.

[Ding, 2013] Ding, X. J. (2013). Research on collision detection algorithm based on obb. In *Applied Mechanics and Materials*, volume 433, pages 936–939. Trans Tech Publ. https://doi.org/10.4028%2Fwww.scientific.net%2Famm.433-435.936.

[Doosti et al., 2019] Doosti, P., Mahjoob, M., and Dadashzadeh, B. (2019). Finite-time control strategy for the running of a telescopic leg biped robot. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 41:1–10. https://doi.org/10.1007%2Fs40430-019-1697-8.

[Durand et al., 2010a] Durand, B., Godary-Dejean, K., Lapierre, L., Passama, R., and Crestani, D. (2010a). Fault tolerance enhancement using autonomy adaptation for autonomous mobile robots. In *2010 Conference on Control and Fault-Tolerant Systems (SysTol)*, pages 24–29. IEEE. https://doi.org/10.1109%2Fsystol.2010.5676030.

[Durand et al., 2010b] Durand, B., Godary-Dejean, K., Lapierre, L., Passama, R., and Crestani, D. (2010b). Reliability improvement in control architecture for mobile robots: implementation using cotama. In *CAR: Control Architectures of Robots*.

[Durand et al., 2010c] Durand, B., Godary-Dejean, K., Lapierre, L., Passama, R., and Crestani, D. (2010c). Using adaptive control architecture to enhance mobile robot reliability. In *TAROS 2010-11th Annual Conference Towards Autonomous Robotic Systems*, pages 54–61.

[Dynamics, 2023] Dynamics, B. (2023). Spot - the agile mobile robot. https://www.bostondynamics.com/products/spot. Accessed: 2023-04-22.

[El Jalaoui et al., 2005] El Jalaoui, A., Andreu, D., and Jouvencel, B. (2005). A control architecture for contextual tasks management: application to the auv taipan. In *Europe Oceans 2005*, volume 2, pages 752–757. IEEE. https://doi.org/10.1109%2Foceanse.2005.1513150.

[El Jalaoui et al., 2006] El Jalaoui, A., Andreu, D., and Jouvencel, B. (2006). Contextual management of tasks and instrumentation within an auv control software architecture. In *IROS*, pages 3761–3766. https://doi.org/10.1109%2Firos.2006.281760.

[Espiau et al., 1996] Espiau, B., Kapellos, K., and Jourdan, M. (1996). Formal verification in robotics: Why and how? In *Robotics Research*, pages 225–236. Springer. https://doi.org/10.1007%2F978-1-4471-1021-7_26.

[Eto and Asada, 2020] Eto, H. and Asada, H. H. (2020). Development of a wheeled wall-climbing robot with a shape-adaptive magnetic adhesion mechanism. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. https://doi.org/10.1109%2Ficra40945.2020.9196919.

[Fankhauser et al., 2016] Fankhauser, P., Bellicoso, C. D., Gehring, C., Dubé, R., Gawel, A., and Hutter, M. (2016). Free gait—an architecture for the versatile control of legged robots. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 1052–1058. IEEE. https://doi.org/10.1109%2Fhumanoids.2016.7803401.

[Featherstone, 1999] Featherstone, R. (1999). A divide-and-conquer articulated-body algorithm for parallel o (log (n)) calculation of rigid-body dynamics. part 1: Basic algorithm. *The International Journal of Robotics Research*, 18(9):867–875. https://doi.org/10.1177%2F02783649922066619.

[Figueroa, 2021] Figueroa, A. (2021). Rediseño electrónico de romerin++: simplificación y actualización de la placa de control. https://138.100.100.141/share.cgi?ssid=0BNdMVz. Bachelor Thesis. Accessed: 2023-04-28.

[Fu et al., 2022] Fu, Z., Kumar, A., Agarwal, A., Qi, H., Malik, J., and Pathak, D. (2022). Coupling vision and proprioception for navigation of legged robots. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17273–17283. https://doi.org/10.1109%2Fcvpr52688.2022.01676.

[Fukui et al., 2019] Fukui, T., Fujisawa, H., Otaka, K., and Fukuoka, Y. (2019). Autonomous gait transition and galloping over unperceived obstacles of a quadruped robot with cpg modulated by vestibular feedback. *Robotics and Autonomous Systems*, 111:1–19. https://doi.org/10.1016%2Fj.robot.2018.10.002.

[Full and Koditschek, 1999] Full, R. J. and Koditschek, D. E. (1999). Templates and anchors: neuromechanical hypotheses of legged locomotion on land. *Journal of experimental biology*, 202(23):3325–3332. https://doi.org/10.1242%2Fjeb.202.23.3325.

[Galdelli et al., 2022] Galdelli, A., D'Imperio, M., Marchello, G., Mancini, A., Scaccia, M., Sasso, M., Frontoni, E., and Cannella, F. (2022). A novel remote visual inspection system for bridge predictive maintenance. *Remote Sensing*, 14(9):2248.

[Galvez et al., 2003] Galvez, J. A., Estremera, J., and De Santos, P. G. (2003). A new legged-robot configuration for research in force distribution. *Mechatronics*, 13(8-9):907–932. https://doi.org/10.1016%2Fs0957-4158%2803%2900008-4.

[Gambao et al., 2004] Gambao, E., Hernando, M., Hernández, F., and Pinilla, E. (2004). Cost-effective robots for façade cleaning. In *Proceedings of the 21st International Symposium on Automation and Robotics in Construction*, Jeju, Korea. International Association for Automation and Robotics in Construction (IAARC). https://doi.org/10.22260/ISARC2004/0025.

[Gao et al., 2020] Gao, F., Fan, J., Zhang, L., Jiang, J., and He, S. (2020). Magnetic crawler climbing detection robot basing on metal magnetic memory testing technology. *Robotics and Autonomous Systems*, 125:103439. https://www.sciencedirect.com/science/article/pii/S0921889019305664.

[Gao et al., 2023] Gao, Z., Chen, X., Yu, Z., Han, L., Zhang, J., and Huang, G. (2023). Hybrid momentum compensation control by using arms for bipedal dynamic walking. *Biomimetics*, 8(1):31. https://doi.org/10.3390%2Fbiomimetics8010031.

[Ge et al., 2016] Ge, D., Ren, C., Matsuno, T., and Ma, S. (2016). Guide rail design for a passive suction cup based wall-climbing robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5776–5781. IEEE. https://doi.org/10.1109/iros.2016.7759850.

[Gilpin and Rus, 2010] Gilpin, K. and Rus, D. (2010). Modular robot systems. *IEEE Robotics & Automation Magazine*, 17(3):38–55. https://doi.org/10.1109%2Fmra.2010.937859.

[Godon et al., 2023] Godon, S., Kruusmaa, M., and Ristolainen, A. (2023). Maneuvering on non-newtonian fluidic terrain: a survey of animal and bio-inspired robot locomotion techniques on soft yielding grounds. *Frontiers in Robotics and AI*, 10:1113881. https://doi.org/10.3389%2Ffrobt.2023.1113881.

[Godon et al., 2022] Godon, S., Ristolainen, A., and Kruusmaa, M. (2022). An insight on mud behavior upon stepping. *IEEE Robotics and Automation Letters*, 7(4):11039–11046. https://doi.org/10.1109%2Flra.2022.3194667.

[Grieco et al., 1998] Grieco, J., Prieto, M., Armada, M., and de Santos, P. G. (1998). A six-legged climbing robot for high payloads. In *Proceedings of the 1998 IEEE International Conference on Control Applications (Cat. No.98CH36104)*. IEEE. https://doi.org/10.1109%2Fcca.1998.728488.

[Griffin et al., 2019] Griffin, R. J., Wiedebach, G., McCrory, S., Bertrand, S., Lee, I., and Pratt, J. (2019). Footstep planning for autonomous walking over rough terrain. In *2019*

IEEE-RAS 19th international conference on humanoid robots (humanoids), pages 9–16. IEEE. https://doi.org/10.1109%2Fhumanoids43949.2019.9035046.

[Grillner and Wallen, 1985] Grillner, S. and Wallen, P. (1985). Central pattern generators for locomotion, with special reference to vertebrates. *Annual review of neuroscience*, 8(1):233–261. https://doi.org/10.1146%2Fannurev.ne.08.030185.001313.

[Grimminger et al., 2020] Grimminger, F., Meduri, A., Khadiv, M., Viereck, J., Wüthrich, M., Naveau, M., Berenz, V., Heim, S., Widmaier, F., Flayols, T., et al. (2020). An open torque-controlled modular robot architecture for legged locomotion research. *IEEE Robotics and Automation Letters*, 5(2):3650–3657. https://doi.org/10.1109%2Flra.2020.2976639.

[Hao et al., 2019] Hao, X., Ma, W., Liu, C., Li, Y., Qian, Z., Ren, L., Ren, L., et al. (2019). Analysis of spiders' joint kinematics and driving modes under different ground conditions. *Applied bionics and biomechanics*, 2019. https://doi.org/10.1155%2F2019%2F4617212.

[Hartikainen et al., 1992] Hartikainen, K. K., Halme, A., Lehtinen, H., and Koskinen, K. (1992). Mecant i: A six legged walking machine for research purposes in outdoor environment. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 157–163. IEEE. https://doi.org/10.1109%2Frobot.1992.220319.

[Hayakawa and Matsuno, 2022] Hayakawa, T. and Matsuno, F. (2022). Autonomous distributed system for single-legged modular robots to traverse environments by adaptive reconfiguration. *Robotics and Autonomous Systems*, page 104152. https://doi.org/10.1016/j.robot.2022.104152.

[Hernando et al., 2021a] Hernando, M., Alonso, M., Prados, C., and Gambao, E. (2021a). Behavior-based control architecture for legged-and-climber robots. *Applied Sciences*, 11(20):9547. https://doi.org/10.3390%2Fapp11209547.

[Hernando et al., 2019] Hernando, M., Brunete, A., and Gambao, E. (2019). ROMERIN: A modular climber robot for infrastructure inspection. *IFAC-PapersOnLine*, 52(15):424–429. https://doi.org/10.1016%2Fj.ifacol.2019.11.712.

[Hernando et al., 2022] Hernando, M., Gambao, E., Prados, C., Brito, D., and Brunete, A. (2022). ROMERIN: A new concept of a modular autonomous climbing robot. *International Journal of Advanced Robotic Systems*, 19(5):17298806221123416. https://doi.org/10.1177/17298806221123416.

[Hernando et al., 2021b] Hernando, M., Gómez, V., Brunete, A., and Gambao, E. (2021b). CFD modelling and optimization procedure of an adhesive system for a modular climbing robot. *Sensors*, 21(4):1117. https://doi.org/10.3390%2Fs21041117.

[Herzog et al., 2016] Herzog, A., Rotella, N., Mason, S., Grimminger, F., Schaal, S., and Righetti, L. (2016). Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid. *Autonomous Robots*, 40(3):473–491. https://doi.org/10.1007%2Fs10514-015-9476-6.

[Hirose, 1981] Hirose, S. (1981). The basic motion regulation system for quadruped walking vehicle. In *Proc. ASME Design Engineering Technical Conf.*

[Hirose et al., 1986] Hirose, S., Kikuchi, H., and Umetani, Y. (1986). The standard circular gait of a quadruped walking vehicle. *Advanced Robotics*, 1(2):143–164. https://doi.org/10.1163%2F156855386x00058.

[Hollerbach, 1980] Hollerbach, J. M. (1980). A recursive lagrangian formulation of maniputator dynamics and a comparative study of dynamics formulation complexity. *IEEE Transactions on Systems, Man, and Cybernetics*, 10(11):730–736. https://doi.org/10.1109%2Ftsmc.1980.4308393.

[Honda, 2022] Honda (2022). P2 robot. https://hondanews.com/en-US/photos/p2-robot. Accessed: 2023-04-22.

[Hong et al., 2016] Hong, S., Choi, D., Kang, S., Lee, H., and Lee, W. (2016). Design of manually reconfigurable modular manipulator with three revolute joints and links. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. https://doi.org/10.1109%2Ficra.2016.7487728.

[Huang et al., 2017] Huang, H., Li, D., Xue, Z., Chen, X., Liu, S., Leng, J., and Wei, Y. (2017). Design and performance analysis of a tracked wall-climbing robot for ship inspection in shipbuilding. *Ocean Engineering*, 131:224–230. https://www.sciencedirect.com/science/article/pii/S0029801817300033.

[Humza et al., 2009a] Humza, R., Scholz, O., Mokhtar, M., Timmis, J., and Tyrrell, A. (2009a). Towards energy homeostasis in an autonomous self-reconfigurable modular robotic organism. In *2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns*, pages 21–26. https://doi.org/10.1109/ComputationWorld.2009.83.

[Humza et al., 2009b] Humza, R., Scholz, O., Mokhtar, M., Timmis, J., and Tyrrell, A. (2009b). Towards energy homeostasis in an autonomous self-reconfigurable modular robotic organism. In *2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns*. IEEE. https://doi.org/10.1109%2Fcomputationworld.2009.83.

[Hutter, 2013] Hutter, M. (2013). *StarlETH & Co.: Design and control of legged robots with compliant actuation*. PhD thesis, ETH Zurich.

[Hutter et al., 2012] Hutter, M., Gehring, C., Bloesch, M., Hoepflinger, M. A., Remy, C. D., and Siegwart, R. (2012). Starleth: A compliant quadrupedal robot for fast, efficient, and versatile locomotion. In *Adaptive Mobile Robotics*, pages 483–490. World Scientific. https://doi.org/10.1142%2F9789814415958_0062.

[Hutter et al., 2016] Hutter, M., Gehring, C., Jud, D., Lauber, A., Bellicoso, C. D., Tsounis, V., Hwangbo, J., Bodie, K., Fankhauser, P., Bloesch, M., Diethelm, R., Bachmann, S., Melzer, A., and Hoepflinger, M. (2016). ANYmal - a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. https://doi.org/10.1109%2Firos.2016.7758092.

[Hutter et al., 2017] Hutter, M., Gehring, C., Lauber, A., Gunther, F., Bellicoso, C. D., Tsounis, V., Fankhauser, P., Diethelm, R., Bachmann, S., Bloesch, M., Kolvenbach, H., Bjelonic, M., Isler, L., and Meyer, K. (2017). ANYmal - toward legged robots for harsh environments. *Advanced Robotics*, 31(17):918–931. https://doi.org/10.1080%2F01691864.2017.1378591.

[Iida, 2007] Iida, F. (2007). Autonomous robots: From biological inspiration to implementation and control. *Artificial Life*, 13(4):419–421. https://doi.org/10.1162%2Fartl.2007.13.4.419.

[Industries and AIST, 2002] Industries, K. and AIST (2002). Hrp-2. https://robots.ieee.org/robots/hrp2/. Accessed: 2023-04-22.

[Irawan et al., 2015] Irawan, A., Razali, A. R., Ishak, W. W., Arshad, M. R., and Yin, T. Y. (2015). Development of hexaquad robot: Modeling and framework. *ARPN Journal of Engineering and Applied Sciences*, 10:17506–17513.

[Irawan and Yin, 2014] Irawan, A. and Yin, T. Y. (2014). Optimizing hexapod robot reconfiguration using hexa-quad transformation. *IAES International Journal of Robotics and Automation*, 3(2):139. https://doi.org/10.11591%2Fijra.v3i2.5936.

[Iwasaki and Zheng, 2006] Iwasaki, T. and Zheng, M. (2006). Sensory feedback mechanism underlying entrainment of central pattern generator to mechanical resonance. *Biological cybernetics*, 94(4):245–261. https://doi.org/10.1007%2Fs00422-005-0047-3.

[Jagannathan, 1989] Jagannathan, V. (1989). *Blackboard architectures and applications*. Elsevier.

[Jakimovski et al., 2009] Jakimovski, B., Meyer, B., and Maehle, E. (2009). Self-reconfiguring hexapod robot oscar using organically inspired approaches and innovative robot leg amputation mechanism. In *International Conference on Automation, Robotics and Control Systems, ARCS-09, Orlando, USA*.

[Jamroga and Ågotnes, 2006] Jamroga, W. and Ågotnes, T. (2006). What agents can achieve under incomplete information. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 232–234. https://doi.org/10.1145%2F1160633.1160672.

[Jet Propulsion Laboratory, 2022] Jet Propulsion Laboratory, N. (2022). Pdm: Planetary dexterous manipulators. https://www-robotics.jpl.nasa.gov/what-we-do/research-tasks/pdm-planetary-dexterous-manipulators/. Accessed: 2021-01-22.

[Jian-yong et al., 2011] Jian-yong, L., Xiang-yan, L., Yao-juan, Z., and Jing, Z. (2011). The development of rotation investigation wall-climbing robot. In *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*, pages 632–635. https://doi.org/10.1109/MEC.2011.6025545.

[Jiménez and Segura, 2008] Jiménez, J. J. and Segura, R. J. (2008). Collision detection between complex polyhedra. *Computers & Graphics*, 32:402–411. https://doi.org/10.1016%2Fj.cag.2008.05.002.

[Jorgensen et al., 2004] Jorgensen, M., Ostergaard, E., and Lund, H. (2004). Modular atron: modules for a self-reconfigurable robot. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 2, pages 2068–2073 vol.2. https://doi.org/10.1109/IROS.2004.1389702.

[Jourdan, 1995] Jourdan, M. (1995). Integrating formal verification methods of quantitative real-time properties into a development environment for robot controllers. *RAPPORT DE RECHERCHE-INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE*.

[Kajita and Espiau, 2008] Kajita, S. and Espiau, B. (2008). Legged robot. In *Springer handbook of robotics*, pages 361–389. Springer. https://doi.org/10.1007%2F978-3-540-30301-5_17.

[Kajita et al., 2001] Kajita, S., Matsumoto, O., and Saigo, M. (2001). Real-time 3d walking pattern generation for a biped robot with telescopic legs. In *Proceedings 2001 ICRA. IEEE international conference on robotics and automation (Cat. no. 01ch37164)*, volume 3, pages 2299–2306. IEEE. https://doi.org/10.1109%2Frobot.2001.932965.

[Kalouche et al., 2014] Kalouche, S., Wiltsie, N., Su, H., and Parness, A. (2014). Inchworm style gecko adhesive climbing robot. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2319–2324. https://doi.org/10.1109/IROS.2014.6942876.

[Kane and Levinson, 1983] Kane, T. R. and Levinson, D. A. (1983). The use of kane's dynamical equations in robotics. *The International Journal of Robotics Research*, 2(3):3–21.

[Kang et al., 2003] Kang, T., Kim, H., Son, T., and Choi, H. (2003). Design of quadruped walking and climbing robot. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 1, pages 619–624. IEEE. https://doi.org/10.1109%2Firos.2003.1250698.

[Kasem et al., 2015] Kasem, H., Tsipenyuk, A., and Varenberg, M. (2015). Biomimetic wall-shaped hierarchical microstructure for gecko-like attachment. *Soft Matter*, 11(15):2909–2915. https://doi.org/10.1039/c4sm01916d.

[Katayama and Ohtsuka, 2022] Katayama, S. and Ohtsuka, T. (2022). Whole-body model predictive control with rigid contacts via online switching time optimization. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8858–8865. IEEE. https://doi.org/10.1109%2Firos47612.2022.9981790.

[Katz et al., 2008] Katz, D., Kenney, J., and Brock, O. (2008). How can robots succeed in unstructured environments. In *In Workshop on Robot Manipulation: Intelligence in Human Environments at Robotics: Science and Systems*. Citeseer.

[Kee et al., 2014] Kee, V., Rojas, N., Elara, M. R., and Sosa, R. (2014). Hinged-tetro: A self-reconfigurable module for nested reconfiguration. In *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. IEEE. https://doi.org/10.1109%2Faim.2014.6878302.

[Kelly et al., 2005] Kelly, R., Davila, V. S., and Perez, J. A. L. (2005). *Control of robot manipulators in joint space*. Springer Science & Business Media.

[Kennedy et al., 2006] Kennedy, B., Okon, A., Aghazarian, H., Badescu, M., Bao, X., Bar-Cohen, Y., Chang, Z., Dabiri, B. E., Garrett, M., Magnone, L., and Sherrit, S. (2006). Lemur IIb: a robotic system for steep terrain access. *Industrial Robot: An International Journal*, 33(4):265–269. https://doi.org/10.1108%2F01439910610667872.

[Kerscher et al., 2008] Kerscher, T., Roennau, A., Ziegenmeyer, M., Gassmann, B., Zoellner, J., and Dillmann, R. (2008). Behaviour-based control of the six-legged walking machine lauron ivc. In *Advances In Mobile Robotics*, pages 736–743. World Scientific. https://doi.org/10.1142%2F9789812835772_0089.

[Kim et al., 2019] Kim, D., Di Carlo, J., Katz, B., Bledt, G., and Kim, S. (2019). Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control. *arXiv preprint arXiv:1909.06586*.

[Kim et al., 2005] Kim, H., Kang, T., Loc, V. G., and Choi, H. R. (2005). Gait planning of quadruped walking and climbing robot for locomotion in 3d environment. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2733–2738. IEEE. https://doi.org/10.1109%2Frobot.2005.1570527.

[Kim et al., 2008a] Kim, H., Kim, D., Yang, H., Lee, K., Seo, K., Chang, D., and Kim, J. (2008a). Development of a wall-climbing robot using a tracked wheel mechanism. *Journal of Mechanical Science and Technology*, 22(8):1490–1498. https://doi.org/10.1007%2Fs12206-008-0413-x.

[Kim et al., 2017] Kim, J., Alspach, A., and Yamane, K. (2017). Snapbot: A reconfigurable legged robot. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5861–5867. IEEE. https://doi.org/10.1109%2Firos.2017.8206477.

[Kim et al., 2008b] Kim, S., Spenko, M., Trujillo, S., Heyneman, B., Santos, D., and Cutkosky, M. R. (2008b). Smooth vertical surface climbing with directional adhesion. *IEEE Transactions on robotics*, 24(1):65–74.

[Kitano et al., 2016] Kitano, S., Hirose, S., Horigome, A., and Endo, G. (2016). Titan-xiii: sprawling-type quadruped robot with ability of fast and energy-efficient walking. *Robomech Journal*, 3:1–16. https://doi.org/10.1186%2Fs40648-016-0047-1.

[Klein and Kittivatcharapong, 1990] Klein, C. A. and Kittivatcharapong, S. (1990). Optimal force distribution for the legs of a walking machine with friction cone constraints. *IEEE Transactions on Robotics and Automation*, 6(1):73–85. https://doi.org/10.1109%2F70.88119.

[Klosowski et al., 1998] Klosowski, J. T., Held, M., Mitchell, J. S., Sowizral, H., and Zikan, K. (1998). Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE transactions on Visualization and Computer Graphics*, 4(1):21–36. https://doi.org/10.1109%2F2945.675649.

[Koo et al., 2013] Koo, I. M., Trong, T. D., Lee, Y. H., Moon, H., Koo, J., Park, S. K., and Choi, H. R. (2013). Development of wall climbing robot system by using impeller type adhesion mechanism. *Journal of Intelligent & Robotic Systems*, 72(1):57–72. https://doi.org/10.1007/s10846-013-9820-z.

[Kwon and Lee, 1998] Kwon, W. and Lee, B. H. (1998). A new optimal force distribution scheme of multiple cooperating robots using dual method. *Journal of Intelligent and Robotic Systems*, 21:301–326.

[LAAS, 2022] LAAS (2022). Jido. http://www.laas.fr/robots/jido/data/en/jido.php. Accessed: 2023-04-22.

[Larionova et al., 2007] Larionova, S., Jakimovski, B., Auf, A. E. S., Litza, M., Moesch, F., Maehle, E., and Brockmann, W. (2007). Toward a fault tolerant mobile robot: mutual information for monitoring of the robot health status. In *International Workshop on*

Technical Challenges for Dependable Robots in Human Environments, IARP, EURON, IEEE/RAS, pages 1–5.

[Lattanzi and Miller, 2017] Lattanzi, D. and Miller, G. (2017). Review of robotic infrastructure inspection systems. *Journal of Infrastructure Systems*, 23(3):04017004. https://doi.org/10.1061%2F%28asce%29is.1943-555x.0000353.

[Laurenzi et al., 2018] Laurenzi, A., Hoffman, E. M., and Tsagarakis, N. G. (2018). Quadrupedal walking motion and footstep placement through linear model predictive control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2267–2273. IEEE. https://doi.org/10.1109%2Firos.2018.8593692.

[Lee et al., 1983] Lee, C., Lee, B., and Nigam, R. (1983). Development of the generalized d'alembert equations of motion for mechanical manipulators. In *The 22nd IEEE Conference on Decision and Control*, pages 1205–1210. IEEE. https://doi.org/10.1177/027836498300200301.

[Lee et al., 2012] Lee, G., Wu, G., Kim, J., and Seo, T. (2012). High-payload climbing and transitioning by compliant locomotion with magnetic adhesion. *Robotics and Autonomous Systems*, 60(10):1308–1316. https://www.sciencedirect.com/science/article/pii/S0921889012000930.

[Li et al., 2022] Li, J., Gao, H., Wan, Y., Humphreys, J., Peers, C., Yu, H., and Zhou, C. (2022). Whole-body control for a torque-controlled legged mobile manipulator. In *Actuators*, volume 11, page 304. MDPI. https://doi.org/10.3390%2Fact11110304.

[Li et al., 2018] Li, X., Wang, W., Wu, S., Zhu, P., Zhao, F., and Wang, L. (2018). The gait design and trajectory planning of a gecko-inspired climbing robot. *Applied bionics and biomechanics*, 2018. https://doi.org/10.1155%2F2018%2F2648502.

[Li et al., 2021] Li, X.-P., Wang, X., and Feng, B. (2021). Modeling and control of a novel facade cleaning robot with four-ducted fan drive. *International Journal of Advanced Robotic Systems*, 18(3):1729881420985721. https://doi.org/10.1177/1729881420985721.

[Liang et al., 2012] Liang, C., Ceccarelli, M., and Takeda, Y. (2012). Operation analysis of a chebyshev-pantograph leg mechanism for a single dof biped robot. *Frontiers of Mechanical Engineering*, 7:357–370. https://doi.org/10.1007%2Fs11465-012-0340-5.

[Liedke et al., 2013] Liedke, J., Matthias, R., Winkler, L., and Worn, H. (2013). The collective self-reconfigurable modular organism (CoSMO). In *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. IEEE. https://doi.org/10.1109%2Faim.2013.6584059.

[Liu et al., 2013] Liu, A., Newsom, J., Schunn, C., and Shoop, R. (2013). Students learn programming faster through robotic simulation. *Tech Directions*, 72(8):16.

[Liu et al., 2015] Liu, Y., Sun, S., Wu, X., and Mei, T. (2015). A wheeled wall-climbing robot with bio-inspired spine mechanisms. *Journal of Bionic Engineering*, 12(1):17–28. https://doi.org/10.1016%2Fs1672-6529%2814%2960096-2.

[Longo and Muscato, 2006] Longo, D. and Muscato, G. (2006). The alicia/sup 3/climbing robot: A three-module robot for automatic wall inspection. *IEEE Robotics & Automation Magazine*, 13(1):42–50. https://doi.org/10.1109/mra.2006.1598052.

[Luh et al., 1980] Luh, J. Y. S., Walker, M. W., and Paul, R. P. C. (1980). On-line computational scheme for mechanical manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 102(2):69–76. https://doi.org/10.1115%2F1.3149599.

[Lyder et al., 2008] Lyder, A., Garcia, R. F. M., and Stoy, K. (2008). Mechanical design of odin, an extendable heterogeneous deformable modular robot. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 883–888. Ieee. https://doi.org/10.1109%2Firos.2008.4650888.

[Maas et al., 2012] Maas, R., Maehle, E., and Großpietsch, K.-E. (2012). Applying the organic robot control architecture orca to cyber-physical systems. In *2012 38th Euromicro Conference on Software Engineering and Advanced Applications*, pages 250–257. IEEE. https://doi.org/10.1109%2Fseaa.2012.74.

[Machado and Silva, 2006] Machado, J. T. and Silva, M. F. (2006). An overview of legged robots. In *International symposium on mathematical methods in engineering*, pages 1–40. MME Press Ankara, Turkey.

[Maehle et al., 2011] Maehle, E., Brockmann, W., Grosspietsch, K.-E., Auf, A. E. S., Jakimovski, B., Krannich, S., Litza, M., Maas, R., and Al-Homsy, A. (2011). Application of the organic robot control architecture orca to the six-legged walking robot oscar. In *Organic Computing—A Paradigm Shift for Complex Systems*, pages 517–530. Springer. https://doi.org/10.1007%2F978-3-0348-0130-0_34.

[Marder and Bucher, 2001] Marder, E. and Bucher, D. (2001). Central pattern generators and the control of rhythmic movements. *Current biology*, 11(23):R986–R996. https://doi.org/10.1016%2Fs0960-9822%2801%2900581-4.

[McGeer et al., 1990] McGeer, T. et al. (1990). Passive dynamic walking. *Int. J. Robotics Res.*, 9(2):62–82.

[Menendez et al., 2018] Menendez, E., Victores, J. G., Montero, R., Martínez, S., and Balaguer, C. (2018). Tunnel structural inspection and assessment using an autonomous robotic system. *Automation in Construction*, 87(November 2017):117–126. https://doi.org/10.1016/j.autcon.2017.12.001.

[Mir-Nasiri et al., 2018] Mir-Nasiri, N., J, H. S., and Ali, M. H. (2018). Portable autonomous window cleaning robot. *Procedia Computer Science*, 133:197–204. https://www.sciencedirect.com/science/article/pii/S1877050918309682.

[Miripour, 2010] Miripour, B. (2010). *Climbing and Walking Robots*. IntechOpen, Rijeka. https://doi.org/10.5772/200.

[Montes et al., 2017] Montes, H., Mena, L., Fernández, R., and Armada, M. (2017). Energy-efficiency hexapod walking robot for humanitarian demining. *Industrial Robot: An International Journal*, 44(4):457–466. https://doi.org/10.1108%2Fir-11-2016-0281.

[Murata et al., 2002] Murata, S., Yoshida, E., Kamimura, A., Kurokawa, H., Tomita, K., and Kokaji, S. (2002). M-tran: Self-reconfigurable modular robotic system. *IEEE/ASME transactions on mechatronics*, 7(4):431–441. https://doi.org/10.1109%2Ftmech.2002.806220.

[Murphy and Arkin, 1992] Murphy, R. and Arkin, R. (1992). Sfx: An architecture for action-oriented sensor fusion. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1079–1086. https://doi.org/10.1109%2Firos.1992.594524.

[Murphy and Hershberger, 1996] Murphy, R. R. and Hershberger, D. (1996). Classifying and recovering from sensing failures in autonomous mobile robots. In *Proceedings of the National Conference on Artificial Intelligence*, pages 922–929. Citeseer.

[Murphy and Hershberger, 1999] Murphy, R. R. and Hershberger, D. (1999). Handling sensing failures in autonomous mobile robots. *The International Journal of Robotics Research*, 18(4):382–400. https://doi.org/10.1177%2F02783649922066286.

[Murray IV et al., 2005] Murray IV, T. J., Pham, B. N., and Pirjanian, P. (2005). Hardware abstraction layer for a robot. US Patent 6,889,118.

[Muscettola et al., 2002] Muscettola, N., Dorais, G. A., Fry, C., Levinson, R., Plaunt, C., and Clancy, D. (2002). Idea: Planning at the core of autonomous reactive agents.

[Musliner et al., 1993] Musliner, D. J., Durfee, E. H., and Shin, K. G. (1993). Circa: A cooperative intelligent real-time control architecture. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(6):1561–1574. https://doi.org/10.1109%2F21.257754.

[Nabulsi et al., 2009] Nabulsi, S., Sarria, J. F., Montes, H., and Armada, M. A. (2009). High-resolution indirect feet–ground interaction measurement for hydraulic-legged robots. *IEEE Transactions on Instrumentation and Measurement*, 58(10):3396–3404. https://doi.org/10.1109%2Ftim.2009.2017650.

[Nahon and Angeles, 1991a] Nahon, M. and Angeles, J. (1991a). Real-time force optimization in parallel kinematic chains under inequality constraints. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 2198–2199. IEEE Computer Society. https://doi.org/10.1109%2Frobot.1991.131956.

[Nahon and Angeles, 1991b] Nahon, M. A. and Angeles, J. (1991b). Optimization of dynamic forces in mechanical hands. *Journal of Mechanical Design*, 113(2):167–173. https://doi.org/10.1115%2F1.2912765.

[Nansai and Mohan, 2016] Nansai, S. and Mohan, R. E. (2016). A survey of wall climbing robots: recent advances and challenges. *Robotics*, 5(3):14. https://doi.org/10.3390%2Frobotics5030014.

[NASA, 1998] NASA (1998). Deep space 1. https://www.jpl.nasa.gov/missions/deep-space-1-ds1. Accessed: 2023-04-23.

[NASA, 2003] NASA, J. P. L. (2003). Jpl robotics: The rocky 8 rover. https://www-robotics.jpl.nasa.gov/how-we-do-it/systems/the-rocky-8-rover/. Accessed: 2023-09-18.

[Nishi, 1991] Nishi, A. (1991). A wall climbing robot using propulsive force of propeller. In *Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments*, pages 320–325 vol.1. https://doi.org/10.1109/ICAR.1991.240633.

[Openrobots, 2022] Openrobots (2022). irobot atrv platform. https://www.openrobots.org/morse/doc/1.4/user/robots/atrv.html. Accessed: 2023-04-22.

[Orin and Oh, 1981] Orin, D. E. and Oh, S. Y. (1981). Control of force distribution in robotic mechanisms containing closed kinematic chains. *Journal of Dynamic Systems, Measurement, and Control*, 103(2):134–141. https://doi.org/10.1115%2F1.3139653.

[Pacheco et al., 2015] Pacheco, M., Fogh, R., Lund, H. H., and Christensen, D. J. (2015). Fable II: Design of a modular robot for creative learning. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. https://doi.org/10.1109%2Ficra.2015.7140060.

[Palmer and Grimsdale, 1995] Palmer, I. J. and Grimsdale, R. L. (1995). Collision detection for animation using sphere-trees. In *Computer Graphics Forum*, volume 14, pages 105–116. Wiley Online Library. https://doi.org/10.1111%2F1467-8659.1420105.

[Pan et al., 2012] Pan, J., Chitta, S., and Manocha, D. (2012). Fcl: A general purpose library for collision and proximity queries. In *2012 IEEE International Conference on Robotics and Automation*, pages 3859–3866. IEEE. https://doi.org/10.1109%2Ficra.2012.6225337.

[Papadimitriou et al., 2019] Papadimitriou, A., Andrikopoulos, G., and Nikolakopoulos, G. (2019). Development and control of a differential wall climbing robot based on vortex adhesion. In *2019 18th European Control Conference (ECC)*, pages 1610–1615. IEEE. https://doi.org/10.23919/ecc.2019.8796154.

[Park et al., 2006] Park, I.-W., Kim, J.-Y., and Oh, J.-H. (2006). Online biped walking pattern generation for humanoid robot khr-3 (kaist humanoid robot-3: Hubo). In *2006 6th IEEE-RAS International Conference on Humanoid Robots*, pages 398–403. IEEE. https://doi.org/10.1109%2Fichr.2006.321303.

[Parness et al., 2017] Parness, A., Abcouwer, N., Fuller, C., Wiltsie, N., Nash, J., and Kennedy, B. (2017). Lemur 3: A limbed climbing robot for extreme terrain mobility in space. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 5467–5473. IEEE. https://doi.org/10.1109%2Ficra.2017.7989643.

[Paul, 1981] Paul, R. P. (1981). Robot manipulators, mathematics, programming and control, cambridge, ma: Mass. *Inst Tech*.

[Pazouki et al., 2012] Pazouki, A., Mazhar, H., and Negrut, D. (2012). Parallel collision detection of ellipsoids with applications in large scale multibody dynamics. *Mathematics and Computers in Simulation*, 82(5):879–894. https://doi.org/10.1016%2Fj.matcom.2011.11.005.

[Peidró et al., 2019] Peidró, A., Tavakoli, M., Marín, J. M., and Reinoso, Ó. (2019). Design of compact switchable magnetic grippers for the HyReCRo structure-climbing robot. *Mechatronics*, 59:199–212. https://doi.org/10.1016%2Fj.mechatronics.2019.04.007.

[Pell et al., 1996] Pell, B., Bernard, D. E., Chien, S., Gat, E., Muscettola, N., Nayak, P. P., Wagner, M. D., and Williams, B. C. (1996). Remote agent prototype for spacecraft autonomy. In *Space Sciencecraft Control and Tracking in the New Millennium*, volume 2810, pages 74–90. SPIE. https://doi.org/10.1117%2F12.255150.

[Peters et al., 2010] Peters, G., Pagano, D., Liu, D., and Waldron, K. (2010). A prototype climbing robot for inspection of complex ferrous structures. In *Emerging Trends in Mobile Robotics*. World Scientific. https://doi.org/10.1142%2F9789814329927_0020.

[Pfotzer et al., 2014] Pfotzer, L., Ruehl, S., Heppner, G., Roennau, A., and Dillmann, R. (2014). KAIRO 3: A modular reconfigurable robot for search and rescue field missions. In *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*. IEEE. https://doi.org/10.1109%2Frobio.2014.7090331.

[Phares et al., 2004] Phares, B. M., Washer, G. A., Rolander, D. D., Graybeal, B. A., and Moore, M. (2004). Routine highway bridge inspection condition documentation accuracy and reliability. *Journal of Bridge Engineering*, 9(4):403–413. https://doi.org/10.1061%2F%28asce%291084-0702%282004%299%3A4%28403%29.

[Picardi et al., 2023] Picardi, G., Astolfi, A., Chatzievangelou, D., Aguzzi, J., and Calisti, M. (2023). Underwater legged robotics: review and perspectives. *Bioinspiration and Biomimetics*, 18(3):031001. https://doi.org/10.1088%2F1748-3190%2Facc0bb.

[Plasberg et al., 2021] Plasberg, C., Kogel, D., Roennau, A., and Dillmann, R. (2021). Adaptive walking pattern generation with damaged legs for the six-legged robot lauron v. In *2021 20th International Conference on Advanced Robotics (ICAR)*, pages 422–427. IEEE. https://doi.org/10.1109%2Ficar53236.2021.9659374.

[Prados et al., 2022a] Prados, C., Hernando, M., Gambao, E., and Brunete, A. (2022a). MoCLORA—an architecture for legged-and-climbing modular bio-inspired robotic organism. *Biomimetics*, 8(1):11. https://doi.org/10.3390%2Fbiomimetics8010011.

[Prados et al., 2022b] Prados, C., Hernando, M., Gambao, E., and Brunete, A. (2022b). ROMERIN: Organismo robótico escalador basado en patas modulares con ventosas activas. *Revista Iberoamericana de Automática e Informática industrial*, 20(2):175–186. https://doi.org/10.4995%2Friai.2022.18749.

[Prados et al., 2023] Prados, C., Hernando, M., Gambao, E., and Brunete, A. (2023). Torque-based control of a bio-inspired modular climbing robot. *Machines*, 11(7):757. https://doi.org/10.3390%2Fmachines11070757.

[Prados Sesmero et al., 2021] Prados Sesmero, C., Buonocore, L. R., and Di Castro, M. (2021). Omnidirectional robotic platform for surveillance of particle accelerator environments with limited space areas. *Applied Sciences*, 11(14):6631. https://doi.org/10.3390%2Fapp11146631.

[Prahlad et al., 2008] Prahlad, H., Pelrine, R., Stanford, S., Marlow, J., and Kornbluh, R. (2008). Electroadhesive robots—wall climbing robots enabled by a novel, robust, and electrically controllable adhesion technology. In *2008 IEEE International Conference on Robotics and Automation*, pages 3028–3033. https://doi.org/10.1109/ROBOT.2008.4543670.

[Pratt et al., 2001] Pratt, J., Chew, C.-M., Torres, A., Dilworth, P., and Pratt, G. (2001). Virtual model control: An intuitive approach for bipedal locomotion. *The International Journal of Robotics Research*, 20(2):129–143. https://doi.org/10.1177%2F02783640122067309.

[Provancher et al., 2010] Provancher, W. R., Jensen-Segal, S. I., and Fehlberg, M. A. (2010). Rocr: An energy-efficient dynamic wall-climbing robot. *IEEE/ASME Transactions on Mechatronics*, 16(5):897–906. https://doi.org/10.1109%2Ftmech.2010.2053379.

[Qadir, 2015] Qadir, R. H. (2015). *Self-Sufficiency of an Autonomous Reconfigurable Modular Robotic Organism*. Springer, Cham. https://link.springer.com/book/10.1007/978-3-319-10289-4#about.

[Qiao et al., 2012] Qiao, G., Song, G., Zhang, J., Sun, H., Wang, W., and Song, A. (2012). Design of transmote: A modular self-reconfigurable robot with versatile transformation capabilities. In *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE. https://doi.org/10.1109%2Frobio.2012.6491153.

[Qiaoling et al., 2019] Qiaoling, D., Yan, L., and Sinan, L. (2019). Design of a micro pole-climbing robot. *International Journal of Advanced Robotic Systems*, 16(3):1729881419852813. https://doi.org/10.1177%2F1729881419852813.

[Raibert and Tello, 1986] Raibert, M. H. and Tello, E. R. (1986). *Legged Robots That Balance*, volume 1. Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109%2Fmex.1986.4307016.

[Regenstein et al., 2007] Regenstein, K., Kerscher, T., Birkenhofer, C., Asfour, T., Zollner, M., and Dillmann, R. (2007). Universal controller module (ucom)-component of a modular concept in robotic systems. In *2007 IEEE International Symposium on Industrial Electronics*, pages 2089–2094. IEEE. https://doi.org/10.1109%2Fisie.2007.4374930.

[Roennau et al., 2014] Roennau, A., Heppner, G., Nowicki, M., and Dillmann, R. (2014). LAURON v: A versatile six-legged walking robot with advanced maneuverability. In *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. IEEE. https://doi.org/10.1109%2Faim.2014.6878051.

[Roennau et al., 2013] Roennau, A., Heppner, G., Pfotzer, L., and Dillmann, R. (2013). Lauron v: Optimized leg configuration for the design of a bio-inspired walking robot. In *Nature-Inspired Mobile Robotics*, pages 563–570. World Scientific. https://doi.org/10.1142%2F9789814525534_0071.

[Romanishin et al., 2022] Romanishin, J., Bern, J. M., and Rus, D. (2022). Self-reconfiguring robotic gantries powered by modular magnetic lead screws. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4225–4231. IEEE. https://doi.org/10.1109%2Ficra46639.2022.9811863.

[Rostro-Gonzalez et al., 2015] Rostro-Gonzalez, H., Cerna-Garcia, P. A., Trejo-Caballero, G., Garcia-Capulin, C. H., Ibarra-Manzano, M. A., Avina-Cervantes, J. G., and Torres-Huitzil, C. (2015). A cpg system based on spiking neurons for hexapod robot locomotion. *Neurocomputing*, 170:47–54. https://doi.org/10.1016%2Fj.neucom.2015.03.090.

[Roth, 1990] Roth, Z. (1990). *Machines that walk: The adaptive suspension vehicle*, volume 25. Elsevier BV. https://doi.org/10.1016%2F0094-114x%2890%2990072-r.

[Rus and Vona, 2000] Rus, D. and Vona, M. (2000). A physical implementation of the self-reconfiguring crystalline robot. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 1726–1733. IEEE. https://doi.org/10.1109%2Frobot.2000.844845.

[Ryland and Cheng, 2010] Ryland, G. G. and Cheng, H. H. (2010). Design of iMobot, an intelligent reconfigurable mobile robot with novel locomotion. In *2010 IEEE International Conference on Robotics and Automation*. IEEE. https://doi.org/10.1109%2Frobot.2010.5509359.

[Saha, 1999] Saha, S. K. (1999). Dynamics of serial multibody systems using the decoupled natural orthogonal complement matrices. *Journal of Applied Mechanics*, 66(4):986–996. https://doi.org/10.1115%2F1.2791809.

[Sakagami et al., 2002] Sakagami, Y., Watanabe, R., Aoyama, C., Matsunaga, S., Higaki, N., and Fujimura, K. (2002). The intelligent asimo: System overview and integration. In *IEEE/RSJ international conference on intelligent robots and systems*, volume 3, pages 2478–2483. IEEE. https://doi.org/10.1109%2Firds.2002.1041641.

[Schmidt et al., 2013] Schmidt, D., Wettach, J., and Berns, K. (2013). 3d realtime simulation framework for a wall-climbing robot using negative-pressure adhesion. In *ICINCO (2)*, pages 184–191. https://doi.org/10.5220%2F0004444401840191.

[Schwarz et al., 2016] Schwarz, M., Rodehutskors, T., Schreiber, M., and Behnke, S. (2016). Hybrid driving-stepping locomotion with the wheeled-legged robot momaro. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5589–5595. IEEE. https://doi.org/10.1109%2Ficra.2016.7487776.

[Seo and Sitti, 2013] Seo, T. and Sitti, M. (2013). Tank-like module-based climbing robot using passive compliant joints. *IEEE/ASME Transactions on Mechatronics*, 18(1):397–408. https://doi.org/10.1109/TMECH.2011.2182617.

[Seriani et al., 2019] Seriani, S., Scalera, L., Caruso, M., Gasparetto, A., and Gallina, P. (2019). Upside-down robots: Modeling and experimental validation of magnetic-adhesion mobile systems. *Robotics*, 8(2):41. https://doi.org/10.3390/robotics8020041.

[Sesmero et al., 2021] Sesmero, C. P., Lorente, S. V., and Castro, M. D. (2021). Graph SLAM built over point clouds matching for robot localization in tunnels. *Sensors*, 21(16):5340. https://doi.org/10.3390%2Fs21165340.

[Shah et al., 2012] Shah, S., Saha, S., and Dutt, J. (2012). Modular framework for dynamic modeling and analyses of legged robots. *Mechanism and Machine Theory*, 49:234–255. https://doi.org/10.1016%2Fj.mechmachtheory.2011.10.006.

[Shang and Shen, 2022] Shang, Z. and Shen, Z. (2022). Flight planning for survey-grade 3d reconstruction of truss bridges. *Remote Sensing*, 14(13):3200. https://doi.org/10.3390%2Frs14133200.

[Shen et al., 2006] Shen, W.-M., Krivokon, M., Chiu, H., Everist, J., Rubenstein, M., and Venkatesh, J. (2006). Multimode locomotion via SuperBot reconfigurable robots. *Autonomous Robots*, 20(2):165–177. https://doi.org/10.1007%2Fs10514-006-6475-7.

[Shen et al., 2007] Shen, Y., Salemi, W., Rus, B., Moll, D., and Lipson, M. (2007). Modular self-reconfigurable robot systems. *IEEE Robotics and Automation Magazine*, 14(1):43–52.

[Shim et al., 2023] Shim, J., Mastalli, C., Corberes, T., Tonneau, S., Ivan, V., and Vijayakumar, S. (2023). Topology-based mpc for automatic footstep placement and contact surface selection. *arXiv preprint arXiv:2303.13726*. https://doi.org/10.1109%2Ficra48891.2023.10160333.

[Simon et al., 1993] Simon, D., Espiau, B., Castillo, E., and Kapellos, K. (1993). Computer-aided design of a generic robot controller handling reactivity and real-time control issues. *IEEE Transactions on Control Systems Technology*, 1(4):213–229. https://doi.org/10.1109%2F87.260267.

[Singh et al., 2021] Singh, M., Fuenmayor, E., Hinchy, E. P., Qiao, Y., Murray, N., and Devine, D. (2021). Digital twin: Origin to future. *Applied System Innovation*, 4(2):36. https://doi.org/10.3390%2Fasi4020036.

[Site, 2022] Site, T. O. R. W. (2022). Heathkit hero 2000 - the old robot's web site. http://www.theoldrobots.com/hero2k.html. Accessed: 2023-09-18.

[Smith et al., 2006] Smith, J. A., Sharf, I., and Trentini, M. (2006). Paw: a hybrid wheeled-leg robot. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 4043–4048. IEEE. https://doi.org/10.1109%2Frobot.2006.1642323.

[Sprowitz et al., 2010] Sprowitz, A., Pouya, S., Bonardi, S., Kieboom, J. V. D., Mockel, R., Billard, A., Dillenbourg, P., and Ijspeert, A. J. (2010). Roombots: Reconfigurable robots for adaptive furniture. *IEEE Computational Intelligence Magazine*, 5(3):20–32. https://doi.org/10.1109%2Fmci.2010.937320.

[Spröwitz et al., 2018] Spröwitz, A. T., Tuleu, A., Ajallooeian, M., Vespignani, M., Möckel, R., Eckert, P., D'Haene, M., Degrave, J., Nordmann, A., Schrauwen, B., et al. (2018). Oncilla robot: a versatile open-source quadruped research robot with compliant pantograph legs. *Frontiers in Robotics and AI*, 5:67. https://doi.org/10.3389%2Ffrobt.2018.00067.

[Spröwitz et al., 2014] Spröwitz, A., Moeckel, R., Vespignani, M., Bonardi, S., and Ijspeert, A. (2014). Roombots: A hardware perspective on 3d self-reconfiguration and locomotion with a homogeneous modular robot. *Robotics and Autonomous Systems*, 62(7):1016–1033. https://doi.org/10.1016%2Fj.robot.2013.08.011.

[Stelzer et al., 2012] Stelzer, A., Hirschmüller, H., and Görner, M. (2012). Stereo-vision-based navigation of a six-legged walking robot in unknown rough terrain. *The International Journal of Robotics Research*, 31(4):381–402. https://doi.org/10.1177%2F0278364911435161.

[Suh et al., 2002] Suh, J., Homans, S., and Yim, M. (2002). Telecubes: mechanical design of a module for self-reconfigurable robotics. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, pages 4095–4101. IEEE. https://doi.org/10.1109%2Frobot.2002.1014385.

[Sun and Sartoretti, 2022] Sun, G. and Sartoretti, G. (2022). Joint-space cpg for safe foothold planning and body pose control during locomotion and climbing. *IEEE Robotics and Automation Letters*, 7(4):9889–9896. https://doi.org/10.1109%2Flra.2022.3192657.

[Svoboda et al., 2019] Svoboda, E. et al. (2019). Your robot surgeon will see you now. *Nature*, 573(7775):S110–S111. https://doi.org/10.1038%2Fd41586-019-02874-0.

[Tache et al., 2007] Tache, F., Fischer, W., Siegwart, R., Moser, R., and Mondada, F. (2007). Compact magnetic wheeled robot with high mobility for inspecting complex shaped pipe structures. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 261–266. https://doi.org/10.1109/IROS.2007.4399116.

[Tan et al., 2006] Tan, K. C., Wang, L., Lee, T. H., and Vadakkepat, P. (2006). Evolvable hardware in evolutionary robotics. In *World Scientific Series in Robotics and Intelligent Systems*, pages 33–62. World Scientific. https://doi.org/10.1142%2F9789812773142_0002.

[Tan et al., 2021] Tan, W., Fang, X., Zhang, W., Song, R., Chen, T., Zheng, Y., and Li, Y. (2021). A hierarchical framework for quadruped locomotion based on reinforcement learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8462–8468. IEEE. https://doi.org/10.1109%2Firos51168.2021.9636757.

[Tanaka et al., 2021] Tanaka, Y., Shirai, Y., Lacey, Z., Lin, X., Liu, J., and Hong, D. (2021). An under-actuated whippletree mechanism gripper based on multi-objective design optimization with auto-tuned weights. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6139–6146. IEEE. https://doi.org/10.1109%2Firos51168.2021.9635872.

[Tanaka et al., 2022] Tanaka, Y., Shirai, Y., Lin, X., Schperberg, A., Kato, H., Swerdlow, A., Kumagai, N., and Hong, D. (2022). Scaler: A tough versatile quadruped free-climber robot. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5632–5639. IEEE. https://doi.org/10.1109%2Firos47612.2022.9981555.

[Tavakoli et al., 2013a] Tavakoli, M., Viegas, C., Marques, L., Pires, J. N., and de Almeida, A. T. (2013a). Omniclimbers: Omni-directional magnetic wheeled climbing robots for inspection of ferromagnetic structures. *Robotics and Autonomous Systems*, 61(9):997–1007. https://www.sciencedirect.com/science/article/pii/S0921889013001024.

[Tavakoli et al., 2013b] Tavakoli, M., Viegas, C., Marques, L., Pires, J. N., and de Almeida, A. T. (2013b). OmniClimbers: Omni-directional magnetic wheeled climbing robots for inspection of ferromagnetic structures. *Robotics and Autonomous Systems*, 61(9):997–1007. https://doi.org/10.1016%2Fj.robot.2013.05.005.

[Teng et al., 2022] Teng, S., Chen, D., Clark, W., and Ghaffari, M. (2022). An error-state model predictive control on connected matrix lie groups for legged robot control. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8850–8857. IEEE. https://doi.org/10.1109%2Firos47612.2022.9981282.

[Tiwari et al., 2020] Tiwari, S., Kazi, R., Nambisan, A., and Jani, P. (2020). Morphius: A self-configuring modular robot that navigates through cracks and debris for inspection. In *Proceedings of the 21st International Conference on Distributed Computing and Networking*. ACM. https://doi.org/10.1145%2F3369740.3372754.

[University, 2022] University, W. (2022). Humanoid history -wabot-. https://www.humanoid.waseda.ac.jp/booklet/kato˙2.html. Accessed: 2023-04-22.

[Unver and Sitti, 2010] Unver, O. and Sitti, M. (2010). Tankbot: A palm-size, tank-like climbing robot using soft elastomer adhesive treads. *The International Journal of Robotics Research*, 29(14):1761–1777. https://doi.org/10.1177%2F0278364910380759.

[Valero, 2021] Valero, J. (2021). Romerin: Diseño de un efector final inteligente de adhesión a superficies. http://138.100.100.141:8080/share.cgi?ssid=0357Lq9. Bachelor Thesis. Accessed: 2023-04-28.

[Vásquez and Matía, 2020] Vásquez, B. P. E. A. and Matía, F. (2020). A tour-guide robot: Moving towards interaction with humans. *Engineering Applications of Artificial Intelligence*, 88:103356. https://doi.org/10.1016%2Fj.engappai.2019.103356.

[Verimag, 2022] Verimag (2022). Dala robot. https://www-verimag.imag.fr/Dala-Robot.html. Accessed: 2023-04-22.

[Vidoni and Gasparetto, 2011] Vidoni, R. and Gasparetto, A. (2011). Efficient force distribution and leg posture for a bio-inspired spider robot. *Robotics and Autonomous Systems*, 59(2):142–150. https://doi.org/10.1016%2Fj.robot.2010.10.001.

[Vilches and Ugarte, 2017] Vilches, V. M. and Ugarte, I. Z. (2017). Envisioning the future of robotics. *Hackernoon March*, 9:2017.

[Villard et al., 1994] Villard, C., Gorce, P., and Fontaine, J.-G. (1994). Study of a distributed control architecture for a quadruped robot. *Journal of Intelligent and Robotic Systems*, 11:269–291. https://doi.org/10.1007%2Fbf01254014.

[Vladareanu et al., 2015] Vladareanu, V., Boscoianu, C., Munteanu, R. I., Yu, H., and Vladareanu, L. (2015). Dynamic control of a walking robot using the versatile intelligent portable robot platform. In *2015 20th International Conference on Control Systems and Computer Science*, pages 38–45. IEEE. https://doi.org/10.1109%2Fcscs.2015.48.

[Volpe et al., 1996] Volpe, R., Balaram, J., Ohm, T., and Ivlev, R. (1996). The rocky 7 mars rover prototype. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS'96*, volume 3, pages 1558–1564. IEEE. https://doi.org/10.1109%2Firos.1996.569020.

[Volpe et al., 2001] Volpe, R., Nesnas, I., Estlin, T., Mutz, D., Petras, R., and Das, H. (2001). The claraty architecture for robotic autonomy. In *2001 IEEE Aerospace Conference Proceedings (Cat. No. 01TH8542)*, volume 1, pages 1–121. IEEE. https://doi.org/10.1109%2Faero.2001.931701.

[Waldron and McGhee, 1986] Waldron, K. and McGhee, R. (1986). The adaptive suspension vehicle. *IEEE Control Systems Magazine*, 6(6):7–12.

[Wang et al., 2021] Wang, C. C., Wang, M., Sun, J., and Mojtahedi, M. (2021). A safety warning algorithm based on axis aligned bounding box method to prevent onsite accidents of mobile construction machineries. *Sensors*, 21(21):7075. https://doi.org/10.3390%2Fs21217075.

[Wang et al., 2020] Wang, M., Su, Y., Liu, H., and Xu, Y. (2020). WalkingBot: Modular interactive legged robot with automated structure sensing and motion planning. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE. https://doi.org/10.1109%2Fro-man47096.2020.9223474.

[Wang et al., 2015] Wang, W., Wu, S., Zhu, P., and Liu, R. (2015). Analysis on the dynamic climbing forces of a gecko inspired climbing robot based on gpl model. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3314–3319. IEEE. https://doi.org/10.1109%2Firos.2015.7353838.

[Wang et al., 2019] Wang, W.-C., Lu, Z., and Tsui, C. L. (2019). A design for autonomous self-building blocks. *Journal of Reliable Intelligent Environments*, 5(2):115–128. https://doi.org/10.1007%2Fs40860-019-00082-1.

[Wang et al., 2011] Wang, Z., Wang, J., Ji, A., Li, H., and Dai, Z. (2011). Movement behavior of a spider on a horizontal surface. *Chinese science bulletin*, 56:2748–2757. https://doi.org/10.1007%2Fs11434-011-4584-y.

[Webster et al., 2022] Webster, C., Kong, F. H., and Fitch, R. (2022). Bio-inspired 2d vertical climbing with a novel tripedal robot. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1239–1246. IEEE. https://doi.org/10.1109%2Firos47612.2022.9981088.

[Wermelinger et al., 2016] Wermelinger, M., Fankhauser, P., Diethelm, R., Krüsi, P., Siegwart, R., and Hutter, M. (2016). Navigation planning for legged robots in challenging terrain. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1184–1189. IEEE. https://doi.org/10.1109%2Firos.2016.7759199.

[Wilcox et al., 2007] Wilcox, B. H., Litwin, T., Biesiadecki, J., Matthews, J., Heverly, M., Morrison, J., Townsend, J., Ahmad, N., Sirota, A., and Cooper, B. (2007). Athlete: A cargo handling and manipulation robot for the moon. *Journal of Field Robotics*, 24(5):421–434. https://doi.org/10.1002%2Frob.20193.

[Wisth et al., 2019] Wisth, D., Camurri, M., and Fallon, M. (2019). Robust legged robot state estimation using factor graph optimization. *IEEE Robotics and Automation Letters*, 4(4):4507–4514. https://doi.org/10.1109%2Flra.2019.2933768.

[Wolfe et al., 2012] Wolfe, K. C., Moses, M. S., Kutzer, M. D., and Chirikjian, G. S. (2012). M3express: A low-cost independently-mobile reconfigurable modular robot. In *2012 IEEE International Conference on Robotics and Automation*. IEEE. https://doi.org/10.1109%2Ficra.2012.6224971.

[Wooldridge, 1999] Wooldridge, M. (1999). Intelligent agents. *Multiagent systems*, 6.

[Xu et al., 2018a] Xu, F., Wang, B., Shen, J., Hu, J., and Jiang, G. (2018a). Design and realization of the claw gripper system of a climbing robot. *Journal of Intelligent & Robotic Systems*, 89:301–317. https://doi.org/10.1007%2Fs10846-017-0552-3.

[Xu et al., 2018b] Xu, J., Xu, L., Liu, J., Li, X., and Wu, X. (2018b). Survey on Bioinspired Adhesive Methods and Design and Implementation of A Multi-Mode Biomimetic Wall-Climbing Robot. In *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 688–693. https://doi.org/10.1109/AIM.2018.8452318.

[Yang et al., 2020] Yang, Y., Caluwaerts, K., Iscen, A., Zhang, T., Tan, J., and Sindhwani, V. (2020). Data efficient reinforcement learning for legged robots. In *Conference on Robot Learning*, pages 1–10. PMLR.

[Yi and Lee, 2016] Yi, S.-J. and Lee, D. D. (2016). Dynamic heel-strike toe-off walking controller for full-size modular humanoid robots. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 395–400. IEEE. https://doi.org/10.1109%2Fhumanoids.2016.7803306.

[Yim et al., 2000] Yim, M., Duff, D. G., and Roufas, K. D. (2000). Polybot: a modular reconfigurable robot. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 514–520. IEEE. https://doi.org/10.1109%2Frobot.2000.844106.

[Yim et al., 2002] Yim, M., Zhang, Y., and Duff, D. (2002). Modular robots. *IEEE Spectrum*, 39(2):30–34. https://doi.org/10.1109%2F6.981854.

[Yoshida and Ma, 2010] Yoshida, Y. and Ma, S. (2010). Design of a wall-climbing robot with passive suction cups. In *2010 IEEE International Conference on Robotics and Biomimetics*, pages 1513–1518. IEEE. https://doi.org/10.1109/robio.2010.5723554.

[Yoshikawa, 1985] Yoshikawa, T. (1985). Manipulability of robotic mechanisms. *The international journal of Robotics Research*, 4(2):3–9. https://doi.org/10.1177%2F027836498500400201.

[Zhang et al., 2014] Zhang, J., Gao, F., Han, X., Chen, X., and Han, X. (2014). Trot gait design and cpg method for a quadruped robot. *Journal of Bionic Engineering*, 11(1):18–25. https://doi.org/10.1016%2Fs1672-6529%2814%2960016-0.

[Zhang et al., 2023] Zhang, Y., Qiao, G., Wan, Q., Tian, L., and Liu, D. (2023). A novel double-layered central pattern generator-based motion controller for the hexapod robot. *Mathematics*, 11(3):617. https://doi.org/10.3390%2Fmath11030617.

[Zhao et al., 2011] Zhao, J., Cui, X., Zhu, Y., and Tang, S. (2011). A new self-reconfigurable modular robotic system UBot: Multi-mode locomotion and self-reconfiguration. In *2011 IEEE International Conference on Robotics and Automation*. IEEE. https://doi.org/10.1109%2Ficra.2011.5980293.

[Zhou and Li, 2018] Zhou, Q. and Li, X. (2018). Experimental investigation on climbing robot using rotation-flow adsorption unit. *Robotics and Autonomous Systems*, 105:112–120. https://www.sciencedirect.com/science/article/pii/S0921889016307710.

[Zhuang et al., 2014] Zhuang, H., Gao, H., Deng, Z., Ding, L., and Liu, Z. (2014). A review of heavy-duty legged robots. *Science China Technological Sciences*, 57:298–314. https://doi.org/10.1007%2Fs11431-013-5443-7.

[Zribi et al., 1999] Zribi, M., Loulin, H., and Piu, C. S. (1999). Position and force control of two constrained robotic manipulators. *Journal of Intelligent and Robotic Systems*, 24:1–22. `https://link.springer.com/article/10.1023/A:1008067612611`.

[Zykov et al., 2004] Zykov, V., Bongard, J., and Lipson, H. (2004). Evolving dynamic gaits on a physical robot. In *Proceedings of Genetic and Evolutionary Computation Conference, Late Breaking Paper, GECCO*, volume 4, page 2004. Citeseer.

[Zykov et al., 2007] Zykov, V., Chan, A., and Lipson, H. (2007). Molecubes: An open-source modular robotics kit. In *IROS-2007 Self-Reconfigurable Robotics Workshop*, pages 3–6. Citeseer.