MAGICIAN
AUTONOMOUS DEFECTS DETECTION AND REPAIR IN MANUFACTURING

PROJECT NO

101120731

PROJECT ACRONYM

MAGICIAN

PROJECT TITLE:

IMMERSIVE LEARNING FOR IMPERFECTION DETECTION AND REPAIR THROUGH HUMAN-ROBOT INTERACTION

CALL/TOPIC:

HORIZON-CL4-2022-DIGITAL-EMERGING-02-07

START DATE OF PROJECT:

01.10.2023

DURATION:

48 MONTHS

DELIVERABLE D4.1

Human-Robot interfaces and intelligence

DUE DATE OF DELIVERABLE:

30.09.2024

ACTUAL SUBMISSION DATE:

30.09.2024

Work Package	**WP4 - Robotic platform and interfaces**
Associated Task	**T4.1, T4.2, T4.3, T4.4, T4.5, T4.6,**
Deliverable Lead Partner	**IIT**
Main author(s)	**Nikolaos Tsagarakis, Luca Muratore, Gionata Salvetti, Nicole D'Aurizio, Domenico Prattichizzo, Luigi Palopoli, Geert Driessen, Roos van Dongen**
Internal Reviewer(s)	**Daniele Fontanelli**
Version	**1.0**

DISSEMINATION LEVEL		
PU	Public	X
SEN	Sensitive - limited under GA conditions	

CHANGE CONTROL

DOCUMENT HISTORY

VERSION	DATE	CHANGE HISTORY	AUTHOR(S)	ORGANISATION
0.1	24.09.2024	First Draft	Nikos Tsagarakis	IIT
0.2	27.09.2024	Internal Review	Daniele Fontanelli	UNITN
1.0	30.09.2024	Final Version	Nikos Tsagarakis	IIT

Co-funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them.

This deliverable is part of a project that has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement no. 101120731.

EXECUTIVE SUMMARY

D4.1 presents the progress made within WP4 towards the realization of human-robot interfaces, specialized end-effectors, control and planning methodologies and overall software and control architecture for the robotic system under development in MAGICIAN. The report introduces the advancements for the above topics during the first year of the project providing details on the development of dedicated wearable tactile devices, the overall control framework, which leverages the XBot2 middleware, presenting the current status of the motion control and task planning and scheduling algorithms that are considered to enable the motion and interaction performance needed as well to permit the robotic platform to prioritize the tasks to be executed related to defect detection and reworking. The presented algorithmic and technological tools will continue evolve in the following period of the project as the robotic system develops and the first experimental trials are carried out.

The objective of this deliverable is to present and provide details of the advancement made in the development of the first version of the technological and algorithmic tools considered within WP4 for addressing the MAGICIAN platform requirements and challenges in terms of human-robot interfaces, end-effector tools, software, control and planning components and their overall integration within the MAGICIAN software and control framework.

The main achievements and findings associated with the work progress performed in WP4 and reported in D4.1 includes the followings:

- Tactile perception modules have been developed, including a wearable tactile device and a handheld tactile device, each designed to account for different user needs and applications. The devices permit to carry out scanning actions resembling the tactile exploration typically performed during surface inspections.
- The first version of control methodologies for providing adaptive robot interactions have been realized and tested in simulation, including the realization of motion and impedance regulation tools for both human-robot and environment-robot interaction.

- The XBot-based framework has been explored as an overall soft and control integration framework and its first integration with the commercial robotic system has been performed.
- Planning and scheduling algorithms and tools have been explored and developed to optimize the robot's defect detection and reworking tasks.
- Motion generation tools have been realized to allows to generate and perform complex motion tasks by leveraging a simple, auto-generated ROS-based interface.

DEVIATIONS

No deviations to report.

TABLE OF CONTENTS

LIST OF TABLES

NO TABLE OF FIGURES ENTRIES FOUND.

LIST OF FIGURES

LIST OF ABBREVATIONS

ACRONYM	DESCRIPTION
D	Deliverable
EC	European Commission
WP	Work package
WT	Work task
CR	Cleaning robot
SR	Sensing robot

1 INTRODUCTION

This deliverable presents the progress made during the first year of the MAGICIAN project, focusing on the development of a robotic platform and its interfaces designed for defect detection and reworking processes. The interfaces serve primarily to acquire tactile and motion data, which will then be replicated by the robotic platform to perform autonomous defect sensing and cleaning operations. Key challenges addressed include the design of intuitive interfaces for precise defect detection and the implementation of impedance and force control methods for enhanced safety and adaptability. Additionally, the deliverable outlines the development of advanced planning and scheduling algorithms that optimize task sequencing for defect detection and reworking, ensuring efficient and collision-free operations in dynamic environments.

1.1 PURPOSE AND SCOPE

The purpose of this deliverable is to provide an overview, outlining the progress made in the development of human-robot interfaces, end-effectors, and control methodologies for the robotic platform, which constitute the core technological components of the MAGICIAN project for autonomous defect detection and reworking processes. These systems are designed to enhance the robot's ability to autonomously perform defect sensing, cleaning, and reworking tasks in industrial environments. This document outlines the design and implementation of these technologies during the first year of the MAGICIAN project.

The human-robot interfaces developed include a wearable and a handheld tactile device. The wearable device, equipped with sensors on the palm, allows for intuitive surface inspections, using interchangeable probe tips to simulate different tactile interactions for detecting defects and imperfections on the car body. The handheld device, featuring a scallop-shaped probe, provides flexibility by covering larger surface areas while detecting smaller defects. Both devices have interchangeable end-effectors and tactile sensors, making them adaptable to different use cases and complementing the camera system by enhancing detection accuracy, especially in challenging areas such as edges or larger surfaces.

In terms of control methodologies, the deliverable highlights the development of impedance and force control strategies implemented via the XBot2 middleware. These strategies allow the robot to adapt to varying external forces, ensuring safety and precision in dynamic environments. The impedance control is further enhanced by introducing a gravity-compensated teaching mode, which enables intuitive human interaction for pose or trajectory teaching, with the robot remaining compliant with the external forces while compensating for gravity.

This deliverable also discusses the development of task planning and scheduling algorithms that allow the robotic platform to prioritize and execute defect detection and reworking tasks efficiently. The system optimizes the sequence of operations based on

defect severity, likelihood, and time constraints, while motion planning ensures collision-free navigation, accounting for environmental factors such as human presence. The deliverable highlights roadmap-based scheduling techniques, dynamic motion primitives, and trajectory adjustments, laying the groundwork for a fully autonomous robotic system that enhances efficiency and safety in industrial operations.

This document sets the foundation for subsequent iterations and refinements, laying the groundwork for a fully autonomous robotic system capable of defect detection, reworking, and cleaning operations in industrial processes.

1.2 CONTRIBUTION TO PROJECT OBJECTIVES

The progress reported in this deliverable directly contributes to the broader goals of the MAGICIAN project, which encompass scientific, technological, and demonstration objectives. Specifically, the objectives that this work supports are:

Scientific and Technological Objectives

- O1: A robotic perception module integrating visual and tactile sensors for defect analysis and classification.
- O2: A robotic cleaning module with a specialized end-effector for defect reworking.
- O3: A software robotic platform integrating services for perception and cleaning modules.
- O4: A closed-loop defect detection and avoidance system for robotic and welding processes.
- O5: Development of two TRL 7 integrated prototypes for defect analysis and reworking.

Social Sciences and Humanities (SSH) Objectives

- O6: A human-centred approach to human-robot collaboration, promoting usability, safety, and trustworthiness.

Demonstration Objectives

- O7: Demonstration of the prototypes in operational scenarios.
- O8: Expansion of MAGICIAN's scope and applicability via Financial Support to Third Parties (FSTP).

The development of human-robot interfaces and control methodologies detailed in this

deliverable is crucial to the MAGICIAN platform capabilities. These advancements enable the platform to perform defect detection, cleaning, and reworking operations autonomously. Implementing impedance and force control systems enhances safety and adaptability during complex industrial processes, while the planning and scheduling algorithms ensure optimal task execution. These contributions align with the overall project goals of delivering a highly automated, adaptable, and efficient system for defect handling and reworking, ultimately advancing the automation of industrial processes and demonstrating the effectiveness of the MAGICIAN platform in operational environments.

1.3 RELATION TO OTHER WORK PACKAGES

As the work done in this WP supports all the objectives of the MAGICIAN project, as detailed in the previous section, the relation is quite tight with all the other WPs. In particular, WP4 is entirely devoted to the definition and implementation of the robotic platform and its interfaces, developing the control, planning and scheduling algorithms for both the CR and the SR. A synthetic list of the most important relations is offered next.

- **WP2 – Use case definition and platform design**: The robotic solutions developed in the WP are primarily defined by the specific requirements of the automotive use case, which is the MAGICIAN main focus. This use case imposes unique challenges related to the system in terms of end-effector design, reworking effectiveness, safety, integration with the perception module, and time and cost constraints. These challenges, being addressed in the first developments of the WP4 work, are briefly outlined in this report.
- **WP3 – Data acquisition and skills learning:** WP4 covers the main technological developments on the MAGICIAN CR and SR robots, hence its activities are closely interconnected with the work carried out in WP3. Planning and scheduling rely on defect analysis and predictions of human operator movements, which is covered in T3.1. Motion control and active sensing directly use data processed through the perception pipeline of T3.2. The motion strategies and the robot control have an intimate relation with the way the operators carry out their job, which is observed and understood in T3.3. However, the robotic platform capabilities will also reshape the work of WP3, since the robotic arm motion capabilities have a direct impact on the available perception strategies and on their effectiveness in due course.
- **WP5 - Integration and performance analysis**: The components developed in WP4 and outlined in this document will be integrated into the final platform (T5.1) and included in the demonstrator (T5.2), thus contributing to the project's KPIs.
- **WP6 – Cascade funding management**: As the robotic platform will be utilized, with the necessary adaptations, in subprojects from the cascade funding scheme, the WP findings will be essential in providing support and technical assistance

(T6.4).

1.4 STRUCTURE OF THE DOCUMENT

This document is structured into eight main sections, detailing the development and progress of the robotic platform, human-robot interfaces, motion control, and planning components as part of the MAGICIAN project. Each section addresses key aspects of the system's design, implementation, and integration, with a final section dedicated to future development and planning.

Chapter 2 focuses on the development of human-robot interfaces, specifically the tactile perception modules. It reviews the state of the art in human-robot interaction, defines the objectives and requirements for the interface, and presents the methodologies employed. The chapter also highlights the integration of the tactile modules into the robotic platform, along with preliminary results and key challenges encountered during development.

Chapter 3 covers the design and implementation of control methodologies, including impedance and motion control for safe and adaptive robot interactions. It describes the state of the art in control technologies and introduces the XBot-based control framework used in the project. Additionally, it discusses the gravity-compensated teaching/interaction mode developed to enhance user-robot collaboration.

Chapter 4 outlines the planning and scheduling algorithms developed to optimize the robot's defect detection and reworking tasks. It details the work scheduling component, motion planning module, and the integration of these elements to ensure efficient, collision-free task execution. Chapter 5, instead, focuses on motion generation and active sensing; this chapter discusses the development of motion generation modules, highlighting task-driven motion and impedance modulation principles.

Chapter 6 summarizes the key outcomes and achievements of the first year of development in WP4. It discusses the current state of the robotic platform and its components, as well as challenges and areas for improvement identified during testing.

Lastly, Chapter 7 outlines the future direction of WP4, including further development and refinement of the robotic platform, interfaces, and control methodologies. It highlights the next steps for integration and testing, with a focus on achieving fully autonomous defect detection and reworking operations.

A comprehensive list of references used throughout the document is reported in Chapter 8, which covers the scientific and technological foundations that support the research and development in WP4.

1.5 SYSTEM OVERVIEW

The robotic system developed within the MAGICIAN project is built around a collaborative robotic platform, specifically the Doosan H2515 cobot. This robot is designed for tasks requiring precision and safety, offering advanced defect detection

and reworking capabilities. The Doosan H2515 features six-axis control, an extended reach of 1500 mm, a payload capacity of 25 kg, and a repeatability of 0.1 mm, making it ideal for handling complex tasks in industrial environments. The cobot's advanced safety features, including six load cells for real-time force sensing, ensure minimal contact forces and high precision during operation, enhancing both safety and performance in collaborative settings.

The system also integrates tactile sensors into wearable and handheld devices, which allow the robot to acquire detailed surface information and autonomously replicate human-like movements during defect detection and cleaning operations.

1.5.1 REQUIREMENTS AND SPECIFICATIONS

The system requirements for the robotic platform emphasize safety, precision, adaptability, and integration. The Doosan H2515 robot was selected for its robust capabilities in handling industrial tasks, including defect detection and reworking. Key specifications include:

- **Payload capacity**: 25 kg, allowing the robot to handle a variety of tools and parts.
- **Reach**: 1500 mm, enabling the robot to cover large surface areas.
- **Repeatability**: 0.1 mm, ensuring high precision in tasks such as grinding and defect cleaning.
- **Safety features**: Six-axis force sensors ensuring a contact force of 0.2 N, guaranteeing high sensitivity in defect detection.
- **Communication protocols**: Ethernet (TCP/IP), ModBUS, and Profinet IO, providing flexible connectivity options.
- **Programming**: The cobot supports intuitive block-based programming and pre-configured routines, simplifying the setup for different tasks. Additionally, the robot's flange is equipped with connectors (6+6 I/O) for secure and efficient tool integration.
- **Compliance**: The system adheres to international safety standards, including EN ISO 13849-1 and EN ISO 10218-1.

These specifications ensure the platform can be easily integrated with other hardware and systems developed in the project while maintaining compliance with industry standards.

2 HUMAN-ROBOT INTERFACE

2.1 INTRODUCTION

The Human-Robot Interface addresses the critical task of defect detection and

reworking in car-body manufacturing through advanced robotics and sensing technologies. A key component of this effort is the acquisition of a comprehensive tactile dataset, crucial for enhancing the accuracy of defect classification and reworking processes. To achieve this, we developed different haptic interfaces, one wearable and one hand-held, to realistically acquire the interaction forces and accelerations that workers experience during the inspection phase. The interfaces have been designed to ensure they do not impede the operator's ability to detect and classify defects. Despite utilizing basic sensors, these interfaces have demonstrated promising results in defect detection and identification, enabling the collection of detailed data that accurately represents the forces involved. In addition to the haptic interfaces, the grinder tool used for defect reworking is equipped with force/torque sensors. This setup supports a learning-by-demonstration approach, allowing the cleaning robot to learn the proper reworking procedures from human operators. Both the defect detection and grinding tools will feature fiducial markers for precise trajectory tracking. After deploying the system, a haptic ring will provide discrete feedback to operators, informing them about the outcomes of post-reworking inspections and verifications.

This report details the development and evaluation of the tactile interface and its integration with the robotic system, aiming to enhance the overall efficacy of defect detection and reworking operations in the MAGICIAN project.

2.2 STATE OF THE ART

The tactile interfaces developed in MAGICIAN leverage cutting-edge wearable haptic technologies. Drawing on IIT's expertise, we developed highly accurate and responsive haptic solutions. To capture the detailed nature of defects, we acquired both static, force-based signals using force sensors [Chinello2012, Pacchierotti2017] and dynamic, acceleration-based signals [Kappassov2015]. Our approach aims to surpass current state-of-art capabilities by transferring human defect-detection skills to a robotic platform. This involves developing active sensing [Seminara2019, Pape2012], where the robot's motion is guided by acquired data to seek further information on defect presence and characteristics. The primary goal is to endow the robot with closed-loop sensorimotor abilities through multi-modal Learning from Demonstration, allowing us to harness human expertise effectively. Since vision-based defect detection has its limitations, the sense of touch remains crucial. To transfer human expertise to robots, we combine tracked hand motions with tactile data, capturing the forces and acceleration applied during defect inspection. Our haptic technology relies on state-of-the-art solutions [Prattichizzo2013], designed to realistically render the interaction forces experienced during inspection.

2.3 OBJECTIVES AND REQUIREMENTS

The aim is to develop a tactile sensing interface that is both user-friendly for operators and easily adaptable for integration into the robotic platform. Particular attention is

being paid to design a system that seamlessly fits into the operator's normal workflow, avoiding any interference or added complexity during defect detection tasks. At the same time, the interface must be modular, allowing for easy extraction and transfer to the robotic platform. This modularity is crucial for enabling the robot to "learn" from the operator's expertise, by replicating the same tactile signals and feedback mechanisms used by the operator to detect and refine surface defects. The challenge lies in ensuring that the interface captures the nuanced forces and vibrations the operator relies on, while remaining flexible enough to be applied across different environments and robotic systems. Achieving this will help streamline the transition between human and robotic inspection, enhancing both the accuracy and efficiency of the defect detection and reworking process.

2.4 TACTILE PERCEPTION MODULE

The tactile sensors proposed in D3.1 have been integrated into two distinct types of tactile devices: a wearable tactile device and a handheld tactile device, each designed to account for different user needs and applications.

The wearable tactile device has been designed to be worn on the user's hand, with the sensors strategically placed on the palm. This configuration allows the user to maintain natural scanning movements, closely mimicking the tactile exploration typically performed during surface inspections. The device is equipped with a probe that features interchangeable pulps, each with different textures, Figure 1. This variety in texture is intended to enhance the detection of surface defects by simulating different tactile sensations, providing a more comprehensive and nuanced analysis. Each pulp, with a radius of 8 mm, is designed to fit seamlessly with the force sensor's dimensions. These pulps are 3D printed using PLA, a material selected for its strength and durability. This ensures that the tips can withstand repeated use during surface scanning without causing scratches or damage. Additionally, the choice of PLA helps maintain the integrity of the force and vibration signals, allowing them to be transmitted clearly to the tactile sensors without interference or signal loss.

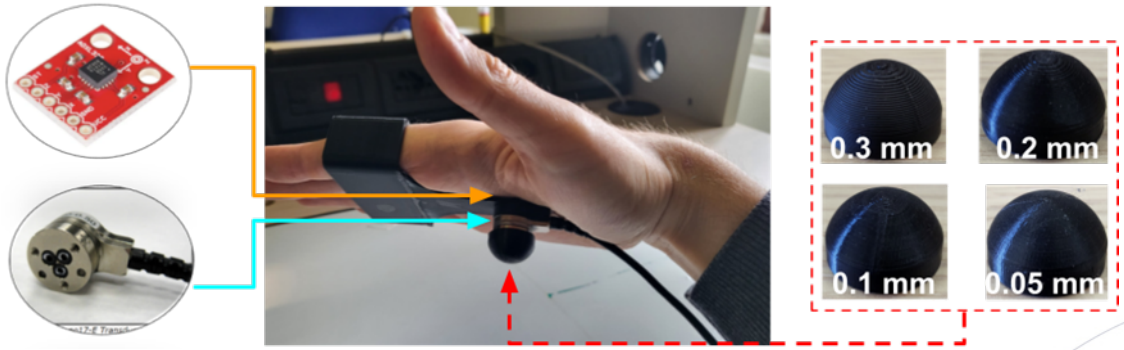


Figure 1: The wearable interface is equipped with a PLA tip, which can be 3D printed with varying texture resolutions (0.3, 0.2, 0.1, and 0.05 mm). These different resolutions are designed to enhance the magnification of tactile signals, allowing for more precise detection of the defect's features.

The second device has been designed as a handheld tool, enabling the operator to perform scanning movements that are slightly modified compared to freehand exploration, while still allowing for freedom of movement, Figure 2. This handheld approach allows for greater freedom and flexibility in the design of the probe. For instance, in this design, a scallop-shaped probe was chosen. This shape increases the surface area that can be scanned in a single acquisition, while simultaneously miniaturizing the contact point with the surface. Such a design is particularly advantageous for detecting smaller, more subtle defects, thus enhancing the overall effectiveness of the tactile scanning process.

Although both designs utilize the same tactile sensor configurations, with interchangeable end effectors between the two devices, they are suitable options to optimize different requirements of the MAGICIAN project, including integration with the camera system described in D3.1. The different end effectors shown in

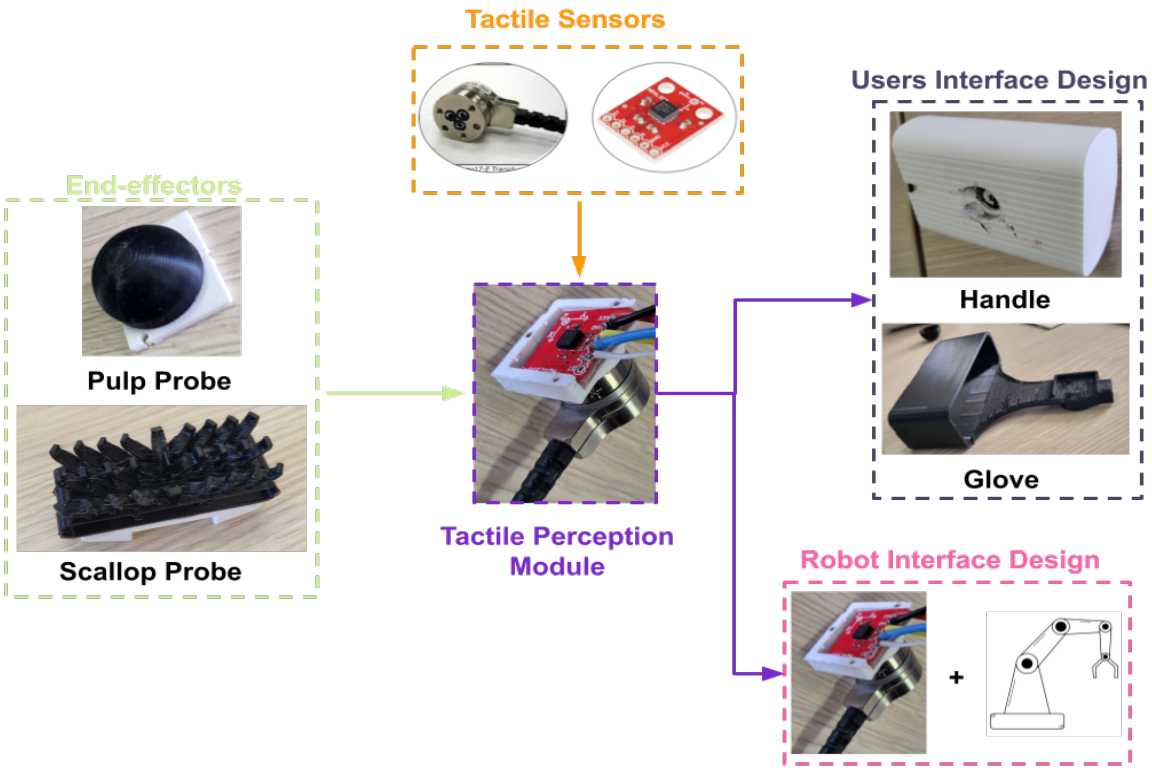


Figure 3 are designed to complement the camera system, particularly in situations where the camera encounters challenges in defect classification, such as uncertain results or areas of car parts that are difficult to access.

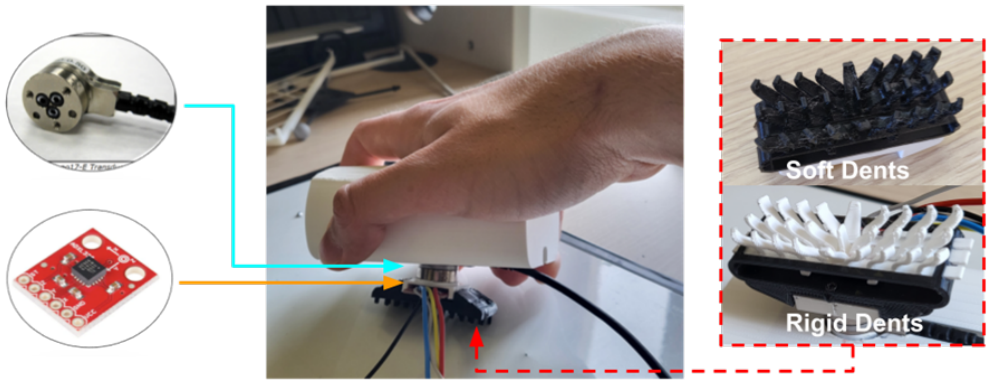


Figure 2: The handheld interface equipped with a scallop component. The scallop is 3D printed using a combination of ABS and TPU materials to ensure effective defect detection while maintaining optimal contact between the surface and the scallop. Different patterns for the teeth arrangement can be employed to customize the detection process, allowing the interface to accurately identify various types of defects.

For instance, the probe tips with varied textures are particularly useful for detecting defects that are hidden from the camera, such as those located on the edges of car parts. The precise, localized scanning capability of these tips enhances the classification process in such scenarios. Conversely, the scallop-shaped end effectors are advantageous when the camera system identifies potential defects on larger surface areas but with high uncertainty. In these cases, the scallop design help to maximizes the scanned area, improving operational efficiency and ensuring more accurate defect detection.

2.5 INTEGRATION IN THE HUMAN-ROBOT INTERFACE

The tactile perception module has been designed with an emphasis on modularity, ensuring seamless integration across the various interfaces required by the MAGICIAN project. This modularity allows the module to be easily adapted for different applications, making it versatile enough to meet the diverse needs of defect detection in both human-operated and robotic systems. At the core of the module there is the tactile perception module, which has been engineered to be compatible with a wide range of end effectors. The flexible design of the perception module ensures that it can be mounted on multiple interfaces, whether the latter are wearable devices for a human operator or robotic platforms for automated inspection tasks. Furthermore, the tactile module is designed to accomplish the specific requirements of each scenario, such as the precision needed for human-driven inspections or the robustness necessary for robotic operations. This adaptability makes it an essential component in achieving the project's goal of transferring human expertise in defect detection to robotic systems, allowing the robot to replicate the tactile feedback that humans rely on.

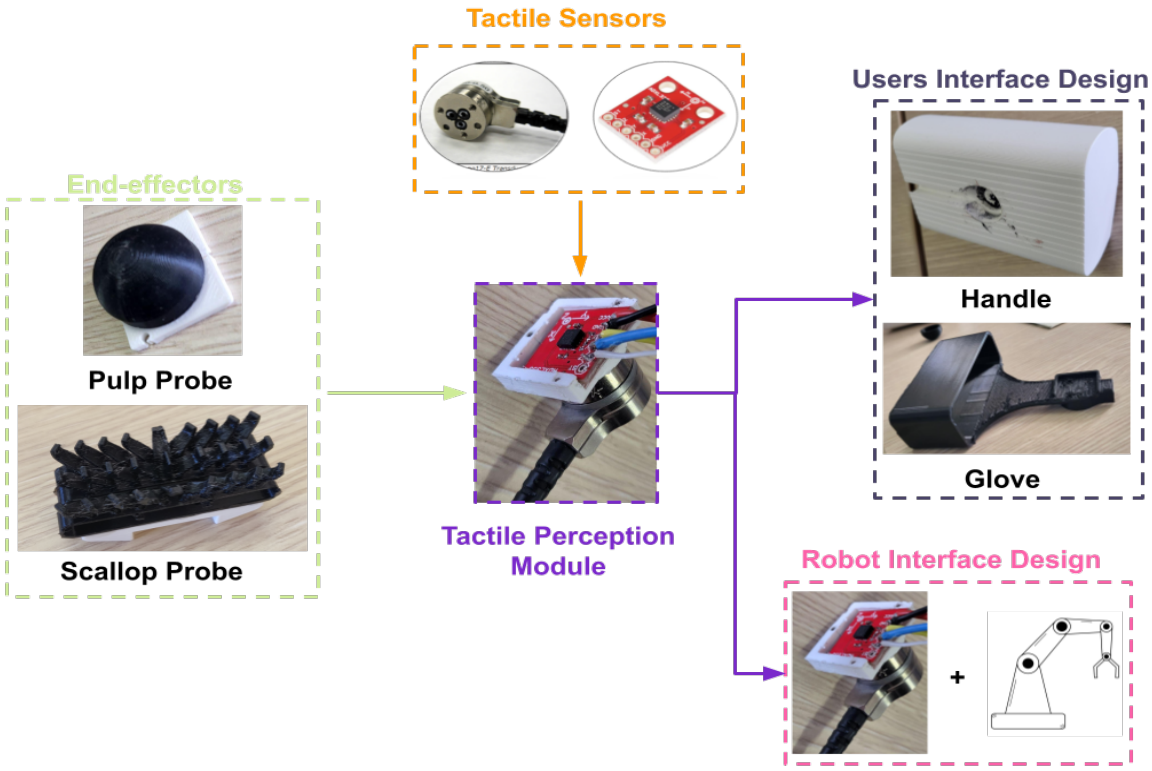


Figure 3: The tactile system is composed of various modules, each serving a specific function within the overall setup. The tactile perception module is designed to be versatile, allowing it to be equipped with different end effectors tailored for surface scanning. This module can be integrated into various interfaces, which are specifically designed either for use by human operators or for mounting on robotic platforms.

2.6 CHALLENGES AND LIMITATIONS

The main challenges and limitations of the tactile perception module come from the same constraints associated with wearable devices. Despite being designed with a strong emphasis on wearability and comfort, the human interfaces do not fully guarantee that operators can maintain their usual range of motion or workflow during defect detection tasks. This is particularly important, as the tactile sensors need to facilitate natural scanning movements without hindering the operators' dexterity.

Further research is required to better understand operator preferences between two key options: a wearable solution which allows operators to perform scanning movements like their traditional workflow but prevents direct hand contact with the car parts, and a hand-held device that is potentially more comfortable but introduces changes to the scanning procedure (but with the same limitation related to the avoided hand contact). Each approach offers advantages and challenges, and it remains to be seen which one will be favoured by operators in terms of ease of use and efficiency.

In addition to addressing these human interface challenges, more work needs to be done to ensure proper integration between the tactile perception module and the robotic platform. Establishing a seamless interconnection is critical for achieving the

project's goal of transferring human expertise to the robot, particularly in defect detection and reworking. The tactile data collected by the human-operated interface must be accurately translated into actionable information for the robot, enabling it to replicate the same scanning techniques and defect identification strategies used by skilled workers.

3 MOTION AND INTERACTION CONTROL

3.1 INTRODUCTION

A robot control system is nowadays a distributed entity, which can be divided into two components: (i) one that runs closer to the hardware and therefore benefits from the least possible latency in accessing the underlying field bus, and (ii) a remote portion that consists of several processes running on multiple machines, or even cloud-based systems. The focus of this report is to propose a solution for the implementation of the first type of software, which we refer to as the real-time middleware XBot2 [Laurenzi2023].

In this report, by real-time middleware, we mean the software framework that runs closest to the target hardware and allows users to customize its behaviour to suit their own needs, typically via a plugin-based architecture. For instance, the firmware inside a digital signal processor (DSP) does not fall into this category, as it cannot be used to invoke custom control code.

The proposed infrastructure will expose multiple integration points to enable, in the context of MAGICIAN, multiple components to cooperate seamlessly at both the high-rate real-time layer and the slower, non-real-time (non-RT) planning level. Our main goals are the seamless support for mixed hardware topologies consisting of both real-time and non-real-time devices, a component-based design, as well as a highly modular architecture that promotes the reusability of its components.

Finally, in our view, components running under real-time constraints should have access to the same high-level, easy-to-use APIs as their non-real-time counterparts. It is the framework's responsibility to provide a real-time-capable toolbox, along with monitoring and troubleshooting tools to simplify debugging, even at the real-time layer.

Real-time performance is usually not a requirement for a general-purpose operating system or kernel, which tends to optimize overall system throughput, possibly penalizing CPU-bound applications. Limiting ourselves to the open-source Linux ecosystem, deterministic time behaviour can be achieved through two different approaches. The first approach is to patch the "vanilla" Linux kernel to enable pre-emptive, priority-based scheduling. This means that a high-priority thread can ideally execute as soon as it is ready (e.g., wakes up from a timed sleep), by immediately halting any lower-priority thread that may be running. This is the approach followed by the well-known PREEMPT_RT patch (described at https://wiki.linuxfoundation.org/realtime/start). The

second approach is the so-called dual-kernel method, where a companion co-kernel intercepts all interrupts and schedules its own processes before the standard Linux kernel. This strategy is followed by the notable Xenomai project (described at https://source.denx.de/Xenomai/xenomai/-/wikis/Start_Here). From the developer's perspective, the main difference is that a dual-kernel approach comes with its own "native" API to interact with the co-kernel, whereas the single-kernel approach requires no custom API beyond standard UNIX/POSIX primitives.

In this section, we present a novel real-time middleware for robotic applications under the name of XBot2. Compared to the previous version, XBotCore [Muratore2020], the XBot2 framework is characterized by the following features:

- A fully dynamic hardware abstraction layer (HAL), supporting on-the-fly device auto-discovery, along with the ability to generate high-level APIs for a simpler, more transparent integration with behaviours, i.e., the user's custom code.
- An improved and more flexible, multi-threaded plugin system for the implementation of periodic behaviours.
- A set of operating system service abstractions that serve as building blocks for the whole architecture and facilitate portability across different Real-Time Operating Systems (RTOSs). Notably, due to a lack of appropriate abstractions, the previous version, XBotCore, was limited to supporting only the Xenomai RT development framework.
- A set of user-land facilities allowing internal components to communicate with each other in a fully decoupled way. Lock-free synchronous and asynchronous paradigms are provided. Leveraging these facilities, components (both at the HAL and behaviour levels) can be seamlessly relocated across different execution threads.
- A more robust and flexible dual-process, client-server approach to allow XBot2 to communicate with robot hardware (or simulators), as opposed to the previous single-process architecture. This layer simplifies XBot2's support for multiple simulators and robotic systems, as demonstrated by our Gazebo, PyBullet, and MuJoCo integrations. This approach avoids a single point of failure and provides process separation in XBot2.

In the following section, we present details and design ideas for each of these key contributions. We also perform validation experiments on different robotic platforms, both real and simulated via various simulation engines. To validate the framework's real-time performance, we provide data from experimental sessions involving both a Xenomai-based host machine and a Linux/PREEMPT_RT one.

3.2 STATE OF THE ART

While other real-time middlewares have been developed by the research and industrial

communities, none has established itself as a de facto standard, unlike the Robot Operating System (ROS), which is by far the most common non-RT integration framework for robotics. The OROCOS project, for instance, has been used in several projects but is now rarely maintained or upgraded, making it difficult for third-party organizations to adopt. Other examples include OpenRTM and PODO. Notably, the successor to ROS, the ROS2 framework, plans to add real-time support as a feature, thanks to its integration with the Data Distribution Service (DDS) as the transport layer, along with careful design. However, RT support is still in its early phases. Other works have adopted a mixed RT/non-RT architecture, where ROS coexists with RT-capable components, similar to how XBot2 integrates with ROS.

This work builds on IIT's experience in developing the precursor to XBot2, namely the XBotCore framework. While a detailed description of XBotCore is beyond the scope of this report, we will briefly mention its main limitations, which we aim to overcome with XBot2. XBotCore provides a framework to execute real-time control code within a single process running on a Xenomai-based host machine while also integrating with the ROS framework (and others). It relies on a static threading model consisting of a real-time thread and a "companion" non-RT thread. The real-time thread executes a basic hardware abstraction layer (HAL) and schedules user modules (known as plugins), while the non-RT thread provides a ROS-based API for external components. However, due to a lack of suitable abstractions, plugins cannot rely on a unified, clear way to configure themselves or communicate with each other, limiting effective code reuse and component decoupling. Therefore, we set new base requirements to overcome these limitations, as follows:

1. Seamless (i.e., completely managed by the framework) multi-thread support; a component should be relocatable to a different thread without needing to adapt its code.
2. Ability to develop decoupled components through appropriate configuration and communication primitives.
3. Ability to adapt to any host OS or RTOS, through an appropriate operating system abstraction layer.

3.3 OBJECTIVES AND REQUIREMENTS

This section presents a novel real-time middleware for robotic applications under the name of XBot2. Compared to our previous iteration, XBotCore, the XBot2 framework is characterized by the following requirements:

- A fully dynamic hardware abstraction layer (HAL), with support for on-the-fly device auto-discovery, as well as the ability to generate high-level APIs for simpler and more transparent integration with behaviours, i.e., the user's custom code.

- An improved and more flexible, multi-threaded plugin system for the implementation of periodic behaviours.
- A set of operating system service abstractions that serve as building blocks for the whole architecture and facilitate its portability across different Real-Time Operating Systems (RTOSs). In this regard, note that—due to the lack of appropriate abstractions—the previous version of the architecture, XBotCore, had the strong limitation of supporting only the Xenomai RT development framework.
- A set of user-land facilities allowing internal components to communicate with each other in a fully decoupled way; lock-free synchronous and asynchronous paradigms are provided. Leveraging these facilities, it is possible to realize components (both at the HAL and behaviour level) that can be seamlessly relocated across different execution threads.
- A more robust and flexible dual-process, client-server approach that allows XBot2 to communicate with the robot hardware (or simulator), as opposed to the previous single-process architecture. Thanks to this layer, it is simple for XBot2 to support multiple simulators and robotic systems (either custom or commercial), as exemplified by our Gazebo, PyBullet, and MuJoCo integrations. This avoids a single point of failure and provides process separation for XBot2.

3.4 OVERALL CONTROL FRAMEWORK BASED ON XBOT

The MAGICIAN control framework is depicted in Figure 4. Starting from the lowest level, we are going to have a layer to control both the Doosan H2515 cobot and the MAGICIAN End-Effector: we will use respectively the Real-Time DRFL API (Doosan Robotics Framework Library, https://github.com/doosan-robotics/API-DRFL) and a custom implementation of the SOEM library (Simple Open EtherCAT Master Library, https://openethercatsociety.github.io/doc/soem/).

Exploiting the XBot2 software architecture, we are capable to control the full MAGICIAN robot system in a transparent way either in simulation or in the real hardware: we will highlight in the next subsections the features of the XBot2 that makes this possible.

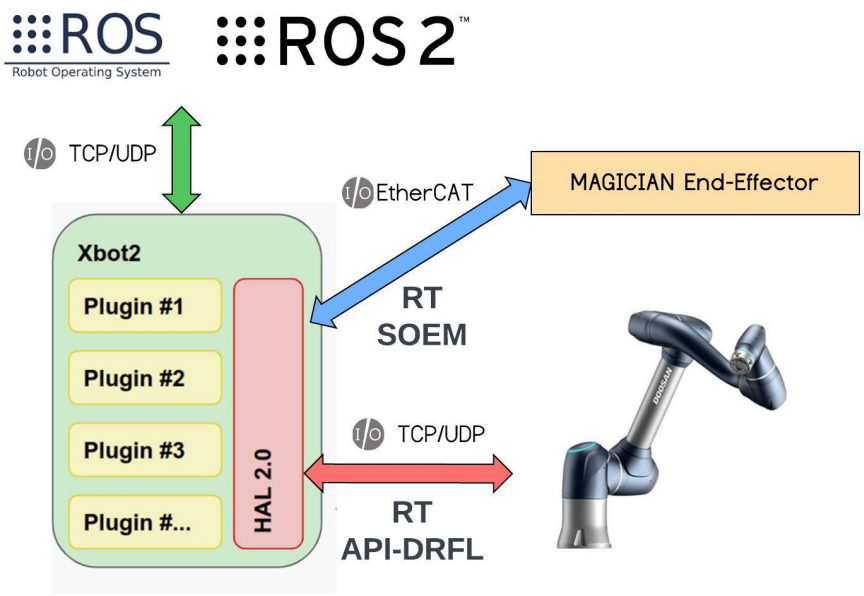


Figure 4: Overview of the MAGICIAN framework.

3.4.1 HARDWARE ABSTRACTION LAYER

The role of a hardware abstraction layer is twofold. First, to grant access - both in transmission and reception - to some external device under suitable performance requirements that depend on the specific device and use case. Whenever this is made possible by the hardware interface, it should also perform automatic discovery of connected device instances to reduce the configuration burden to a minimum for the user. In the context of XBot2, we use the word 'hardware' in a relaxed sense, i.e. this could mean either (i) direct access to the device's fieldbus (such as EtherCAT, Profibus, CAN, etc.), or (ii) usage of a network interface (such as a UDP socket) connecting to the robot's control box (as is often the case when dealing with commercial robots for research) or any other - direct or indirect - way to reach the hardware or simulator.

Secondly, it must present the user of the framework with a programming interface that abstracts away unnecessary details, therefore promoting the reuse of software components whenever the hardware changes in a compatible way. For instance, the user (or client) of the HAL system must not be required to modify his/her code if e.g., (i) passing from simulation to experiments, (ii) changing a device vendor, or (iii) changing the robot vendor entirely.

Finally, the HAL system must comply with our general requirements of Section 1.2, i.e. the HAL should be relocatable to any execution thread in a transparent way, as well as the HAL clients. To address this condition, XBot2's HAL system resorts to a client-server approach, where two sides are defined:

- The DeviceDriver side is a unique per-device component that connects and

communicates with the hardware. This acts as the server side of a device's HAL.

- The DeviceClient side, i.e. the component in charge to provide the proper API to the user. Multiple instances of the client side usually exist in the system.

The XBot2 framework leaves maximum flexibility to the developer in choosing a way to connect and synchronize the driver and client components. On the other hand, it also provides suitable base classes that provide ready-to-use mechanisms that rely on a set of primitives, therefore ensuring satisfaction of our generic requisites in a standardized way. The implemented class hierarchy is structured as follows, with the relevant UML diagram being shown in Figure 5.

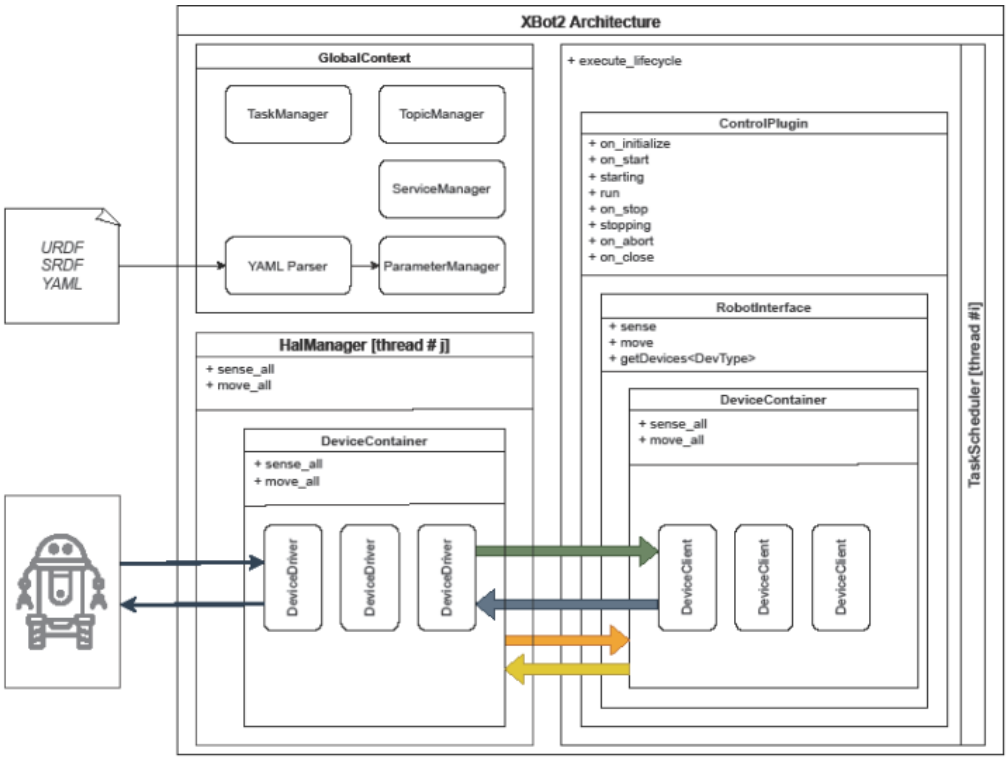


Figure 5: Xbot2 implemented class hierarchy and related UML diagram.

From the user perspective, device drivers and clients are automatically loaded by XBot2's HalManager during the framework initialization phase and based on a configuration file. Available device clients can be inspected and used from a control module via the *RobotInterface* object, as it will be explained in the next section.

As a final remark, device drivers and clients are loaded inside so-called *containers*, which group together all devices of the same kind (e.g., all joints of a given type). Containers can be employed to carry out operations that involve all devices at once, depending on the characteristics of the hardware interface.

3.4.2 IMPLEMENTING BEHAVIORS

Ultimately, the goal of XBot2 is to foster a simple and dynamic use of devices from control (or monitoring) modules to realize some desired behaviour. The implementation thereof happens inside a component called *ControlPlugin*, where the word ``plugin'' refers to the ability to load the module itself inside the system at runtime dynamically.

The XBot2 ControlPlugin is made of three components, i.e.:

- A finite state machine describing the module's lifecycle;
- A set of facilities for named resource resolution;
- A *RobotInterface* object which grants access to the HAL's client side;

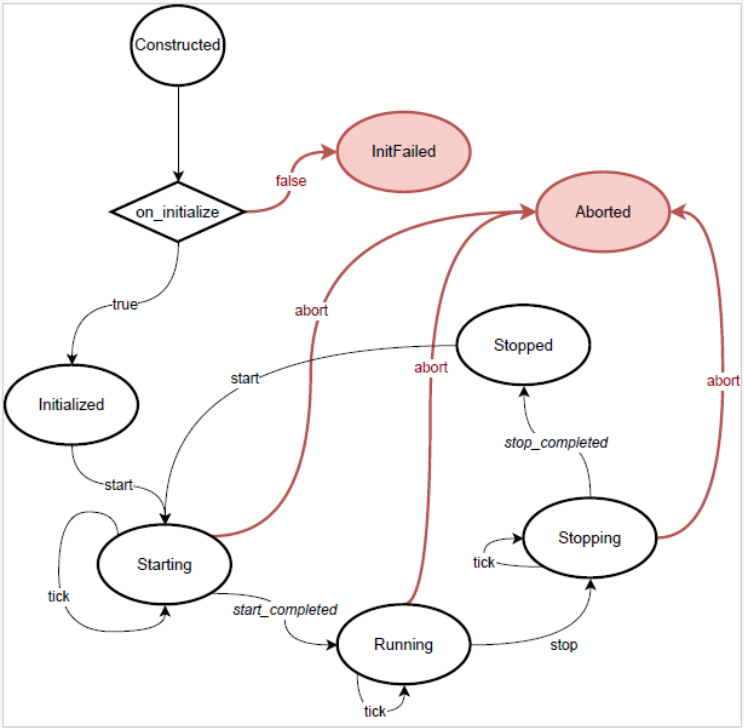


Figure 6: States and permitted transitions of the ControlPlugin lifecycle.

The ControlPlugin lifecycle describes all possible states for the component, and for each state enforces a set of allowed transitions, as depicted in the figure below. The implementer of a plugin can generate transitions and react to them as well, via callbacks. For instance, a periodic task is implemented by assigning a callback to the *Run* state, which is called upon every clock tick, according to the required period. The TaskManager component is ultimately responsible for (i) the plugin execution according to the defined lifecycle, (ii) the broadcast of information about the running modules state as well as some relevant statistics (such as CPU time), and (iii) for providing a service that allows other components of the system to emit events to change the plugin state (e.g., to start a plugin).

XBot2's *RobotInterface* has been introduced as a part of our previous iteration XBotCore, and it has been upgraded to be the main access point for the user to the HAL's client side. The RobotInterface component is part of the more general *XBotInterface* package, which comprises (i) an XBotInterface base class, and two derived classes, namely (ii) the *ModelInterface*, and (iii) the *RobotInterface*. Based on the observation that the generic state of a multi-limbed robot, which is often made of over thirty degrees of freedom, is a difficult quantity for humans to interpret and manipulate, the XBotInterface class leverages the URDF and SRDF in order to present an interface to the robot state (e.g. motor positions, velocities, torques, gains, and others) in a chain-by-chain fashion. Indeed, each chain is usually of a manageable length and is characterized by a natural ordering, i.e. from the base link to the tip link.

The *ModelInterface* class inherits such a chain-wise structure and adds on top a suite of methods to retrieve several kinematic and dynamic quantities corresponding to the robot state as described by the base class. A concrete implementation based on the RBDL library is provided with the framework.

The *RobotInterface* class enables the connection of the inherited robot state to a robot via a sense/mode pair of virtual methods. Two concrete implementations are provided, namely (i) a ROS-based client library called *RobotInterfaceRos*; it connects to XBot2's ROS API via suitable ROS topics, services, and actions and (ii) the XBot2's HAL client library, called *RobotInterfaceRt*, will use the HAL client side as described in the section above instead.

Thanks to this abstraction layer, control code that runs inside an XBot2 ControlPlugin is moveable to a remote ROS node with minimal effort. Clearly, any real-time guarantee will be lost in such a case.

For each ControlPlugin loaded into the system, an instance of RobotInterface is created. On construction, it will dynamically load the client side of all defined HAL devices. The user can then examine the HAL system via the *getDevices* template method, which will return all device clients conforming to a given virtual interface. A generic control module can be written by (i) using the most generic interface that allows the carrying out of the task at hand and (ii) running optional code that requires a more specific interface only if that interface is available on the target hardware.

3.5 MOTION/IMPEDANCE CONTROL

Implemented through the XBot middleware, various modules have been designed to control the robot in different tasks and environments. One such module is a motion generation system based on impedance control. The importance of such control scheme is supported by various demonstrations from literature, beginning with the work of [Hogan1985]. Given that modern robots are expected to operate in dynamic and evolving environments, such the ones considered in the project, it is of fundamental importance to account for the interaction forces experienced by the manipulator. These forces are not only inevitable during manipulating objects or interacting with the environment, but

they are also essential to consider for ensuring safety in the case of unexpected collisions. This is crucial for both the protection of the robot, and, especially, for the safety of humans working in proximity of the robot and/or collaborating with it. These considerations form the basis for developing human-centred technologies, which are the core mission of this project.

The impedance control can be implemented at different levels, i.e. at the task/operational space and at the joint space. While the first is discussed later in the Section 5.5, the latter is based on the following formula:

$$\tau = K(q_d - q) + D(\dot{q_d} - \dot{q}) + M(q)(\ddot{q}_d) + C(q, \dot{q})\dot{q} + g(q) + J^T(q)f$$

where K is a matrix of stiffness gains, D is a matrix of damping gains, $M(q)$ is the mass matrix, $C(q,\dot{q})\dot{q}$ are Coriolis torques, $g(q)$ are gravity torques, $J^T(q)$ is the transposed Jacobian matrix, and f any kind of other external forces due, for example, to interaction with objects or people. By modulating the impedance parameters, it is possible to obtain different levels of compliance, according to the needs. As mentioned earlier, this control module is implemented using the XBot middleware and can be leveraged in real-time control plugins when needed.

3.5.1 GRAVITY COMPENSATED TEACHING/INTERACTION MODE

A practical application of the implemented impedance control module is the *gravity compensated teaching/interaction mode*. By appropriately tuning accordingly the gains of the above formula, the robot behaviour can be fully compliant to external forces, i.e. forces applied by the human, but still compensating the gravity, hence remaining still when no external forces are present. This mode is useful to teach the robot particular poses or trajectory, safely and effortlessly by intuitively moving the manipulator in the wanted positions [Muratore2023].

3.6 CO-DESIGN OF ROBOT-GRINDER INTERFACE

A crucial aspect of the work is to assess the need for an interface between the end-effector and the grinder. Such an interface could be fundamental in reducing vibrations and better distributing contact forces, thus preventing damage to the robot and improving the accuracy of operations. However, designing such as interface is not a simple task. For instance, making it too stiff or not stiff enough could negatively affect the performance of the entire system. For this reason, we believe that using co-design could be our best option to design this interface in the best feasible way. Co-design is based on the simultaneous optimization of hardware and control parameters, to achieve top performance at a specific task. Therefore, we need to be able to evaluate the behaviour of the robot accurately and efficiently for any given combination of hardware and control parameters. In this study, we analyse our ability to simulate the behaviour of the robot performing a grinding operation, using state-of-the-art numerical integration techniques, which would then provide the foundation for our co-design framework.

3.6.1 CONTACT FORCE MODELING

In grinding operations, the end-effector of the robot is subject to various forces. The two main forces are:

- friction forces, generated when the end-effector contacts the surface,
- vibration forces, resulting from the internal dynamics of the grinder.

Friction forces emerge when the grinder touches the surface, opposing the motion. These forces include the normal force, which acts perpendicularly to the surface, and the tangential force, which opposes motion along the plane of contact. At the same time, the grinder itself introduces vibrations into the system. These vibrations have different amplitudes and frequencies, depending on the type of tool and its operating characteristics. In our model, we simplified the forces generated by the grinder by representing them as sinusoidal forces along the three Cartesian axes (x, y, z).

This approach, while being a simplification, effectively captures the effect of vibrations on the robot, helping to understand the impact of these forces on overall system performance. Vibrations can adversely affect accuracy, generating unwanted oscillations that the control system must be able to compensate for.

3.6.2 MODELING THE VIBRATIONS GENERATED BY THE GRINDER

To simulate the vibrations generated by the grinder mounted on the end-effector of the robot, sinusoidal forces along the three Cartesian axes x, y, and z were assumed. We chose to consider a frequency range of 10 to 200 Hz. This choice is based on existing studies that analyse vibrations typical of sanding tools, such as orbital sanders. In particular, a study by Radwin et al. (1990) shows that tools such as the “palm grip orbital sander” produce significant accelerations up to 150 Hz. However, to include potential variations in vibrational behaviour and ensure complete modelling of system dynamics, an overestimation up to 200 Hz was considered.

In the following list, we summarize the frequency-weighted arms accelerations for a common sander tool:

- Tool: Palm Grip Orbital
- Frequency: 150 Hz
- Acceleration RMS along X: 25.4 m/s^2
- Acceleration RMS along Y: 30.3 m/s^2
- Acceleration RMS along Z: 45.6 m/s^2

Frequency-weighted accelerations, expressed in m/s^2, were used to calculate the forces applied on the robot end-effector through the relationship $F = ma$, where m represents the combined mass of the end-effector and the grinder.

3.6.3 NORMAL FORCE

The normal force F_n represents the reaction of the surface to contact with the grinder. It is modelled by combining an elastic and a viscous component:

$$F_n = K(\Delta z) + Cv$$

where:

- K is the elastic constant (stiffness) of the surface.
- Δz is the deformation of the surface along the z axis due to contact.
- C is the viscous damping coefficient.
- v is the relative velocity along the z axis.

3.6.4 TANGENTIAL FRICTIONAL FRICTION FORCE (FRICTION)

The tangential friction force F_r is a force acting along the plane tangent to the contact surface. It depends on the normal force and the relative tangential velocity of the grinder on the surface. The tangential friction force is modelled as:

$$F_r = -k_f F_n \frac{v_t}{\|v_t\|}$$

where:

- k_f is a constant describing the relationship between the normal force and the tangential friction force.
- F_n is the normal force calculated as described above.
- $v_t = (v_x, v_y)$ is the tangential relative velocity vector between the grinder and the surface.
- $\|v_t\|$ is the norm of the tangential velocity.

The negative sign indicates that the friction force is always opposite to the direction of the grinder's tangential motion.

3.6.5 SIMULATION SETUP

Simulations were carried out using Python and the Adam library (Automatic Differentiation for Rigid-Body-Dynamics AlgorithMs). Adam was chosen for its ability to efficiently compute robot dynamics, using automatic differentiation to generate gradients, Jacobians, and Hessians of dynamic quantities. These derivatives were not necessary for carrying out these simulations, but they could come in handy for the co-design of the robot-grinder hardware interface. All simulations were based on the Doosan h2515 robot model.

The PD controller was implemented with a time step of 1 ms, which allowed the control actions to be updated with a high frequency. The simulation was instead performed with a smaller time step of 1/16 ms, to ensure accurate modelling of the system dynamics and

minimize numerical integration errors.

To model the contact between the end-effector and the surface, we considered a stiffness of the surface $K = 3 \cdot 10^4$ N/m, typical for a standard car chassis. The damping coefficient C was set equal to the square root of K.

The reference trajectory for the end-effector is a sine wave with varying amplitudes, designed to test the robot's ability to follow smooth motions.

3.6.6 COMPARISON OF ACCURACY OF INTEGRATION METHODS

In this section, we analyse the accuracy of fourth-order Runge-Kutta (RK4) and Euler numerical integration methods, when controlling the robot with an Operational Space Control (OSC) method. The goal was to determine how the accuracy of the simulation varies with the choice of integration method and time step, with the latter being varied from 1/32 ms up to 1 ms, doubling the value at each iteration. The accuracy of each test was measured as the difference between the trajectory calculated with a certain time step and the ideal trajectory, obtained with the lowest time step (1/64 ms). This difference is expressed in terms of the infinity norm of the error between the two trajectories.

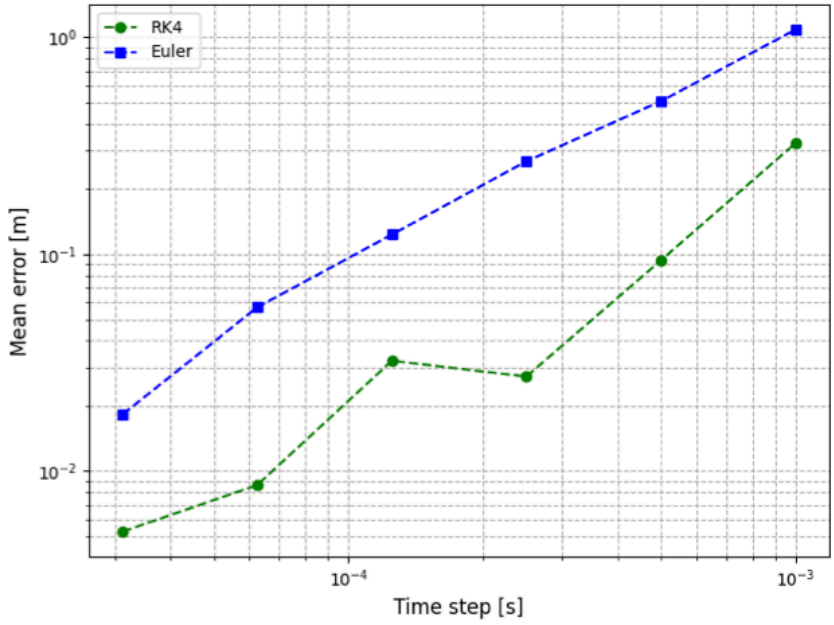


Figure 7: Integration error as a function of integration time step for the Euler and RK4 numerical integration methods.

Our results, depicted in Figure 7, show that when using the typical integration time step of 1 ms, the resulting integration error is about 1 for Euler and 0.3 for RK4. These errors are both too large for considering the simulation results reliable. The plot shows that to get to a simulation error below 0.01, which we could consider low enough for our purposes of co-design, we should use an integration step of 1/16 ms with RK4, and lower than 1/32 ms with Euler. However, these integration steps are both extremely small, which means that the resulting simulations would be extremely computationally

demanding. Moreover, in these simulations we have not considered the hardware interface between the grinder and the end-effector, which could make things even worse. For this reason, we believe that it would be better to investigate the use of more advanced integration techniques, that could provide better accuracy with lower computation times, such as exponential integrators.

4 PLANNING AND SCHEDULING

4.1 INTRODUCTION

The objective of this activity is to develop planning solutions to allow the robot to execute its tasks with a sufficient level of performance and within adequate safety margins.

The objectives of the activity are best understood if we refer to the figures that describe the overall architecture of the system.

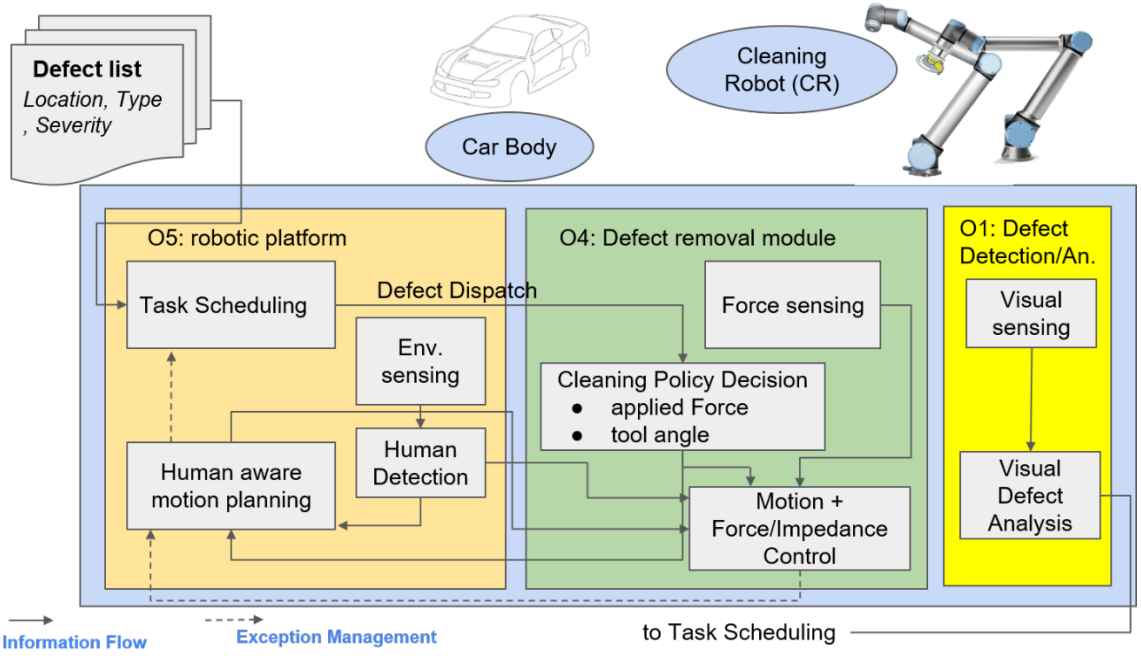


Figure 8: The conceptual architecture of the cleaning robot.

In Figure 8, we have reported the conceptual architecture describing the Cleaning Robot. The task scheduling component (TS) receives a list of defects that need to be reworked. These defects are classified by severity, type and location. The TS decides a sequence of defects to treat considering: 1. The type of cleaning policy that needs to be applied, 2. The degree of severity and hence the importance of each defect, 3. The constraint on time. The decision is implemented by a motion planner that decides the best trajectory between two adjacent tasks in the sequence. The motion planning is human-aware since it is connected to an environment monitoring system. Whenever a human is detected in the workspace of the robot, her/his motion is predicted using the

techniques discussed in D4.1 and the motion plan is adjusted to minimise the risk of collision or accidents.

The architecture of the sensing robot is shown in Figure 9.

In this case the task scheduling component must generate a trajectory that visits and identifies the defects. The area where the defects are most likely to be found can be guessed using the statistical data coming from previous execution. Indeed, a defect is with a good probability the result of a systematic problem that is created by common causes (e.g., two electrodes have not been re-dressed recently and generate a substantial number of sparks during the welding process). However, the task scheduler must combine the visit of areas where defects have been found in the past with the exploration of new areas (things could change over time).

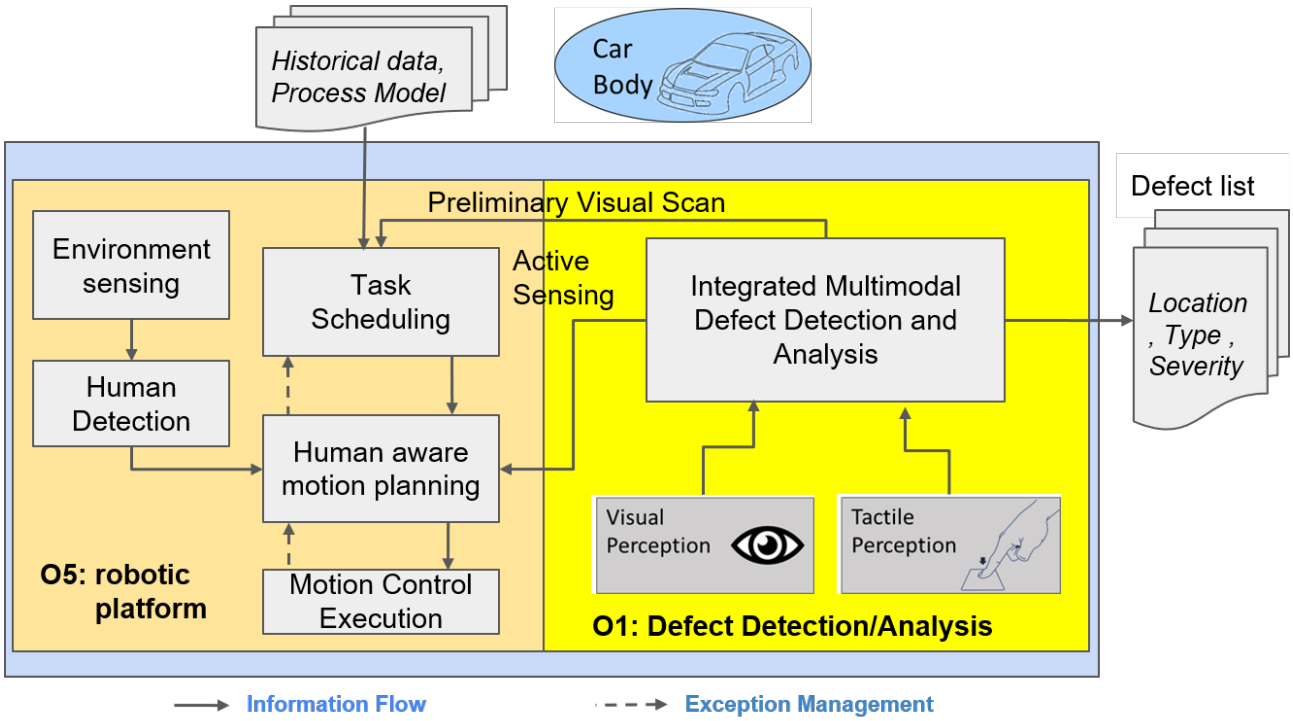


Figure 9: The logical architecture of the sensing robot.

The workings of the human-aware motion planner are essentially the same as for the CR. To summarise, the project requires the development of two algorithms for task scheduling that support the different phases of the project, and of a human aware motion planner.

4.2 STATE OF THE ART

The state of the art in task scheduling and in human aware motion planning is quite rich.

Broadly speaking, a robot planning algorithm is used to decide a sequence of actions that accomplish a task. The simplest type of robot planning is called motion planning,

and its purpose is to find a sequence of elementary motion that enable the robot to move from a point A to a point B. This task can be far from trivial for the presence of possible occlusions and for the possible occurrence of self-collisions (i.e., the robot's linking colliding with each other).

The motion planning problem was formulated in the late seventies of the past century [Lozano79] as a search through the robot configuration space. A larger number of papers (see [Lav06]) have attacked this problem from several directions. Some of the most popular approaches fall in the class of sampling-based approaches [Kav96, Lav01], or in iterative optimisation [Rat09, Sch14].

Collision-free motion planning is certainly important, but not sufficient to enable the robot's operation in a complex space. Indeed, a robot is not simply supposed to operate in the environment, but also to operate changes. This simple fact requires a behaviour that is no longer modelled as continuous in time. Depending on the topology of the underlying space and on the state of the objects that are immersed in it, the robot may have to perform different actions to achieve the same tasks. In the terms of Alami et al. [Al90], Branicky et al. [Bra02], and Hauser et al. [Ha10], robot planning has an inherently multimodal structure. In simpler terms, robot planning is "best viewed as a hybrid discrete–continuous search problem that involves selecting a finite sequence of discrete mode types (e.g., which objects to pick and place), continuous mode parameters (such as the poses and grasps of the movable objects), and continuous motion paths within each mode to a configuration that is in the intersection with the subsequent mode" [Gar21].

In the AI community, the planning problem has been modelled as finding the best sequence of discrete actions that move a transition system into a final desired state [Gh16]. Let us consider a system with:

- A state space X,
- An initial state x_I
- A final desired state x_G
- An action space $U(x)$, which contains all the actions that are active at state x
- A transition function $x' = f(x, u)$), which specifies the new state x', starting from a state x, and applying the action u

The planning problem is about finding a sequence of input that guarantees to move the system from the state x_I into x_G. A more sophisticated version can also require that the solution is optimal in some sense (e.g., time employed, energy spent, etc.). A robot planning problem is translated into a discrete system by means of factoring techniques. The typical approach is to create a graph-based representation, in which nodes represent possible states of the system, and transitions are associated with robot actions. Once a problem is formulated in discrete terms, we can solve the problem using classic graph search algorithms. However, given the huge dimension of the typical state-space, finding a solution within an acceptable time requires very effective heuristics [Bo01, Ho01].

The problem of motion planning and task planning can be approached using a joint optimisation approach that takes decisions both on the discrete variables and on the continuous variables [Tou15]. Although an interesting development, this type of approach falls outside the scope of the project's activities.

The activities that we will develop in MAGICIAN have important connection with the following areas:

- Selecting a number of cleaning tasks to execute within a time budget. As discussed later, this problem can be cast into the framework of orienteering problems [Gun16]
- Selecting a trajectory for the sensing robot that covers an area visiting the area where the defects are more likely to be found. This problem can be addressed within the framework of ergodic control [Dong23]
- Human aware motion planning: the problem is finding a trajectory for the robot that does not compromise the safety and the psychological well-being of the person. This step requires integrating human motion prediction see (D3.1) within a motion planner. In the literature, we find model-based approaches [Ole24] and other approaches that exploit neural networks [Wan24]. In the section below we will discuss our specific solution.

4.3 PLANNING AND WORK SCHEDULING FOR THE CR

As apparent from Figure 8, the Task Scheduling component receives as its input a set of locations $C = \{C_0, \dots, C_{n-1}\}$on the car body where defects are likely to be found. Each location is associated with the following parameters:

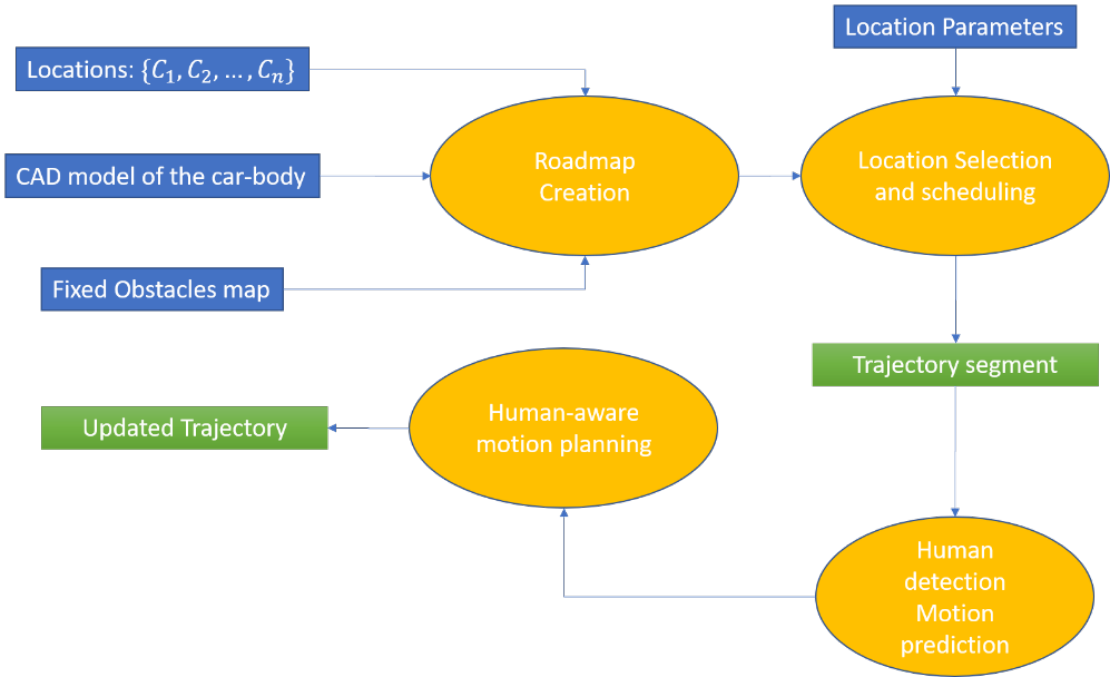


Figure 10: Planning pipeline for MAGICIAN "classic approach".

- A position in the workspace of the robot; this position is a “standard position” for the robot to commence the cleaning operations. For instance, the robot end-effector is required to be at a given distance from the surface (e.g., 10cm) and with a pose orthogonal to the surface;
- A reward S_i associated with the possible removal of the defect; this parameter includes a component related to the degree of severity of the defect and a component related to the importance (which is a function of its position);
- A probability π_i that the defect is actually present; our SR cannot guarantee 100% accuracy, and the presence of the defect can be associated with a measure of confidence;
- A processing time τ_i required to remove the defect; this time is estimated based on the severity of the defect and on its location within the robot’s workspace.

Very importantly, we have a maximum time T_m to complete the exploration (dictated by the so-called Takt time). At the moment of the writing, we have set-up a procedure which follows a quite standard decomposition between motion and task planning, which is illustrated in Figure 10.

4.3.1 ROADMAP CREATION

The first step is finding a roadmap, i.e., a graph-based representation joining the different locations. This step can be performed using standard planning algorithm to connect the points pairwise. We are experimenting sampling-based methods such as RRT [Kuf00], RRT* [Kar11] and optimisation-based solution such as CHOMP [Rat09] and

model predictive control [Ole24, Sle21]. The construction of a roadmap amounts to the creation of a graph in which:

- The nodes correspond to the locations $C = \{C_0, \dots, C_{n-1}\}$
- The arcs correspond to the pairwise motion between two nodes, with each arc having a travel cost τ_{ij}. This time is evaluated by a point-to-point motion planner. We remind that nodes are associated with an additional working time τ_i .

Ideally, the graph is completely connected, but in practice a node could be unreachable from another time within a maximum time. In this case, some of the transitions could be missing.

Once the robot arm has reached the operation area associated with each location, the robot starts executing dynamic motion primitives [Sav23] to remove the defect. Such primitives are the result of a dynamic equation of the form:

$$\tau \dot{z} = \alpha_z(\beta_z(g - y) - z) + f(x),$$

$$\tau \dot{y} = z,$$

$$\tau \dot{x} = -\alpha_x x$$

where y is the desired position (e.g., of the end-effector), z is associated with velocity $f(x)$ is a forcing term and x is an auxiliary variable, which vanishes with exponential rate and is used to modulate in time the forcing action. The latter is expressed as a combination of Gaussian kernels:

$$f(x) = \frac{\sum_{i=1}^{N} w_i \Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)} x,$$

$$\Psi_i(x) = \exp\left(-h_i(x - c_i)^2\right)$$

Each kernel starts its forcing action a modulated time c_i and has a duration determined by h_i. In essence a dynamic motion primitive generates a damped oscillation with a forcing term given by a sum of Gaussian Kernel. DMPs can be used to imitate human operations and their parameters can be learned by observing the human (imitation learning). This aspect is analysed in depth in D3.1.

4.3.2 LOCATION SELECTION AND SCHEDULING

The location selection and scheduling problem is a typical problem of graph optimisation. One of the best-known problems of this kind is the so-called **Traveling Salesman Problem (TSP)** [Jun95], in which an agent is required to visit a set of nodes. The nodes are connected by arcs associated with a temporal cost. The TSP amounts to finding a sequence of nodes that the agent should visit so that all the nodes are eventually visited (1) and the total travel time is minimised (2).

The TSP has been shown to be strongly NP-hard, meaning that no pseudo-polynomial algorithm exists to solve it. However, a large literature has identified many efficient heuristics to produce good sub-optimal solutions. The TSP is widely applicable in fields such as logistics and manufacturing, where minimizing travel costs and times is crucial.

In our case, the problem takes a different form because of the maximum time budget constraint. This modified problem is known as the **Orienteering Problem (OP)** and is defined as follows [Gol87]. Consider a set of nodes connected by a graph, each one associated with a reward that is earned if the node is visited. Suppose an agent starts its travel at a first node (1) and finishes at a final destination node (n). The problem amounts to finding a selection and a sequence of nodes so that the total travel time remains below T_m (1) and the total reward is maximised (2).

The OP is essentially a combination of the TSP and the knapsack problem, where the focus is on selecting a subset of nodes that maximizes the overall profit or reward within the given time constraint [Kant et al., 2022]. Unlike the TSP, the OP allows for flexibility, as not all nodes need to be visited. This makes the OP more suitable for scenarios like ours, where only certain nodes need to be visited based on priority and within a time budget.

The problem can be modelled as an Integer Linear Programming (ILP) problem or solved using heuristic solutions. When the number of nodes is reasonably small, like in our case, the solution time is very low even for a full ILP formulation [Archetti et al., 2007].

4.3.2.1 MATHEMATICAL FORMULATION OF THE ORIENTEERING PROBLEM

Consider a complete graph $H = (A, B)$, where $A = \{1, 2, 3, \dots, n\}$ represents the set of nodes (defects) and B represents the edges (paths between defects). Let node 1 and node n represent the start and end depots, respectively. The profit associated with cleaning defect x is ϕ_x, and ψ_{xy} is 1 if the edge between node x and node y is selected; otherwise, it is 0. The ordinal position of node i is u_i and the travel time between nodes x and y is T_{xy}. The time required to fix defect i is t_i.

The Integer Linear Programming (ILP) formulation for this OP can be described as follows *[Kan22]*:

Objective Function

Maximize the total profit of selected nodes:

$$\text{Maximize} \sum_{x=2}^{n-1} \sum_{y=2}^{n} \phi_x \psi_{xy}$$

Constraints

1. Ensure the route starts and ends at the given depots:

$$\sum_{x=2}^{n} \psi_{1x} = 1$$
$$\sum_{x=1}^{n-1} \psi_{xn} = 1$$

2. Ensure each node is visited at most once:

$$\sum_{x=1}^{n-1} \psi_{xk} = \sum_{y=2}^{n} \psi_{ky} \leq 1 \quad \forall k = 2, \dots, n-1$$

3. Ensure the total travel time does not exceed the time budget T_{Max}:

$$\sum_{x=1}^{n-1} \sum_{y=2}^{n} T_{xy} \psi_{xy} + \sum_{x=2}^{n-1} t_i \sum_{y=1}^{n} \psi_{ij} \leq T_{\text{Max}}$$

4. Sub-tour elimination constraints:

$$2 \leq u_x \leq n \quad \forall x = 2, \dots, n$$
$$u_x - u_y + 1 \leq (n-1)(1 - \psi_{xy}) \quad \forall x \neq y, \forall x, y = 2, \dots, n$$

5. Ensure binary decision variables:

$$\psi_{xy} \in \{0,1\} \quad \forall x, y = 1, \dots, n$$

In this formulation, constraints ensure the route starts and ends at the designated depots, no node is visited more than once, the total travel time is within the allowable budget, and sub-tours are eliminated. The objective function aims to maximize the profit associated with the visited nodes, reflecting the prioritization of defects. This approach aligns well with the problem requirements, allowing the cleaning robot to focus on the most critical defects within the given time constraint.

4.3.3 INPUT PARAMETERS AND DATA TRANSFORMATION

The formulated orienteering problem requires a set of input parameters which need to be derived from the output of the sensing robot and the robotic movement data. The input required by the model is as follows:

- Time Constraint (T_{Max})
- Travel Times Matrix (T_{xy})
- Estimated Cleaning Times (t_i)
- Profit for Cleaning Defect (ϕ_i)

While the time constraint is a constant, the other inputs have to be derived by three different modules, as visualized in Figure 11.

4.3.3.1 TRAVEL TIMES MATRIX

To create the travel times matrix (T_{xy}), as visualized in Figure 11 as the "Distance Matrix Calculator", we need to follow these detailed steps:

1. **Collect Defect Locations**: Gather the coordinates of each defect identified by the sensing robot. The locations should be in the form of a list of tuples, ($[(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)]$).

2. **Calculate Euclidean Distances**: Compute the Euclidean distance between each pair of defects using the formula:

$$d_{xy} = \sqrt{\{(x_y - x_x)^2 + (y_y - y_x)^2\}}$$

This will give you a distance matrix D where each entry D_{xy} represents the distance between defect x and defect y.

3. **Convert Distances to Travel Times**: Transform the distance matrix into a travel time matrix by applying a conversion factor based on the robot's speed. If the robot's speed is v (distance per unit time), the travel time T_{xy} can be calculated as:

$$T_{xy} = \frac{D_{xy}}{v}$$

The resulting travel time matrix T will contain the time required for the robot to travel between each pair of defects.

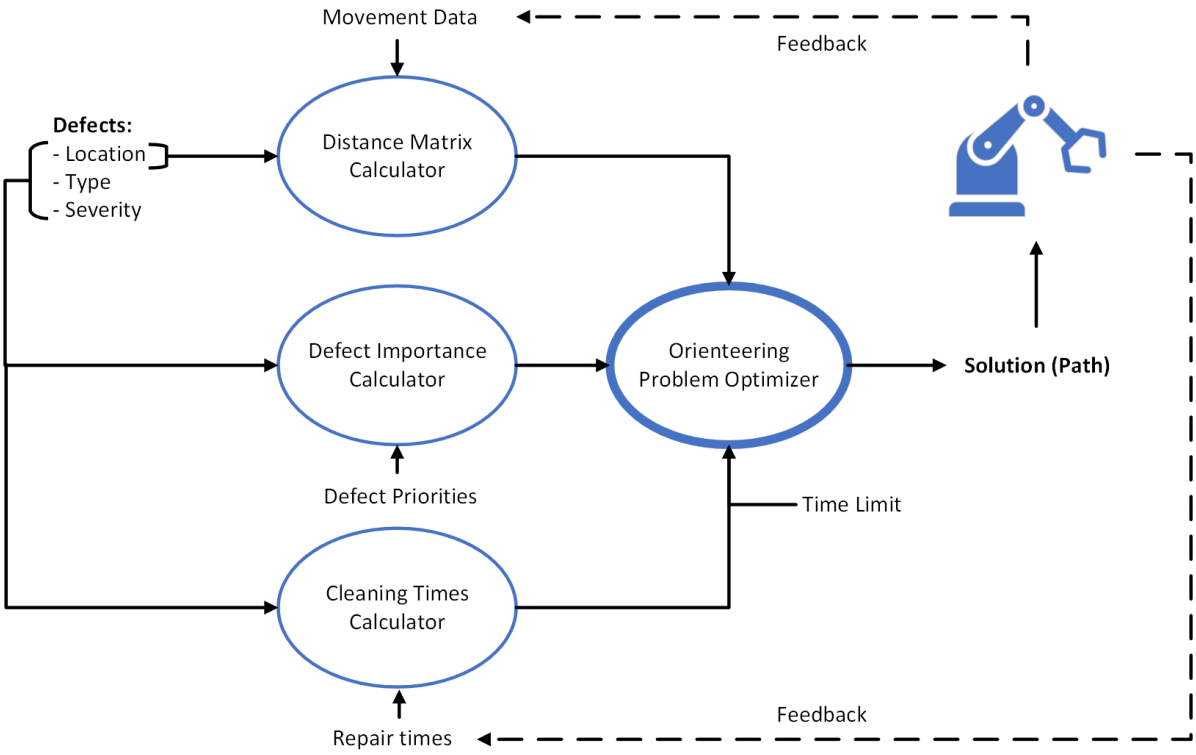


Figure 11: Path Optimizer Framework.

4.3.3.2 PROFIT BASED ON PRIORITIES

The next step is to calculate the importance of cleaning each defect in terms of profit, visualized as "Defect Importance Calculator" in Figure 11. To determine the profit ϕ_i for cleaning each defect, follow these steps:

1. **Identify Defect Types and Severities**: From the sensing robot's output, obtain the type and severity of each defect. These should be formatted as lists, $[type_1, type_2, \dots, type_n]$ and $[severity_1, severity_2, \dots, severity_n]$ respectively.

2. **Define Priority Formula:** Develop a formula to calculate the priority of cleaning each defect based on its type and severity. For example, a simple weighted sum could be used:

$$\phi_i = \alpha \cdot type_i + \beta \cdot severity_i$$

where α and β are weights that reflect the relative importance of defect type and severity.

4.3.3.3 ESTIMATED CLEANING TIMES

Lastly, to estimate the cleaning times (t_i) for each defect for the "Cleaning Times Calculator" in Figure 11, use the following procedure:

1. **Obtain Cleaning Time Data**: Based on historical data or expert input, determine the

average time required to fix defects of different types and severities. This information can be compiled into a reference table.

2. **Calculate Individual Cleaning Times**: Using the defect type and severity data, lookup or calculate the estimated cleaning time for each defect. This could be done using a predefined function or lookup table. For example:

$$(t_i) = f(type_i, severity_i)$$

where f is a function or table that returns the cleaning time based on the defect's type and severity.

3. **Format the Cleaning Times**: Compile the estimated cleaning times into a list $[t_1, t_2, \ldots, t_n]$, ensuring that all times are in a consistent unit (e.g., seconds or minutes).

By following these steps, you can transform the raw data from the sensing robot into the required input parameters for the orienteering problem model. This structured approach ensures the inputs are accurate and formatted correctly for effective optimization.

4.3.4 SOLUTION APPROACHES

We have implemented the algorithm in the context of MAGICIAN and are now in the process of testing and benchmarking. We are also considering a stochastic generalization of the problem to account for the probability of a false positive, which makes the reward a stochastic variable [Gun16]. Given the critical importance of fast computation in real-time applications, especially in robotic systems, we need to explore methods that balance speed and solution quality effectively.

In this context, **exact optimization** methods such as those offered by solvers like Gurobi and SCIP (commonly used for solving orienteering problems) are known for their precision in solving Integer Linear Programming (ILP) problems. However, while these methods can guarantee an optimal solution, their high computational requirements make them less practical for real-time applications where solutions must be computed in seconds.

To meet the real-time demands of systems like MAGICIAN, we are also considering **heuristic methods**, which prioritize speed and simplicity. Techniques such as nearest neighbour, greedy algorithms, or local search generate approximate solutions more rapidly. Although these methods may not always yield the optimal solution, they are more suitable for real-time decision-making due to their ability to produce sufficiently good solutions within stringent time constraints.

Additionally, **genetic algorithms** (GAs) and other evolutionary algorithms (EAs) present a promising alternative. These algorithms, inspired by natural evolution, excel at exploring large solution spaces and often find near-optimal solutions within reasonable time limits. Their parallel nature makes them particularly attractive for rapid computation in real-time applications.

Given the need for both speed and solution quality, we are exploring a hybrid approach

that combines heuristic methods with evolutionary algorithms. This approach leverages the speed of heuristics and the robust solution quality of genetic algorithms, allowing us to achieve quick and reliable results within the stringent time constraints of our system. By integrating these strategies, we aim to optimize the cleaning robot's path while ensuring efficiency and responsiveness in real-time applications.

4.3.4.1 MODEL EXTENSIONS

The problem lends itself to useful generalisations. A first useful generalisation is the so-called Team Orienteering Problem (TOP) [Ar07], in which several paths are synthesized in parallel and executed by different agents. In this scenario, the reward is earned only during the first visit to a node, incentivizing the algorithm to find non-intersecting paths. This is particularly relevant for our work with multiple robotic manipulators operating simultaneously. When robotic arms are involved, we must also account for the possibility of collisions. This can be mitigated by either imposing a separation of the time windows during which transitions in potential conflict occur or partitioning the graph to minimize intersections, assigning each subgraph statically to a single agent.

Multi-Agent Systems represent one of the most important extensions of this problem. By incorporating multiple robotic agents (e.g., cleaning robots or manipulators), the system can execute parallel processes, significantly improving efficiency and reducing overall completion times. This extension aligns with the Team Orienteering Problem, as it allows for the optimization of multiple paths for multiple agents, ensuring an optimal distribution of tasks while preventing collisions. Collision avoidance can be achieved through dynamic coordination, further enhancing the performance of the system.

Additional model extensions can further improve the problem's applicability to more complex, real-world scenarios:

- Confidence Intervals for Sensing Outputs: By integrating uncertainty into the model, particularly regarding the reliability of defect identification, we can better prioritize tasks. This approach would involve adding confidence intervals for sensing outputs to ensure that tasks are selected based on the most reliable data.
- Variable Cleaning or Repair Times: Instead of assuming a fixed time to repair or clean each defect, the model could be extended to allow for variable times. This would enable a cost-benefit analysis, helping the system decide how long to stay at a defect based on its severity and the expected improvement in repair quality.
- Dynamic Learning Mechanisms: Over time, the model could incorporate learning mechanisms, adapting based on real-time feedback regarding travel speeds, cleaning times, or repair success. Machine learning techniques could further refine route planning and task prioritization, enabling the system to evolve with changing conditions.
- 3D Distance Calculations: Currently, the model assumes that defects are mapped on a 2D plane. However, if actual 3D distances better reflect travel times, incorporating these calculations into the model could improve accuracy.

- Stochastic Extensions: The problem can also incorporate stochastic elements, such as probabilistic travel times or uncertain repair outcomes. This would align the model with the Stochastic Orienteering Problem (St-OP), which is valuable for managing uncertainty and maximizing expected profit.

Moreover, variations like the Clustered Orienteering Problem (COP), where defects are grouped into clusters, could also simplify the problem, allowing the system to focus on clusters of defects rather than individual ones, as suggested by Kant et al. (2022).

By implementing these extensions, including the Team Orienteering Problem framework and multi-agent coordination, we can significantly improve both the efficiency and accuracy of the system while ensuring that operations such as robotic cleaning or manipulation are carried out optimally in real-world, complex environments.

4.4 MOTION PLANNING MODULE

As clear from Figure 10 and from the discussion above the execution of the primitives that move the robot between two locations and, to a lesser extent, of the DMPs adopted for defect removal, have to consider the possible presence of humans because the working environment is promiscuous.

Our approach relies on the presence of a module that predicts the motion of the human for the next 1.5/2 s. If a possible accident is foreseen, the approach modifies the trajectory choices to mitigate the problem.

We are currently considering two different approaches:

- Roadmap manipulation
- Replanning.

At the time of this writing, we are still in the stage of evaluating different approaches to find the most suitable to our application. In the write-up below we report the main techniques that we are considering.

4.4.1 ROADMAP MANIPULATION

The starting point is a roadmap, which can be built joining the locations as discussed above. We can enrich the roadmap with other “transition points” with the sole purpose of facilitating the motion between two different locations.

The presence of a human and her/his predicted motion makes some of the arcs temporarily unusable because of the possible collisions. We can deal with this problem by adapting the orienteering problem to operate with time varying constraints and time windows.

The idea is that the travel time of an arc τ_{ij} depends on the time in which the arc is travelled [Ver14] and each node can be visited within a time frame [Ver13].

This approach can be set-up as an Integer Linear Program, but very efficient heuristics have been proposed in the literature. The most demanding activity is the query to decide

when an arc is potentially interested by a collision. We can take this decision in a "lazy" way, i.e., while the exploration is being made [Boh01]. We are currently trying to adapt the algorithm in [Ver13] to this modality.

A different approach is based on the technique shown in [Hu22]. The idea is that the PRM is extended with temporal information. The idea is that we start from a PRM and then, based on the presence of moving obstacles, we decide a time interval in which each node is available. Then, during the query phase, we apply a variant of the A* algorithm with time-varying costs.

4.4.2 REPLANNING

In this case, we assume that the detection of human motion takes place after trajectory connecting two points has been decided. In this case, we can apply directly state-of-the-art model based [Ole24] or neural based [Wan24] approaches to design the trajectory that joins the two points avoiding accidents.

Another possibility is to use an approach similar to that in [Jai04]. We create our PRM using only the static information. Then, while we explore the graph, we can make a lazy evaluation of possible collisions. When we find one, we can try to reconstruct a different strategy connecting the point before and the point after the collision by using RRT.

Finally, we are looking at reactive versions of RRT and RRT*, which can be found respectively in [Cef19] and [Ot16]. In this case the trajectory is changed using very efficient solutions to account for the presence of obstacles. The reactive behaviour is easy to implement but cannot take advantage of the prediction of the human motion, and the result has potentially a lower performance. Indeed, the robot is forced to move slowly to avoid a possible collision that can occur at any time, while exploiting the prediction allows for an anticipatory behaviour and move along trajectories that are reasonably collision-free.

4.5 PLANNING AND SCHEDULING FOR THE SR

As regards the scheduling activities for the SR, we are in the typical case for which we need a correct balance between exploitation and exploration. On the one hand, we seek the defects in the area where they are most likely to be found based on the historical data (exploitation), on the other we need to explore the car-body in search of new areas where defects could be found (e.g., due to changes in the condition of the production phases preceding the analysis).

The two phases have to be properly orchestrated. Indeed, given the fixed amount of time available, the exploitation phase will have to account for the following facts.

1. The visual inspection has to be concentrated in the areas where, based in the historical data, the defects are most likely to be found. In addition, defects should receive a larger priority according to the area where they are;
2. The tactile inspection has to be concentrated in areas that are not easily reachable

by the camera (e.g., the area underneath the doors or on their side);

3. Some slack time has to be reserved for tactile inspection for defects that are classified as uncertain during visual inspection.

The exploration phase should be very efficient in discovering potentially new areas with defects and at avoiding areas that have been carefully analysed during the exploitation phase.

In our first development run, we have decided to statically partition the duration of the two phases and use specific solutions for each of them. In the future we plan to explore a more dynamic interaction between the two phases.

4.5.1.1 EXPLOITATION

For the exploitation phase, a possibility is to use again a variation of the orienteering methods described above. Based on the previous history, the system can identify a number of locations associated with a high level of criticality and with a probability that a defect will materialise.

We can again construct a roadmap joining all the different locations distinguishing between fly-over phases, in which we move very quickly between two areas, and inspection phases, where we inspect the area remaining at a fixed distance from the car-body. The framework is very similar to the one that we have described for defect removal. The only remarkable difference is that for the inspection phase we can utilize two types of sensors (visual and tactile). The implications are illustrated below.

4.5.1.2 INTEGRATING DIFFERENT SENSORS IN SR OPERATION SCHEDULING

Let D be a defect at a given location and let V denote the simple use of the visual sensor, and T denote the use of the tactile sensor. Suppose we know, from the previous runs, a good estimate of the probability $P(D) = L$ that a defect will appear in that location. If we know

- the accuracy and the probability of a false positive for the visual sensor $P(V|D) = M$ and $P(V|\overline{D}) = m$,
- the accuracy of the tactile sensor $P(T|D) = N$ and the probability of a false positive $P(T|\overline{D}) = n$,
- assuming that the two readings are independent,

we can estimate the probability that the defect will appear to each sensor or to their combined use by respectively using one of the following equations:

$$P(\mathrm{D}|\mathrm{V}) = \frac{P(V|D)P(D)}{P(V|D)P(D) + P(V|\overline{D})P(\overline{D})} = \frac{ML}{ML + (1-L)m}$$

$$P(\mathrm{D}|\mathrm{T}) = \frac{P(T|D)P(D)}{P(T|D)P(D) + P(T|\overline{D})P(\overline{D})} = \frac{NL}{NL + n(1-L)}$$

$$P(\mathrm{D}\,|\mathrm{T}\,\mathrm{V}) = \frac{P(V|D)P(T|D)P(D)}{P(V|D)P(T|D)P(D) + P(V|\overline{D})P(T|\overline{D})P(\overline{D})} = \frac{NML}{NML + nm(1-L)}$$

It is easy to see that the combined use of the tactile sensor combined with the visual sensor gives advantages over the latter only if $N \geq n$. As an example, we plot the ratio $\frac{P(D|TV)}{P(D|V)}$ as a function of $P(T|\overline{D})$, for $M = 0.5, m = 0.1, N = 0.4, L = 0.8$.

As intuitive, we have convenience in using the second sensor only if its accuracy is greater that the probability of false negative (see Figure 122). The degree in which this choice can be convenient depends on the different probabilities and can be estimated online based on the frequencies of the different events.

Generally speaking, we have convenience in using multiple sensors iff:

$$P(D|\overline{\mathrm{TV}}) \leq P(D|\overline{\mathrm{V}}) \leq P(D) \leq P(D|V) \leq P(D|VT)$$

It can be shown that this is verified if $P(V|D) \geq P(V|\overline{D}) \wedge P(T|D) \geq P(T|\overline{D})$. In essence, we are saying that the presence of a positive answer from each sensor improves the knowledge on the probability of the presence or of the absence of a defect. The computation shown above allows us to compute the amount of the improvement.

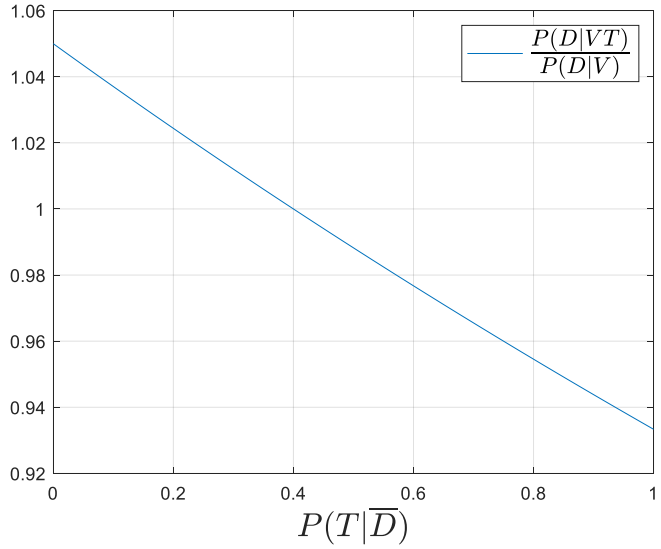


Figure 12: Ratio of the relation between the probability of using visual + tactile sensor over using only the visual sensors. The ratio is computed for different values of the false negatives.

Clearly, the processing time and the probability associated with the different locations depends on this choice. At the beginning of each work cycle, we can compute for each location the probability that we will identify the defect with the sole use of the visual sensor. If this probability is lower than a desired threshold, we can opt for the combined use of vision and tactile sensor. This choice has a direct impact on the choice of the parameters that are used to set up the orienteering problem.

4.5.1.2.1 EXPLORATION

As regards the exploration phase, one of the most promising approaches that we are evaluating is based on computational geometry. In the past years, the UNITN group has proposed the application of Llyod based methods to many distributed control problems [Bol22], including rendezvous, dynamic coverage, motion of agents with line-of-sight preservation constraints. The idea is to decompose the environment into Voronoi cells and implement a distributed gradient descent method choosing a cost function fitted to our purposes. For instance, if the desired task is exploration, we can utilise a cost function that penalises the places that robot visited already. The approach can be used for a single agent, but it generalises nicely to multi-agent systems without suffering deadlock or livelock conditions. We are now adapting the idea to the manipulator setting, where issues like possible interference between the different kinematic structures take centre stage.

5 MOTION GENERATION AND ACTIVE SENSING

5.1 INTRODUCTION

As we discussed above, once the SR reaches an area of potential interest, it collects information on the potential presence of defects. We use the term “active sensing” to denote feedback control strategy that guise the system in order to maximise the amount of information collected. In some sense, we can consider the feedback control strategy as a means to generate feedback controlled motion primitives that execute the sensing task. In this section, we mention the strategies we are putting in place to implement this idea.

5.2 STATE OF THE ART

Active sensing is the problem of control design for carrying out sensing tasks [Kreu05, Cai09]. Applications are in a large number, and include object classification/recognition [Den02, Arb99] and next viewpoint selection [Vaz01].

Generally speaking, we can model an active sensing algorithm as going through four different phases, Figure 13. Based on the previous information we update our belief state (e.g., our best guess on where are the defects). The typical way beliefs are created and updated are through filtering techniques. Kalman filters can be used effectively when the posterior distribution is known to be Gaussian [Leu06]. In our case, we cannot make any such assumption. Therefore, we will adopt general Bayesian filtering [Mil15] because it allows us to consider non-Gaussian Distribution and easily integrate the physical information on the dynamics of the robot performing the search.

The step B is to evaluate the information that can be collected given our sensing state and our state belief. Given a state, the possible ways to evaluate the utility of a

measurement can be different. Very informative is for our case the Fisher information matrix, which quantifies the ability of a random variable associated with a measurement to estimate an unknown parameter [Em98]. In some sense, the fisher information predicts where the derivative of the expected signal to the variance of the noise is high producing more salient data.

The next step is to decide a control action that drives robot trajectories following the distribution maps. Following [Mil15], we will apply ergodic control as described in the next section.

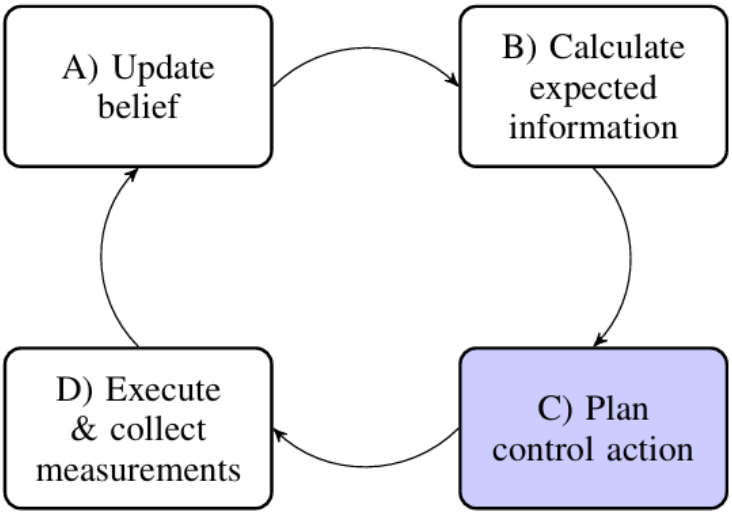
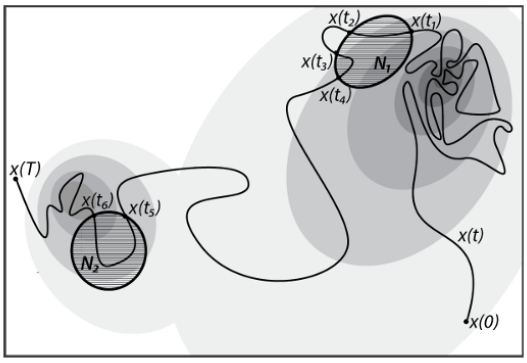


Figure 13: Typical phases of information Based sensing algorithm (courtesy [Mil15])

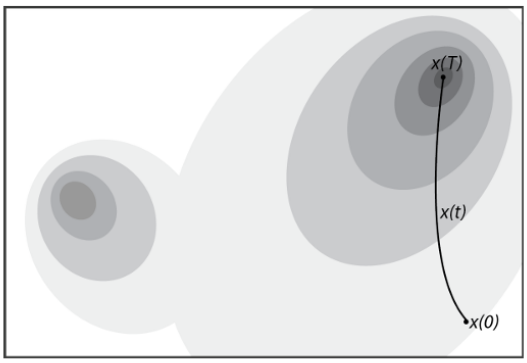
5.3 ERGODIC CONTROL

In addition to the "classic" approaches described earlier, we are approaching the problem of defect search adopting an active sensing paradigm based on ergodic control [Mil15]. By using this the exploration and exploitation phases are strongly harmonised.

Ergodic control is a technique that determines a motion policy for a robot such that statistics given by a prior distribution and the spatial statistics averaged through time converge.



(a) Ergodic trajectory



(b) Information maximizing trajectory

Figure 14: Example of ergodic control (a) as opposed to information maximisation (b). [Courtesy [Mil15])

It is telling to look at example in Figure 14. In the bottom part, we see that the trajectory seeks the points with the maximum information, whilst in the top part (ergodic control) the objective is to fill the space according to the prior distribution. In our case the prior distribution is given by the historical probability of finding defects.

Given our search domain X

$$\mathrm{C(x)} = \frac{1}{\mathrm{T}}\int_0^{\mathrm{T}} \delta\big(x - x(t)\big)dt$$

with $\delta(.)$ being the Dirac delta, represents the spatial statistics of the process $x(t)$. Our goal is to find a control law to make these statistics converge to the estimated probability density function EID(x).

If we represent $C(x)$ by its Fourier coefficients $c_k\big(x(t)\big)$, and EID(x) by its Fourier coefficients ϕ_k, we can express the deviation of the trajectory x(t) from erogidicity as [Mat11]

$$\mathcal{E}(\boldsymbol{x}(t)) = \sum_{\boldsymbol{k}=0\in\mathbb{Z}^n}^{\boldsymbol{K}\in\mathbb{Z}^n} \Lambda_k[c_{\boldsymbol{k}}(\boldsymbol{x}(t)) - \phi_{\boldsymbol{k}}]^2$$

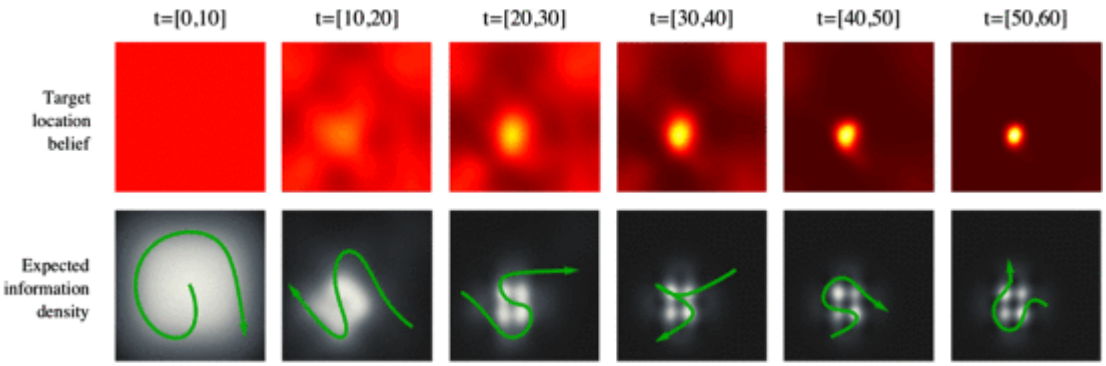
.

The use of the Fourier coefficients makes the distance from ergodicity differentiable in $x(t)$ allowing us to set up an optimal control problem where the cost function is given

by:

$$J(\boldsymbol{x}(t)) = \underbrace{\gamma \mathcal{E}[x(t)]}_{\text{ergodic cost}} + \underbrace{\int_0^T \frac{1}{2} u(\tau)^T R u(\tau).}_{\text{control effort}}$$

Classic active sensing solutions, divide the exploration and the exploitation phase switching between two cost functions. The application of ergodic control allows us to treat the two phases in the same way. The only differentiation is in the way we collect measurements. During the exploration phase, we choose our sampling policy to prioritise coverage, while in the exploitation phase we increase our sampling rate in the proximity of the hotspots. We report an example from [Mil15]



In the example we show the location of a 2D target. In the top row, the figure shows the update of the belief state, whilst in the bottom row we show the trajectory computed by ergodic control. The trajectory allows us to update the information and progressively refine the pdf EID(x) where the target can be found.

The application of ergodic control is not straightforward: it requires some research efforts that we are putting in place. For this reason, we will first adopt the more classic approach based on orienteering and Llyod-based navigation described above.

5.4 MOTION GENERATION MODULE

MAGICIAN control architecture will employ the CartesI/O cartesian control framework developed at IIT [Laurenzi2019]: it allows the untrained user to perform complex motion tasks with robotics platforms by leveraging a simple, auto-generated ROS-based interface.

Contrary to other motion control frameworks (e.g., ROS MoveIt!), CartesI/O focuses on the execution of Cartesian trajectories that are specified online rather than planned in advance. Moreover, it addresses the problem of generating such motions within a hard Real-Time (RT) control loop.

As highlighted in the picture below, CartesI/O:

- Provides a uniform way to programmatically interact with a Cartesian controller;

- Automatically generates a complete ROS API for sending references to the controller;
- Allows the use of the ROS-based API also in the case that the *solver* is running inside a real-time thread.

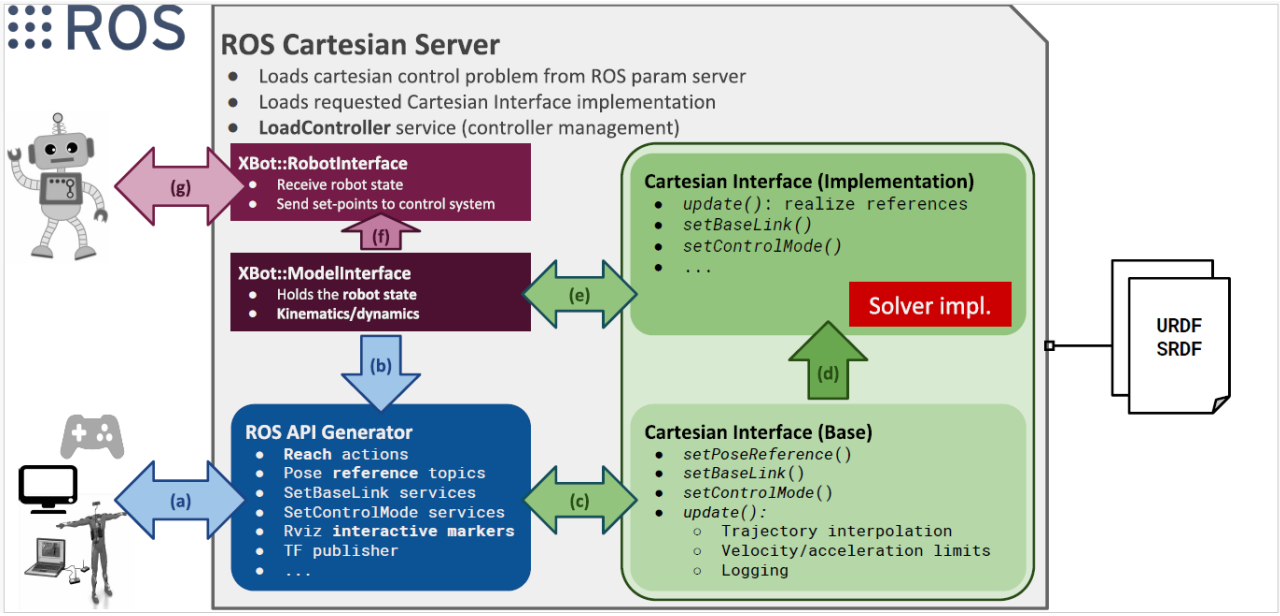


Figure 15: The main components of the software architecture with its motion modules.

5.5 TASK AND HUMAN PRESENCE DRIVEN MOTION/IMPEDANCE MODULATION PRINCIPLES

An example of what is achievable with CartesI/O motion generation and the XBot middleware is a *Cartesian Impedance modulation* principle, as anticipated in Section 3.5. Acting in the task space, the control law formula is now the following:

$$\tau = J^T \left(K \left(x_d - x \right) + D \left(\dot{x_d} - \dot{x} \right) + f_{ff} \right)$$

where K is the Cartesian stiffness gain, D is the Cartesian damping gain, x, x_d are the actual and the desired Cartesian tool control point pose, respectively, and f_{ff} an optional *feed-forward* force.

Acting in the Cartesian space, it is possible to modulate the manipulator compliance in specific Cartesian direction, tackling the physical interaction in a more convenient and flexible way. As an example, for a grinding/polishing task where the manipulator follows a rigid surface, the stiffness along the normal direction can be lowered, assuring to keep the contact with the object, while still following with a high precision the desired path in the other directions. Figure 16 introduces such an example, with a mobile collaborative robot following a curved surface in a simulated environment (Gazebo). The plugin, being implemented with the XBot tools, is ready to be used in a real scenario.

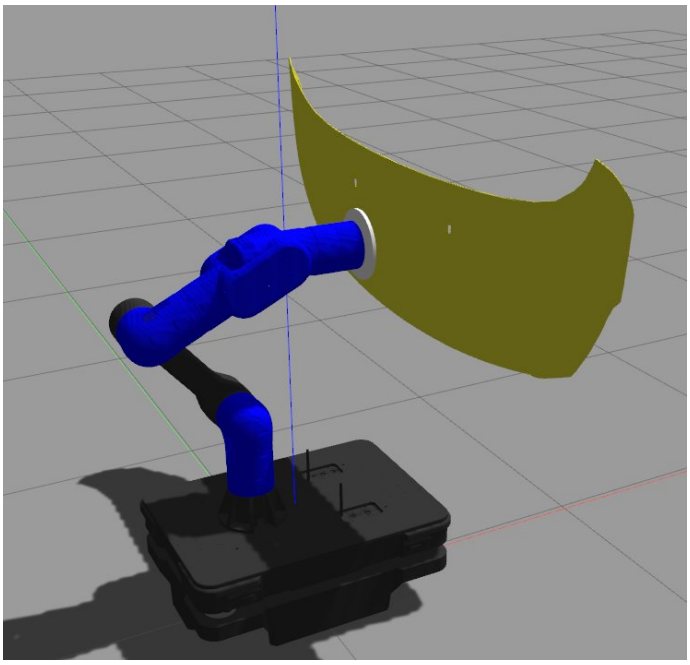


Figure 16: A mobile robot performing a surface following task in the Gazebo simulation.

In general, with the developed modules, the stiffness (and damping) gains can be set online, depending on the status of the task in execution. Eventually, such parameters would be set automatically, with techniques of variable impedance modulation, based on the status of the tasks and on the task requirements [Bertoni2022].

5.5.1 SELF-COLLISIONS AVOIDANCE MODULE

The above-mentioned software framework employed, allowed to easily integrate in the important modules for the optimization and safety of the robot generated motions. One such capability of the CartesI/O motion generation module is a self-collision avoidance system. Implemented as a constraint in the stack of tasks defined, such module takes into consideration the collision meshes of the robot model (usually simpler shapes that envelop the real robot meshes). During the robot motion, the module checks online for eventual collisions among the robot's link and modifies accordingly the trajectories for a safe robot motion.

6 CONCLUSIONS

The algorithmic methodologies and technological solutions presented in this report represent the foundation tools considered and are currently under development for the realization of the robotic system of MAGICIAN.

The development of these tools is expected to continue in the next period as their implementation and testing on the MAGICIAN robotic platform will progress. Further extensions and upgrades in the functionalities of the robotic solution will be performed based on the outcome of their implementation and validation in the field during the execution of the defect detection and reworking use case tasks.

The tuning of the control tools will be carried out considering the performance measured and the requirements imposed by the use case tasks. For example, additional control methodologies and/or tuning may be necessary to address issues imposed by the interaction and end-effector intrinsic vibrations.

The physical interaction parameters of the robotic platform such as impedance settings and contact force regulation for satisfying the necessary task performance will also form and interesting topic of the follow up activities.

Similarly, adaptation and upgrades on the task planning and scheduling tools will be guided by the results obtained and observed execution efficiency.

Finally, developments on the interfaces of the different tools will be necessary to facilitate their integration within the overall software and control framework and the communication among the different algorithmic and technological components.

7 REFERENCES

[Chinello2012] Chinello, Francesco, et al. "A three DoFs wearable tactile display for exploration and manipulation of virtual objects." 2012 IEEE Haptics Symposium (HAPTICS). IEEE, 2012.

[Pacchierotti2017] Pacchierotti, Claudio, et al. "Wearable haptic systems for the fingertip and the hand: taxonomy, review, and perspectives." IEEE transactions on haptics 10.4 (2017): 580-600.

[Kappassov2015] Kappassov, Z., Corrales, J. A., & Perdereau, V. (2015). Tactile sensing in dexterous robot hands. Robotics and Autonomous Systems, 74, 195-220.

[Seminara2019] Seminara, Lucia, et al. "Active haptic perception in robots: a review." Frontiers in neurorobotics 13 (2019): 467142.

[Pape2012] Pape, L., Oddo, C. M., Controzzi, M., Cipriani, C., Förster, A., Carrozza, M. C., & Schmidhuber, J. (2012). Learning tactile skills through curious exploration. Frontiers in neurorobotics, 6, 6.

[Prattichizzo2013] Prattichizzo, D., Chinello, F., Pacchierotti, C., & Malvezzi, M. (2013). Towards wearability in fingertip haptics: a 3-dof wearable device for cutaneous force feedback. IEEE Transactions on Haptics, 6(4), 506-516.

[Laurenzi2023] Arturo Laurenzi, Davide Antonucci, Nikos G. Tsagarakis, Luca Muratore, “The XBot2 real-time middleware for robotics,” Robotics and Autonomous Systems, Volume 163, 2023, 104379, ISSN 0921-8890, https://doi.org/10.1016/j.robot.2023.104379.

[Muratore2020] L. Muratore, A. Laurenzi, E. Mingo Hoffman and N. G. Tsagarakis, "The XBot Real-Time Software Framework for Robotics: From the Developer to the User Perspective," in IEEE Robotics & Automation Magazine, vol. 27, no. 3, pp. 133-143, Sept. 2020, doi: 10.1109/MRA.2020.2979954.

[Laurenzi2019] A. Laurenzi, E. M. Hoffman, L. Muratore and N. G. Tsagarakis, "CartesI/O: A ROS Based Real-Time Capable Cartesian Control Framework," 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 2019, pp. 591-596, doi: 10.1109/ICRA.2019.8794464.

[Hogan1985] N. Hogan, “Impedance control: An approach to manipulation: Part I-theory". Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME, 107(1). https://doi.org/10.1115/1.3140702

[Muratore2023] L. Muratore, A. Laurenzi, A. De Luca, L. Bertoni, D. Torielli, L. Baccelliere, E. Del Bianco, N. G. Tsagarakis, “A Unified Multimodal Interface for the RELAX High-Payload Collaborative Robot” Sensors 2023, 23, 7735. https://doi.org/10.3390/s23187735

[Bertoni2022] L. Bertoni, L. Muratore, A. Laurenzi and N. G. Tsagarakis, "Task Driven Online Impedance Modulation," 2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids), Ginowan, Japan, 2022, pp. 865-872, doi: 10.1109/Humanoids53995.2022.10000215.