



H2020-ICT-25-2016-2017

HYbrid FLying rollIng with-snakeE-aRm robot for contact inSpection

HYFLIERS

D3.3

Visual-force control techniques of a hybrid mobile-aerial system endowed with a robotic arm for autonomous inspection in stable contact with the environment

Contractual date of delivery	30 Nov 2021
Actual date of delivery	
Editor(s)	Vincenzo Lippiello (CREATE)
Author(s)	Vincenzo Lippiello (CREATE), Jonathan Cacace (CREATE)
Workpackage	WP3
Estimated person-months	
Dissemination level	PU
Type	R
Version	1.0
Total number of pages	29

Abstract:

This document reports the progress made on Task 3.2 of the HYFLIERS project. In particular, in this task, the snake robot of T3.1 has been endowed with environmental awareness capabilities like 2D and 3D mapping, while a hierarchical task composition framework has been deployed to allow the control of the hyper-redundant platform

Keywords:

Deliverable, snake robot, hyper-redundant robotic arm, UT inspection, visual inspection, UT probe, tube inspection, 2D mapping, 3D mapping, navigation, force control, visual feedback control, task composition.

Executive summary

This document summarizes the activities made in Task 3.2 for Work Package 3. In this context, the hyper redundant robotic arm developed in Task 3.1 and presented in the deliverable D3.1 and the mobile carrying module are used to implement the visual and force-based controller to enable autonomous and semi-autonomous inspection tasks.

To support the inspection task, both dense and flat environment reconstruction have been deployed in order to enable safe navigation along the inspection pipe and obstacle avoidance during the motion of the hyper redundant arm. The contributions detailed in this document are summarized and introduced in the following:

- *Generation of a 2D flat map of the environment*: the first part of the document focuses on the definition of localization techniques for the motion of the robot over the pipe in order to use such information for the generation of an accurate 3D local map of the industrial pipe based on dense reconstruction methods. In particular, in this case odometry measures generated by the encoders of the wheels are fused with a set of laser measurements collected by the range finders placed at the bottom of the omnidirectional wheels, as already defined in D2.1 for WP2.
- *Generation of a 3D map of the inspection pipe*: the sensor fusion method used to generate a 2D map allows the robot to localize and generate a 2D flat map of pipe correctly. The 3D map of the environment is composed considering the 3D point cloud data generated by a depth sensor attached to the end effector of the hyper-redundant arm presented in D3.1. This information is further used to generate a 3D grid map of the pipe to exploit this information in the planning and execution phase of the inspection task.
- *Definition of a hierarchical task formulation for redundancy control*: to fully exploit the high number of degrees of freedom of the snake robot manipulator, the projector onto the null space of the system Jacobian has been exploited. In particular, following the proposed approach, a set of tasks can be defined with different levels of priority in order to assure the convergence of the highest priority tasks and, if possible, achieve a set of secondary, low priority tasks. The following set of tasks has been defined:
 - *Pose task*: position and orientation task to assure the correct positioning of the end effector of the robot.
 - *Joint limit task*: task to assure that the arm does not collide with its structure.
 - *Obstacle avoidance*: task to assure that the robotic arm does not collide with external obstacles.
 - *On pipe stabilization*: task to assure the stabilization of the carrying module during the inspection operation over the pipe.
- *Stabilization and control on a pipe of a wheeled mobile manipulator with a snake-like arm*: a Model Predictive Control (MPC) approach has been deployed to perform the stabilization of the wheeled robot on the pipe. When a saturation on the wheel's torque occurs, the stabilization task leverages the resulting propagating force on the wheeled robot given by the snake-like arm's dynamics. The significant number of degrees of freedom given by the snake-like arm allows the use of a prioritized redundancy resolution scheme to inspect adjacent pipes while avoiding self-collisions and impacts on the environment.

The approaches discussed in this deliverable have been extensively tested CoppeliaSim and Gazebo ROS simulation environments.

Abbreviations and symbols

2D, 3D	Two-dimensional, Three-dimensional
AMCL	Adaptive Monte Carlo Localization
COM	Center Of Mass
DOF	Degree Of Freedom
FTS	Force and Torque Sensor
LIDAR	Light Detection And Ranging
MPC	Model Predictive Control
NDT	NonDestructive Test
RGB-D	Red Green Blue Depth
ROS	Robot Operating System
RTAB-Map	Real-Time Appearance-Based Mapping
SLAM	Simultaneous Localization And Mapping
ToF	Tim of Flight
UT	Ultrasound Transducer
WUAV	Wheeled Unmanned Aerial Vehicle

Table of Contents

1. Introduction	6
2. Generation of local maps	6
2.1 2D local map generation	6
2.2 Wheeled base navigation	8
2.3 Dense reconstruction of the pipe	10
3. Hierarchical Task-Priority Control framework	12
3.1 Position and Orientation task	14
3.2 Joint limit task	14
3.2 Center of Mass task	15
3.3 Obstacle avoidance task	16
4. Stabilization and Control on a Pipe of a Wheeled Mobile Manipulator with a Snake-like Arm	18
4.1 MPC Controller for platform stabilization	23
4.2 Case studies	26
5. Conclusion	29

List of Figures

Figure 1: Joint structure of the mobile wheeled robotic base. The ground rover consists of two joints and four laser sensors.	7
Figure 2: Distance sensors projected in the base frame of the robot to reconstruct the shape of the pipe.	8
Figure 3: Industrial mockup for 2D map generation and navigation.	8
Figure 4: Map of the inspecting pipe. The blue trajectory represents the reference path interpolated with a cubic spline.	9
Figure 5: Map data involved in the localization and navigation task. The walls of the pipe (obstacles) are depicted with their middle points (C_p) and reference trajectory (R_{traj}). l_{pf} and l_{pr} are the the centre position of the ToF measured distances by the front.	10
Figure 6: Intel RealSense D435 depth sensor plugged on the end effector of the hyper redundant manipulator.	11
Figure 7: Octomap generated with the depth sensor of the hyper-redundant arm considering two different pipe structures.	12
Figure 8: Variation of joint limits error with (red) and without (blue) the joint limit task enabled in the task stack.	15
Figure 9: The carrying module of Task 2.1 is able to rotate over the pipe.	15
Figure 10: Position of the Center of Mass (green sphere) of the whole robot in a particular configuration of the hyper-redundant arm.	16
Figure 11: Obstacle avoidance task.	17
Figure 12: Distances between more close links and obstacles.	18
Figure 13: The hybrid robot performs an inspection task in a cluttered scenario.	19
Figure 14: Snapshot extracted from the simulation environment, with the pipe-rack, the rover and the snake-like arm. The frames are depicted. The arm's CoM is represented in green; the rover's CoM is in red; and the whole robot's CoM is in yellow.	20
Figure 15: Potential grid-map built with the navigation function algorithm. On the top, the change of colour from blue to yellow shows different value of potential. On the bottom, the red circles indicate the collision-free region bounds and the path in terms of cells is shown in blue connecting the two dots representing initial (in red) and final configuration (in green). The trajectory is computed in the Y-Z plane fixing the end-effector X position.	22
Figure 16: 2-D sketch of the rover on a pipe, with the illustration of the symbols employed to derive the dynamic model of the rover and it constrains.	23
Figure 17: Overall control scheme.	25
Figure 18: Case study 1. stabilization test. (a)-Time history of the state vector x . (b)-Time history of the wheels control inputs u . (c-f)-Time histories to check the fulfilment of the constraints: with the dashed lines are represented the boundaries.	27
Figure 19: Case study 2: inspection test. (a)-Optimized 2-D path: the two pipes are approximated as two circumferences; with the dashed line the limits of the collision-free region are shown. (b) Linear planned trajectory. (c) Angular planned trajectory.	28
Figure 20: Case study 2: Robustness test. (a)-Time history of the state vector x affected by white noise. (b)-Time history of the control input u . (c-f)-Time histories to check the fulfilment of the constraints: with the dashed lines are represented the boundaries.	28

1. Introduction

The goal of the WP3 is to design, develop and support pipe inspection tasks exploiting a hybrid wheeled robotic manipulator able to safely operate in a partially observable environment. This deliverable takes into consideration the work done in Tasks 2.1 and 3.1 in which a mobile platform able to stabilize over an industrial pipe and a hyper-redundant manipulator have been designed and deployed respectively. Differently, this deliverable focuses on the use of these mechanical systems modules to perform a given task. The rest of the document is organized as follow.

First, we address the problem of generating local maps of the pipe. In particular, a 2D planar map to navigate the pipe with the wheeled mobile based is generated. After the generation of the map, a navigation experiment, considering a curved pipe section is performed in order to demonstrate the motion capabilities of the rover on a known generated map. Then, a 3D map is generated considering the point-cloud data from a depth sensor plugged on the snake arm. This map is based on Octomap data format.

After the elaboration of the sensor data, two control modes are described. The first control technique is based on the well-known hierarchical task composition. This approach is based on a decentralized control technique that aims to control in two different ways the wheeled base and the hyper-redundant arm. The proposed approach consists in exploiting the high number of degrees of freedom of the mechanical system to accomplish multiple tasks.

The previous approach just considers the kinematic of the system. Differently, a Model Predictive Control (MPC) strategy has been deployed to both stabilize the rover upon the duct and use the hyper redundant arm.

2. Generation of local maps

This section presents advances in the generation of local maps of the pipe to inspect. The overall goal of this task is to provide relevant information to the autonomous system of the robot to achieve inspection tasks. In particular, two different maps will be generated, a 3D dense map and a 2D flat map. In this latter case, the map is used to enable safe navigation of the wheeled rover upon the pipe, in order to navigate the eventual bending of the pipes as well. Differently, the dense reconstruction of the pipe is used to perform the inspection tasks and allow the hyper-redundant arm to avoid obstacles during the inspection operation. In the following, both the generation of the map and the navigation over a curved pipe are discussed. Then, the 3D reconstruction of the pipe is presented.

2.1 2D local map generation

In order to remotely perform NDT measurements, the inspection robot must be able to autonomously navigate over the industrial pipe to inspect. With this aim, the wheeled platforms proposed in D2.1 and developed in Task 2.1 must be able to adapt their kinematic configuration to follow the shape of the pipe during the navigation. In this context, this problem is addressed endowing the robot with SLAM capabilities. In this context, a ROS package has been deployed to generate a flat laser-based map of the pipe. To generate a flat map of the pipe, a set of ToF (Time of Flight) sensors placed in the proximity of the rover wheels are used. The configuration of such sensors is depicted in Figure 1, in which the kinematic structure of the robot is represented using the **tf** ROS package (<http://wiki.ros.org/tf>). In this figure, the front and rear frames *front_link* and *rear_link* are represented along with the four sensor frames *laser_front_sx*, *laser_front_dx*, *laser_rear_sx* and *laser_rear_dx*.

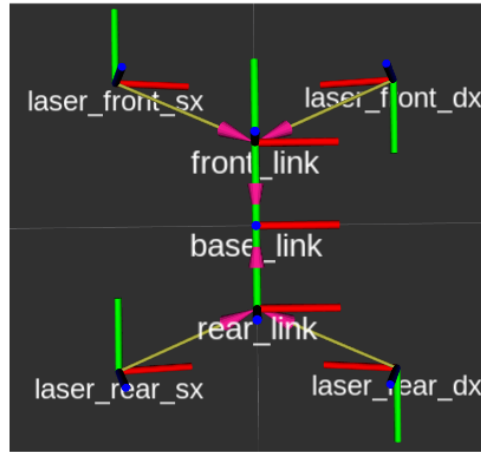


Figure 1: Joint structure of the mobile wheeled robotic base. The ground rover consists of two joints and four laser sensors.

In this context, each ToF sensor gives the current distance with the pipe as shown in Figure 2 for the front and rear parts of the rover. These distances are later projected in the base frame of the robot and remain almost constant along the straight paths of the pipe and change on pipe-bend sections. Classical SLAM algorithm can be used to create the environmental map. In this work, gmapping (<http://wiki.ros.org/gmapping>) and AMCL (<http://wiki.ros.org/amcl>) ROS packages have been integrated in the rover navigation software.

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = S \begin{bmatrix} V_f \\ \omega_s \end{bmatrix} \quad (1)$$

To properly localize the robot, odometry measurements have been also considered. In particular, the orientation of the robot is directly got by the inertial sensor, while its position is calculated with the inverse kinematics described by equation (1) using the data provided by the wheels encoders.

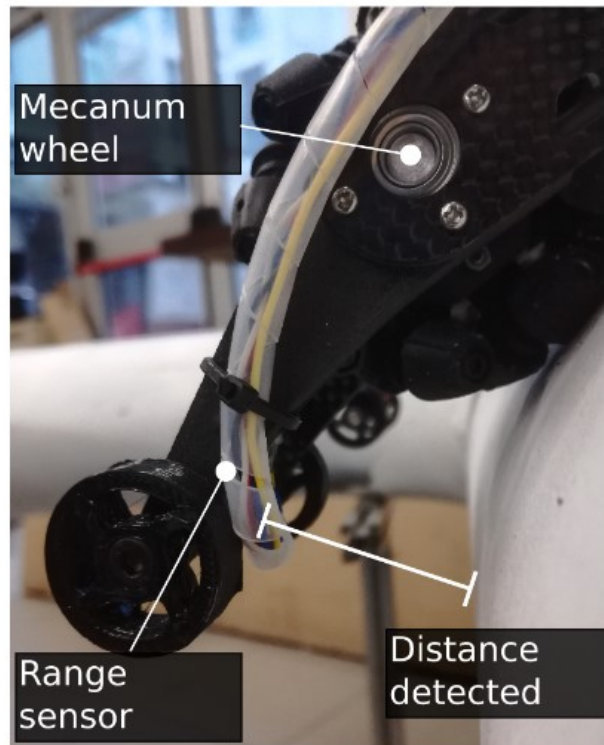


Figure 2: Distance sensors projected in the base frame of the robot to reconstruct the shape of the pipe.

To generate the laser scan data required by the localization package, we chose to implement a virtual LIDAR characterized by 360 degree of scanning range to properly fit the measurements taken by four ToFs. In this context, the output of the virtual LIDAR represents the external faces of the pipe with respect to the body frame of the robot. For this reason, the map used in the localization algorithm is projected on the tube section plane identified by reflecting points of ToF sensors.

2.2 Wheeled base navigation

The map generation and the localization capabilities have been tested in an industrial pipe mockup, as shown in figure 3.



Figure 3: Industrial mockup for 2D map generation and navigation.

In the proposed setup, a navigation task has been implemented to allow the wheeled robot to move along the pipe, reaching its end after crossing the curved part. In this context, the front and rear joints

of the robot have been commanded to adapt the rover configuration to the curvature of the tube. With this aim, the middle points of the pipe are calculated from its external sides and later interpolated with a cubic spline. The generated path is used to find the angular rotation to command the rover joints during the navigation. An example of the generated path over the pipe shown in Figure 3 is reported in Figure 4.

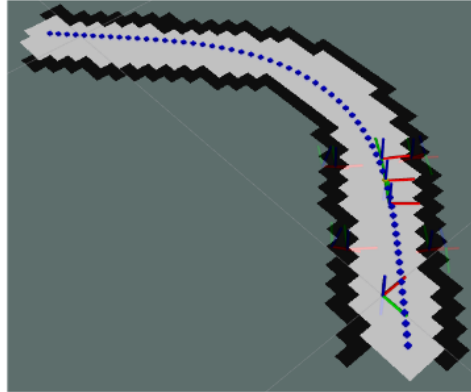


Figure 4: Map of the inspecting pipe. The blue trajectory represents the reference path interpolated with a cubic spline.

In this case, the crawling capability of the rover has been assessed in a navigation experiment in which the rover has to autonomously walk along the industrial pipe where the robot forward velocity has been fixed at 0.4 m/s. Differently, the front and rear joint commands are calculated during the robot motion. In particular, this controller exploits the information provided by the localization and mapping module of the robot to find the closest position between the reference path and the rover joints. At this point, the direction of the reference path is evaluated with respect to the positions of the front and rear joints. A proportional control law has been implemented to nullify the error between the current configuration of the joints and the direction of the path. All the elements involved in the navigation and localization task are depicted in Figure 5. In particular, the walls of the pipe are represented as black circles and are used to calculate the middle points of the pipe (dashed green line). Such points are later interpolated to generate the desired path (blue line). To better show the navigation behaviour of the rover, we also reported the centre position of the ToF measured distances by front (red star) and rear (black star) wheels. In this way is possible to see how the rover actually configures its shape to adhere to the pipe curvature.

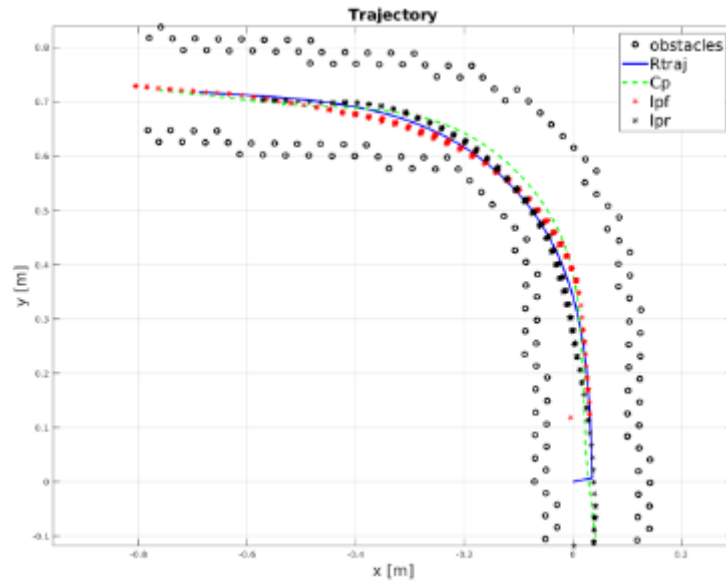


Figure 5: Map data involved in the localization and navigation task. The walls of the pipe (obstacles) are depicted with their middle points (Cp) and reference trajectory (Rtraj). lpf and lpr are the the centre position of the ToF measured distances by the front.

2.3 Dense reconstruction of the pipe

Localization capabilities introduced in previous sections are used to generate the 3D model of the industrial pipe. For this scope, a 3D depth sensor has been mounted on the end effector of the hyper-redundant in order to exploit the generated point cloud for the map reconstruction. In particular, RTAB-Map (Real-Time Appearance-Based Mapping) ROS package (<http://wiki.ros.org/rtabmap>) is used to generate an accurate 3d map fusing the odometry information introduced in section 2.1 and the point cloud of the depth sensor. In particular, RTAB-Map is an RGB-D Graph SLAM approach based on a global Bayesian loop closure detector. This approach has been tested in the Gazebo Ros simulator and CoppeliaSim. In this context, a sketch of the new sensorized manipulator is depicted in Figure 6. Thanks to its precision compact size, an Intel RealSense D435 has been considered to accomplish this task.

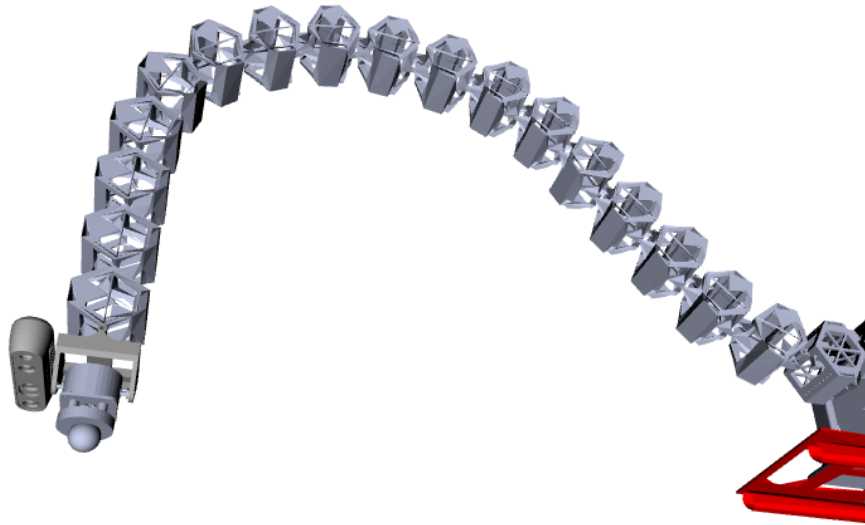


Figure 6: Intel RealSense D435 depth sensor plugged on the end effector of the hyper redundant manipulator.

The generated point cloud can be used to accomplish different tasks. In one case, it can be used to avoid obstacles during the arm motion, while it can be used also to implement the hybrid force-vision approach for contact control used to perform the NDT measures. Starting from the 3D data get by the depth sensor and the localization of the robot over the pipe, a 3D Octomap is generated. In particular, an Octree-based OctoMap map is built, as shown in Figure 7. In this figure, two different pipe structures are reconstructed, a horizontal pipe, and a vertical pipe.

The 3D map is created using the *octomap_server* ROS package (http://wiki.ros.org/octomap_server). This package builds and distributes volumetric 3D occupancy maps as OctoMap binary stream and in various ROS-compatible formats e.g. for obstacle avoidance or visualization. The map is incrementally built from incoming range data like the PointCloud2 of the depth sensor. In this context, the created octomap can be used to plan safe trajectories for the manipulator during the inspection tasks and also to correctly calculate the contact point to perform the NDT measure ad discussed in the next sections.

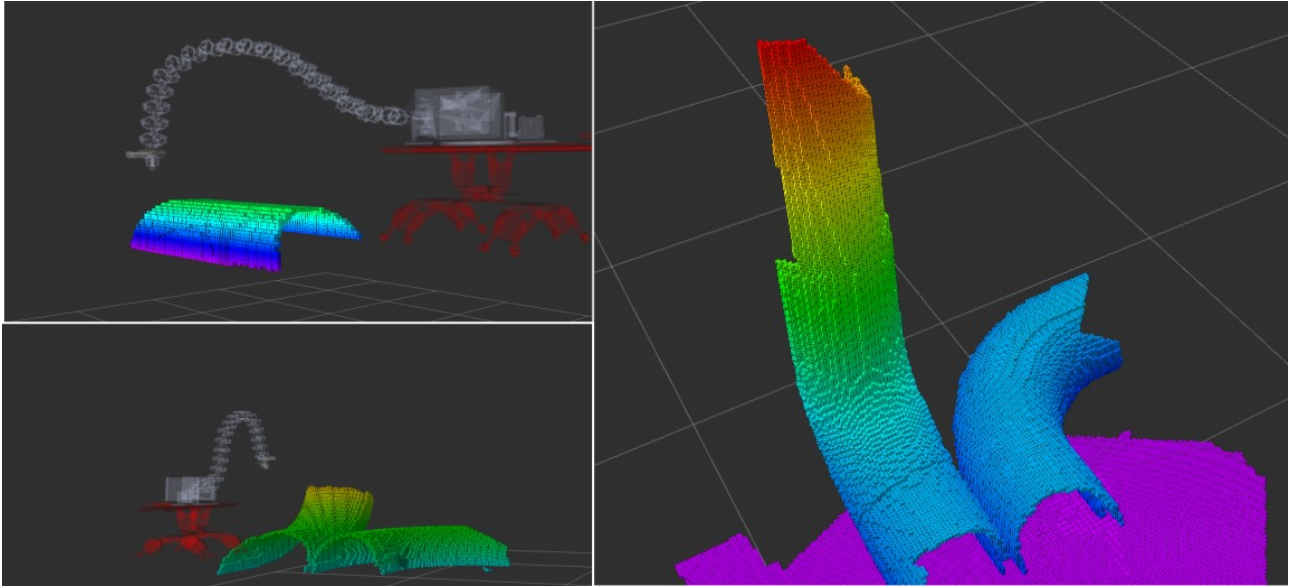


Figure 7: Octomap generated with the depth sensor of the hyper-redundant arm considering two different pipe structures.

3. Hierarchical Task-Priority Control framework

In order to fully exploit the huge number of DOF of the hyper-redundant manipulator, a hierarchical task-priority control framework has been deployed. In this way, the internal motion of the arm can be used to allow a reconfiguration of the arm achieving different tasks at the same time with a different level of priority. In this section, the formulation of the framework is presented along with a set of tasks already set up for robot control. Finally, the use of such a framework to avoid the obstacles of the scene during the arm motion and to perform the NDT measures is discussed.

Let n be the number of DOFs of the robotic arm. The kinematic redundancy of the system can be exploited by defining the dimension of the main task less than n . In this way, it is possible to achieve some secondary tasks in terms of priority, while preserving the principal task, exploiting the extra DOFs. Hence, in a redundant manipulator, the same position of the end-effector can be reached through different joint configurations, letting the robotic arm to be suitably reconfigured by using internal motions. Let $q = [q_1 \cdots q_n] \in R^n$ be the arm joint vector, describing the arm configuration, where q_i is the i -th joint variable, where $i = 1, \dots, n$. The goal of the kinematic control framework is to accomplish the primary and, eventually, the secondary tasks generating the control velocity $\dot{q} \in R^n$ representing the velocity command of the robot joints. In this context, let $\sigma_0 := f_0(q) \in R^{\mu_0}$ be the main task, the following differential relationship holds:

$$\dot{\sigma}_0 := \frac{\partial f_0(q)}{\partial q} \dot{q} = J_0(q) \dot{q},$$

where $J_0(q)$ represents the main task Jacobian. In order to fulfil the regulation problem of σ_0 to the desired value σ_0^* , let $\tilde{\sigma} = \sigma_0 - \sigma_0^*$ be the main task error. The previous equation can be inverted the following equation is used to generate the velocity command:

$$\dot{q}^* = J_0(q)^\dagger \Lambda_0 \tilde{\sigma},$$

where $\Lambda_0 \in R^{\mu_0 \times \mu_0}$ is a positive-defined gain matrix, while $J_0(q)^\dagger$ is the generalized pseudo-inverse of $J_0(q)$. Under such considerations, the error dynamics is achieved considering the following equation in order to obtain an asymptotic convergence of the main task to its desired value.

$$\dot{\tilde{\sigma}}_0 = -\Lambda_0 \tilde{\sigma}_0,$$

Considering that the dimension of the first task is lower than the total DOF of the manipulator, the projection into the null space of the Jacobian of the first task can be exploited to formulate further tasks. One or more secondary tasks can be defined as to $\sigma_{-1} := f_1(q) \in R^{\mu_1}$, where again $\mu_1 < n$. In this case, the joint velocity command can be achieved with the following equation:

$$\dot{q}^* = J_0(q)^\dagger \Lambda_0 \tilde{\sigma}_0 + \sum_{i=1}^{\eta} N_{0|\dots|i-1}(q) J_i(q)^\dagger \Lambda_i \tilde{\sigma}_i,$$

where $N_{0|\dots|i}(q)$ is the projector onto the null space of the augmented Jacobian $J_{0|\dots|i}(q)$ of the i -th task, where $i=0, \dots, n-1$. The Jacobian and the null-space projector are defined as follow:

$$J_{0|\dots|i}(q) = [J_0(q)^\top \dots J_i(q)^\top]^\top$$

$$N_{0|\dots|i}(q) = I_n - J_{0|\dots|i}(q)^\dagger J_{0|\dots|i}(q).$$

The proposed priority-based task formulation ensures that the execution of the main task is not affected by the remaining tasks in the assigned order. In other words, the execution of the lower priority tasks is subordinated to the execution of the higher priority ones. In this context, the lower priority task will be satisfied if enough and qualified DOFs are available, while the complete fulfilment of the main task is instead always guaranteed. Notice that the convergence of the task errors depends on the annihilation and independence properties of the task Jacobians.

With this formulation, the considered tasks are put in a prioritized stack that can be online modified either by insert/removing a task in/from the stack, or by switching the priority order of one or more tasks. In order to address possible discontinuities of the control input that could be generated by the modification of the stack, a time-vanishing smoothing term is adopted to manage the transitions. Specifically, suppose that the transition phase starts at $t=0$ and the r -th task of the stack must be deactivated and substituted by the $(r+1)$ -th task. During the transition, the velocity command is computed with the following equation.

$$\dot{q}^*(t) = \dot{q}_{r+1}^*(t) + e^{-\frac{t}{\tau}} (\dot{q}_r^*(0) - \dot{q}_{r+1}^*(0)),$$

where τ is a positive time constant determining the transition phase duration, and \dot{q}^* is the velocity command corresponding to the k -th task of the stack. When t becomes sufficiently higher than τ , the r -th task stack is entirely removed, and a new transition can start. Similarly, the previous equation can be employed to switch the priority between two tasks in the stack. Consider that such an equation only affects the computed commanded velocity during the transition of different stacks after their composition. The main aim of this equation is to make smoother the activation/deactivation of given tasks. In order to accomplish safely the NDT measurement task, several secondary tasks have been defined and will be discussed in the following.

- End effector position task: the goal of this task is to command the position of the end effector in order to follow a given trajectory.
- End effector orientation task: the goal of this task is to command the orientation of the end effector in order to adapt the orientation of the problem placed on the end effector of the robot considering the shape of the pipe to inspect.
- Joint limit task: this task is used to prevent the violation of the joint mechanical joint limits of the actuator of the arm.

- Center of Mass task: this task is used to maintain the center of mass of the whole robotic system closest as possible to the center of the pipe. This is important to prevent the robot to slip down from the pipe.
- Obstacle avoidance task: this task is used to avoid obstacles detected with the depth camera during the motion of the robotic arm.

In the following, more details about the tasks are provided.

3.1 Position and Orientation task

The first tasks in order of priority are the tasks devoted to the control of the end effector. In particular, the position and the orientation are controlled in these tasks. Of course, since the main goal of the manipulator is to achieve the NDT measure, this task is always enabled and the first in the task priority stack. The formulation is trivial. A position and orientation trajectory is calculated in order to put the probe of the manipulator in contact with the pipe. Starting from the trajectory the position and orientation errors are calculated while the entire Jacobian matrix of the robotic arm is used to generate the joint velocity.

3.2 Joint limit task

This task supports the controller in avoiding joint limits during the manipulation task. In order to bring the joints as far as possible from their mechanical limits, the robot is solicited to keep default and desired configuration. Therefore, different reference configurations have been defined for different kinds of working configurations. In this context, the task function is defined as follow:

$$\sigma_l(\mathbf{q}) = \frac{1}{2n} \sum_{i=1}^n \left(\frac{q_i - q_i^*}{q_{H_i} - q_{L_i}} \right)^2,$$

where q^* is the reference configuration for a given working configuration while q_{H_i} and q_{L_i} are the minimum and maximum joint angles. This task is programmed to be always enabled in the task stack with lower priority with respect to the position and orientation task. The error for this task has been monitored to demonstrate the performance of the manipulation task. In Figure 8 the variation of the joint limit error when this task is enabled and disabled respectively is reported.

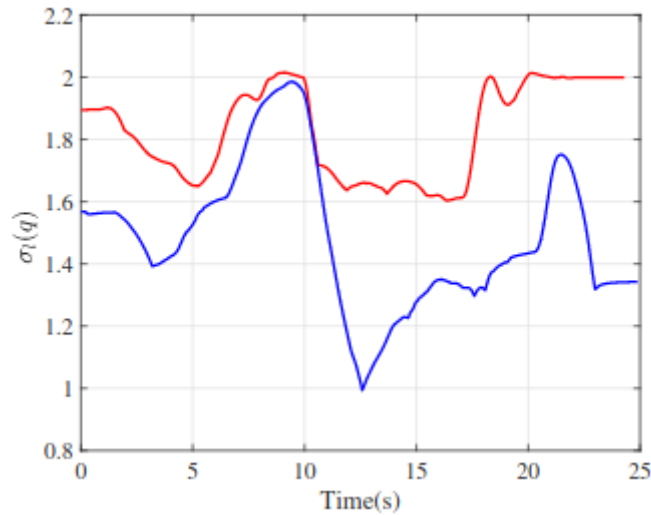


Figure 8: Variation of joint limits error with (red) and without (blue) the joint limit task enabled in the task stack.

3.2 Center of Mass task

The hyper-redundant arm has been attached to the carrying module developed in Task 2.1. In this way, the arm can be used to consider the DOF provided by the mobile base as well. In particular, considering the design of the carrying module, the robotic system is able to stay on the pipe (see Figure 9) stabilizing it and preventing the robot from slipping down.

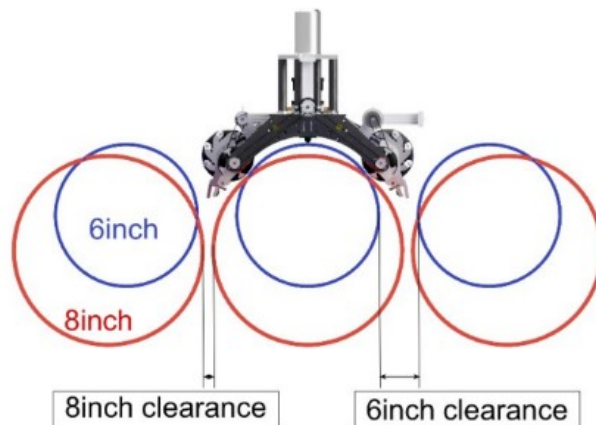


Figure 9: The carrying module of Task 2.1 is able to rotate over the pipe.

During the manipulation tasks, the center of mass (COM) of the robotic structure is far from the geometrical barycentre of the robot causing the slipping down it from the pipe. For this reason, the center of mass task aims to maintain the COM of the whole robotic system aligned with the center of the pipe. The dynamic model of the robot is used to calculate the position of the COM, as shown in Figure 10, where the green sphere represents the COM in the current configuration of the hyper-redundant arm.

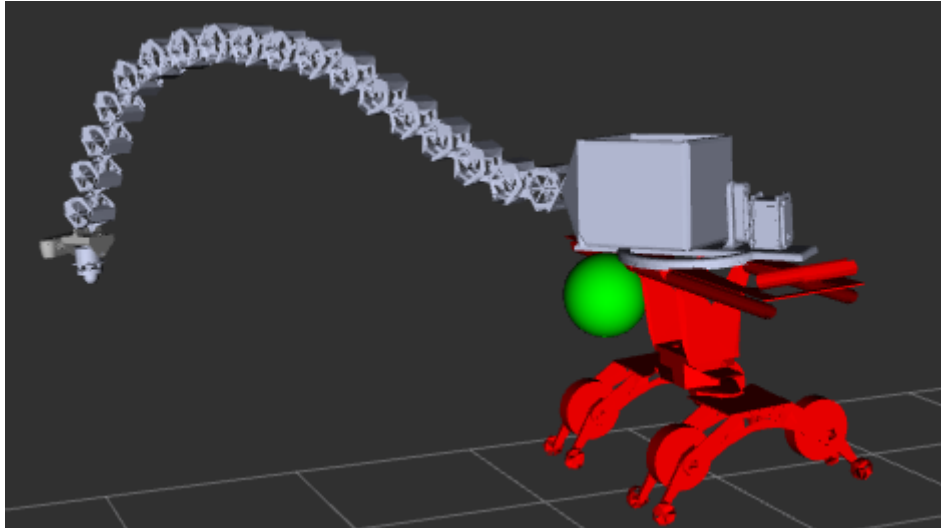


Figure 10: Position of the Center of Mass (green sphere) of the whole robot in a particular configuration of the hyper-redundant arm.

Implementing a torque controller for the actuators of the wheels of the carrying module is possible to control the orientation of the robot over the pipe in order to maintain the COM of the system as closest as possible to the center of the pipe in a certain interval, in order to prevent the robot to fall down from the pipe under the weight of the arm. In this context, the error of this task is calculated considering the initial value of the COM when the manipulation starts.

3.3 Obstacle avoidance task

An additional task considered for the control of the hyper-redundant arm is represented by the obstacle avoidance task. In this context, the internal motion of the arm is used to avoid the obstacles without affecting the end-effector position and orientation. In this context, the error of the task is represented by the distance between the arm and the pipe structure detected by the depth sensor. To demonstrate the effectiveness of the proposed approach a set of simulation test cases have been defined. In particular, Figure 11 shows a longitudinal measurement procedure along x on the $x - y$ plane, in which the secondary constraint has been implemented in the kinematic inversion to obtain an “obstacle avoidance” task. In this case, the robot exploits internal motion to avoid the obstacles its motion without affecting the desired path for its end effector.

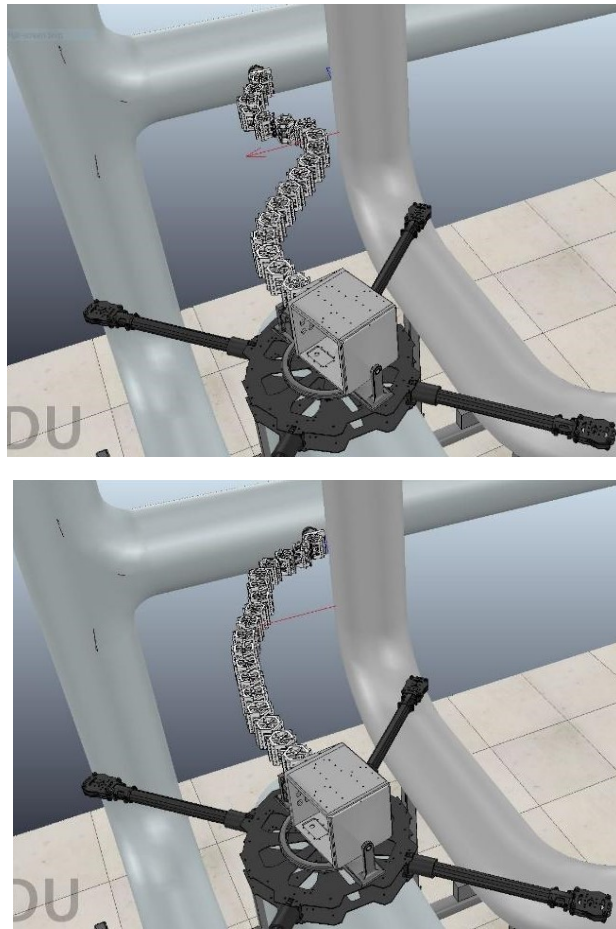


Figure 11: Obstacle avoidance task.

Differently from the other tasks discussed in this section, the obstacle avoidance task is activated only when the robot is working in a narrow region of the workspace. In this case, the distance between the arm and the obstacle during the manipulation is reported in Figure 12.

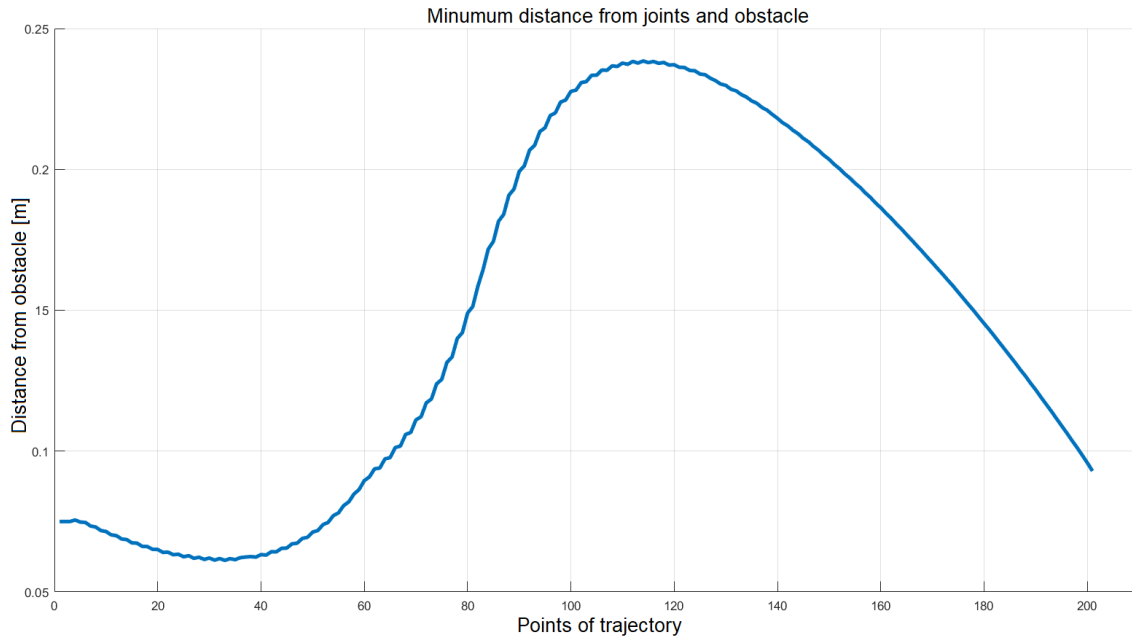


Figure 12: Distances between more close links and obstacles.

4. Stabilization and Control on a Pipe of a Wheeled Mobile Manipulator with a Snake-like Arm

To solve the control problem of the wheeled mobile manipulator with a snake-like arm, we have simulated a control solution involving the arm and the mobile base to preserve the overall dynamic stability over the pipe during the arm's motion.

The addressed task is highlighted in Figure 13: in a cluttered refinery, the platform must perform an inspection task on the adjacent pipe while staying on the other. During the arm's movements, the mobile basis can be destabilized. From preliminary realistic simulation tests with a physics engine simulator (Gazebo), it has been possible to verify that the wheels' torque can saturate. Therefore, the mobile base can slip on the pipe, losing stability and jeopardizing the task achievement. The proposed solution exploits the hyper-redundant arm's dynamic to achieve the task with the end-effector and stabilize the mobile platform. Indeed, the internal dynamic motions of the hyper-redundant arm, transmitted to the mobile base, can help the stabilization of the platform. It means that the ground projection of the centre of mass of the overall device intersects the pipe, guaranteeing static stability. Dynamic stability is also preserved during the movement through theoretical findings in the closed-loop model-based control.

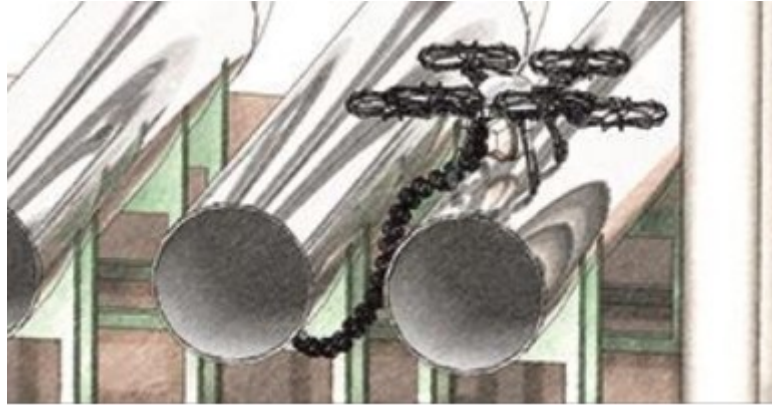


Figure 13: The hybrid robot performs an inspection task in a cluttered scenario.

The considered mobile manipulator is made by a wheeled base (rover) and a snake-like arm. The rover mounts four custom omnidirectional wheels allowing the omnidirectional robotic motion along the pipe. This hyper-redundant manipulator comprises ($n \gg 6$) revolute joints, giving a high movement capability in cluttered environments. In this context, the following assumptions are made.

- A1: It is supposed that the robot is already landed on the pipe rack and the propellers are turned off for the entire duration of the task.
- A2: The rover can move only in its sagittal plane. This means that the eight omnidirectional wheels are kept fixed.
- A3: The wheels and the pipes are rigid, and a Coulomb friction model is assumed, while the rolling friction is negligible.
- A4: The environment is structured. This means that the pipe radius, size, and distances in the pipe rack are known.
- A5: The inertia of the wheels is negligible compared to the inertia of the rover's rotation around the center of the pipe.

The devised architecture comprises three elements, namely, an offline trajectory planner for the end-effector of the snake-like arm; an MPC controller to stabilize the rover on the pipe; and an inverse dynamic control scheme with prioritized redundancy resolution to track the trajectory while retaining the rover balance without colliding with the environment and avoiding self-collisions.

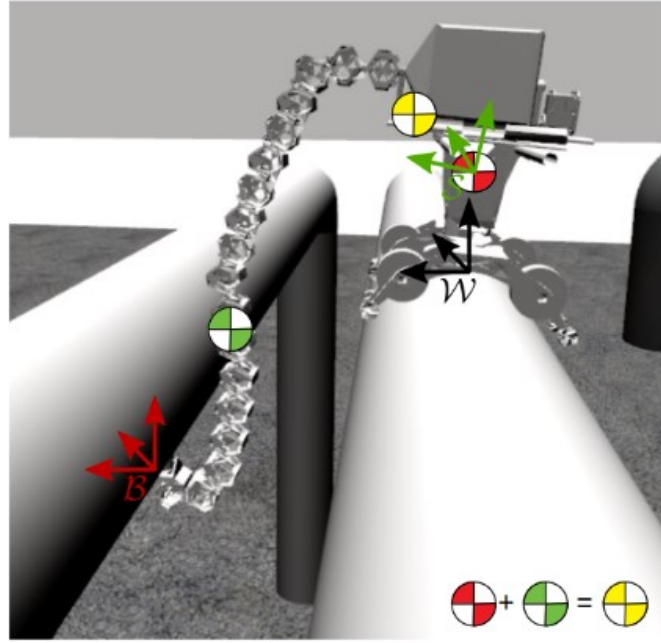


Figure 14: Snapshot extracted from the simulation environment, with the pipe-rack, the rover and the snake-like arm. The frames are depicted. The arm's CoM is represented in green; the rover's CoM is in red; and the whole robot's CoM is in yellow.

With reference to Figure 14, let \mathcal{W} be the fixed reference frame placed, at the top of the pipe. Let \mathcal{S} be the frame attached to the rover's center of mass (CoM). Without loss of generality, at the beginning of the task, \mathcal{W} and \mathcal{S} have the same orientation. Let \mathcal{B} be the frame attached to the snake-like arm's end-effector. Given the assumption A2, it is possible to decouple the kinematics and dynamics of the snake-like arm from the one of the rover. Indeed, the rover has only one DoF given by $\theta \in \mathbb{R}$ which concisely describes the orientation $R_s(\theta) \in SO(3)$ of \mathcal{S} with respect to \mathcal{W} . Therefore, the pose of \mathcal{S} in \mathcal{W} is dictated by $T_s(\theta) = (p_s, R_s) \in SE(3)$, with $p_s \in \mathbb{R}^3$ the position of \mathcal{S} with respect to \mathcal{W} .

The pose of \mathcal{B} with respect to \mathcal{S} is given by $T_b^s(q) = (p_b^s, R_b^s) \in SE(3)$, with $p_b^s \in \mathbb{R}^3$ and $R_b^s \in SO(3)$ the position and the orientation of \mathcal{B} with respect to \mathcal{S} , respectively, and $q = [q_1 \ \cdots \ q_n]^T \in \mathbb{R}^n$ the vector collecting the snake-like arm's joint values. The direct kinematic problem for the arm can be solved using the product of exponentials formula

$$T_b^s(q) = e^{[s_1]q_1} e^{[s_2]q_2} \dots e^{[s_n]q_n} M,$$

where $s_i = [\omega_i^T \ -(\hat{\omega}_i b_i)^T]^T \in \mathbb{R}^6$, with $i = 1, \dots, n$, is the screw axis associated to the revolute joint q_i ; $[s_i] = \begin{bmatrix} \hat{\omega}_i & -\hat{\omega}_i b_i \\ 0_3^T & 0 \end{bmatrix} \in se(3)$ is the matrix representation of the screw axis; $\hat{\omega}_i \in so(3)$ is the skew-matrix associated to the vector $\omega_i \in \mathbb{R}^3$ that represents the unit vector in the positive direction of the i -th joint axis expressed in \mathcal{S} ; $b_i \in \mathbb{R}^3$ is any arbitrary point on the i -th joint axis expressed in \mathcal{S} ; 0_x is zero vector of proper dimensions; and $M \in SE(3)$ expresses the pose of \mathcal{B} in \mathcal{S} when the arm is at its home position. The single exponential map $e^{[s_i]q_i} \in SE(3)$.

The pose of the end-effector in \mathcal{W} can be easily obtained as $T_b(q, \theta) = T_s(\theta)T_b^s(q) \in SE(3)$.

The differential kinematics for the snake-like arm is expressed by the relation $v_b^s = J^s(q)\dot{q}$, with $v_b^s \in \mathbb{R}^6$ the twist of the end-effector related to \mathcal{S} and $J^s(q) \in \mathbb{R}^{6 \times n}$ the spatial Jacobian expressed in \mathcal{S} . The Jacobian is obtained iteratively, and its i -th column can be obtained through

$$J_i^s(q) = \text{Ad}_{e^{[s_1]q_1}e^{[s_2]q_2}\dots e^{[s_{i-1}]q_{i-1}}} S_i \in \mathbb{R}^6,$$

for $i = 2, \dots, n$ and $J_{s,1}(q) = S_1$. The adjoint matrix, $\text{Ad}_T \in \mathbb{R}^{6 \times 6}$, is used to change representation frame, given a generic transformation matrix $T \in SE(3)$. It is thus possible to express the spatial Jacobian $J^s(q)$ in \mathcal{W} as $J(q) = \text{Ad}_{T_s(\theta)} J^s(q)$. The dynamic model of the manipulator can be obtained through the recursive Newton-Euler approach exploiting again the screw theory. The compact form of the dynamic model is expressed by

$$\tau = B(q)\ddot{q} + h(q, \dot{q}),$$

where $B(q) \in \mathbb{R}^{n \times n}$ is the inertia matrix; $h(q, \dot{q}) \in \mathbb{R}^n$ accounts for the gravitation, centrifugal, and Coriolis terms; and $\tau \in \mathbb{R}^n$ is the joint torques vector.

Given the assumption A4, the scope of the planner is to offline compute the desired trajectory for the end effector in \mathcal{W} , avoiding the environmental obstacles. Let $T_{b,i} = (p_{b,i}, R_{b,i}) \in SE(3)$ be the initial pose of the robot at $t = t_i \geq 0$, with $p_{b,i} \in \mathbb{R}^3$ and $R_{b,i} \in SO(3)$ the initial position and orientation of \mathcal{B} in \mathcal{W} , respectively. Let $T_{b,f} = (p_{b,f}, R_{b,f}) \in SE(3)$ be the desired pose of the robot at $t = t_f > t_0$, with $p_{b,f} \in \mathbb{R}^3$ and $R_{b,f} \in SO(3)$ the desired position and orientation of \mathcal{B} in \mathcal{W} , respectively. The desired end-effector's trajectory in each instant of time $t \in [t_i, t_f]$ is given by

$$T_{b,d}(t) = \begin{bmatrix} R(t) & p(t) \\ 0_3^T & 1 \end{bmatrix}, \quad \dot{T}_{b,d} = \begin{bmatrix} \dot{R}(t) & \dot{p}(t) \\ 0_3^T & 0 \end{bmatrix},$$

Notice that it is possible to compute the desired end-effector twist as $v_{b,d} = \dot{T}_{b,d} T_{b,d}^{-1}$.

The planned position $p(t) \in \mathbb{R}^3$ is obtained using a path primitive of the desired trajectory, which is the analytical representation of the path in function of the so-called arc length, $s(t) \in \mathbb{R}$. In case of a rectilinear path, the representation

$$p(t) = p_{b,i} + \frac{s(t)}{\|p_{b,f} - p_{b,i}\|} (p_{b,f} - p_{b,i})$$

is used, where $\|\cdot\|$ is the Cartesian norm, and the evolution of the arc length $s(t)$ is imposed using a polynomial approach with the expression whose coefficients must be found imposing the initial and final conditions the trajectory. In this context, a 5-th order polynomial has been employed.

The orientation planning $R(t) \in SO(3)$ is carried out in this way. First, the matrix $R_{b,f}^{b,i} = R_{b,i}^T R_{b,f} \in SO(3)$ is computed. Then, the axis/angle representation is applied to $R_{b,f}^{b,i}$ extracting the axis $r \in \mathbb{R}^3$ and the angle $\alpha_f \in \mathbb{S}^1$. The matrix $R(t)$ is in turn obtained as $R(t) = R_{b,i} R^i(t)$, where $R^i(t) \in SO(3)$ is constructed through an axis/angle representation at each instant of time, with axis r and angle $\alpha(t)$, such that $\alpha(t_i) = 0$ and $\alpha(t_f) = \alpha_f$. Again, a polynomial representation can be used for $\alpha(t)$. Through this choice it is possible to obtain $R(t_i) = I_3$, with I_\times the identity matrix of proper dimensions, and $R(t_f) = R_{b,f}$.

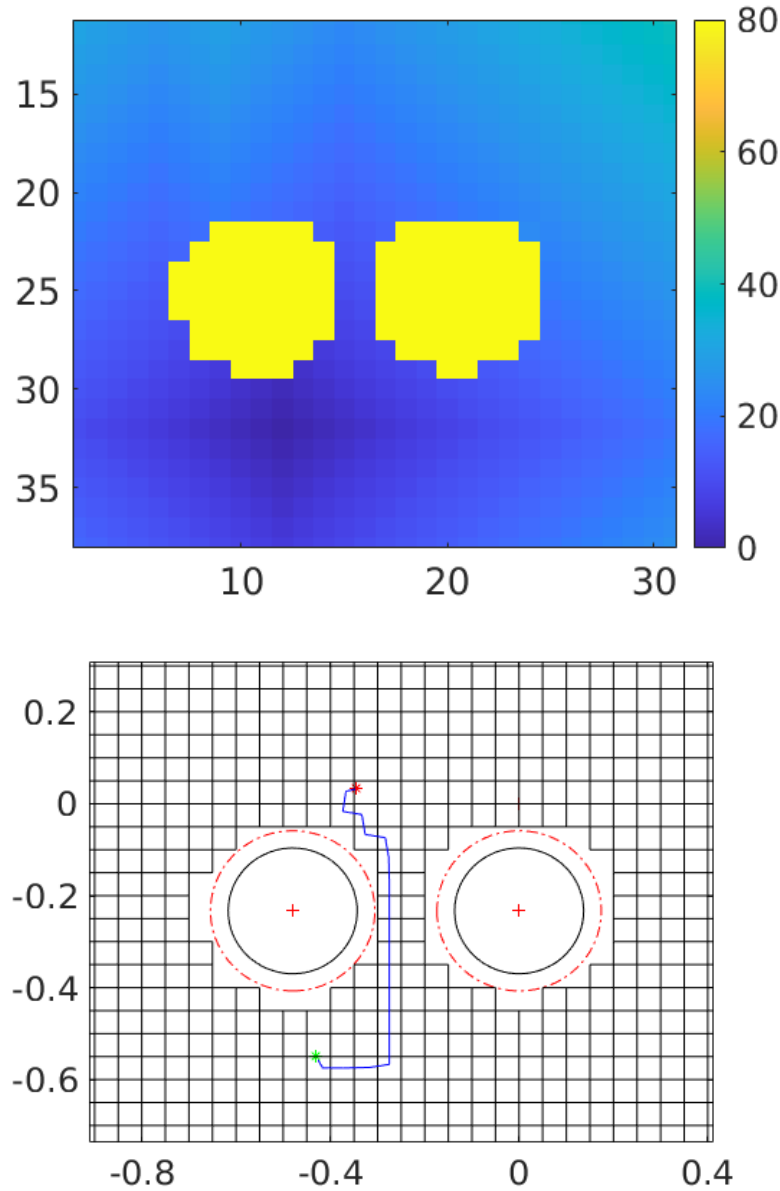


Figure 15: Potential grid-map built with the navigation function algorithm. On the top, the change of colour from blue to yellow shows different value of potential. On the bottom, the red circles indicate the collision-free region bounds and the path in terms of cells is shown in blue connecting the two dots representing initial (in red) and final configuration (in green). The trajectory is computed in the Y-Z plane fixing the end-effector X position.

The entire trajectory is obtained as a concatenation of rectilinear paths computed using the previous approach. For simplicity, the planning for the inspection task is carried out on the sagittal planes. Two adjacent pipes of the pipe-rack are considered (see Figures 14 and 15). Each pipe is approximated to a circle whose radius is $R_p > 0$. In order to have a collision-free trajectory, the navigation function technique is used. This method is based on dividing the workspace into a grid, eliminating those cells associated with the presence of an obstacle. A potential/score value is then associated with each grid cell, and the zero value is associated with the cell containing the goal. Then, starting from it, a map of potentials is constructed, considering an adjacency law to explore the grid. To come by a collision-free trajectory, the cells related to the obstacles are excluded from the planning, receiving a high score. The final trajectory is then obtained connecting the cells starting from the initial to the final

one along a path associated with decreasing potentials. An additional step is considered to minimize the number of segments of this path at the end of the algorithm: the extremities of all the segments are analyzed, removing unnecessary sections, reducing the length of the whole trajectory without neglecting the collision-free hypothesis. Besides, the last segment must be considered to connect the obtained path with the desired position under the pipe to inspect it with the correct orientation. An illustrative example of the navigation function result is given in Figure 14.

4.1 MPC Controller for platform stabilization

An MPC controller is devised to stabilize the rover on the pipe. In this context, the robot has a snake-like manipulator whose dynamics help in stabilizing the rover. To this purpose, the CoM of the whole robot (see Figure 14) is employed. Indeed, this quantity is a function of both the rover's CoM and the arm's CoM.

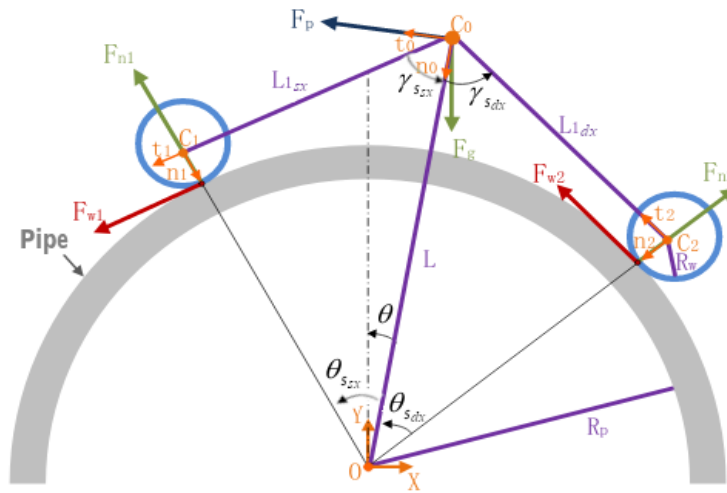


Figure 16: 2-D sketch of the rover on a pipe, with the illustration of the symbols employed to derive the dynamic model of the rover and its constraints.

With reference to Figure 14, the rover is approximated as in the schematic representation of Figure 16 thanks to the symmetry with respect to the sagittal plane and the A2 assumption. Therefore, the rover is assumed to be controlled by the equivalent of two wheels (each resulting wheel is the pair of two actual wheels) through the torques $\tau_{w1} \in \mathbb{R}$ and $\tau_{w2} \in \mathbb{R}$, respectively. The effect of the snake-arm dynamics is seen as a resulting force $F_p \in \mathbb{R}^2$ in the sagittal plane. The input vector to stabilize the rover on the pipe is thus given by $u = [\tau_{w1} \quad \tau_{w2} \quad \|F_p\|^T]^T$. The MPC will compute the optimal u such as to avoid the rover slipping and fall from the pipe, taking into account many constraints. The wheel torques τ_{w1} and τ_{w2} are directly applied to the rover, while the motion controller explained in the next section takes care of controlling the snake-like arm to follow the desired end-effector trajectory and, in parallel, give the resulting F_p at the whole robot's CoM.

With reference to Figure 16, let C_0 be the whole robot's CoM in the sagittal plane. Let $L > 0$ be the distance between the center of the pipe and the center of mass of the WUAV, $\gamma_s, \theta_{s,xx}, \theta_{s,dx} \in \mathcal{S}^1$ be the angles depending on the geometry of the device and the pipe, $L_{1,dx}, L_{1,sx} > 0$ the distances from C_0 to the first and second wheel, respectively. Since C_0 varies due to the arm's movement, the parameters $L_{1,dx}, L_{1,sx}, \theta_{s,dx}, \theta_{s,xx}, \gamma_{s,dx}$ and $\gamma_{s,xx}$ that appear in Figure 16 must be computed in each cycle through straightforward geometric relationships, here omitted for brevity. They basically depend on the manipulator's CoM that is computed in each control step using the theory of rigid

bodies through $CoM_{manip} = \frac{\sum_{i=1}^n m_i r_i}{\sum_{i=1}^n m_i} \in \mathbb{R}^3$, where $m_i > 0$ is the mass of each link and $r_i \in \mathbb{R}^3$ is the position of the CoM of each link in \mathcal{S} .

Let $\theta \in \mathcal{S}^1$ be the angle of C_0 with respect to the vertical axis of \mathcal{W} . Define the state vector $x = [x_1 \ x_2]^T = [\theta \ \dot{\theta}]^T \in \mathbb{R}^2$. Notice that θ can be obtained from the knowledge of the position of C_0 in the sagittal plane through the $Atan2$ function. The dynamics describing the rotation of the structure around the tube can be written in a discrete-time form with time $k \in \mathbb{Z}$ as

$$f(x(k), u(k)) = \begin{bmatrix} x_1(k) + T_s x_2(k) \\ x_2(k) + T_s \left[\frac{g \sin(x_1(k))}{L} + \frac{u_1}{mL} + \frac{\cos(\theta_{s,dx})}{mLR_w} u_2 + \frac{\cos(\theta_{s,sx})}{mLR_w} u_3 \right] \end{bmatrix}$$

where $m > 0$ is the total mass of the mobile manipulator, $g > 0$ is the gravity acceleration, and $T_s > 0$ is the sampling time period.

Define the vectors $F_p = [-\|F_p\| \cos\theta \ -\|F_p\| \sin\theta]^T$, $F_{w1} = [-F_{w1} \cos(\theta_{s,sx} + \theta) \ -F_{w1} \sin(\theta_{s,sx} + \theta)]^T$, and $F_{w2} = [-F_{w2} \cos(\theta_{s,dx} - \theta) \ F_{w2} \sin(\theta_{s,dx} - \theta)]^T$, with $F_{wj} = \tau_{wj}/R_w$, with $j = \{1, 2\}$ and $R_w > 0$ the radius of each wheel. The lumped term acting at C_0 is given by $F_s = F_g + F_p + F_{w1} + F_{w2}$, with $F_g = [0 \ -mg]^T$. The constraints to be imposed to the MPC to prevent wheels slippage and rover falling from the pipe are

$$\left\{ \begin{array}{l} u \in \mathbb{U} \\ x \in \mathbb{X} \\ F_s^T n_0 > 0 \\ |F_s^T t_0| \leq \tan(\gamma_s) F_s^T n_0 \\ |\tau_{w1}| \leq \frac{\mu mg R_w L_{1,sx} \cos(\theta)}{L_{1,sx} \cos(\theta_{s,dx}) + L_{1,dx} \cos(\theta_{s,sx})} \\ |\tau_{w2}| \leq \frac{\mu mg R_w L_{1,dx} \cos(\theta)}{L_{1,sx} \cos(\theta_{s,dx}) + L_{1,dx} \cos(\theta_{s,sx})} \end{array} \right.,$$

where $\mathbb{X} \subset \mathbb{R}^2$ and $\mathbb{U} \subset \mathbb{R}^3$ takes into account the saturation limits of the actuators and the physical limits of the joints, $\mu > 0$ is the static friction coefficient between each wheel and the pipe, $\gamma_s = \frac{\gamma_{s,dx} + \gamma_{s,sx}}{2}$, and $n_0 = [\sin(\theta) \ -\cos(\theta)]^T$. However, the constraints and the dynamics are linearized in each step around the operative point (\bar{x}, \bar{u}) given by the actual state \bar{x} and the previous optimal input \bar{u} . The result is a linear MPC, in which the prediction is carried out on the linearized system. A Cartesian-space inverse dynamic control algorithm is designed to track the desired trajectory. The redundancy of the snake-like arm is later exploited to impose internal motions used to solve tasks with lower priority. The overall architecture is depicted in Figure 17.

representing a security bound. This threshold is given by the sum of the radius of the circle surrounding the obstacle $d_0 > 0$ and a safety distance $d_{min} > 0$. Besides, in the previous equation the term $h(d_0, d_{min}, d_{start}) > 0$ is the amplitude of the repulsive forces, which can have different shapes like the exponential or sigmoidal trends. Here, the first shape has been used to model the collision avoidance force against the static objects in the environment, whereas the second one is used for the generation of the self-collision avoidance forces. The terms $p_{a,c} \in \mathbb{R}^3$ and $p_{b,c} \in \mathbb{R}^3$ are the coordinates of the two points which violates the collision constraint returning $d_{min} < d_0 + d_{start}$, respectively. As a result, the collision avoidance torques and the relative accelerations for the third task can be evaluated as $u_{av} = B^{-1}(q) \sum_{i=1}^{n_c} J_i^T(q) f_{av_i}$, where $n_c > 0$ is the number of the detected collisions and J_i is the Jacobian mapping the generalized forces acting on the joint, which is the one at the minimum distance from the obstacles, into the torques that must be applied on the previous $(j - 1)$ joints. Finally, the total control input for the hyper-redundant manipulator which satisfies all the three tasks is $u_q = u_v + (I - J^\#(q)J(q))u_{stab} + (I - J_{aug}^\#(q)J_{aug}(q))u_{av}$.

4.2 Case studies

Some case studies are now presented to show the behavior of the developed architecture. It will be supposed that the rover is placed with its wheels directly in contact with the pipe, considering an initial angle around the tube different from zero. The parameters that appear in the model of the mobile robot are retrieved from the HYFLIERS prototype. They are $m = 10.6$ kg, $L = 0.23$ m, $R_w = 0.035$ m, $\mu = 0.85$, $R_p \simeq 1.1367$ m. Furthermore, the control inputs and the state variables must lie within the following bounds considered:

$$\mathbb{X} := \{x \in \mathbb{R}^2 : -20^\circ \leq x_1 \leq 20^\circ, x_2 \in \mathbb{R}\}$$

$$\mathbb{U} := \{u \in \mathbb{R}^3 : -10 \text{ N} \leq u_1 \leq 10 \text{ N}, -1 \text{ Nm} \leq u_2, u_3 \leq 1 \text{ Nm}\}$$

The snake-like arm is composed of $n = 21$ joints. The mass of the first link is 0.0518 kg, the mass of the second link is 0.2540 kg, whereas the masses of the following 18 modules until the end-effector are 0.02 kg. The presented tests are performed using a standard PC, with an i7-8750H CPU and 8 Gb of RAM. The OS chosen is Ubuntu, with its release 18.04 LTS Bionic Beaver, and the communication between the software is managed using ROS to its version Melodic Morenia. The simulations are carried out using MATLAB to its version R2020a and Gazebo v7.0, and a synchronization technique between the two software is required to properly exchange control inputs and feedback measurements. The sampling time is set to $T_s = 0.01$ s, and the gains of the MPC and the motion controller have been tuned using a trial and error procedure, which resulted in $Q = 0.001I_2$, $R = [0.1 \ 0.2 \ 0.2]I_3$, $K_p = 270I_6$ and $K_D = 35I_6$. Moreover, the parameters for the asymptotic stability of the MPC controller $\sigma = 10$ and $\delta = 0.05$.

The simulations are performed in order to show how the stabilizing term F_p is necessary to balance the structure when the wheels torque saturate, and moreover to show the performances of the whole inspection task against parametric uncertainties and noisy measures.

In this first case, it is shown how the torques applied by the wheels can saturate, requiring an additional stabilizing term, F_p , to overcome issues that can arise when those torques cannot counteract the gravity effect. The manipulator is placed in a rest configuration wrapped around the rover, letting the CoM of the entire structure to be fixed in time with respect to the robot base frame. The initial angle of the rover around the pipe is set to $\theta_{rover} \simeq -0.1745$ rad, and both the control and prediction horizons are set equal to $K = 10$. The computational time required by MATLAB and Gazebo to complete the simulation is 12 s.

The evolution of the quantities related to this simulation are represented in Figure 18, in which in subfigure (a) is depicted the evolution of the state in function of the optimal control input shown in subfigure (b). The fulfilment of the constraints is shown in subfigures (c-f), in particular those constraints are expressed through $F_{n1} := F_s^T n_0$ (c), $F_{n2} := |F_s^T t_0| - \tan(\gamma_s) F_s^T n_0$ (d), $F_{n3} := |\tau_{w1}| - \frac{\mu m g R_w L_{1,sx} \cos(\theta)}{L_{1,sx} \cos(\theta_{s,dx}) + L_{1,dx} \cos(\theta_{s,sx})}$ (e), $F_{n4} := |\tau_{w2}| - \frac{\mu m g R_w L_{1,dx} \cos(\theta)}{L_{1,sx} \cos(\theta_{s,dx}) + L_{1,dx} \cos(\theta_{s,sx})}$ (f).

Notice how the control inputs related to the actuation of the wheels saturates at the beginning of the simulation. For this reason, the stabilizing input F_p should be used to help stabilizing the structure. The steady-state error is nullified at the end of the control, despite the saturation that afflicts the control inputs.

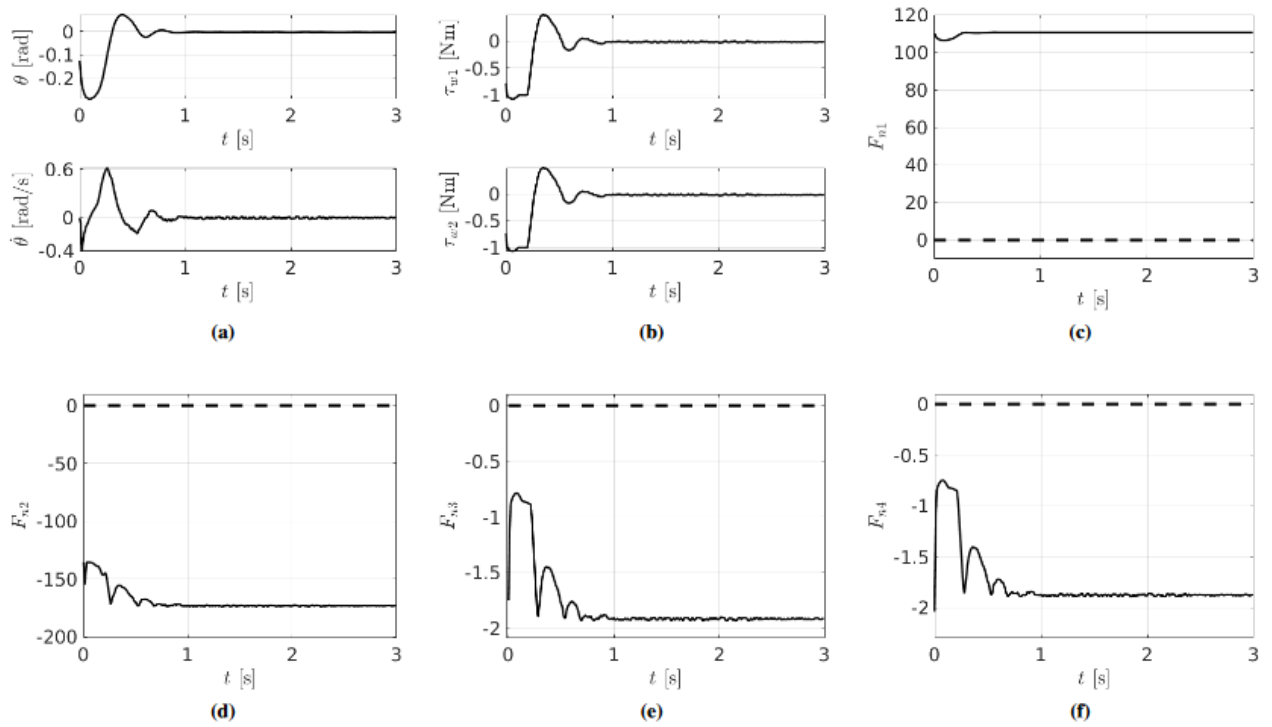


Figure 18: Case study 1. stabilization test. (a)-Time history of the state vector x . (b)-Time history of the wheels control inputs u . (c-f)-Time histories to check the fulfilment of the constraints: with the dashed lines are represented the boundaries.

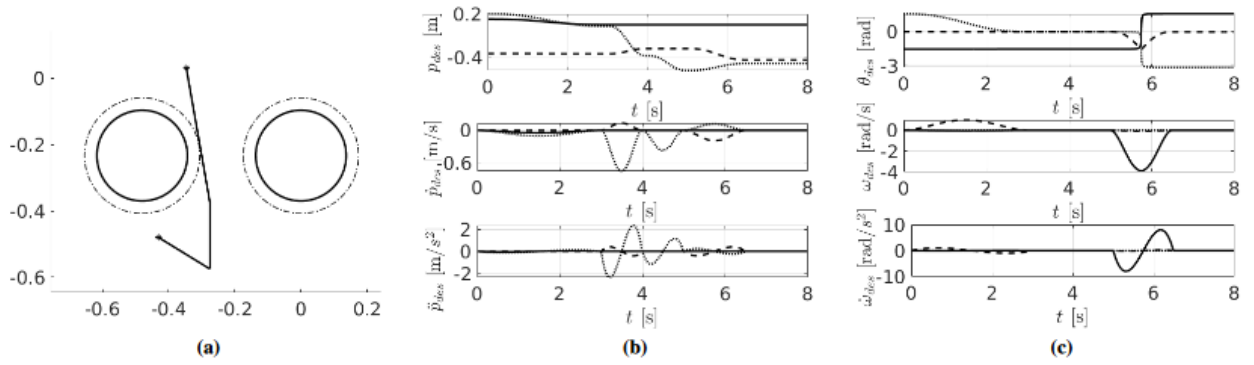


Figure 19: Case study 2: inspection test. (a)-Optimized 2-D path: the two pipes are approximated as two circumferences; with the dashed line the limits of the collision-free region are shown. (b) Linear planned trajectory. (c) Angular planned trajectory.

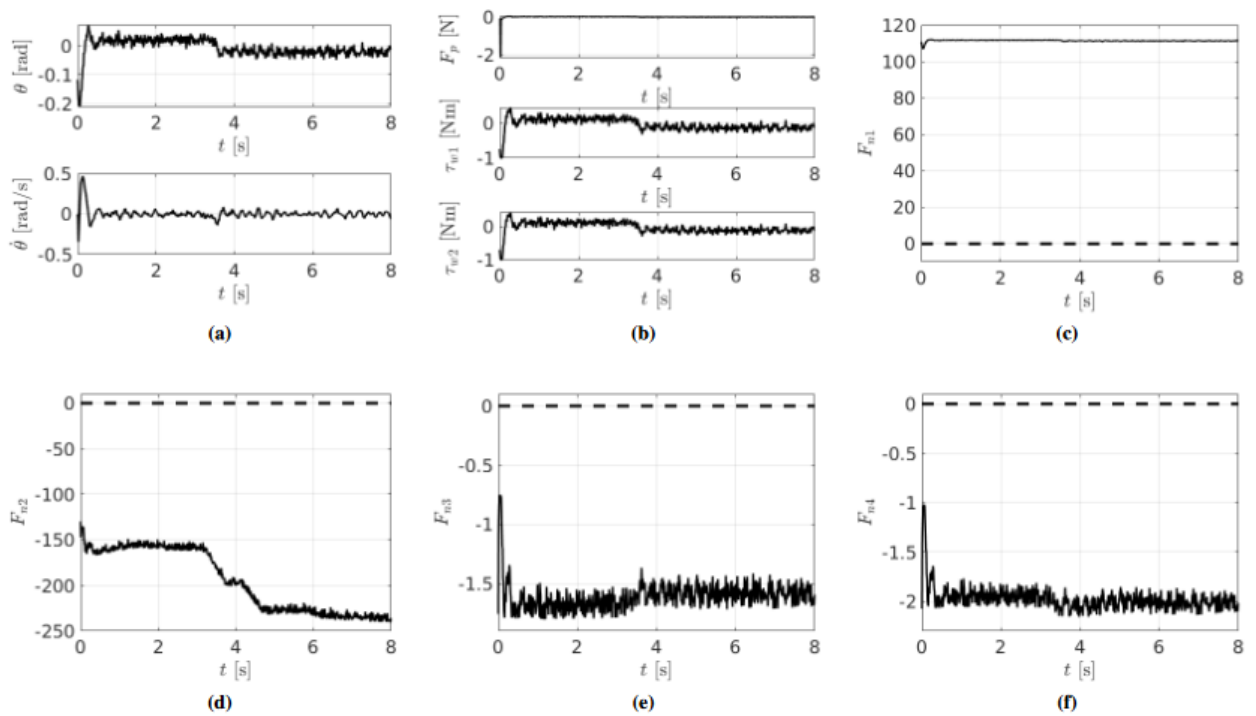


Figure 20: Case study 2: Robustness test. (a)-Time history of the state vector x affected by white noise. (b)-Time history of the control input u . (c-f)-Time histories to check the fulfilment of the constraints: with the dashed lines are represented the boundaries.

In case study 2, the desired inspection task is addressed. The goal is to control the manipulator to reach the inspection area under the pipe, avoiding the collisions with the tube and self-collisions of the snake-like arm. At the same time, the robot must be balanced on the pipe itself, taking into account the movement of its CoM caused by the new configurations of the manipulator. The initial angle of the rover around the pipe is set again to $\theta \simeq -0.1745$ rad, with the control and prediction horizons set to $K = 10$. In this case the computational time required by MATLAB and Gazebo to complete the simulation is 215 s, which can be reduced developing the models and controls using programming languages like C++ instead of MATLAB. From the motion planning algorithm previously described, the trajectory in Figure 19 (a) is obtained. In particular, the final trajectory is composed by two sections: the first one of duration 3 s is necessary to move the end-effector towards the starting configuration of the trajectory depicted in Figure 19 (a), the second one of duration 5 s to reach the goal position under the pipe in exam with a collision-free trajectory. The total simulation is planned

to last 8 s needed by the end-effector to reach the desired position. The collision avoidance torques are computed choosing the following gains used in the skeleton algorithm: $k_{1_{pipe}} = k_{2_{pipe}} = 0.8$ to compute the avoidance forces to avoid collision with the pipes and $k_{1_{self}} = 1.5$ and $k_{2_{self}} = 35$ to avoid the self-collisions. $d_{start} = 0.164$ is the starting distance between the object where the force has to act and $d_0 = 0.03$ is a limit distance where a collision may occur. These parameters are obtained after a trial-and-error procedure.

This test is also used to prove the robustness of the designed MPC controller by considering a white noise, whose standard deviation is chosen as $\sigma = 10^{-2}$, on the measurements of the state of the optimization problem. Also, a parametric uncertainty about the mass of the robot is considered (15% more of its real value). In Figure 20, the results of the MPC algorithm during the inspection task are shown. For the instant of time in which τ_{w1} and τ_{w2} saturates, it is show how the additional control input F_p helps the system to recover this situation. The control algorithm takes into account also the movement of the CoM of the robot, keeping it in the stable region. The noise in the measure of the angle θ can be seen as a chattering behavior on the signal plotted in Figure 20 (a-b). Then, the steady-state error in terms of angular position, represented in Figure 20 (a), is 0.0269 rad, whereas in Figure 20 (c-f) are represented the fulfilment of the constraints.

5. Conclusion

To conclude, the problem of performing NDT measurements with a wheeled mobile manipulator equipped with a snake-like arm has been addressed from different points of view. The perception problem has been solved via the generation of octomaps, while a hierarchical task framework has been tested to exploit the total number of DoFs of the system assuming quasi-static conditions. Similarly, the control of the wheeled mobile manipulator during the inspection has been solved by considering an offline planner, to calculate the desired trajectory for the end-effector and an inverse dynamics control technique with a prioritized redundancy resolution to track the desired reference. The considered subtasks are employed to avoid self-collisions of the snake and ensure collision avoidance with the environment. Besides, the stabilization of the wheeled robot on the pipe and the no-slippage condition of the wheels are addressed through an MPC-based approach. Realistic simulations bolstered the performance of the devised approach, also against parametric uncertainties and noisy measurements.