Chapter 4

Algorithms for Removing Asbestos

By Tobias Haschke, Sami Charaf Eddine, Kamal Mohy El Dine, Siddharth Maraje, Fabian Herb, Oriol Orra, Diego Fogliacco, Tim Detert, Claudia Cornely, Daniel Martin, Albert Fullana Puig, Ramón Carrasco Jimenez, Matthias Rabel and Jean-Christophe Fauroux

DOI: 10.1561/9781680837155.ch4

Published in *Robots to Re-Construction – The Roadmap to Robotized Asbestos Removal* by Burkhard Corves, Tobias Haschke and Mathias Hüsing (eds.). 2021. ISBN 978-1-68083-714-8. E-ISBN 978-1-68083-715-5.

Suggested citation: Tobias Haschke, Sami Charaf Eddine, Kamal Mohy El Dine, Siddharth Maraje, Fabian Herb, Oriol Orra, Diego Fogliacco, Tim Detert, Claudia Cornely, Daniel Martin, Albert Fullana Puig, Ramón Carrasco Jimenez, Matthias Rabel and Jean-Christophe Fauroux 2021. "Robotic Systems for Removing Asbestos" in *Robots to Re-Construction – The Roadmap to Robotized Asbestos Removal*. Edited by Burkhard Corves, Tobias Haschke and Mathias Hüsing. pp. 209–357. Now Publishers. DOI: 10.1561/9781680837155.ch4.

now the essence of knowledge

Bots2ReC Robots to Re-Construction

Funded by the European Union

4.1 User Interface

Deliverable 6.1
User Interface for Tele-Operation

Authors: Tobias Haschke, Matthias Rabel

4.1.1 Deliverable in the Context of the Work Program

The User Interface for Tele-operation (UITO) (Task 6.3) is the point of contact for the end user of the robotic system that is developed within Bots2ReC. It is the instance for reducing complex robotics behavior into intuitive, safe and comfortable ways of interaction. Therefore it is embedded into work package 6 where the high-level task planning and the necessary algorithms are developed. The UITO is communicating with the CPCS of the robotic system and creates the first solution for interacting with the robotic system via the self-written control system. It is the first task within WP6 and requires a first system architecture for the basic tele operated movements. Along the development of the UITO it is also necessary to define the first levels of the CPCS for avoiding time consuming adapting operations in the ongoing project. Beside this, it is also necessary to have supervision for the whole project because the user interface is going to be extended in Task 6.4. According to this schedule it is important to create a UI that is easy to maintain and to extend during the ongoing project.

4.1.2 Methods and Work Carried Out

For developing the UITO a systematical approach of designing the software architecture is chosen. The documentation of this process is done with the Unified Modeling Language (UML) which is state of the art for displaying the content of software solutions vivid and comprehensive. The most important advantage of using UML is that the results are understandable for every consortium member even if they do not have further expertise in computer science. Out of this a basic way for discussions and feedback loop is found. Referring to the common workflow for using UML for software development, in a first step the identification of the stakeholders for the project was carried out. Afterwards the targets for the UITO are defined and corresponding Use-Cases are written. Finally the necessary activity diagrams are confirmed for solving the Use-Cases with a well-defined flow of data. During this process multiple feedback loops are used for informing the affected consortium members and for finding a consensus of the right solution. Especially the feedback of the end user Bouygues is essential as they have the expertise in removing asbestos and can provide important details about the conditions on a construction site. The feedback loops are organized via a Microsoft Word Document where the functionality of comments is used for asking questions, setting remarks and sending responses. Beside the documentation of the software architecture there are also discussions about the hardware for displaying and interacting with the robotic unit. These discussions are also part of Task 6.3. For the development of the UITO the hardware definitions of Task 3.1 and the technical requirements of Task 3.2 are taken into account. As the partner Indurad is developing the UITO they also use the input from Task 4.2 for the environmental perception and the first results from Task 6.1.

4.1.3 Identification of Requirements

The identification of requirements is a classical approach to define the content of the UITO on a textual basis. It is a solution to define the use-cases with words so that every participant of the discussion can follow it easily even without deeper knowledge of computer science. At first the stakeholders of the software solution are defined. Out of this the main targets of the UITO and the interaction with the different roles of the stakeholders are outlined. The next steps are models for the systems behaviour with use-case diagrams and corresponding activity diagrams. The Unified Modelling Language (UML) is chosen to document the software product. The typical workflow for defining these aspects can be seen in Figure 4.1.

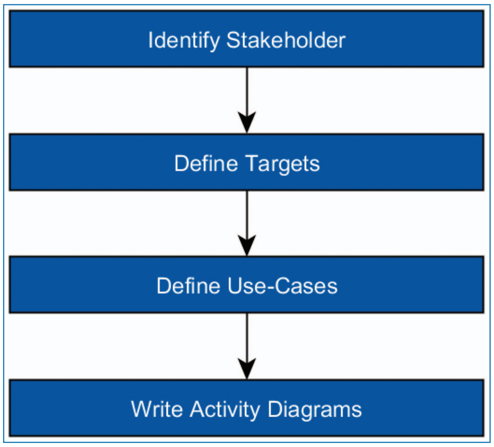


Figure 4.1. Steps for the development process.

4.1.3.1 Stakeholders

The different stakeholders represent the groups of people or systems that are affected by the user interface for tele-operation. For this system there are the following stakeholders:

- Construction worker (User)
- Developer (Engineers of the Bots2ReC Consortium)
- Construction Company (Company that uses the product)
- Product Owner (Company that sells the product)

4.1.3.2 Targets

These targets are meant to be top-level-requirements that need to be fulfilled by the UITO. They are solution-neutral, consistent and relevant for the project.

4.1.3.2.1 Template for Targets

For the definition of targets the following template was used. It is common in the description of tasks in the understanding for UML and is enough to define abstract targets without mentioning the solution.

ID	###
Target	What should be achieved?
Stakeholder	Which stakeholders are linked to this target?
Effect on Stakeholders	Which effects can be expected for the stakeholders?
Boundary conditions	Are there fix boundary conditions for this target?

It is essential to define the stakeholder who is linked to this target and how he is affected by it. Out of these ideas the targets can be used to understand the influence of the software product more precise.

4.1.3.2.2 List of Targets for UITO

ID	**T-1**
Target	Tele-operate the robotic unit inside of construction site
Stakeholder	• User
Effect on Stakeholders	The User does not need to move the robotic unit manually
Boundary conditions	The CPCS can only convert the user input according to the abilities of the mobile platform. Active collision avoidance is also necessary.

ID	**T-2**
Target	Tele-operate the robotic arm for asbestos removal tasks
Stakeholder	User
Effect on Stakeholders	• The User does not need a protective suit any more • The User is not working in a hazardous environment
Boundary conditions	• The Robotic Unit needs to be stable during the whole process • The asbestos removal task can be fulfilled completely (process parameters as necessary)

ID	**T-3**
Target	Get feedback about the robotic system that enables the tele-operation
Stakeholder	• User
Effect on Stakeholders	• The user has a vivid and intuitive feedback of the actual state • The robotic unit can be tele-operated more precisely
Boundary conditions	• The connections between the systems (Wi-Fi, Ethernet, etc.) have data limits that are limiting the amount of data for the interface

ID	T-4
Target	Ensure a pure safe tele-operational process for human and machines
Stakeholder	• User
Effect on Stakeholders	• The User is not able to crash the robotic unit • The Robotic Unit cannot be harmed by wrong input commands of the user
Boundary conditions	–

ID	T-5
Target	Give feedback about the removal result of the robotic unit
Stakeholder	• User

ID	T-5
Effect on Stakeholders	• The User needs to get feedback about the removal success
Boundary conditions	• The connections between the systems (Wi-Fi, Ethernet, etc.) have data limits that are limiting the amount of data for the interface

ID	T-6
Target	Enable the user to manage errors and issues
Stakeholder	• User
Effect on Stakeholders	• The User do not need to enter the hazardous environment for solving expected issues like sensor restarts
Boundary conditions	• The performed actions are limited to the low level access to every machine (e.g. for restarting devices)

ID	T-7
Target	Create a user interface that is easy to use and easy to learn
Stakeholder	• Construction Company
Effect on Stakeholders	• The Construction Company does not need much time in teaching employees (User) for the automated asbestos removal
Boundary conditions	• As easy as possible but as complex as necessary! A common understanding of robotics behaviour is mandatory

ID	T-8
Target	Create a user interface that is easy to maintain and to extend
Stakeholder	• Product Owner
Effect on Stakeholders	• The Product Owner can extend the user interface with new functionalities and maintain it without huge effort
Boundary conditions	• Properties of the used programming language is the limiting factor

4.1.3.3 Use-Cases

Use-Cases describe with normal words (no technical terms) the interactions of actors with the system. Actors are often members of the identified stakeholders as e.g. the User itself. Every Use-Case can be defined with nouns and a single verb. It is solution-neutral and only describes **what** needs to be done by the system and **not how** it is done.

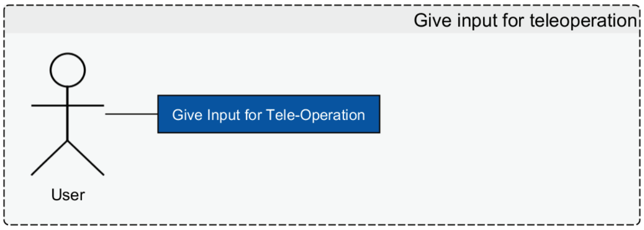
4.1.3.3.1 Template for Use-Cases

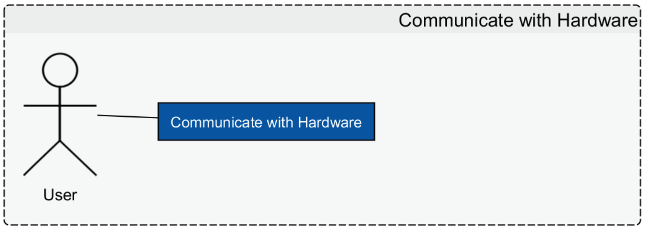
For a clean and readable structure of the Use-Cases the following template is used. It contains all necessary information about the Use-Case and can be read quite easy.

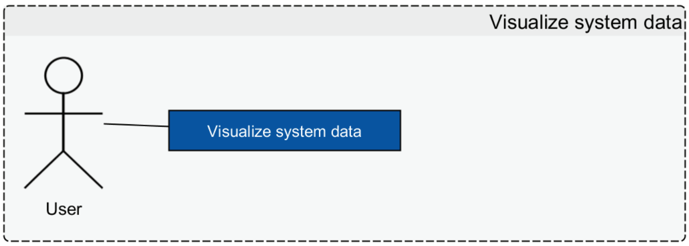
ID	###
Use-Case	<Verb> <Noun>
Brief Description	Brief description of the Use-Case
Prerequisites	What needs to be done before this Use-Case?
Results	What is the outcome of this Use-Case?
Typical Process	Steps towards the results

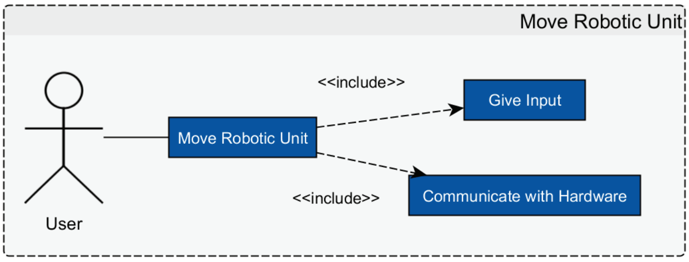
Every project member involved used this template to edit the list of Use-Cases during the documentation of possible Use-Cases.

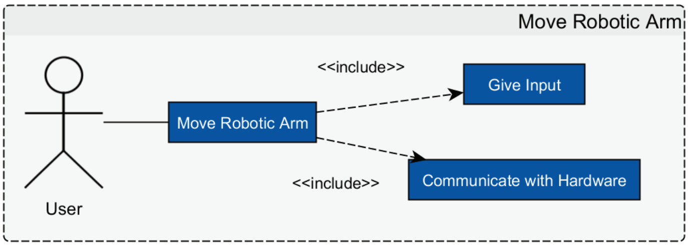
4.1.3.3.2 List of Use-Cases for UITO

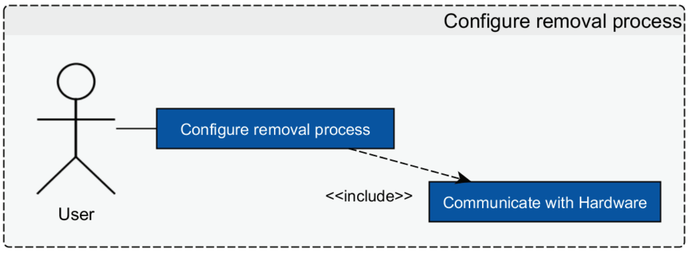
ID	**UC_001**
Use-Case	Give input for tele-operation
Brief Description	Use an input device for giving commands to the UITO
Prerequisites	The system is ready to receive tele-operational commands
Results	The commands are used for the robotic movement
Typical Process	• The commands are read into the system • They are converted to robotic commands
Graph	 Give input for teleoperation Give Input for Tele-Operation User

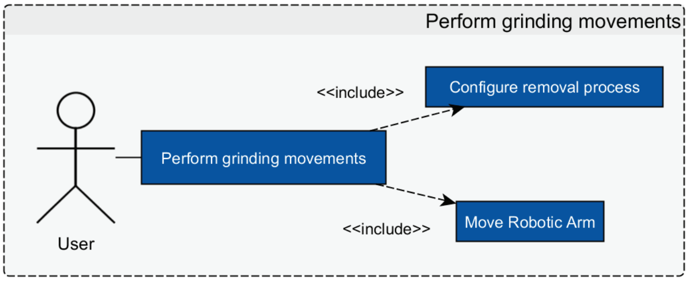
ID	**UC_002**
Use-Case	Communicate with Hardware
Brief Description	Enables the CPCS to communicate with every hardware of the robotic system (sensors, actuators, tools)
Prerequisites	The Robotic Unit is powered on
Results	The CPCS can send specific commands to the robotic unit and gets feedback about it
Typical Process	• Establish connection with the hardware • Check if everything is running as wanted • Give status about the actual situation (everything is fine, errors or warnings)
Graph	 Communicate with Hardware Communicate with Hardware User

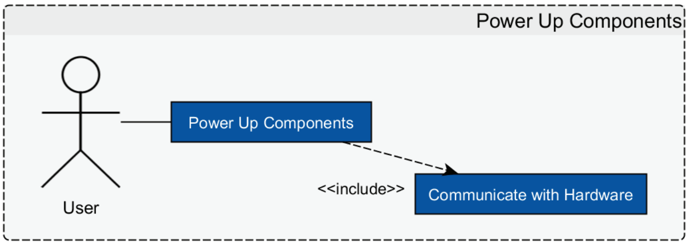
ID	UC_003
Use-Case	Visualize system data
Brief Description	The User is able to see system output on the screen of the UITO
Prerequisites	The System is powered on and connected to the UITO
Results	The User can easily see the actual State of the Robot
Typical Process	• Convert robotic data into visual information like 3D view of scenery
Graph	

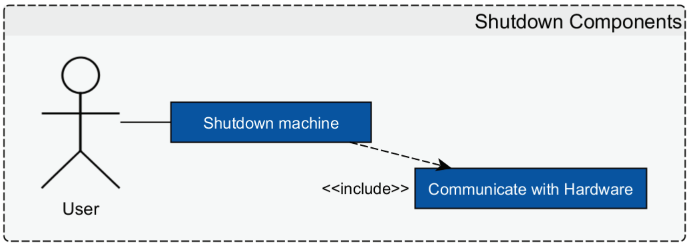
ID	UC_004
Use-Case	Move Robotic Unit
Brief Description	Move the robotic unit in x, y and theta inside of the construction site
Prerequisites	The Robotic Unit is powered on and calibrated.
Results	The Robotic Unit reaches the new position as the user expected
Typical Process	• Recognize User input • Process movement commands • Execute Commands Give Feedback
Graph	 Move Robotic Unit <<include>> Give Input Move Robotic Unit <<include>> Communicate with Hardware User

ID	UC_005
Use-Case	Move Robotic Arm
Brief Description	Move the robotic arm in space to reach walls or to look around with the camera
Prerequisites	The system is powered on and the robotic unit is calibrated
Results	The arm reaches the new position as the user expected
Typical Process	• Read Joystick Input • Convert it to arm movement • Execute motion
Graph	 Move Robotic Arm User Move Robotic Arm <<include>> Give Input <<include>> Communicate with Hardware

ID	UC_006
Use-Case	Configure removal process
Brief Description	Setup the specific process parameters for removing asbestos (disk speed, pressure, max. speed, etc.)
Prerequisites	The system is turned on and ready to be configured properly
Results	Save the configuration for the removal process (dynamically?)
Typical Process	• Open actual configuration • Allow changes Save new configuration
Graph	 Configure removal process User Configure removal process <<include>> Communicate with Hardware

ID	UC_007
Use-Case	Perform grinding movements
Brief Description	Enable the tool and the aspiration for removing asbestos
Prerequisites	The robotic Unit is in 'Ready to Remove' status
Results	The robot allows the execution of tele-operational commands with running tool and aspiration
Typical Process	• Give command for turning on tool and aspiration • Check if command is allowed • Run tool and aspiration and allow tele-operation of the arm
Graph	 Perform grinding movements <<include>> Configure removal process Perform grinding movements User <<include>> Move Robotic Arm

ID	UC_008
Use-Case	Power Up Components
Brief Description	Run the boot process of the robotic system after turning on the power
Prerequisites	The robot has a charged battery
Results	The robotic system is ready to use in tele-operation mode
Typical Process	• Wait for upcoming hardware • Solve errors if necessary
Graph	 Power Up Components Power Up Components User <<include>> Communicate with Hardware

ID	UC_009
Use-Case	Shutdown Components
Brief Description	Shutdown the robotic system without harming it
Prerequisites	The system is turned on
Results	The system is shut down without being harmed
Typical Process	• Wait for shutdown of every component • Allow to switch power off afterwards
Graph	

4.1.4 Activity Diagrams

Every Use-Case can be divided into smaller activities that enable the system to give the desired outcome. These activities and all the corresponding decisions and alternative ways can be shown in an activity diagram. There is only **one** activity diagram per Use-Case. It is a documentation of all allowed processes during the Use-Case. In the case of Bots2ReC it is more convenient to focus on the main tasks as tele-operating the robot and not on the small Use-Cases like communicating with the hardware that are necessary anyway.

Flow Charts can be used for the more vivid description of interaction between different subsystems and their network. It is also a good way to define different stages within the software solution and to manage the behaviour. As seen in Figure 4.2 there are multiple outcomes possible for the boot process. On the one hand it is possible to run into the error handling process to solve upcomming issues during the boot process or it can end up in a successful boot process.

The error handling flow is divided into two parts. The system will try to solve a detected error on its own first and asks the user for help afterwards if there was no solution. The last option of doing the recovery behaviour is a restart of the whole system that might be solved during a new boot process.

Getting the system moving is most essential for fulfilling the task of removing asbestos. The combination of tele-operating the platform and the arm are not used for now in combination. Out of this the tele-operation of the mobile platform and the arm are split into two different workflows. As the movement of the arm needs more stability than the movement of the platform itself there are different bounding conditions for each tele-operational task. Bots workflows are taken into account for the UITO.

If the user enables the system to remove asbestos there are quite a lot constraints that need to be fulfilled before being able to start the tool and the aspiration. Related to this the first step for the removal process is a well-defined question and answer system for the actual state of the robot and if the user shall be granted for starting the removal process. It is necessary that the robot is stabilized firmly for running the removal process. This is represented within the robot mode that is handled by the CPCS. After getting access to the command of running the tele-operational removal task the user is just using the workflow of the robotic arm tele-operating process that was given before.

4.1.5 Basic Information

4.1.5.1 Software Development Tools

In relation to the defined Use Cases the consortium members defined the operational tools for fulfilling the tasks. The programming language C++ is used for the programming of the logical behaviour of the robot. The user interface itself needs to provide more visual effects and complexity that can be dealt with using by mark-up languages. For the software architecture of the high control system the open source framework ROS (Robot Operating System) is used. It is based on the publish-subscribe pattern which makes it highly modular and adjustable for the use

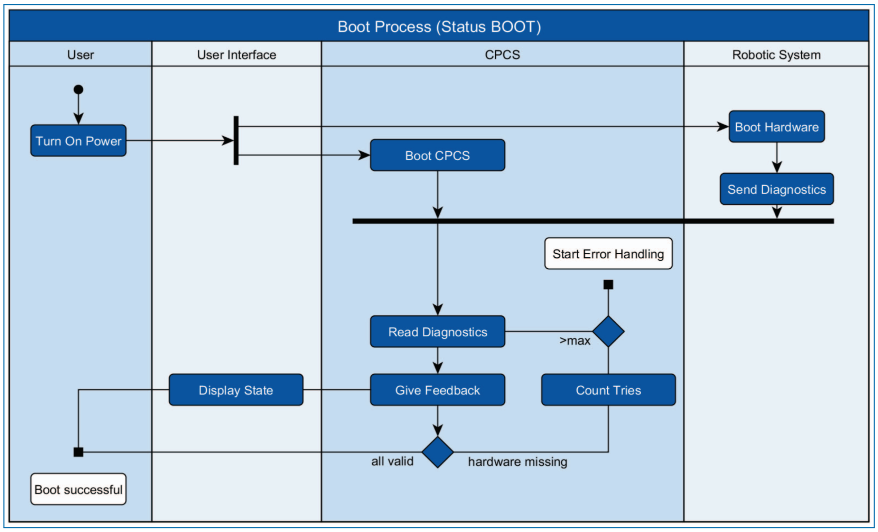


Figure 4.2. Flow Chart of the boot process.

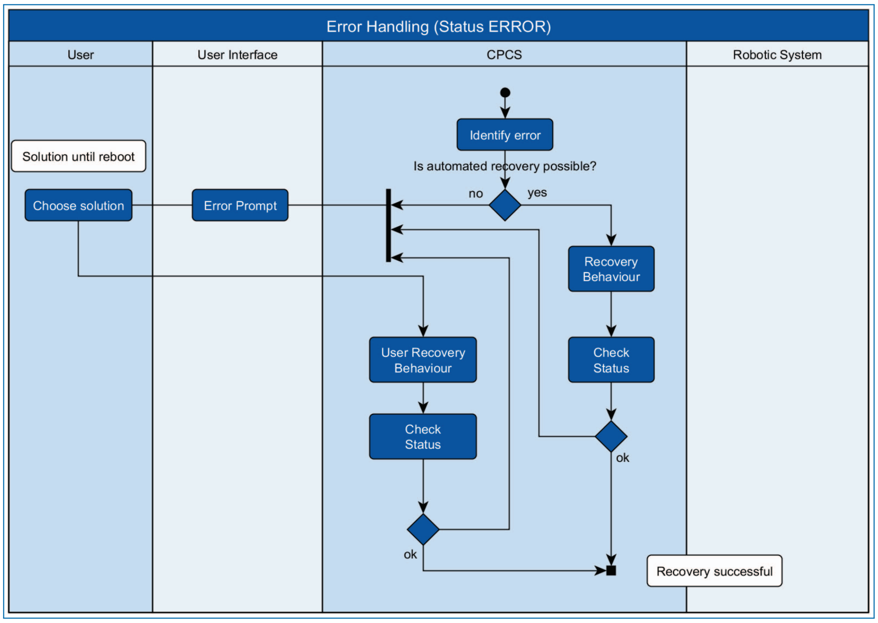


Figure 4.3. Flow Chart of the error handling.

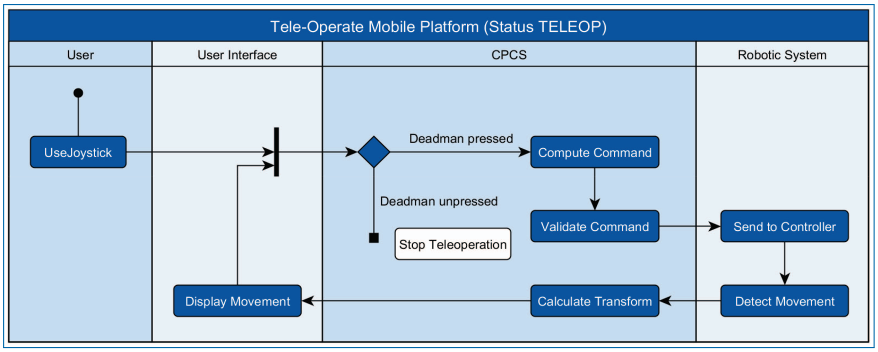


Figure 4.4. Flow Chart of the tele-operation of the mobile platform.

of different kind of robots. The common language of ROS is C++ which was one of the main arguments for choosing this framework because of its hardware-related programming language.

Because the UI is implemented by Indurad the existing framework of Indurad for user interfaces is used. It is a C++ framework exposing a QML Qt Meta Language or Qt Modelling Language interface and it is written with Qt Creator. The output is a standalone C++ Project that can be launched by an executable of ROS later on. Out of this the first integration of the framework into ROS is already manageable.

Out of this the main structure of the UITO (User Interface for Tele-Operation) is already predefined. There is the pure user interface that cares about the

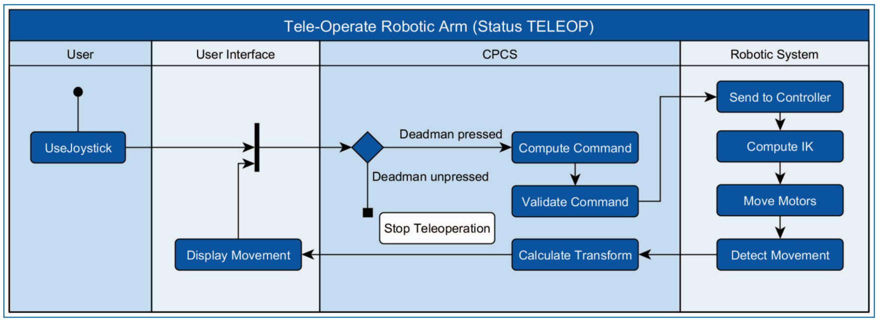


Figure 4.5. Flow Chart of the tele-operation of the robotic arm.

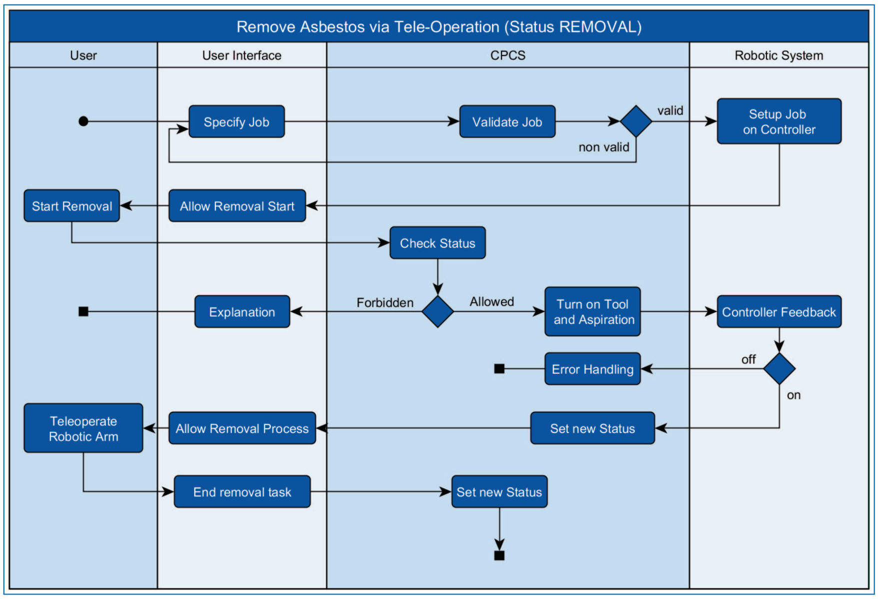


Figure 4.6. Flow Chart of the asbestos removal process.

representation of data and the handling of user interactions (QML). On the ROS site there is a software solution that translates the commands from the user site into technical signals for the robotic system. As an example: The user is moving a joystick to tele-operate the mobile platform and this mentioned ROS code is translating it into velocity and position commands. The interface in between is just to make sure that the QML of Qt Creator and the typical ROS commands can interact with each other. All in all, the very abstract system layout can be defined like this:

Table 4.1. List of the software units their IDEs and the used programming language.

#	Logical Parts of the User Interface	IDE	Language
1	User Interface	Qt Creator 5.2 and higher	C++, QML
2	Logic behind the interface (on the ROS side)	Whatever you prefer e.g. Qt, Eclipse, etc.	C++
3	Interface between ROS and QML	Whatever you prefer E.g. Qt, Eclipse, etc.	C++

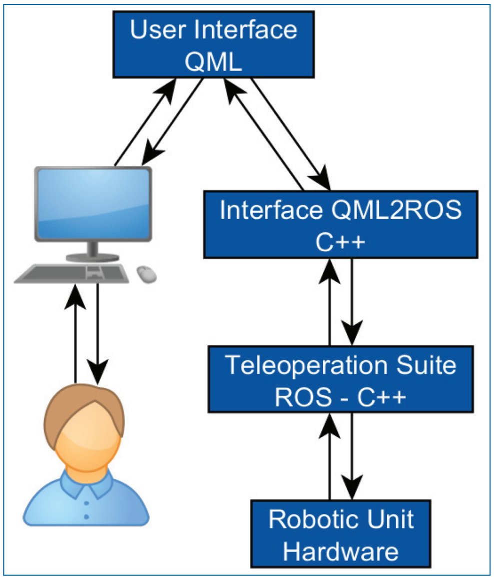


Figure 4.7. Global Layout of the UITO.

4.1.5.2 Interface between Indurad Framework and ROS

The interface “QML2ROS” is defined by typical ROS messages that are used for transferring the data. In the following picture the necessary messages are displayed.

The String messages are necessary for sending text-based information and the Bool messages are most likely to send states or commands between the two software instances. Sending the Int16MultiArray allows the software solution to define error codes and status messages for the user interface. The Pose message is used for showing the 3D environment of the robot. It is necessary to give positions and orientation to the STL-files of the robot. The usage of the different information is explained later on in detail.

As this is the first state of the user interface there need to be extensions during the overall project when the global CPCS is designed. Out of this the shared information between the UI and the CPCS can be extended easily by adding further message types that are used for interaction. A good example can be the usage of environment descriptive objects like boxes, planes or meshes that could be taken from the shape_msgs of ROS. The implementation of these objects types is straight forward and underlines the freedom of development within this software solution.

4.1.6 The UITO

Out of the requirements list and the different Use-Cases a user interface could be realised. In relation to the mentioned requirements the UITO is built out of modular windows, menus and task bars. Here is a list of all the objects that can be handled by the user.

#	**Module**	**Content**
1	3D-Window	Show the virtual representation of the robotic unit and the construction site
2	Status messages	Show warnings, Errors or Task Results
3	Menu bar	Enable the user to choose between different views, actions and tasks. Perhaps to open task specific windows or higher setting options
4	Task specific windows	Specific information about the ongoing action (e.g. the tele-operation of the mobile platform)
5	Robot Camera View	Actual video from the different cameras mounted on the robots
6	Step by Step window	The user is guided through different steps of the removal process. This bar can be found on the ground/bottom of the UI

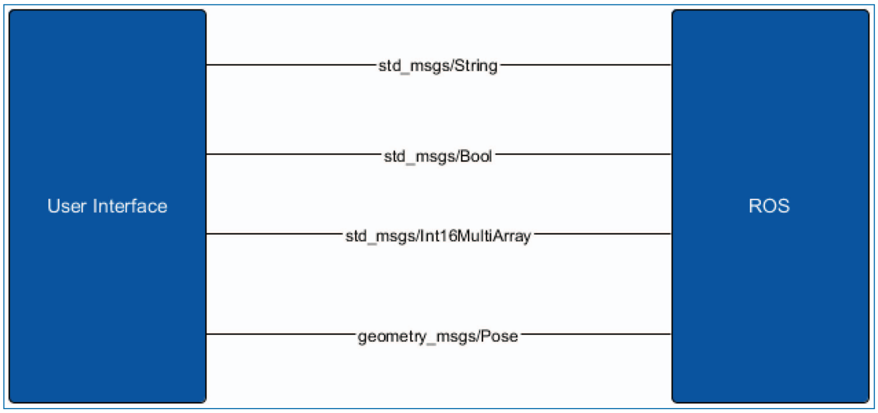
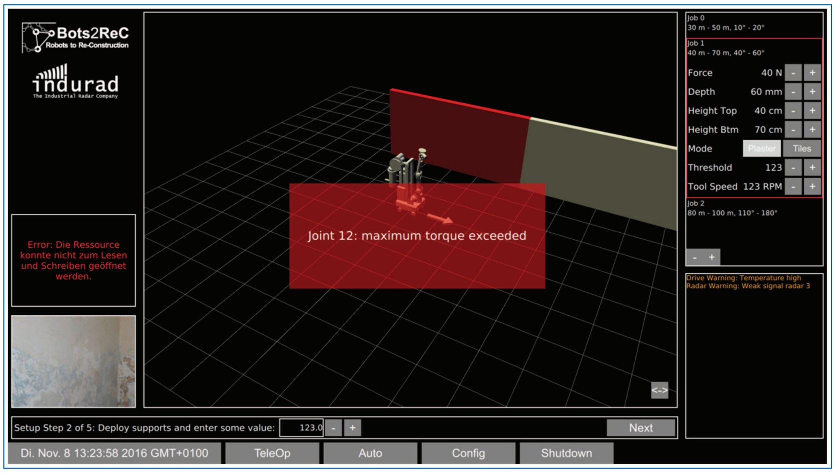


Figure 4.8. QML2ROS.



Error: Die Ressource konnte nicht zum Lesen und Schreiben geöffnet werden.

Joint 12: maximum torque exceeded

Job 0
30 m - 50 m, 10° - 20°

Job 1
40 m - 70 m, 40° - 60°

Force	40 N
Depth	60 mm
Height Top	40 cm
Height Btm	70 cm
Mode	[✓ Plaster] Tiles
Threshold	123
Tool Speed	123 RPM

Job 2
80 m - 100 m, 110° - 180°

Drive Warning: Temperature high
Radar Warning: Weak signal radar 3

Setup Step 2 of 5: Deploy supports and enter some value: 123.0

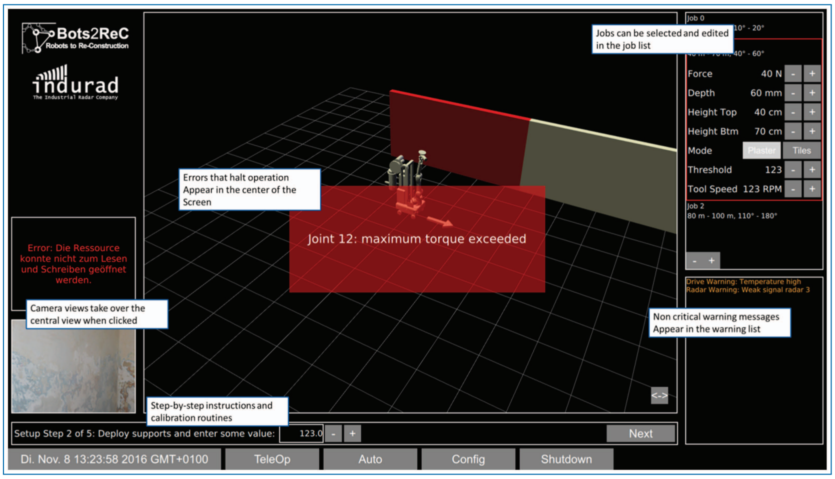
Di. Nov. 8 13:23:58 2016 GMT+0100

Figure 4.9. User interface without comments.

The screenshot in Figure 4.9 gives a first impression about the implemented UI. The centered view is a 3D View of the robotic system and its current state. It is located within a 3D map that can be build by 3D Slamming algorithms or imported by 3D Scans made by professional construction site rebuilders.

This task of autonomous map generation is part of the tasks 6.1 and 6.2 and is going to be clarified in the future. The final data structure for the environmental description is going to be defined during the testing phase of the first prototype. For now the user interface is implemented in a generic way so that it can be adapted to every solution that might be defined for this mentioned data structure in the future. Within the 3D View it is possible to see the current robot state, its position inside of the flat and the properties of the flat itself. The robot positions are updated with the ROS data at runtime and allow the tele-operation in most of the cases without any further details. If the sensor signals are not clear enough and the 3D view might get unprecise it is possible to enlarge the actual view of the mounted cameras to clarify the situation. Beside this two different ways of representing the robot state and its surrounding it is possible to specify the job that is going to be run during the removal task. While only the typical grinding task will be executed and represented during first tests, the user interface can be extended in the future for different tools, processes or surface circumstances.

Within the UITO it is also possible to have the feedback about warnings and errors of the robotic system. In the right lower corner of the screen the warnings are displayed. Warnings are issues that are not perfect but that can be handled by the robotic system. Only when they get worse they can turn into an error. Error messages are written in the center of the screen as shown in Figure 4.10. Errors will force the robot to stop its task execution. Out of this the centered banner in red was chosen for the user interface.



Jobs can be selected and edited in the job list

Errors that halt operation Appear in the center of the Screen

Joint 12: maximum torque exceeded

Error: Die Ressource konnte nicht zum Lesen und Schreiben geöffnet werden.

Camera views take over the central view when clicked

Non critical warning messages Appear in the warning list

Step-by-step instructions and calibration routines

Job 0
10° - 20°
40 m - 70 m, 40° - 60°

Force	40 N
Depth	60 mm
Height Top	40 cm
Height Btm	70 cm
Mode	Plaster / Tiles
Threshold	123
Tool Speed	123 RPM

Job 2
80 m - 100 m, 110° - 180°

Drive Warning: Temperature high
Radar Warning: Weak signal radar 3

Setup Step 2 of 5: Deploy supports and enter some value: 123.0

Di. Nov. 8 13:23:58 2016 GMT+0100

Figure 4.10. User interface with comments and explanations.

Beside this mentioned functionality the user interface of tele-operation is also able to show the different states of the robot. For now there are states as "TeleOp", "Auto", "Config" and "Shutdown" implemented. The best way how to organize the system states for the user will be clarified during the tests of the prototype V1. The aim is to create an easy and robust way to deal with the robotic system. The feedback of Bouygues and its experienced employees will be taken into account for the design of these tools.

4.1.6.1 Hardware for tele-operation

Beside the normal input of words and commands on the screen the system need a device for the tele-operation movement. Of course this is also possible with a common keyboard but probably it is also a good idea to look for a more intuitive and flexible device. During the investigations concerning this device the following possibilities have been checked by the consortium members.

#	Device	Properties	Example
1	Normal Joystick	Typical Desktop Solution **Advantages:** • Cheap • Already implemented in ROS **Disadvantages:** • Only for clean rooms without dust and dirt	

#	Device	Properties	Example
2	Rugged Joystick	Very rugged and safe Joystick boards with additional functions as emergency stop. **Advantages:** • Made for tough conditions • Additional functionality over buttons and switches **Disadvantages:** • Expensive • Not implemented in ROS **Product:** • Control Chief	
3	Game Controller	Typical gaming solution for PS3 or Xbox **Advantages:** • Easy to use • Rugged and cheap • Already implemented in ROS **Disadvantages:** • Perhaps not rugged enough?	
4	Rugged Gaming Controller	Rugged version of the gaming controller **Advantages:** • Rugged setup extra for construction sites **Disadvantages:** • Not implemented in ROS **Product:** • Humanistic Robotics	

#	Device	Properties	Example
5	Keyboard and Mouse	Normal desktop solution **Advantages:** • Already necessary for the normal input of text and klicks • Cheap • Everybody knows it **Disadvantages:** • Not as intuitive as other devices	

After several investigations the PS3 Controller was chosen as the input device for tele-operation. In correspondence with Bouygues it was pointed out, that the area for tele-operation is not as rough as mentioned. Furthermore, the PS3 controller can be covered with plastic foil as well and is cheap in comparison to other devices. Related to other devices it gives a very intuitive solution for tele-operating the arm and the mobile base at the same time. Another positive aspect about the PS3 Controller was the fully integrated functionality in ROS that made the integration straight forward. During the first tests the usefulness could be pointed out.

4.1.7 Outlook and Future Work to be Carried Out

Within the Task 6.3 the UITO is created successfully on software site. The desired functionality is shown and proven in simulation. The manipulability of the mobile platform with the PS3 Controller was shown on hardware site as well. Accordingly to the hardware delay of the robotic arm as pointed out in the reports of WP5 it was not possible to debug the UITO with the complete robotic system in the loop. Therefore these tests are planned for the beginning of March when the whole robotic unit is assembled at Eurecat in Barcelona. The estimated time for debugging this software suite is less than a week and should be straight forward so that there will be a working interface between the UITO and the robotic unit in the mid of March. Finally the task will be reached with a delay of two weeks and there are no needs to adjust the goals of WP6.

4.1.8 Summary and Conclusion

All in the entire user interface for tele-operation was implemented successfully. It is able to represent the current robotic state and to allow the end user to manipulate essential parts of the robotic system in an intuitive way. For reaching this deliverable the methods of the Unified Modeling Language where used for documenting the software architecture and for sharing ideas and thoughts on a vivid basis. A consensus about the software solution was found via sharing the Word Documentation and attached comments. As pointed out in chapter 7 the success of the UITO was shown in simulation for the whole prototype and for the mobile platform on the hardware site as well. The test of the whole unit will be carried out as soon as possible at the beginning of March. In according to the user interface the display methods for environmental objects are developed and improved in Task 6.1. The resulting data sets will be integrated into the UITO continuously. The user interface may be also affected by the development of the CPCS later on but the estimated influences are quite low.

Deliverable 6.4
Interface for job specification layout, implemented and documented

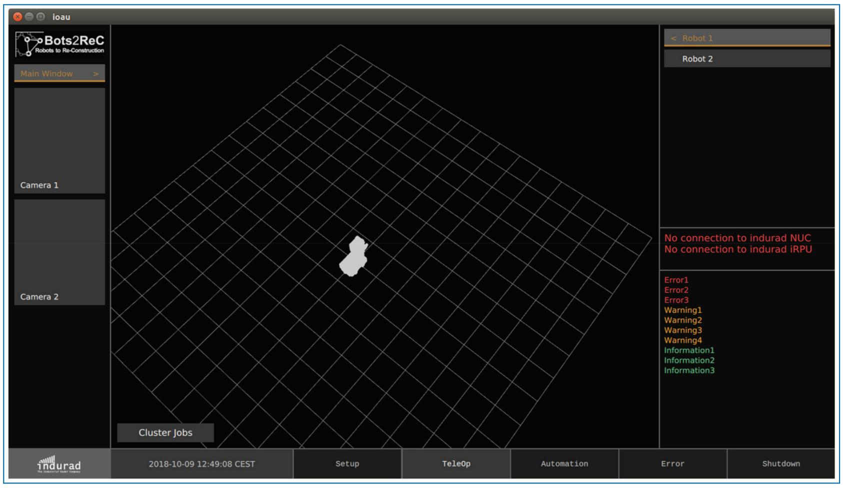
Authors: Tobias Haschke, Fabian Herb

4.1.9 Deliverable in the Context of the Work Program

The User Interface for the Bots2ReC Project is the abstraction layer between the complex robotic data elements and the future operator. According to the development of the CPCS and the Use Case automation the User Interface is a key factor for the acceptance of the product within the market. Related to Task 6.3 it is the consequent progression of the tele-operated version implemented before. Beside the past UI version it is also connected to the development processes of the other tasks within WP 6 as it needs to enable the operator for using the entire robotic system with the desired purpose. Beside the relation to the CPCS development it is also represented to BYCN for the confirmation of usability.

4.1.10 Methods and Work Carried Out

As this version of the user interface is a progression of the tele-operated version the development strategies and the chosen framework remains the same. The whole system is tailored for an All-In-One (AIO) PC from Asus for enabling a tablet-like behavior within a robust case. The AIO is chosen as it allows typical PC hardware interfaces as USB and Ethernet for a robust solution within construction environments.



No connection to indurad NUC
No connection to indurad iRPU

Error1
Error2
Error3
Warning1
Warning2
Warning3
Warning4
Information1
Information2
Information3

Figure 4.11. New straightened design of the User Interface.

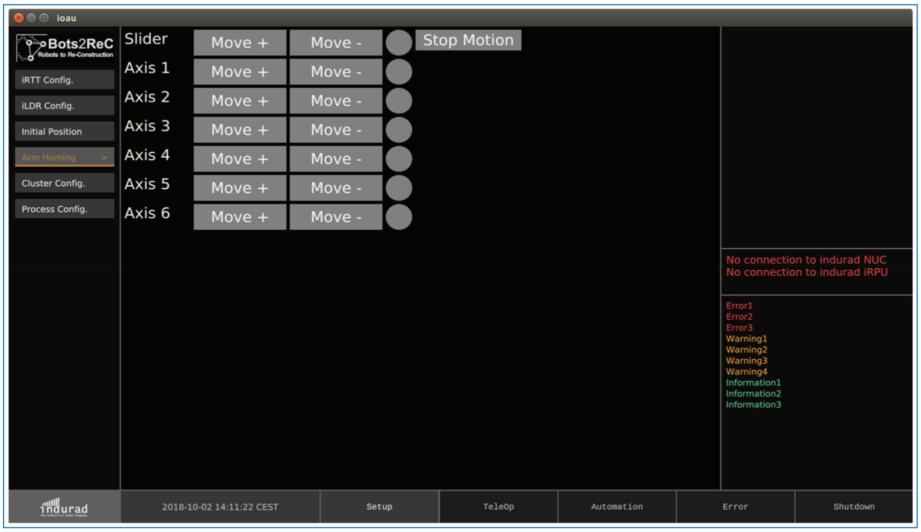
On the software side the AIO is running a minimal linux image with the UI application as startup sequence. Out of this the operator does not recognize anything of the operating system in the background as the only available screens are the ones from the UI. This solution is used for simplifying the use of the system without enabling the operator to mistune the system manually. For the implementation the UI Framework QML by Qt is still in use as it is easily linked to plain C++ code. Out of this the interface between the CPCS programs and the UI can be developed quite flexible as they are based on the same programming languages.

As an addition the QML framework natively supports touch-enabled solutions and well rendered output screens for a vivid, fresh and intuitive usage of the UI. Out of the combination of high range of feature and the high performance of C++ it is often used for custom displays as the one at hand for the Bots2ReC use case.

4.1.11 User Interface for Job Specification (UIJS)

During this development phase the whole design of the UI for tele-operation was cleaned and well-balanced for the new added features. This means that the first solution for tele-operation was still fully functional but the overall appearance had to be straightened out for intuitive workflows for the operator. The main idea of the new design is to keep the usage of the UI as simple as a typical and well-known app on nowadays smartphones. This lead to compact button and panel positions with clean understandable rules.

In comparison to the former solution the already developed panels remained the same. The 3D view for the robots model status is still cantered by default and can



No connection to indurad NUC
No connection to indurad iRPU

Error1
Error2
Error3
Warning1
Warning2
Warning3
Warning4
Information1
Information2
Information3

2018-10-02 14:11:22 CEST

Figure 4.12. Feature Arm Homing.

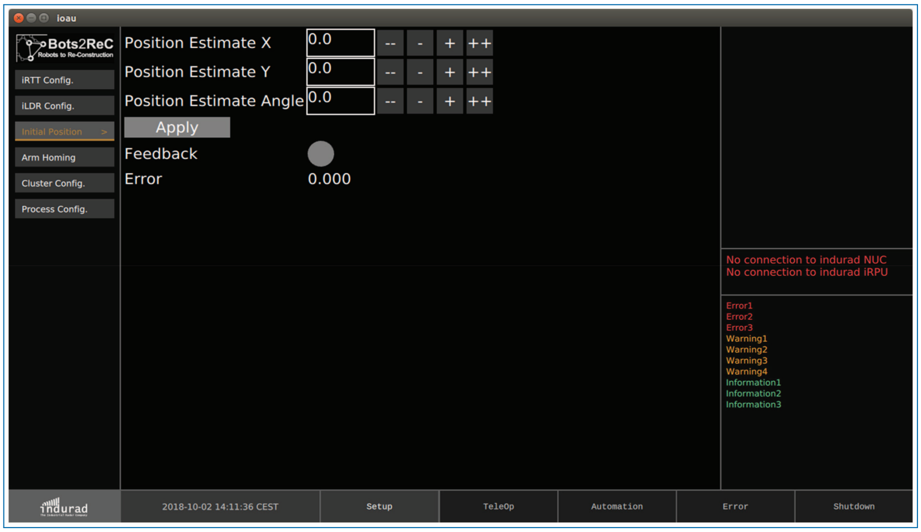
be swapped by the camera input panels from the left side. Also still the same are the boards for the network status and the message board on the lower right side.

A new feature that was necessary is the robot selection menu in the upper right side as the development of V2 forced the development team to control both robots from within the UI. During the CPCS extension in the meantime five discrete stages for the asbestos removal robot where defined that are changeable by the operator:

- Setup – Calibration and Configuration processes
- TeleOp – Teleoperation of the robot in form of the mobile base and the tool
- Automation – Autonomous task execution based on the global environment data model
- Error – Robot is unable to solve a special situation alone and is waiting for external advices
- Shutdown – Status for shutting down the robotic system

Especially the Setup Stage changed a lot for this deliverable, as the setup of the environment model and the initial state of the robots are essential for the task execution process.

Since this release, it is possible to home the arm of V1 manually at the beginning without being an expert in robot engineering. This simplification is one huge step to the intuitive usage of the robotic system through this simple interface panel with direct robotic feedback.



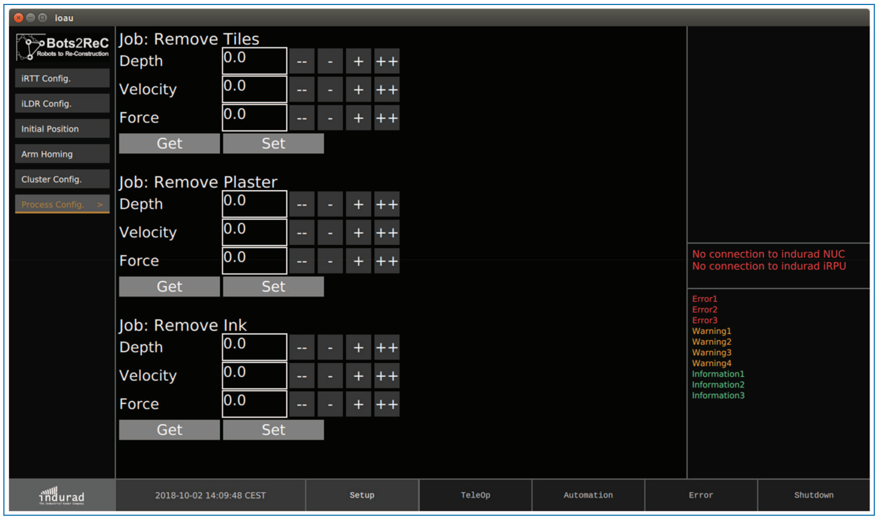
Position Estimate X 0.0
Position Estimate Y 0.0
Position Estimate Angle 0.0
Feedback
Error 0.000

No connection to indurad NUC
No connection to indurad iRPU

Error1
Error2
Error3
Warning1
Warning2
Warning3
Warning4
Information1
Information2
Information3

2018-10-02 14:11:36 CEST

Figure 4.13. Feature Initial Pose.



Job: Remove Tiles
Depth 0.0
Velocity 0.0
Force 0.0

Job: Remove Plaster
Depth 0.0
Velocity 0.0
Force 0.0

Job: Remove Ink
Depth 0.0
Velocity 0.0
Force 0.0

No connection to indurad NUC
No connection to indurad iRPU

Error1
Error2
Error3
Warning1
Warning2
Warning3
Warning4
Information1
Information2
Information3

2018-10-02 14:09:48 CEST

Figure 4.14. Feature Job List.

During the Setup Stage each robot need an initial position within the known map for being able to localize itself. As the mapping and removal execution processes are separated and most likely also run on different robotic systems this step is necessary for every boot sequence on a construction site. To reduce the complexity the current error and a feedback signal are used to give the operator a visual and numeric idea about the current initial pose accuracy that was configured.

As the semantic map is the semantic and geometric data representation after the mapping process this is also used to add the process specification. Each element can be setup with a specific parameter set that can be specified during the Setup Stage. In this case Figure 4.14 shows the possible setup of three processes which can be adapted to the current use case.

4.1.12 Delays and Aspired Compensation

The entire UI is implemented and deployed on the target AIO for the Bots2ReC project. Its functionality is entirely tested within the ROS framework but due to the hardware challenges on V1 a real-scenario test is still outstanding.

4.1.13 Outlook and Future Work to be Carried Out

The tests with the real robot will start immediately after solving the mentioned hardware challenges. Due to the modular software layout, the integration of these features into the real robot should be remarkably small.

4.1.14 Summary and Conclusion

Overall, the user interface has reached a well-tailored solution for the use cases at hand. It is simple, intuitive and represents the workflow of the robotic system. After the integration into the real robot system and the real-scenario validation this should be the final release of this user interface for the Bots2ReC project.

Task 2.4 Cooperative robotic control strategies

The payload (equipment) has been distributed in two bases as the robot footprint could not be increased due to application limitations (size of the apartments to be decontaminated). Both platforms have an umbilical cable between them and need to be moved as a single system. The robot should be easy to operate in manual and automatic modes. The system should keep the distance between the platforms avoiding tension in the cable.

To avoid the cable damage, a mechanical solution was developed by TLABS, a kinematic chain with 3DOF and two steel links that protect the cable and that act as a physical limit if the maximal extension is reached, in this case the structure is strong enough to withstand the motor torques. This structure imposes a limitation in the minimum distance, and also in the relative orientations of both platforms. The minimum distance is about 20 cm between platforms, while the structure allows angle differences of $+/-90^{\circ}$ of each end joint. The mentioned structure can be seen in the next figures:

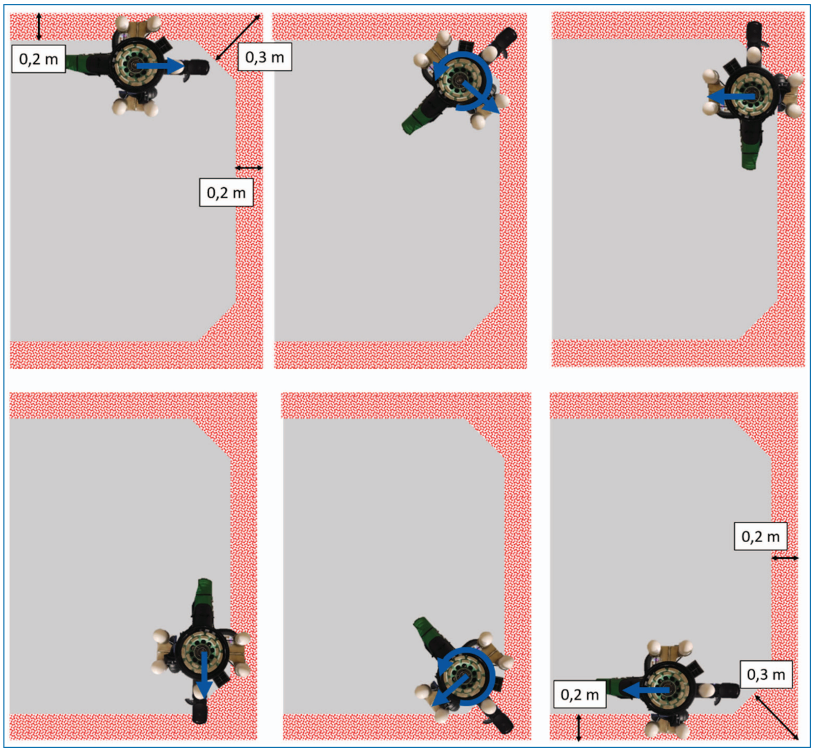


Figure 4.15. Wall coverage floor.

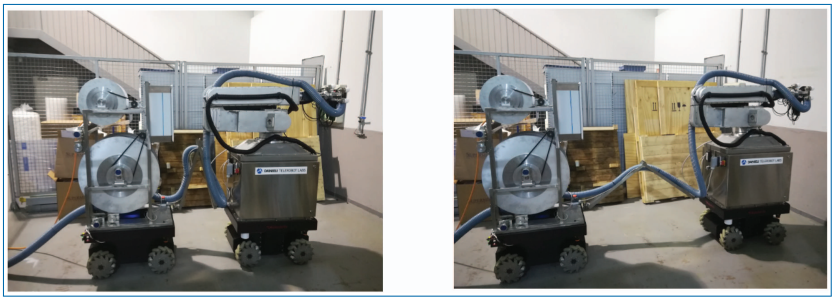


Figure 4.16. Robotic Tandem with cable and aspiration bridge.

With the described tandem system, the objective was the developmet of a control strategy that permits the coordinated control of two platforms. With this objective, the following three concepts have been analysed:

- Concept 1 – **Following reactive**: the rear platform just follows the front one in a reactive way. The rear platform "sees" the front one either by beacons or QR codes, etc. Speed references are commanded in a way that the objectives of following/keeping distance to obstacles are weighted and a resulting speed is computed. This concept can be implemented with or without localization, the differences are explained next.

1.1 Both platforms are localized: the rear platform tries to keep **a fixed relative pose w.r. to the front platform based on the estimated pose.**

1.2 With no localization of the rear platform: in this case the rear platform tries to keep **a fixed relative pose w.r. to the front platform based on the measured relative pose (e.g. beacons)**. The main advantage of this approach that the requirements in terms of localization are relaxed as only the front platform requires localization, while the rear platform is just acting in a reactive mode. A disadvantage is that the rear platform cannot benefit from the local costmap obstacle avoidance of the move_base stack.

- Concept 2 – **Following planning**: both platforms are correctly localized. The rear platform uses a trajectory planner and computes poses relative to the estimated pose of the front platform and updates the navigation goal continuously.

  2.1 The rear platform plans path (with preferred planner) to the updated front platform robot pose or fixed transform (tf) relative to front platform. The standard planners do not introduce restrictions in distance nor orientation. The controller may stop the front platform if the distance between platforms is not kept or will adjust speed.

  2.2 **The rear platform plans to a fixed pose that is relative to the front platform.**

  2.3 **The rear platform follows the historic frame sequence of the front platform**. This should allow e.g. turning a corner.

- Concept 3 – **Pure teleoperation**: both platforms receive the joystick reference and operate synchronized in omnidirectional mode. When the tandem system moves linearly, both platforms receive the same command. This concept has been implemented in a client-server configuration where the front platform connects to a topic that is sent via a rostful server to the rear platform that reads this reference and inputs to its own joystick multiplexor.

  3.1 The angular speed is sent according to a fixed distance and this distance is kept manually by the operator. This leads to some adjustments while the robots move, but turned out to be a simple and effective solution.

  3.2 The angular speed is sent according to a measured distance. This should be easy, as the mentioned markers should permit an update of this value in real time.

The operation of the tandem system is in open loop, speeds are sent and the control and odometry errors can lead to some misalignments of the platforms. The joystick allows to control each platform independently thus permitting

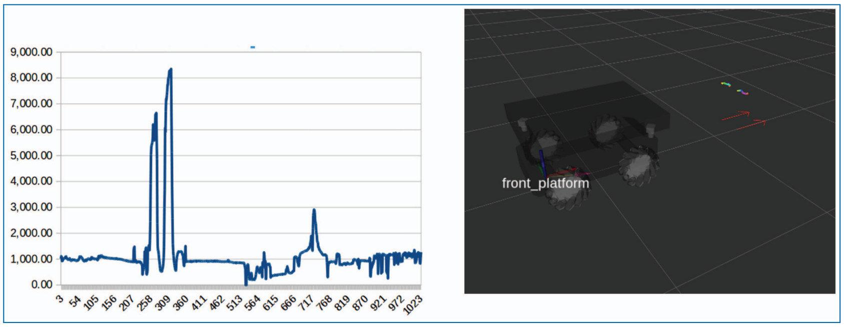


Figure 4.17. Single laser intensity and detected markers on front platform.

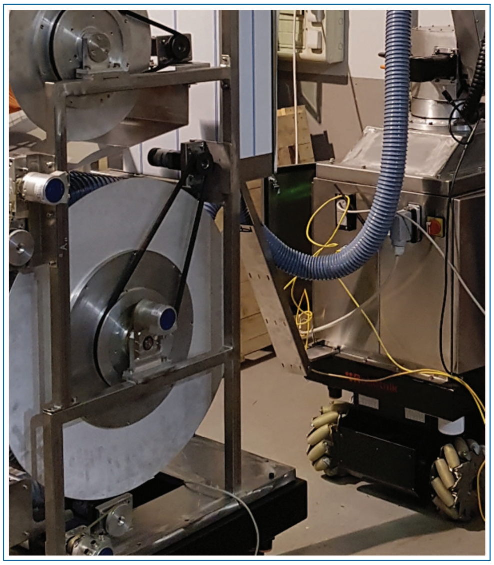


Figure 4.18. Reflectors installed in the front platform.

single motions to align the tandem in orientation, to keep the required distance or to perform complex operations as passing through narrow spaces where the operation in more autonomous modes becomes too complex.

The three described concepts have been implemented and tested in simulation (1.1, 1.2 ,2.1, 2.2) and the concepts 1.2, 2.2 and 3.1 tested with the real robots.

For the implementation of the reactive mode QR codes were initially considered, but finally discarded due to the dust. The final solution selected was based on the detection of markers with a higher reflectivity. This method has proven to operate even in dust conditions. The laser range finders are providing not only the distance but also the intensity of the reflected beams. In a short range, the intensity perceived on the markers is much higher than non reflective objects, even in dust conditions. The next figure shows the intensity of a single laser scan. Two peaks can

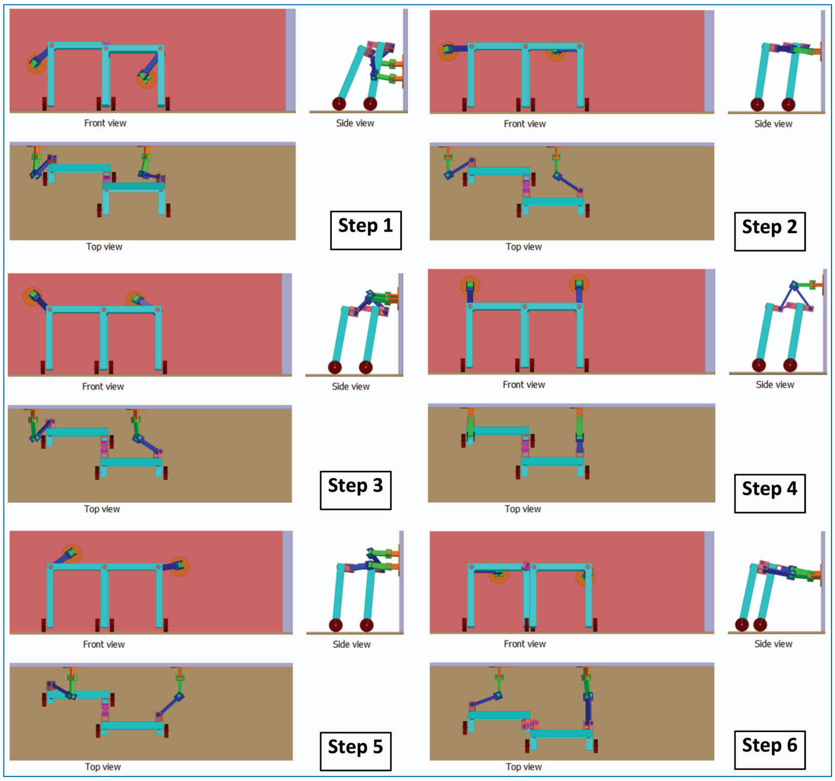


Figure 4.19. Grinding process using a p-bot in side by side connection.

be clearly identified corresponding to the markers. The position of the two markers (x1,y1,x2,y2) permits to compute a transform relative to the laser link and finally to the robot base link. This information is finally used by the reactive system to estimate the front platform pose and to command motion commands.

From the localization point of view: in autonomous mode, each platform is tracking its own estimated pose. The distances of the reactive system can be compared with the distances of the localizations of platform one and two and a divergence should make the system stop. This is the considered software checking routine that is not integrated yet as long-term tests weren´t able to be performed.

Figure 4.18 gives an impression of the real scenario two platform following each other.

4.1.14.1 Work package 4: Customization of the mobile platform: Design, control and implementation

The entire work package 4 was closed within RP2 by the Deliverable 4.3 already submitted and approved by the last review meeting. Nevertheless, the integration of hardware showed that some iterative implemented modifications were necessary.

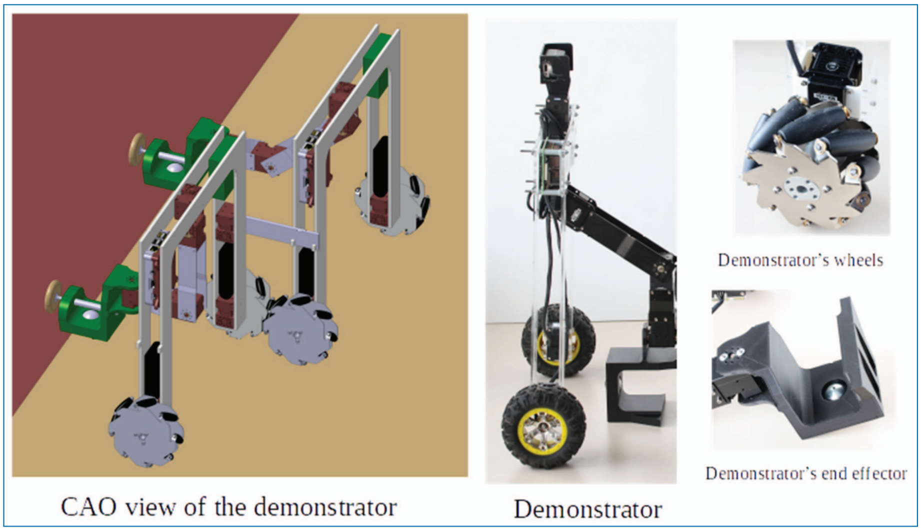


Figure 4.20. Demonstrator of the grinding process.

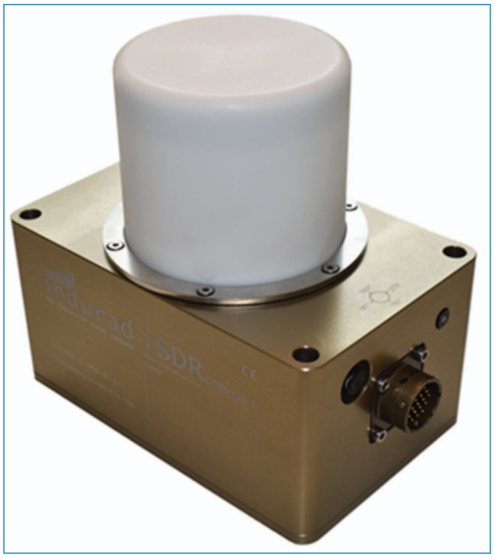


Figure 4.21. iSDR-C.

To keep the form they are documented within the related tasks even though they are officially closed already. In principle, all the modifications and optimizations were caused by necessary improvements during the integration and commissioning phases in WP3 and WP6.

4.2 Semantic Mapping

Deliverable 6.2
D6.2 Sensor fusion and data evaluation implemented

Authors and contributors:
CHARAF EDDINE Sami, RWTH Aachen University

4.2.1 Deliverable in the Context of the Work Program

Deliverable **D6.2** (Sensor fusion and data evaluation implemented and tested) is part of task 6.1 (Environmental perception and mapping). The main objectives of task 6.1 are:

- to **combine data coming from various sensors** via the defined interfaces from task 3.1 (e.g. vision, RGBD cameras, Lidar, radar) used in task 4.2 **to build a map** of the environment
- to **identify processing** areas such as
  - ground surface,
  - walls and
  - ceiling
- to **detect areas** that should be **avoided** during automated asbestos removal:
  - doors,
  - windows and
  - areas of too complex geometry (e.g. pipes, lighting, electric devices)
- to continuously **update the map** by several robots working simultaneously and share the map between these robots

The achievement of these objectives will allow the creation of an intuitive map based user interface for the task allocation by the operator.

4.2.2 Suggested Approach

Solving task 6.1 requires two kinds of maps, one of them for navigation in possibly low-visibility environments, the other one for **task allocation, planning and fulfillment**. The map for navigation will be used solely internally for computational purposes and is investigated in task 6.2, based on the the sensor developments for the mobile platform (iRTT, iSDR, iSDRcompact described in D4.2 and LIDAR described in D4.1). The map for task fulfillment is visible to the user and thus has

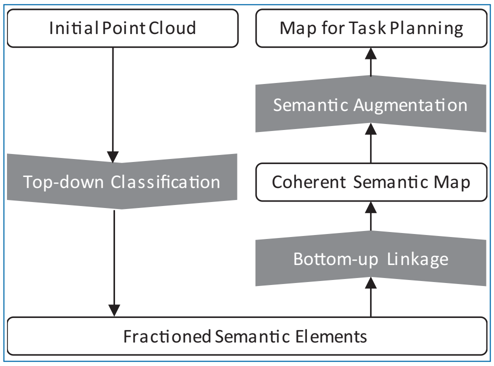


Figure 4.22. Main processing steps of the proposed algorithm.

to be displayed in the user interface. This map is the central element of interaction with the user, as it will provide information about processing and non-processing areas within the contaminated rooms.

The mapping approach presented in this section does not need any sensor fusion as solely a 3D laser scanner, provided by Bouygues, captures the recorded data.

4.2.3 Mapping for Task Fulfilment

We aim to enable task planning and assignment based on a lightweight visualization of the robot's environment within a user interface. To do so, we propose a computationally efficient approach that consists of three consecutive main steps, as shown in Figure 4.22. First, we apply a **top-down classification** (sec. 3.1) to divide the initial point cloud into several clusters and classify them as wall, ceiling or floor, solely based on geometrical considerations. Second, we perform a **bottom-up linkage** (sec. 3.2) and integrate the classified objects, grouped as single room entities, into a hierarchical tree data structure to build a coherent semantic map. Third, we **augment this map** (sec. 3.3) by detecting doors, windows and other elements that might impede asbestos removal. Finally, we infer a lightweight coherent polygon model from the created semantic map in order to visualize it within the user interface. The user is now able to make decisions for further task planning and can assign tasks to certain semantic objects.

4.2.3.1 Top-down classification

The input data of our algorithm is a single point cloud as shown in Figure 4.23. We segment this point cloud into separate clusters as shown in Figure 4.24, using an implementation of the region-growing algorithm provided in PCL.[1] It is worth

1. http://pointclouds.org/

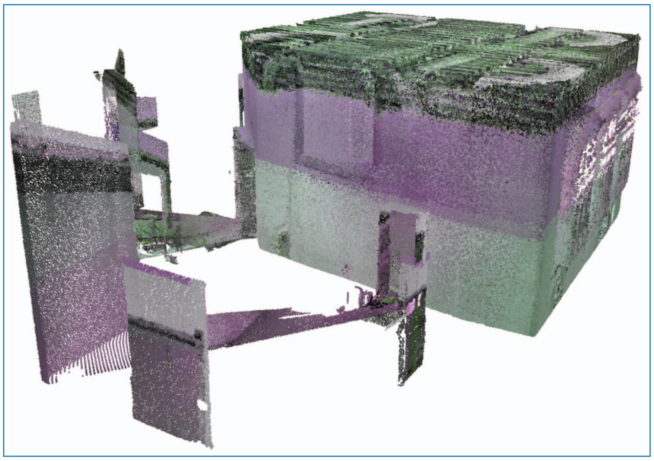


Figure 4.23. Initial point cloud.

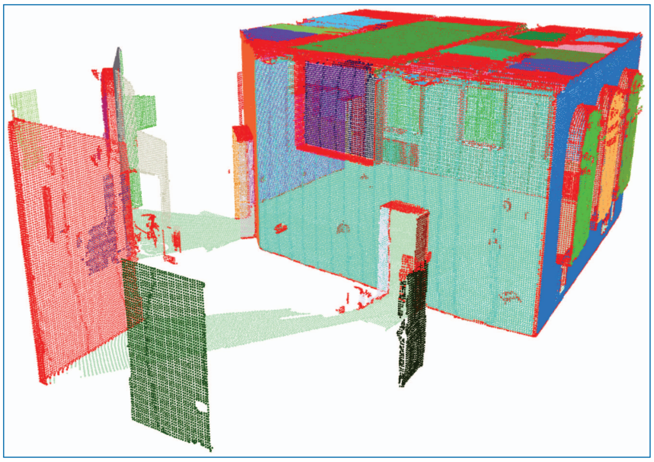


Figure 4.24. Results of the region growing algorithm. Each color represents a certain cluster, the red color indicates dismissed clusters due to too small cluster size.

mentioning, that the walls in Figure 4.24 are clustered correctly, but – and this is crucial to our application – the single walls are not yet grouped as one group feature walls. This is shown by the different colors of the single walls. The same effect happens to the ceiling of the room: Instead of being clustered as one single element, it is clustered into multiple non-coherent elements. We then use the principal component analysis to calculate the respective bounding boxes for each region. Afterwards we classify the estimated bounding boxes according to their geometry and obtain walls, floors and ceilings, as shown in Figure 4.25.

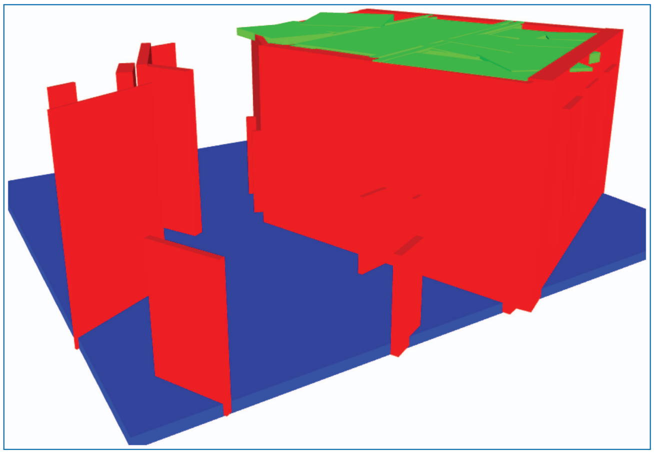


Figure 4.25. Classified bounding boxes of the clusters. Red boxes characterizes walls, green boxes are referred to as ceiling clusters and blue boxes represents floor clusters.

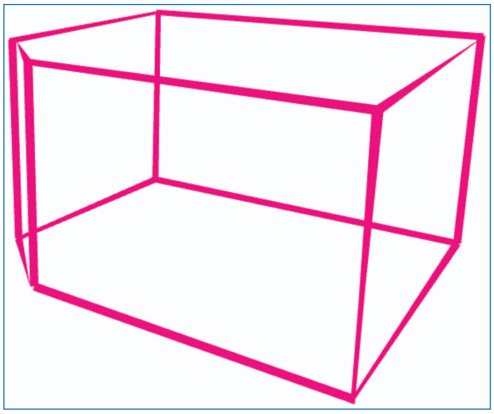


Figure 4.26. Linked and approximated polygon model inferred from the scene.

4.2.3.2 Bottom-up linkage

Our aim is to concatenate the detected, yet fragmented, clusters in order to obtain a coherent semantic map. First, we search for the adjacent bounding boxes of every wall and obtain a list of all wall elements and their neighbored walls, so called **linkage**. Given this description of the overall environment, we then determine single room subsets using a graph cycle search on the wall edges and store this information in a semantic objects tree structure, so called **grouping**. In the last step we **approximate** all regions by their best fitting planes and intersect any neighboring planes with each other. The resulting intersection points represent the semantic map as a coherent polygon model of the robots environment (Figure 4.26), enabling a lightweight visualization in the user interface.

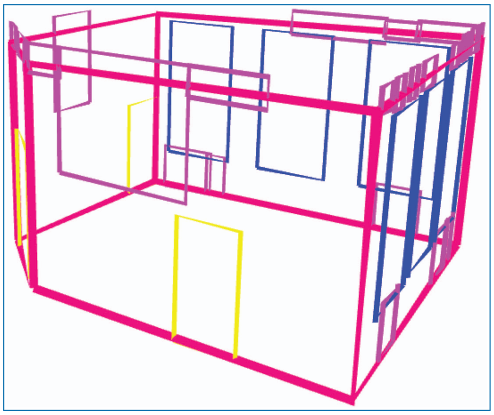


Figure 4.27. Polygon model with extracted doors, windows and other structures visualized. Blue polygons indicates windows, yellow polygons doors and pink polygons shows not further classified noticeable features which should be concerned during task planning.

4.2.3.3 Semantic augmentation

After the creation and linkage of the semantic map we further examine the classified clusters in order to identify subfeatures, e.g doors and windows within wall clusters or lamps within ceiling clusters. The final map is shown in Figure 4.27.

4.2.4 Validation

To demonstrate the performance and accuracy of our deduced model, we finally compare the polygon model from Figure 4.27 to the initial point cloud from Figure 4.23. As shown in Figure 4.28, the polygon model matches the initial point cloud very well.

4.2.5 Outlook and Future Work to be Carried Out

As mentioned in Sec. 2, the current approach relies on the point cloud of a 3D laser scanner from Bouygues that is a) very expensive and b) requires the point cloud to be captured before the removal process. This also implies that the Bouygues 3D laser scanner is a separate unit that is not mounted on the robotic system and needs additional manual operations. To circumvent this drawback, we will try to develop a low-cost solution, integrated to the robotic system, that is able to capture the 3D point cloud required for our semantic mapping approach. As our previous trials using a single Kinect mounted on a Turtlebot were not promising, we plan to develop a Multi-Kinect-sensor and build the overall point cloud representing the environment in the scope of task 6.2.

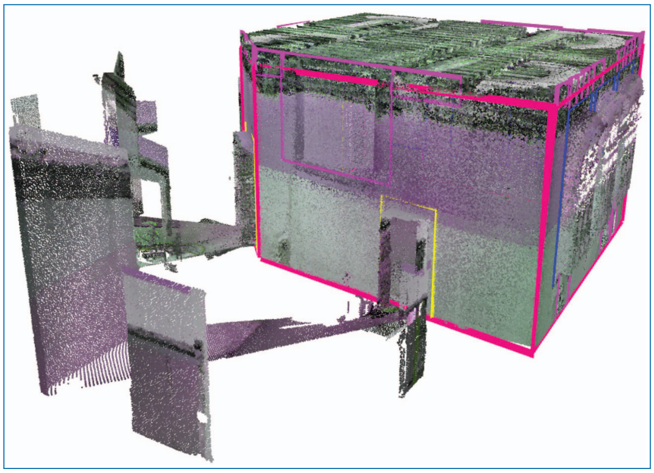


Figure 4.28. Visualization of the initial point cloud and the resulting polygon model.

4.2.6 Summary and Conclusion

The algorithm for semantic mapping based on a 3D point cloud has been developed successfully. The main purpose of the algorithm is to translate the huge initial point cloud to a lightweight and coherent substitute made of polygons and to enable task planning of the asbestos removal. In order to pass the inferred information to the user interface, we transform the detected and classified group features from the semantic map into a coherent polygon model. We obtain:

- appealing coherent 3D representation of the semantic objects
- class labels of every semantic object
- editability of the objects for defining process related properties (e.g. surface condition: plaster, paint or tiles)

4.3 Artificial Task Planning

Deliverable D2.3
Algorithm for precise, dynamic and centralized control strategies

Authors and contributors:
Tobias Haschke, RWTH Aachen
Albert Fullana Puig, EURECAT

4.3.1 Deliverable in the Context of the Work Program

For controlling multiple tethered mobile robots with the aim of removing asbestos from normal flats, a multi-layer control strategy is implemented. Due to the requirements set by the construction industry (Bouygues Construction), the main objective of the task execution is time efficiency. This objective is explained by the enormous costs for an unlet apartment house during the refurbishment time, as the building cannot generate any income during this period. Out of this, the whole control implementation is aiming for a fast and robust task execution.

The dominant layers are *task planning* and *task execution*. Within the *task planning* layer an optimal strategy for solving the whole removal problem is generated. Due to the uncertain environment of construction sites, this optimum may differ during the execution time, which leads to a continuous planning process throughout the whole execution time. This already reconciled to the *task execution* layer, which aims on the one hand for the execution of the pre-planned tasks and needs to deliver valid process data for updating the systems state. This data is used for re-planning the whole execution problem during runtime again to adapt to any changes or mislead assumptions.

In the following chapters, the individual mechanisms of actions are explained for each layer individually and how they interact. Within the first chapter, the underlying model of the environment and the robot is explained. Build on this the next chapter will explain some optimization problems and their solution to fulfill the task planning layer. Finally, yet importantly, the task execution chapter will point out how the currently optimal tasks are executed in a robust and efficient way.

4.3.2 Methods and work carried out

Modeling of the problem

Before an optimization problem can be formulated an adequately exact representation of the problem in case of a mathematical model needs to be developed first. This chapter will explain the different model elements and its context for the following steps. During the completely modelling process, a fully parameterized description was put into focus.

Also the problem formulation can be split in two different layers. The first subproblem deals with the optimization of operational points in front of a wall that needs to be cleaned from asbestos. This can be defined as an optimization problem as different robots can have different properties as reach lengths of the robotic arm, tool size or a different force distribution. Especially within the Bots2ReC Project this property of the problem formulation is essential, as the project will have multiple different prototypes (V1 and V2) that still need to work together.

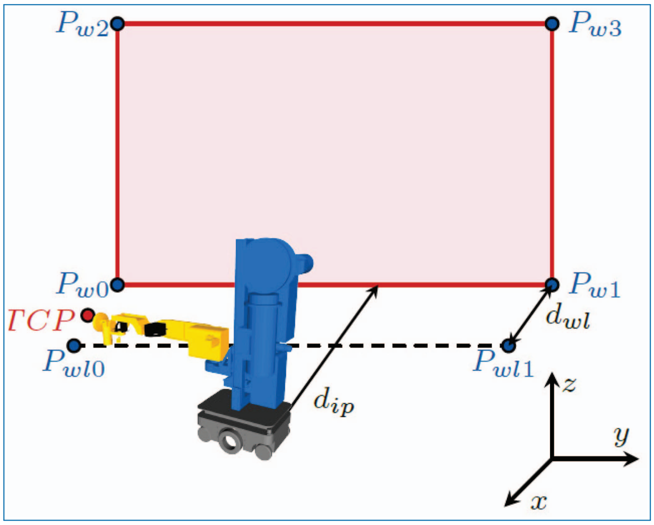


Figure 4.29. Geometric modelling of a wall task.

These different properties will be used for formulating the cost of each robot to fulfill a certain task. The individual costs need to be modelled to enable an optimal task planning. In Figure 4.29 the geometric modelling of a wall task is shown that contains the entire wall as a polygon representation. This data format is equal to the semantic mapping approach developed in WP6 in Tasks 6.1 and 6.2.

The additional information for each robot is the working line distance *dwl* that is represented by *Pwl0* and *Pwl1*. This is an individual distance, which is best for each robot and its reach size. Additionally, each robot is defined with a maximum working width that can be processed. Due to the non-linear force transmission dependent on the distance to the operational point it is also necessary to formulate cost functions for these characteristics.

Those cost functions are shown in Figure 4.30 and are fully parametrized by 3 parameters a, b and c. Out of this it is possible to adapt this quadratic cost function to any new prototype easily without affecting the model or the planning strategy. As each task need to cover a complete area of a wall, ceiling or floor element the vertical and horizontal movements need to be combined to trajectories. As the cost functions for horizontal and vertical movements can be different (e.g. tool is working better in direction of the gravitational vector), two main trajectory options are defined. They are displayed in Figure 4.31.

The trajectories are parametrized over the tool size of each individual robot. Which means that the trajectories will differ for different grinding disc diameters and their maximum speeds. Summing up the first sub-problem the removal task splits each wall element into the desired number of tasks which are optimal for each robot. Afterwards the optimal removal trajectory is selected for each element.

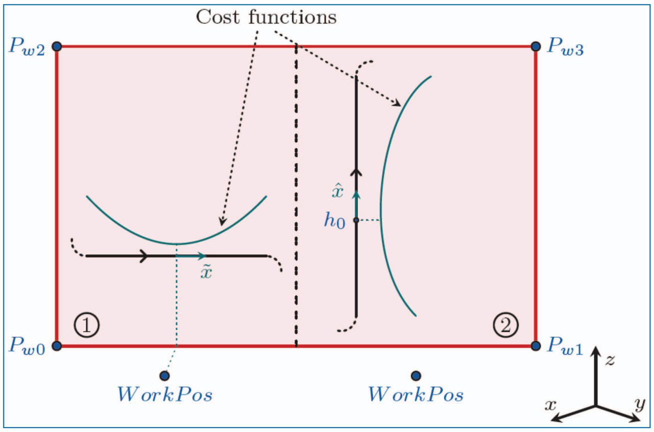


Figure 4.30. Cost functions for coping with higher distances.

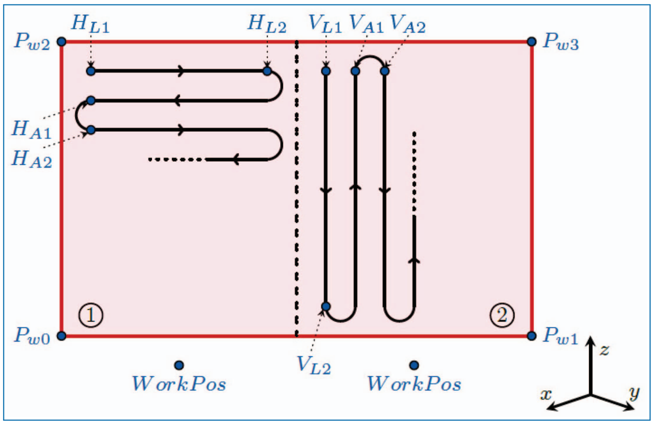


Figure 4.31. Removal trajectories in horizontal and vertical orientation.

The second sub-problem is anchored on a higher abstraction and managing level. It is the formulation of the entire plans that need to be synchronized throughout the whole team of robots. This sub-problem addresses typical questions as “Which robot is executing the task?” and “When is the task executed?”. Before the answers can be given for these questions the model needs to be developed.

The data format for storing the entire plan is represented in Figure 4.32 and can also show the different abstraction levels. The *RobotsPlan* contains the complete global plan which defines each individual movement of each robot. This plan is stored in the CPCS. As it is not really efficient to send the complete tasks to every robot individual sub-levels are introduced. They are called *RobotJob* and contain a list of task for one individual robot. The last two sub-levels are lists of so called *ScheduleActivities* which can contain four different shapings: *Travel*, *WaitBeforeTask*,

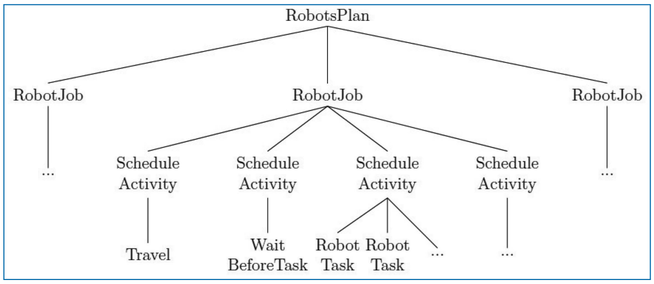


Figure 4.32. Data Format of tasks.

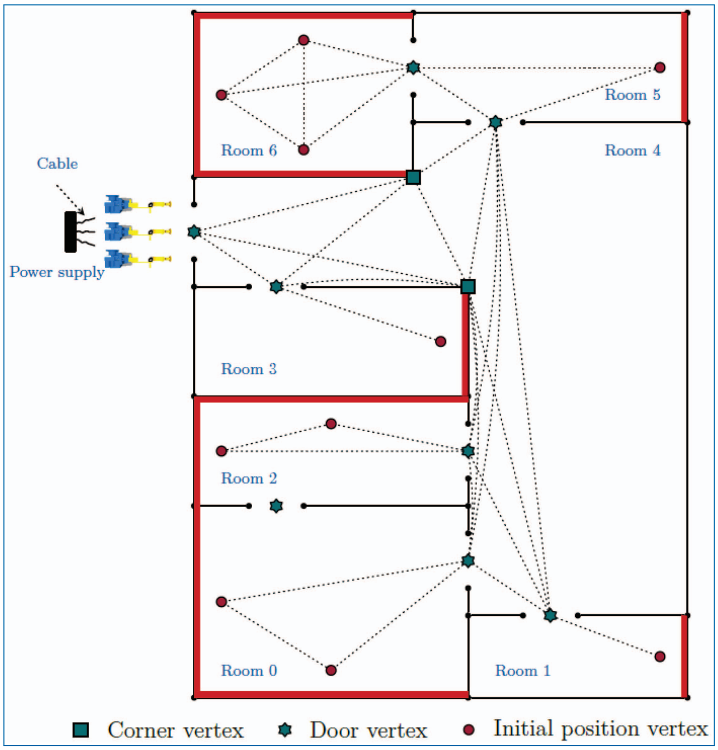


Figure 4.33. Graph generation.

WorkOnTask and *WaitAfterTask*. The robot itself without causing high computational effort can process each individual ScheduleActivity easily.

Beside the data format a graph is generated as a fundamental data model for the task planning of the second sub-problem. This graph represents the entire floor plan with the user-defined tasks, the corresponding initial position for each task and typical characteristics of the flat as doors and corners. The connection of the mentioned vertices generated the entire graph as it is shown in Figure 4.33 that is the

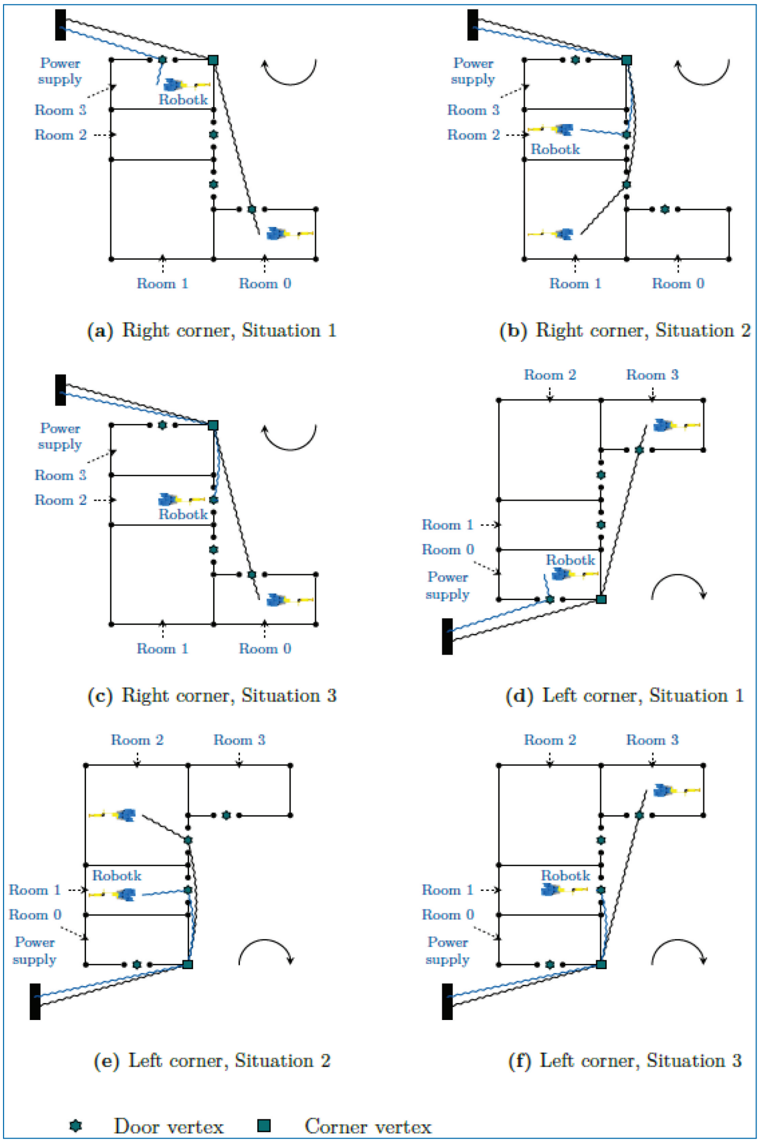

Power supply
Room 3
Room 2
Robotk
Room 1
Room 0
(a) Right corner, Situation 1
(b) Right corner, Situation 2
(c) Right corner, Situation 3
(d) Left corner, Situation 1
(e) Left corner, Situation 2
(f) Left corner, Situation 3
Door vertex
Corner vertex

Figure 4.34. Typical cable crossing problems with corners in hallways.

basis for the optimal solution of the second sub-problem. As the mobile manipulators of the Bots2ReC project are tethered to a fixed power supply during the removal process this cable handling becomes very important for the entire planning process. As an example there are six different situations shown in Figure 4.34 that can occur with embossed corners in hallways. These cable-crossing problems are formulated in logical rules and can be tested for each individual task and removal situation.

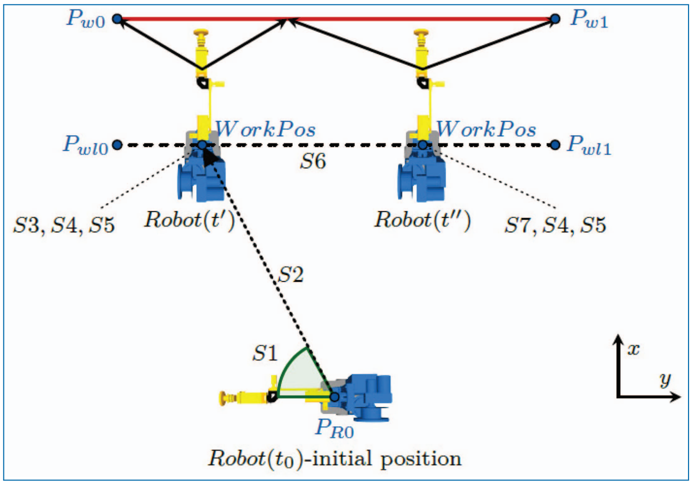


Figure 4.35. Solution for the first sub-problem.

Task Planning

The task planning algorithm solves the different problems that are defined in the previous chapter. For solving the first sub-problem for one individual wall the entire costs for the removal process are added and minimized. This means that the travelling costs and the costs for the trajectory execution are added up to one complete cost value. In this case it is the total amount of time that a robot needs to fulfil the whole task.

In Figure 4.35 a possible solution for one wall is given for a single robot. The wall is split into two areas that are processed after each other. For each area the optimal trajectory (either vertical or horizontal movements) is selected and the correct point of operation in front of the area is defined. Beside this also every translational movement of the mobile base is computed as well. As this is a simple minimization of an objective function the solution converges towards the global minimum and can be solved fast.

The solution of the second sub-problem is more complex due to the time-depending cable situation and the interdependent characteristics of the whole robot fleet. Mathematically it would be possible to set up an entire tree for every possible solution of the problem but this would be very time consuming. Having in mind that the environment on construction sites is unstable and sometimes even unpredictable another method is chosen for solving this problem: an auction between the different robots. This method is much faster compared to the absolute formulation of the problem and can be executed repeatedly with the current process information for adjusting the current plan if planned processes are getting out of scope.

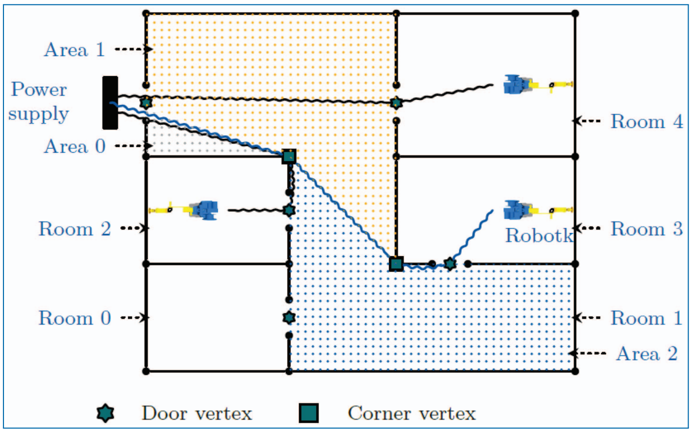


Figure 4.36. Removal situation within a normal flat.

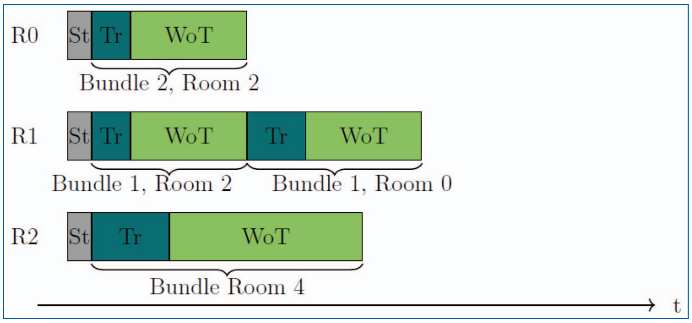
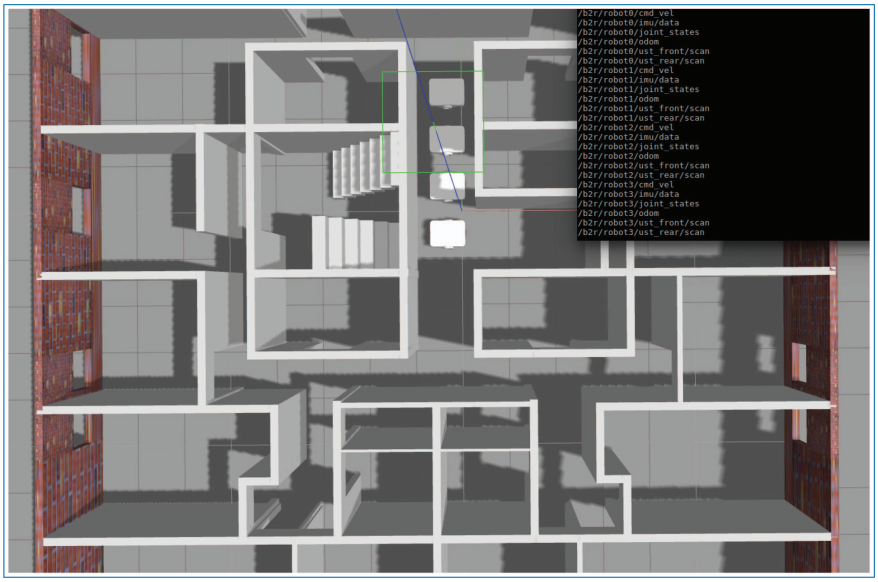


Figure 4.37. RobotsPlan.

This approach is leaned into the direction of reactive behavior that is often used for mobile robots and their task execution.

Out of this the auctioneer on the CPCS knows all the robotic properties and computes their individual costs for fulfilling each known task in the tasks list. Afterwards different heuristics can be used to bring tasks to auction (e.g. the closest task to the entrance, the room with the higher number of tasks, etc.) and each robot will set its cost as an individual bit. The robot with the lowest cost will win the task and can add the necessary *ScheduleActivities* to its schedule. This is done until the auctioneer sells every task. During every bit the cable-crossing conditions are checked and when a forbidden cable situation occurs, a high penalty value is thrown as a bit to ensure that this robot do not win this task. In Figure 4.36 a typical removal situation with three robots in a normal flat is given.

It also displays the different areas of the hallway that are stretched out by the robot cables when they start to touch corner or door vertices. A typical *RobotsPlan* and the individual schedules are shown in Figure 4.37.



```
/b2r/robot0/cmd_vel
/b2r/robot0/imu/data
/b2r/robot0/joint_states
/b2r/robot0/odom
/b2r/robot0/ust_front/scan
/b2r/robot0/ust_rear/scan
/b2r/robot1/cmd_vel
/b2r/robot1/imu/data
/b2r/robot1/joint_states
/b2r/robot1/odom
/b2r/robot1/ust_front/scan
/b2r/robot1/ust_rear/scan
/b2r/robot2/cmd_vel
/b2r/robot2/imu/data
/b2r/robot2/joint_states
/b2r/robot2/odom
/b2r/robot2/ust_front/scan
/b2r/robot2/ust_rear/scan
/b2r/robot3/cmd_vel
/b2r/robot3/imu/data
/b2r/robot3/joint_states
/b2r/robot3/odom
/b2r/robot3/ust_front/scan
/b2r/robot3/ust_rear/scan
```

Figure 4.38. Simulation environment for multi-robot strategy.

Task Execution

The last element within the task handling chain is the *Task Execution* during run-time. In addition, this chapter can be split into two sub-elements that are in control of the execution: the environment and the task managers. Following with the previous description of the multi-robot strategy problems, the behavior of each robot within the environment has a repercussion to other robots. For this reason, an environment manager is executed in the cpcs, which is in charge of updating the information available regarding all robots. Each robot will receive positions of other robots and more importantly, its costmap for autonomous navigation will be updated with the cables position from the power supply to all other robots. This measure will prevent any cable crossing, thus, trapped or fallen robots. In Figure 4.39, the costmap update of one robot is shown in a simulation environment representing an entire flat created for the Bots2ReC project (Figure 4.38).

Once the previously explained algorithm is executed, each robot will have a planned task, and the cpcs will update each robot costmap taking into account the tasks assigned. Note that a robot costmap will now include all robot cables as obstacles except its own cable. When a new task is assigned or the position changed, the robot will notify the cpcs which will update and publish all other costmaps. This communication is implemented using ROS services where the environment manager acts as a server and the robot as a client and when the service is called each costmap is published on each robot costmap topic.

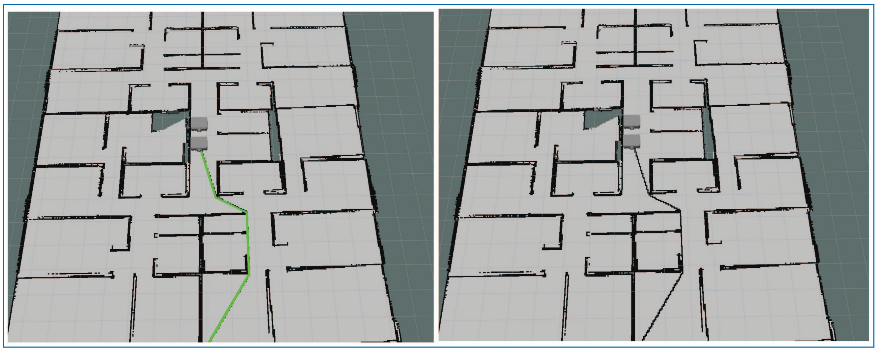


Figure 4.39. Costmap update with cable as obstacle.

Concerning the task manager, different *Modes* are used to establish robust robotic behavior that are managed by a typical *StateMachine* that will survey the global system status. For the Bots2ReC robot the following *Modes* are created:

- Boot
- Calibration
- Teleoperation Base
- Automation Base
- Teleoperation Arm
- Automation Arm
- Error
- Shutdown

Each Mode contains so-called *hierarchies*, which can affect the robotic behavior for each mode directly. The *AutomationBase* mode can be used to explain this strategy. During the autonomous movement of the mobile base to a new goal the whole robotic arm needs to be in the *driving position* to ensure a stable movement throughout the whole movement. If the arm is not in this position it is moved towards it before the base starts moving. Furthermore, every arm movement is prevented during the autonomous movements of the base. When a button on the user interface or the remote control is pressed the mobile base can flip directly into the *TeleoperationBase* mode. All those rules and behaviors are stored within the hierarchies that are strictly linked to each mode.

For simplification of the requirements for changing between different modes a central state machine on each robot takes care of the entire robot data. This means that, for example, the state machine will survey the hardware status of the robot and will generate simple Boolean flags that can be read directly by the execution modes. This reduces the logical handling of the hierarchies and can make the logic

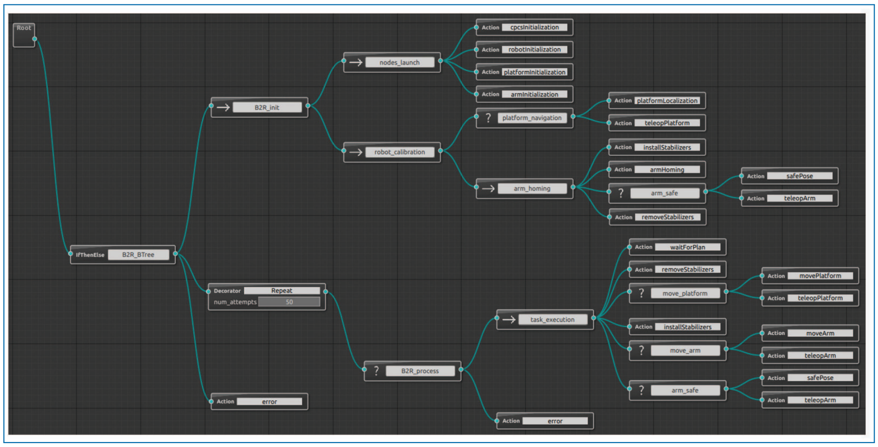


Figure 4.40. Behavior Tree for Bots2Rec robot task execution.

much more understandable. This state machine is implemented using a structure called Behavior Trees. They are extensively used in gaming and AI for representing an agent behavior and are far more readable than classical state machines as a lot of connections between states are implicit within the behavior tree definition. The execution order is from left to right and up to down. Basically there are three types of nodes: Actions, Controls and Decorators.

- *Action*: it will define a specific execution *Mode* or a simple action to develop.
- *Control*: these nodes define a tree behavior, how its child nodes will be executed. For example, we use *Sequence* () (fail if one of its children fails) or *Selectors* (?) (success if one of its children succeeds).
- *Decorator*: with one only child, typically used for invert its child result or repeat it until fail or a number of attempts.

The previous explanation was necessary to understand the Bots2ReC state machine, defining the behavior of a single robot, represented in Figure 4.40.

The first node (B2R_Tree) is a *control node* which manages the higher level of the behavior algorithm. It simply tells: "**If** the initialization (boot and calibration) of the robot succeeds, **then** repeat the robot process until failure. Raise *error mode* to manage this situation otherwise (**else**)". In the task execution there are nodes representing simple waiting actions (waitForPlan, installStabilizers...) and other nodes that describe a movement execution according to a specific *Mode* (movePlatform, moveArm…). When moving the platform or the arm autonomously fails, there's always the possibility of operating the robot manually through the connection to a teleoperation node.

4.3.3 Outlook and Future Work to be Carried Out

The content of this deliverable describes the algorithms at different levels or layers developed to manage and optimize the work of the set of multiple robots to minimize the required time to process the construction site surface according to the different constraints, for example the different characteristics of every robot (tool dimensions, reach, etc.) and the minimization of the crossing of tethering lines.

Once planning algorithms has been defined, developed and tested within the simulation environment, next step is to bring them to the real scenario after second robot unit is built and fine-tune and adapt them according to the physical needs and also considering the results of the different tests being conducted with the robot unit V1 at the testing site.

4.3.4 Summary and Conclusion

As main conclusion from this deliverable we have the fact that the necessary planning algorithms and task allocation basis has been set and tested within a simulation environment. That basis is going to be continuously updated and adapted according new inputs from other task in execution and will finally be implemented within the final demonstrator of the project.

Task 6.5 Development of the central process control system

In reporting periods 1 and 2, the planning procedure and the methods implemented therein for the assignment of tasks to individual robots in a robot fleet were developed. An excerpt of the features contained therein is listed here:

- Consideration of different fleet sizes
- Consideration of different robot types
- Consideration of tethered robots
- Solution for different floor plans
- Optimization of the removal trajectory
- Possibility of continuous re-planning during the removal job

Apart from a continuous improvement during the continuation of the planning process, the parameterization for the second prototype could be carried out and successfully integrated into the existing planning process. In the following an exemplary solution for a given apartment is shown, which is solved by the robot V2:

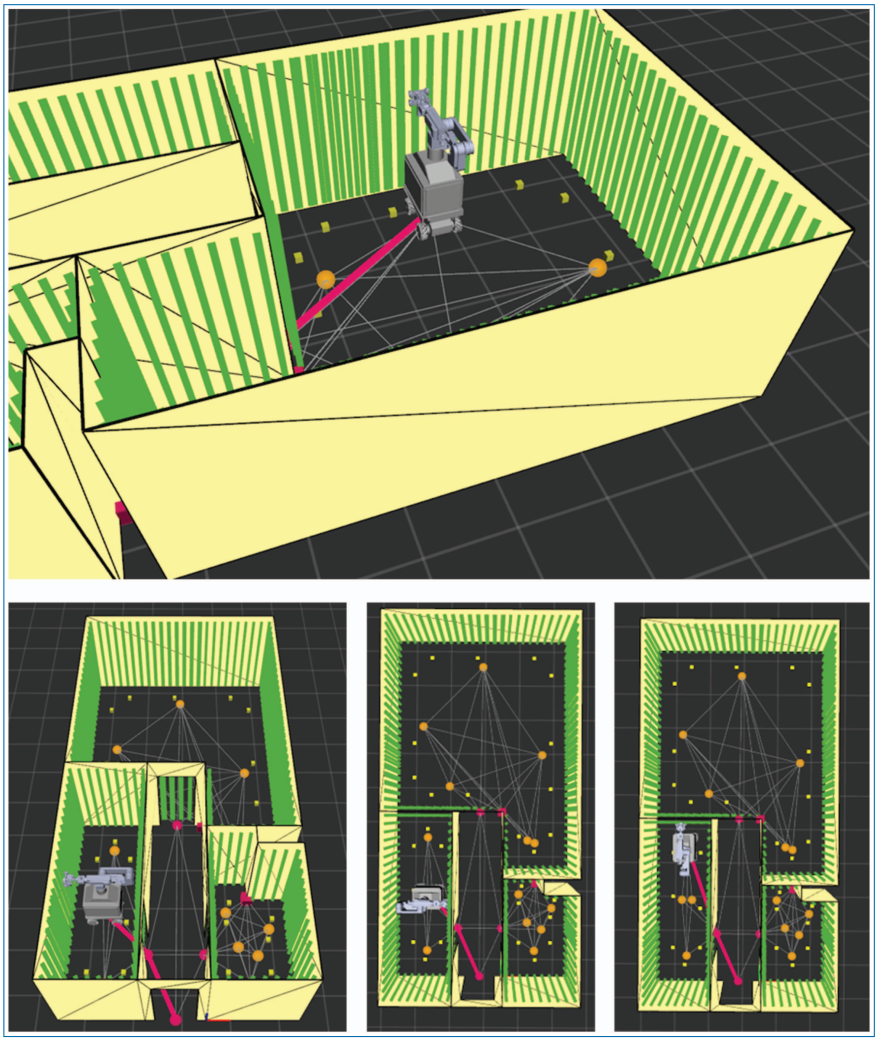


Figure 4.41. Removal plan for single flat by V2 mobile manipulator.

4.4 Mapping and Localization

Task 6.1 Environmental perception and mapping

A comparison of laser-based and RGBD SLAM algorithms has been done. This included a set of experiments developed to evaluate the performance of common SLAM algorithms. We selected GMapping, Hector Mapping, Cartographer and RTAB-Map due to their availability and relatively easy setup in ROS. For all the experiments a Summit XL Steel Robot was used (RB-Kairos base), i.e. the robot mounts 2 Hokuyo UST-20LX laser range finder sensors in the robot edges. The

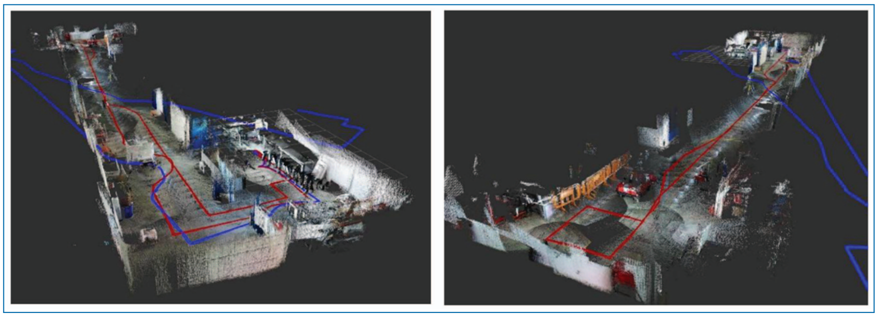


Figure 4.42. Exemplary comparison of RTABMap and the Odometry.

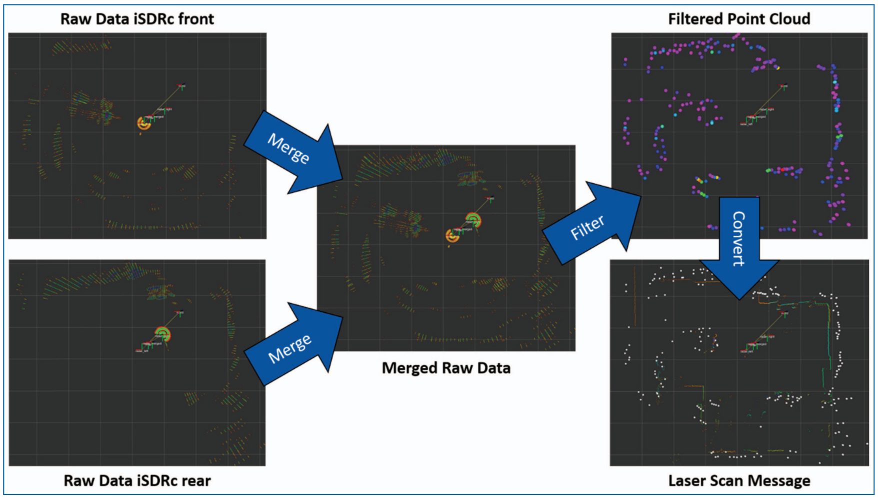


Figure 4.43. Radar Processing steps for localization.

analysis done compares the advantages and limitations of the listed methods and shows the accuracy of the estimated robot pose, as some performance parameters as the CPU or memory usage. Experiments to improve odometry estimation have also been carried out. Different algorithms and devices have been selected to provide alternative odometry sources that can improve the robot wheel odometry. In these experiments the robot standard and non calibrated odometry source was used and the errors of the estimated odometries compared with the ground truth. An example of using RTABMap (red) in comparison to Robot odometry (blue) is given in the following figures.

In addition to the above mentioned findings based on existing algorithms using lidar data, the 2D perception by radar data was further advanced. The consortium had agreed on the redundant solution to localize by radar and lidar in case the problem of the suction quality of V1 could not be eliminated completely. In this way a permanent localization could be achieved even with no 100% extraction despite

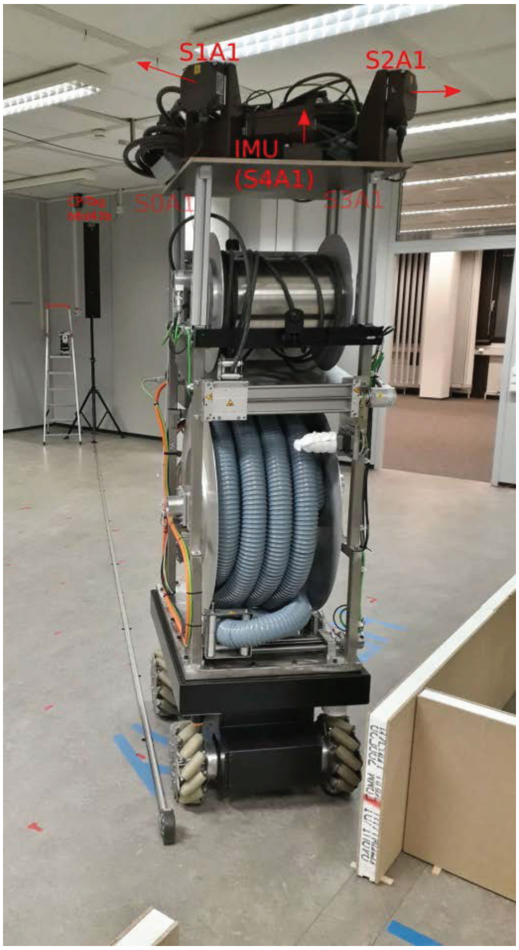


Figure 4.44. Mounting positions and orientation of iRTT-AUs and CP-Tag.

dust particles. In this sense, the results for the radar slam have been extended to a pure localization. This means in detail that the common adaptive monte-carlo localization (AMCL) should be carried out with corresponding radar data. This idea fits seamlessly into the semantic mapping step of the overall process, as it automatically provides a lidar-based 2D map of the environment as a result. A later localization by means of radar signals on this map will considerably reduce the setup time of a robotic system and has been targeted accordingly. For this reason, a corresponding fusion and filter algorithm has been developed for the advanced iSDRc from Indurad, which transforms the radar signal into a classical point cloud based on characteristics of the amplitude frequency response. Afterwards the predefined interface in ROS can be used for ACML. In this way, the advantages of the

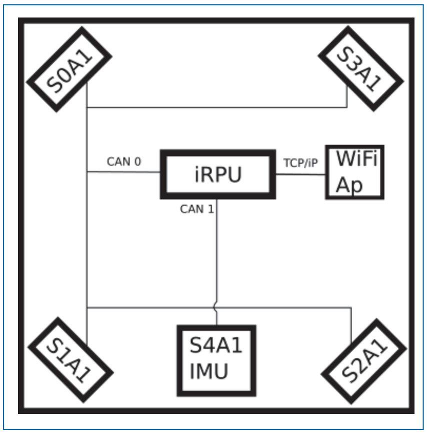


Figure 4.45. Schematic antenna set up.

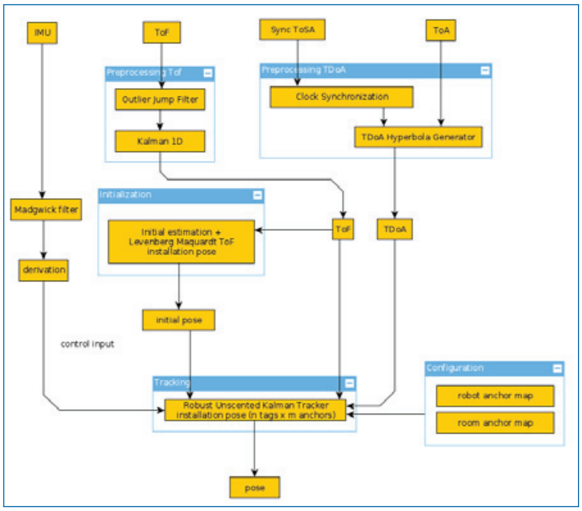


Figure 4.46. Advanced iRTT data flow.

already extensively tested localization method and the penetration of radar signals are efficiently combined.

- Integration of TDoA: measuring reception time differences of radio signals in the 6.4 GHz ultra-wide band with clock synchronization between receptionists over the air.
- New mounting positions on top of the robot as shown in Figures 4.44 and 4.45 with sideward facing antennas for better line of sight to surrounding tag-antennas to reduce multipath effects.

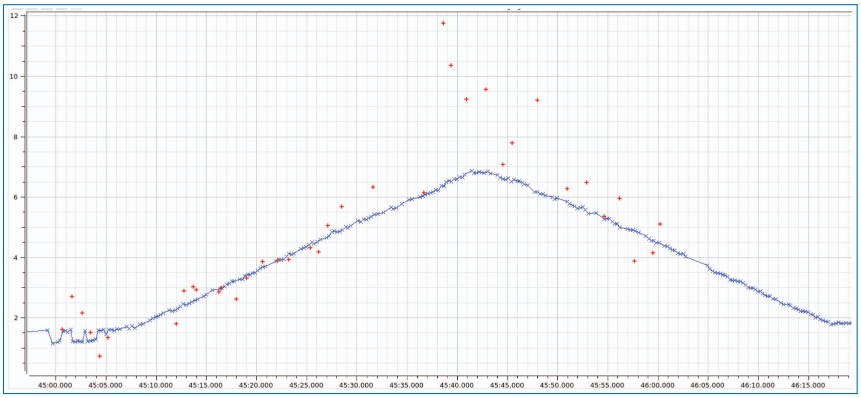


Figure 4.47. Distance points for straight traversal towards CP Tag with front facing antennas. Blue: Distance based on UWB Time-of-Flight measurements. Red: Distance based on ISM Phase Difference.

- Diverse redundant distance measurement system based on detection of radio wave phase differences in the 2.4 GHz ISM (Industrial Scientific Medical) band (at 2,443GHz center frequency).
- Integration of inertia measurement unit (IMU) for supportive pose estimation with Madgwick filter algorithm.
- Robust Kalman Observation Model, with improved convergence of estimated positions and orientations (the pose).
- Integrated Ornstein-Uhlenbeck Process Model (IOU), that deals with stochastic nature of the distance measured with radio ranging systems.
- Improved non-line of sight (NLOS) probability estimation, based on UWB NLOS metrics.

All above mentioned improvements (except for the phase difference measurements in the above mentioned ISM band) and their interconnectivity can also be seen in the following figure.

4.5 Navigation

To verify the correct operation of the robotic unit V1 in terms of navigation and operation, it has been decided to carry out simulations in a virtual environment that recreates the conditions of a real environment. The Gazebo simulator has been chosen, as it offers the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. It is capable of simulating a population of robots, sensors and objects in a three-dimensional world. It generates both realistic sensor feedback and physically plausible interactions between objects.

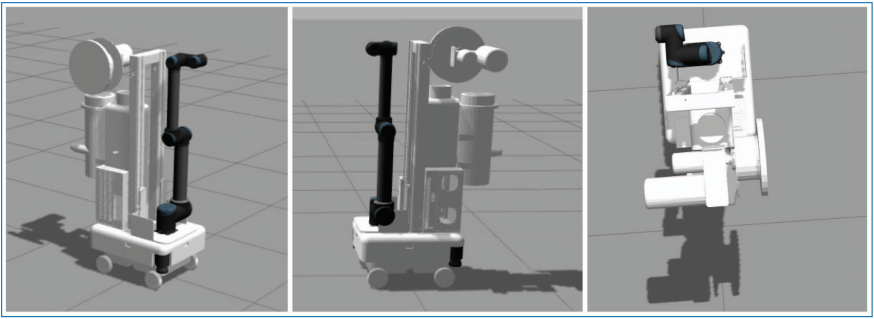


Figure 4.48. Gazebo 3D mock-up model of the robotic unit V1 system in ISO, side and top views.

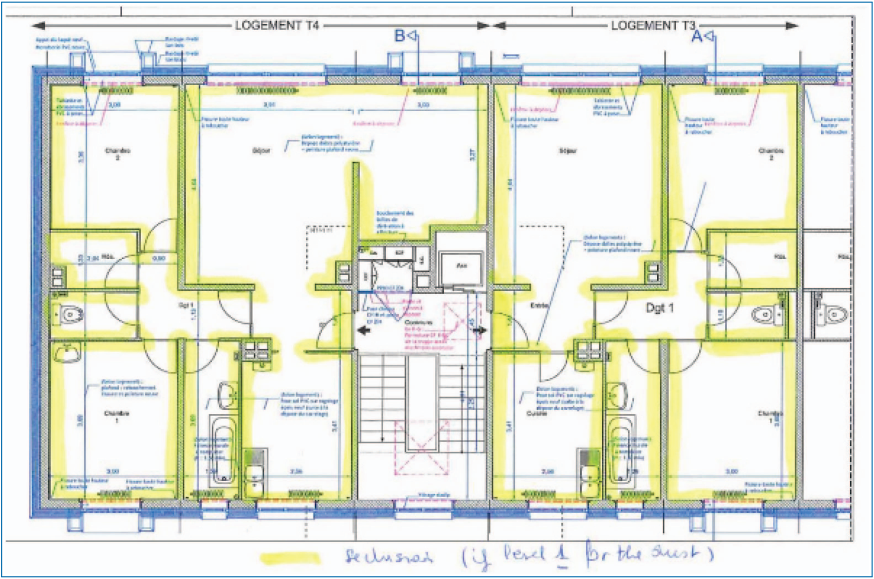


Figure 4.49. Model of a building level with two flats, provided by Bouygues.

A mock-up model of the robotic unit V1 was created with Gazebo to perform simulations in the ROS environment. This model has been generated from the CAD model provided by Telerobot Labs and Robotnik Automation (Figure 4.48).

The environment for this simulation was based on the map of a building level provided by BYCN (Figure 4.49). In order to recreate the simulation environment, it has been used the Gazebo tool building editor. This tool generates a file **.world*, that contains a description of the world to be simulated by Gazebo. It describes the layout of robots, light sources, user interface components, and so on. The world file can also be used to control some aspects of the simulation engine, such as the force of gravity or simulation time step.

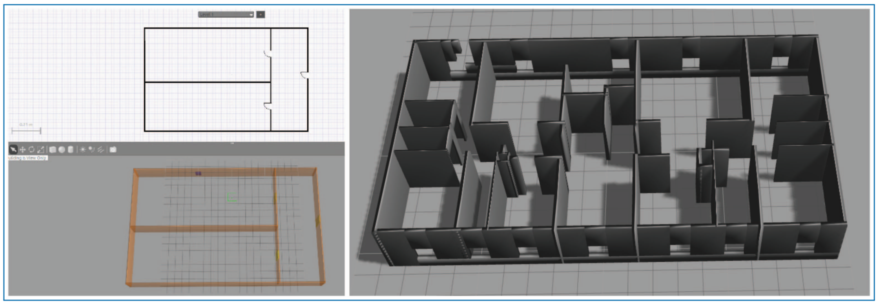


Figure 4.50. Model of a flat created with the Gazebo building editor.

Gazebo world files are written in XML, and can thus be created and modified using a text editor. To simplify those tasks, the Gazebo building editor plug-in has been used (Figure 4.50). This plug-in is very useful to create a scene from scratch. Once the drawing is finished, the world file can be saved for future use into the simulation environment at any time.

To simulate the model in ROS environment, a URDF file (Unified Robot Description File) was generated from the CAD model of the robot. This file contains the physical and mechanical description of the robot, such as the type of joints, the coordinate system of each link, collision elements, and so on. The URDF file has been generated with the SolidWorks to URDF Exporter plug-in of SolidWorks.

Once the robot model for ROS and Gazebo was obtained, a plug-in was added to the Gazebo model to control the movements of the platform within the simulation environment. The chosen plug-in was *libgazebo_ros_planar_move*, for moving the robotic unit in a horizontal plane using geometry_msgs/Twist. The plug-in works by imposing linear velocities in (XY) and an angular velocity (Z) to the controlled bodies at every cycle.

4.5.1 Navigation Tests

In these tests, the robotic unit mock-up model V1 was added into a virtual environment reproducing a flat and, thanks to virtual 2D laser sensors, the proximity of the obstacles around to the mobile platform could be measured (Figure 4.51).

For a better autonomy, the capacity of the robot to re-create a map by itself from its environment was investigated. To generate this map, the *gmapping* ROS package has been used. This package was able to generate a 2D map from the data of the lasers sensors of the platform, as the platform moved through the environment (Figure 4.52). The navigation during the generation of 2D map was tele-operated manually.

Once generated the navigation map for the platform, a simple navigation test has been performed (Figure 4.53). A user-defined goal point was indicated in the

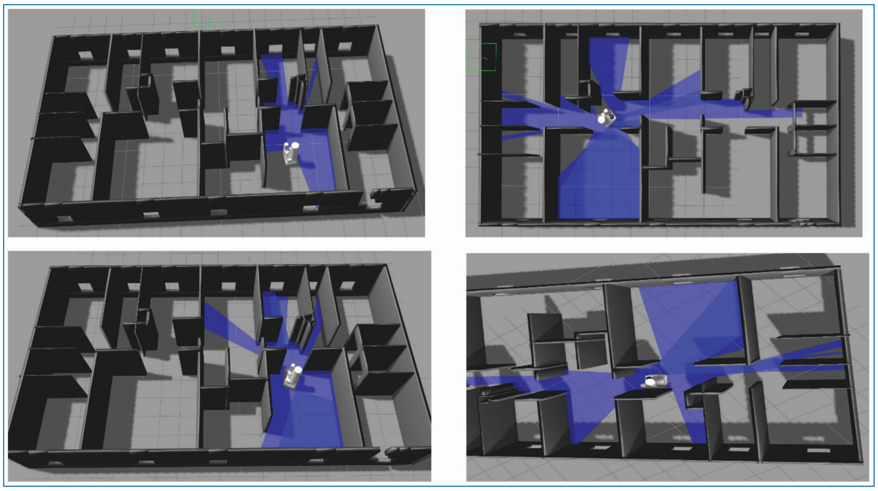


Figure 4.51. Virtual robotic unit (in white) obtaining data from the virtual 2D laser sensors.

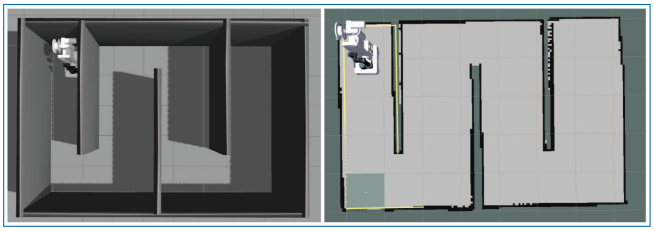


Figure 4.52. Map generated from Gazebo's simulated environment.

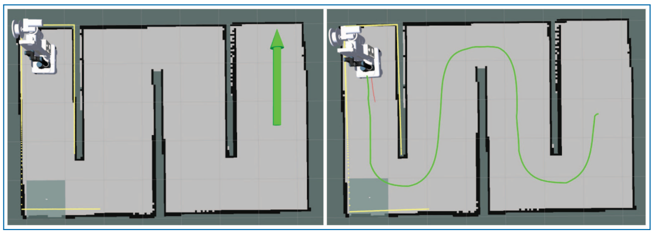


Figure 4.53. (a) Goal for the robotic unit. (b) Path planning by the algorithm of navigation.

map, and then the planner could plan an optimal and free collision path, that was sent to the virtual controller to execute the calculated trajectory.

4.5.2 Bent Corridor Tests

To verify that the robotic unit V1 was able to turn in narrow bent corridors, different tests have been performed with varied sizes of corridors (700x700, 700x800,

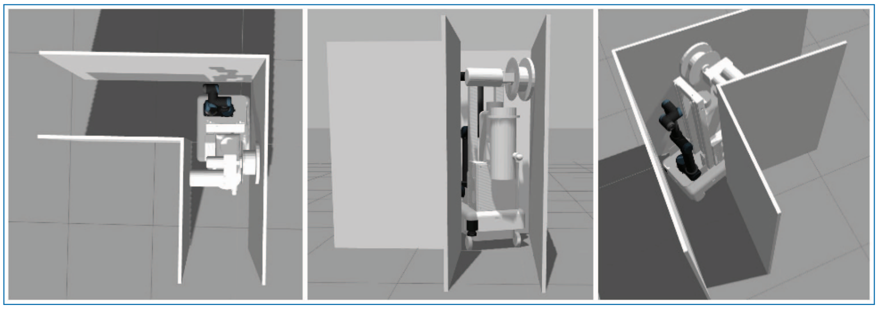


Figure 4.54. Robotic unit turning in a 700x700 corridor.

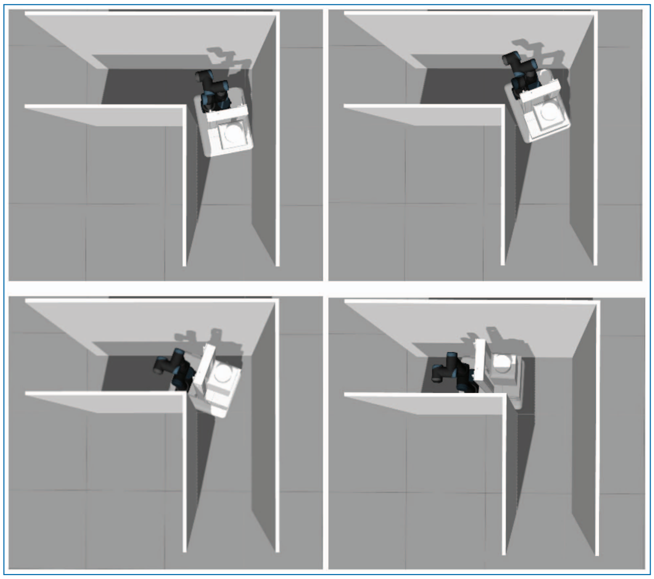


Figure 4.55. Simplified robotic unit turning in a 700x700 corridor.

700x900, 800x800). The first figure indicates the width of the first part of the corridor, whereas the second figure is for the width of the second part of the corridor that connects to the first one. A Gazebo model has been created for each corridor. The navigation along the narrow bent corridors has been made manually to simplify the complexity of the task.

4.5.3 700x700 Corridor Test

In the case of 700x700 corridor, the robotic unit has not been able to turn in the corridor, getting stuck while turning (Figure 4.54).

It was decided to remove the aspiration unit and the cable reel, as shown in Figure Illustration, since they are outside of the footprint of the mobile platform, and check whether the robotic unit could turn in the corridor.

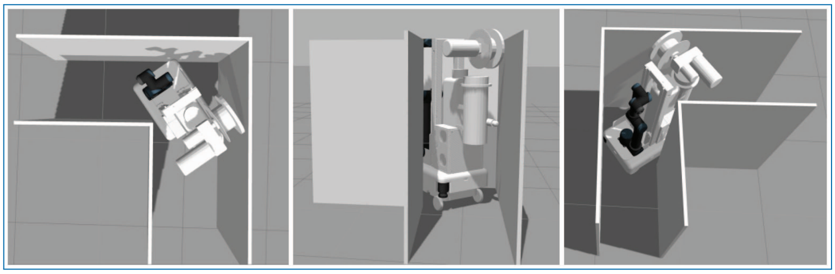


Figure 4.56. Robotic unit turning in a 700x800 corridor.

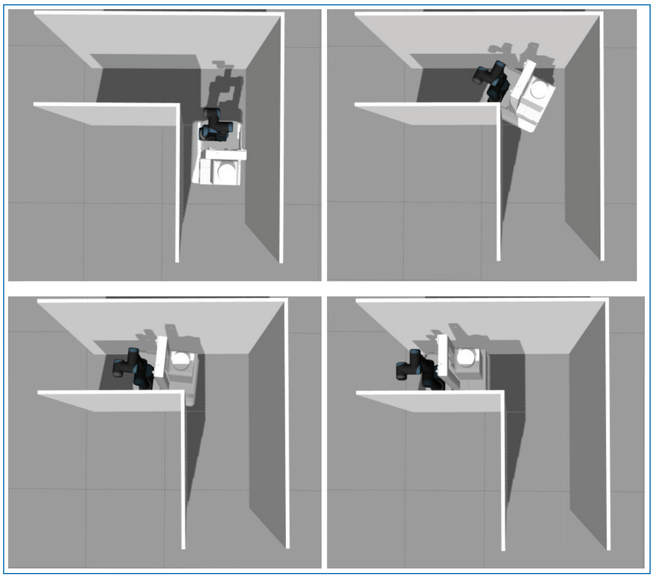


Figure 4.57. Simplified robotic unit turning in a 700x800 corridor.

Without the aspiration unit and the cable reel, the robotic unit has been able to turn the 700x700 corridor, although very tightly along the corridor.

4.5.4 700x800 Corridor Test

With the corridor of 700x800, the robotic unit has not completed turned, because the cable reel stands out of the footprint on the platform, preventing the rotation.

A test without the cable reel worked perfectly (Figure 4.57).

4.5.5 700x900 Corridor Test

In the corridor of 700x900, the robotic unit has been able the turn in the corridor correctly, this time with the cable reel and the aspiration unit as it was in the initial setup (Figure 4.58).

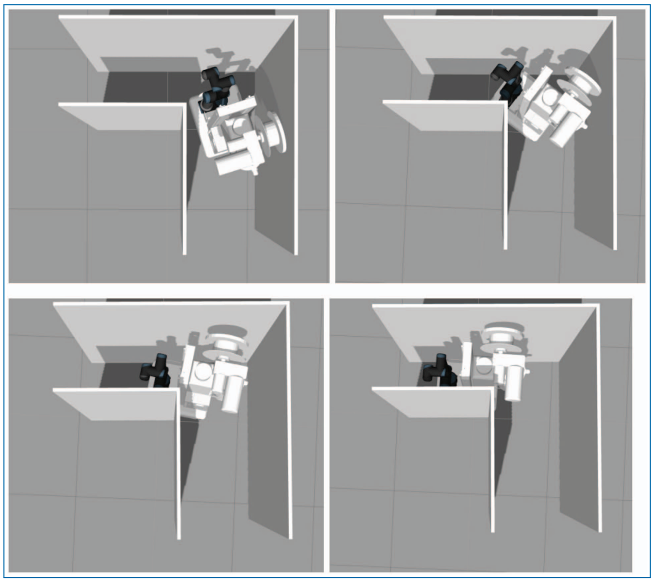


Figure 4.58. Robotic unit turning in a 700x900 corridor.

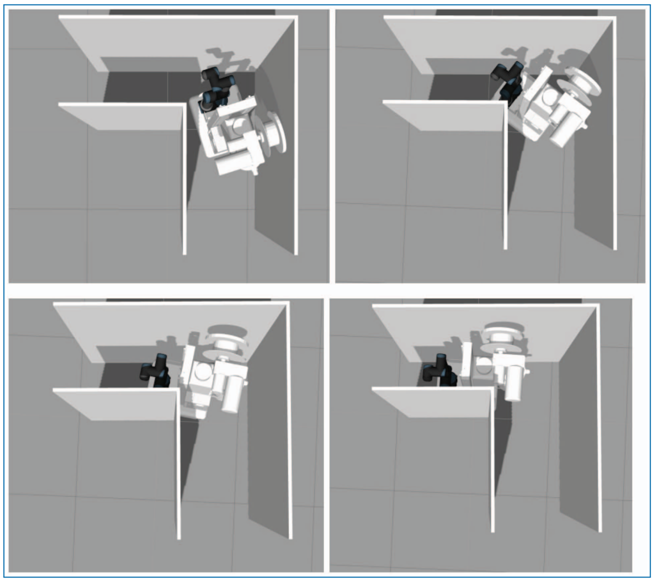


Figure 4.59. Robotic unit turning in a 800x800 corridor.

800x800 corridor test

In the corridor of 800x800, the robotic unit has been able to turn in the corridor correctly, this time with the cable reel and the aspiration unit as it was in the initial setup.

Table 4.2. Summary of bent corridor test results.

Bent corridor Crossing	With robotic unit V1 system	With simplified robotic unit V1
700x700 corridor	No	Yes
700x800 corridor	No	Yes
700x900 corridor	Yes	–
800x800 corridor	Yes	–

4.5.6 Summary of Bent Corridor Test Results

Table 4.2 contains the results of the tests performed in different bent corridors. It is indicated if the platform could turn in the corridors with the complete robotic unit V1 system, or with a simplified robotic unit with less protruding elements.

4.6 Stability Observations

Deliverable 2.1
D2.1 Preliminary simulations of the robotic units

Authors and contributors:
CARRASCO JIMENEZ Ramon, EURECAT
DETERT Tim, RWTH
FAUROUX Jean-Christophe, SIGMA Clermont
MARAJE Siddharth, SIGMA Clermont
MARTIN Daniel, EURECAT
MOHY EL DINE Kamal, SIGMA Clermont

4.6.1 Deliverable in the Context of the Work Program

Work package 2 (WP2) aims to support the other work packages by creating relevant scientific and engineering models in order to bring clear answers to the technical choices to be made during the design process of the robotic unit.

Deliverable D2.1 is the first result issued from WP2, with the ambition to provide simplified models of the robotic unit that is being designed at the beginning of the project. The models constitute the foundations of a design and control simulation environment to be used all along the project.

The deliverable D2.1 takes input from "*Task 1.1 – Specification of asbestos removal tasks*", particularly for the specification of the environment to clean, including narrow corridors, and "*Task 1.2 – Initial trials of operational process*", specifically for the tool requirements and manual productivity.

D2.1 provides a preliminary testing environment with simplified mock-ups of the robotic units implemented in geometric/kinematic/dynamic simulation tools prior to the final design of the robot. Based on the requirements, a tall robotic unit has been considered for preliminary evaluation, made up of a manipulating arm on a mobile platform. A virtual environment was reconstructed from a digital map of the testing area, adding attributes from Task 1.1 to represent the spatial dimensions of the scenario.

The parametrized models developed in D2.1 provide useful design rules for "*Task 2.2 – Stability model and development of mechatronic stabilizers adapted to the task and environment*" and "*Task 2.3 – Virtual scenario of a representative asbestos removal use case*", which will include a virtual mock up and refined models. Further co-simulations will be implemented to address specific calculations like the stability modelling in Task 2.2.

4.6.2 Methods and Work Carried Out

This deliverable required extensive exchanges with the partners for defining and double checking the requirements on the robotic unit and performing the three main following tasks:

- **Extracting a core simplified subset of requirements** critical for feeding the first models. These requirements were gathered during several meetings, including a visit of Bouygues at Clermont-Ferrand (05/04/2016) and a visit of several flats under or after their cleaning process, organized by Bouygues at Clermont-Ferrand (01/06/2016), which showed a realistic environment and its specific difficulties, particularly in terms of confined and restricted space.
- Creating **preliminary models** and extracting results to feed the discussion. The main constraints to consider for the robotic unit were:
  - tool reachability for tall walls of 3 m height, ceiling at 3 m and surface all around the mobile base;
  - non-collision robot-environment in narrow corridors and while passing through doors;
  - singularity avoidance for future control;
  - quasi-static stability in all the workspace
- Exchanging with partners regularly for **banning or confirming architectural choices** during web conferences, phone meetings and brainstorming

sessions. A particular contribution was made for defining the **arm architecture**. Finally, three kinematics were modelled and tested.

The considered approach for analysis was to develop several types of models:

- **Geometrical models**, mostly for assessing tool reachability and collision avoidance. The models can be of two types:
  - Skeleton models, reducing links to a simple average line. The models are represented with the Geogebra software [Geogebra, 2016], an interactive mathematical tool that provides intuitive modelling method both in 2-D and 3-D through algebraic expressions and geometric constructions. Geogebra allows model parametrization and has been proved to be particularly suitable for dimensional synthesis [Prause *et al.*, 2015].
  - 3D CAD models for robot representation, simulating tool paths and avoiding collisions. The links have a volumetric simplified shape. One big advantage of CAD is that dimensional constraints can be used both on the joints (direct geometric model) and on the tool (inverse geometric model). The assembly constraint solver manages them simultaneously, which is particularly useful for redundant models, where the redundant joints can be fixed in the direct geometric model and the remaining joints are positioned by solving the inverse geometric model. The CAD models were made with Catia V5 / V6 (SIGMA) and SolidWorks 2016 (EURECAT)
- **Quasi-static models**, including the masses of the bodies and contacts on the ground, to evaluate the static stability margin of the robotic unit. Algebraic equations were added to a Geogebra model for barycentric calculations and stability margins.
- Preliminary **multi-body models**, including contact forces and inertias, to be re-used in Task 2.3 for evaluating the complete dynamics of the robotic unit during its motion and cleaning tasks. Adams 2013 was used and will take its full interest for Task 2.3 as it allows modelling flexible bodies and as we think the stiffness issues in the robot may compromise its precision.
- **Controllable models**, into a complete realistic environment based on ROS, ready to represent and control the robotic unit V1. This work was performed by EURECAT, as the main integrator of the future prototype V1. A ROS tutorial was also organized by SIGMA in Clermont-Ferrand for improving the ROS competencies of the whole consortium (31/05/2016-01/06/2016).
- **Preliminary simulations of the robotic units**

This section provides a summary about the initial analysis and preliminary models of the robotic unit. It includes requirement extraction, preliminary dimensioning, stability analysis, study of three different arm architectures, force analysis, preliminary dynamic and productivity analysis and finally a controllable model of the robotic unit in its environment.

4.6.2.1 Requirements and constraints on the robotic unit

A core simplified subset of requirements critical for the first models can be found below:

- **Environment** to be cleaned: type of surfaces to clean (plastering, spattling compound, paint, with an option on ceramic tiles and glue), geometry (horizontal/vertical/inclined, flat or curved with a curvature radius of 1.5m) and positioning of the surfaces, minimal door dimensions (70 cm x 2 m), constraints due to seclusion, obstacles and tight environment (particularly 90° angles in narrow 70 cm wide corridors);
- **Robotic unit settings**: maximal acceptable dimensions (Width <0.6 m, Length <0.8 m, Height <1.8 m) and mass (300 kg), approximate masses of the main components, required workspace (reaching a ceiling at 3 m);
- **Cleaning process** and associated tools: typical nature and dimensions of existing tools for manual cleaning (rotating disks of 150-300 mm of diameter, grinding cylinders), involved forces (typically 80 N of normal force, but depends on the tool).

The robotic unit integrates a robotic arm equipped with an abrasive tool mounted on a mobile base will perform asbestos removal operation. The design of this robotic arm will primarily be influenced by the following constraints:

- **Number of Degrees of Freedom (DoF):** For asbestos cleaning scenario, 5 degrees of freedom are needed at least considering the scenario of curved walls: 3 translational DoF and 2 rotational DoF, with axes perpendicular to the tool axis.
- **Reachability:** The dimensions of the room to be cleaned put constraints on the link lengths of the arm. According to the list of specifications, the height wise dimension of the wall to be cleaned is maximum 3 meters. Thus, the robotic arm must be able to reach the top, ground and corner points on the wall.
- **Payload capacity:** The required payload carrying capacity is 50 N maximum since the weight of the tool is around 5 kg. Additionally, the removal process requires at least 80 N of normal force.

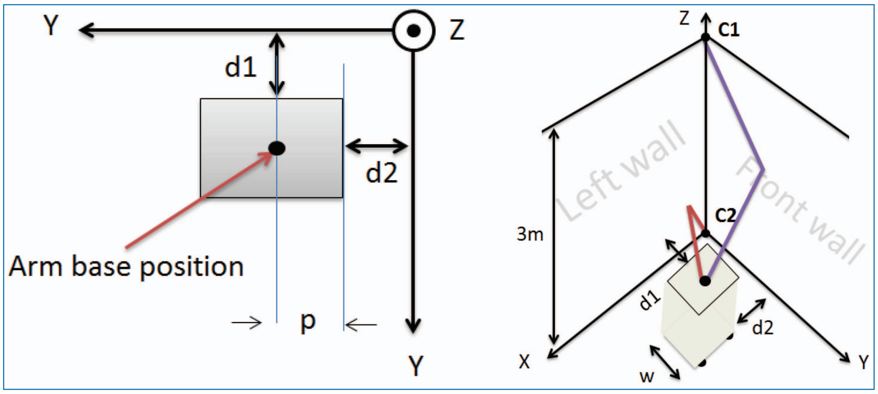


Figure 4.60. Constraints of the arm modelling. (a) Position of the robot with respect to the walls. (b) Constraint points of the arm to reach.

- **Stability of the Robotic Unit**: The robotic unit, while performing removal operation, can lose its stability due to its variable pose and the external forces applied on the robot. Depending on the considered forces, two approaches can be considered:
  - **Static stability:** As the mobile platform has a vertically elongated aspect ratio and is supporting a robotic arm that can overhang, the structural stability of the total system is dependent on the arm structure and pose. This quasi-static model is suitable for slow cleaning motions and mainly considers the wheel-ground contacts.
  - **Dynamic stability**: During cleaning operation, interaction of the robotic unit with the environment generates dynamic forces (normal and tangential reaction forces, frictional forces) on the robotic unit, both on the tool and supporting wheels. Dynamic effects will also be generated due to the high speed cleaning motions with weighting parts. This type of model takes all the forces into account and is suitable for simulating a fast cleaning.

4.6.2.2 Preliminary geometric dimensioning

The first trial of the arm modelling was done based on the reachability approach. The distances of the robot from the left and the front wall, d1 and d2 respectively, are adjusted at their minimum value and considered as constraints (Figure 4.60 Illustration(a)) in order to minimize the lengths of the links of the arm. Assuming the arm base height is h, the constraint for the robot is to reach the farest points in its workspace C1 and C2 is shown in Figure 4.60 Illustration(b). The width of the mobile base is defined as w. Variable p defines the position of the arm base from the front edge of the robot.

According to the robot specifications, the width of the mobile platform is 0.6 m, its length is 0.7 m and the height is 1.5 m. Distances d1 and d2 are fixed at 0.3 m and the arm link lengths are minimal when based on the distance from the arm base to the furthest point, then divided by two since we have two links. A small offset is then added to links to avoid the singularity on fully extended arm. The inverse geometric model validates the lengths by reaching C1 and C2. Hence, the needed link length is 0.95 m for this simplified model. A similar approach will be used for the more detailed models studied in Section 3.3.

4.6.2.3 Stability Analysis of the robotic Unit

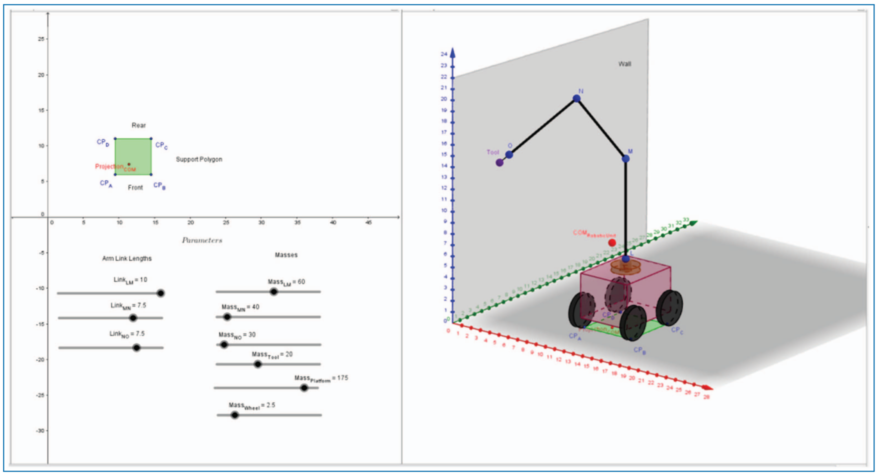
The effect of the static and dynamic factors mentioned in section 3.1 on the stability of the robotic unit must be analysed in order to perform removal operation efficiently. However, for preliminary selection of the arm, the main focus is to assess only the static stability of the system.

To perform the static stability analysis, a suitable stability margin should be considered. For the preliminary analysis, a static stability margin proposed by McGhee and Frank – '***projection of the centre of mass in the support polygon***' [McGhee and Frank, 1968] is used as a stability margin. In order to evaluate the stability, robotic unit must be modelled with realistic weight distributions of its components. The Geogebra software is used for building the skeleton model of the robotic unit.

For stability analysis, the design parameters are ***link lengths*** of the robotic arm, ***dimensions*** of the mobile platform, ***positioning*** of the robotic unit with respect to the wall, ***masses*** of the components from robotic arm and mobile platform.

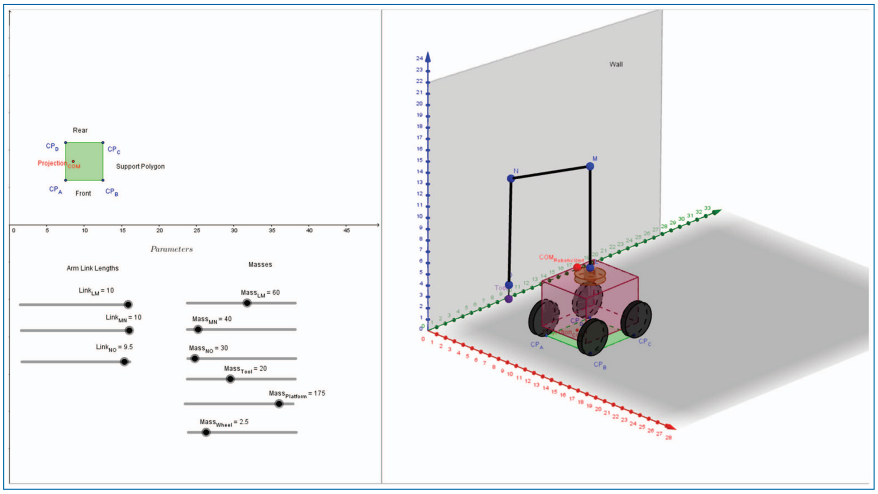
4.6.2.4 3-D skeleton model of the Robotic Unit

A 3-D model of the robotic unit was created in Geogebra and the centre of mass of the total system was evaluated (Figures 4.61–4.62). The objective of the model was to build a **quasi-statically balanced model without external forces for the approximate evaluation of the stability margins and structural design**. The design parameters for this model are arm link lengths (LM, MN, NO)**,** their masses and the mass of the mobile platform. This model allows assessing the effect of design parameters on the variation of the centre of mass of the robotic unit. It revealed that in certain arm configurations, the projection of the centre of mass is close to the edge of the support polygon, as can be seen on the green supporting rectangle on the top left part of Figures 4.61–4.62. This indicates increase in the toppling moments and may cause instability of the system. Thus, the arm configuration and the weight distribution in the arm architecture is a critical factor while achieving stability.



$Link_{LM} = 10$
$Link_{MN} = 7.5$
$Link_{NO} = 7.5$
$Mass_{LM} = 60$
$Mass_{MN} = 40$
$Mass_{NO} = 30$
$Mass_{Tool} = 20$
$Mass_{Platform} = 175$
$Mass_{Wheel} = 2.5$

Figure 4.61. 3D Model of the robotic unit (cleaning a wall).



$Link_{LM} = 10$
$Link_{MN} = 10$
$Link_{NO} = 9.5$
$Mass_{LM} = 60$
$Mass_{MN} = 40$
$Mass_{NO} = 30$
$Mass_{Tool} = 20$
$Mass_{Platform} = 175$
$Mass_{Wheel} = 2.5$

Figure 4.62. 3D Model of the robotic unit (cleaning ground).

The main advantage of this 3D model is that it can cover several cleaning scenarios of the ground/walls/ceiling. If components of the robotic unit change during the project, this model should always be used for a fast evaluation of the stability margin of the modified robotic unit.

4.6.2.5 Stability analysis of robotic unit based on different arm architectures

For the arm dependent stability analysis of the robotic unit, several 2D models of the robotic unit with different arm architectures were built for a simplified and focused analysis of the arm. Stability of the robotic unit was studied during an operation of the robotic unit with the tool moving in the plane of the wall. To quantify stability, the shift in the projection of centre of mass of the overall system from its centre position is expressed in a non-dimensional way, in percentage. Thus, the longitudinal stability can be expressed by the P_{Stab} variable as indicated by equation (1). When P_{Stab} is 100%, the stability is maximal. The robot is at the limit of instability for $P_{Stab} = 0\%$ and unstable under 0%.

$$P_{Stab} = \left(1 - \frac{Dist\left(COM - Centre_{Supportpolygon}\right)}{Length_{Supportpolygon}/2}\right) \times 100 \qquad (4.1)$$

As the stability depends on the nature of the arm, three architectures A-B-C are proposed for consideration and presented below.

Architecture A) 5 DoF classical arm

The first obvious consideration is a classical 5 *DoF* arm. The 2-joint wrist is made by revolute joints with axes perpendicular to the tool axis. For simplifying the analysis, the 2-joint wrist is transformed into a single revolute joint in the 2D model with an axis perpendicular to the model plane. The base revolute joint of the arm is also ignored, reducing the model to a **3R serial arm** in the 2D model.

The link lengths were determined by considering the requirements on reachability. The 2D Geogebra model of the robotic unit was constructed to obtain the required link lengths (Figure 4.63). In this case, links MN and NO have a length of 900 mm each.

The projection of the centre of mass of the overall system was also determined for the top to bottom cleaning trajectory. The robotic unit shows a stable behaviour since the projection of its centre of mass always lies well within the supporting polygon. For example in Figure 4.64(a), the length of the support polygon is 600 mm and the distance between the projection of the centre of mass and the centre of the robotic unit is 77 mm. Thus, for the configuration shown in Figure 4.64(a), P_{Stab} can be calculated by equation (2).

$$P_{Stab} = \left(1 - \frac{77}{350}\right) x100 = 78\% \qquad (4.2)$$

Similarly, for the configuration shown in Figure 4.64(b), P_{Stab} is found to be 71%.

Even if architecture A is capable to generate the cleaning motions along the walls with sufficient quasi-static stability, it cannot be used in our case because of serious collision issues. As can be seen in Figure 4.63, the mobile platform has to be put at a distance $x_A = 350$ mm from the wall to let the end effector pass between the mobile platform and the wall. This safe operational distance from the wall also

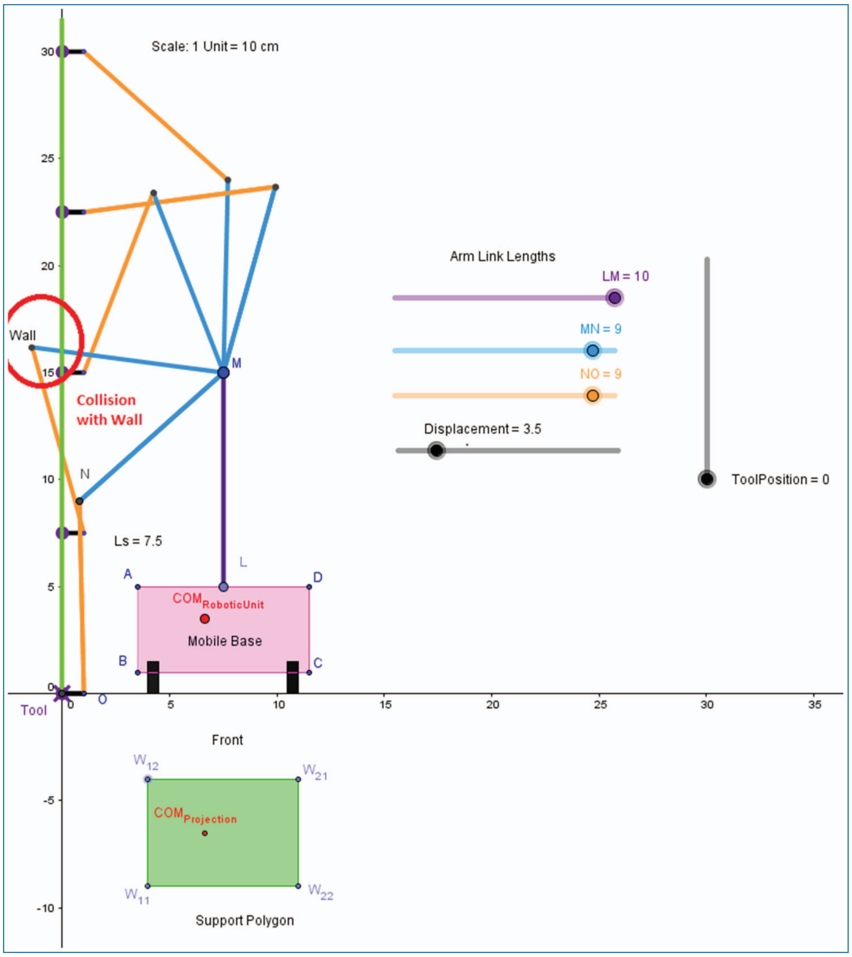


Figure 4.63. Collision with the wall while cleaning from a close distance (x_A= 350 mm).

serves to avoid collision with the wall. Figure 4.63 shows the elbow of the robotic arm colliding with the wall. This means either the safety distance x_A is not long enough or the link lengths MN = NO = 900 mm are too long, even if they must be long enough so the ceiling at 3 m high can be cleaned.

Architecture A has even more collision issues in the case of a narrow corridor, a common feature in the flats to be cleaned. Figure 4.65 shows a 100 cm wide corridor, which is a large value compared to the narrowest corridors of 70 cm. Even in this optimistic case, the model shows not only the elbow collision with the left wall when the tool altitude is $y_O = 120$ cm but also a second elbow-collision with the right wall when $y_O = 240$ cm. This second case shows clearly that architecture A cannot be used for our project.

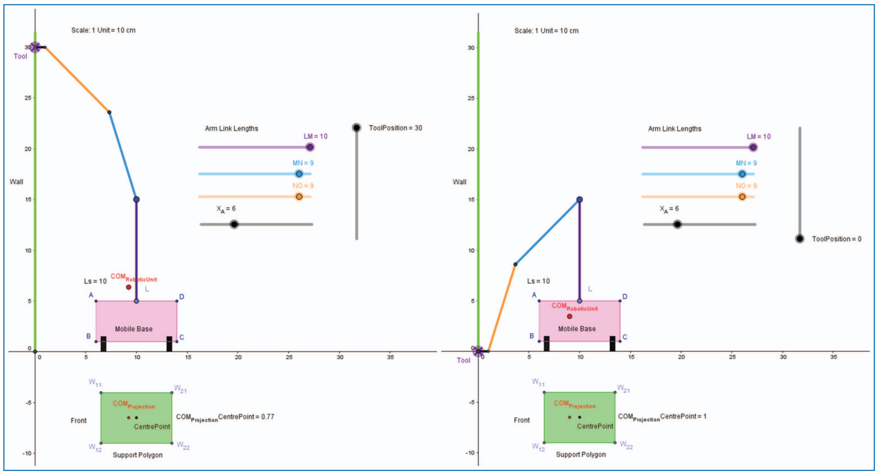


Figure 4.64. 2-R architecture: Wall cleaning scenario. (a) Topmost point. (b) Ground point.

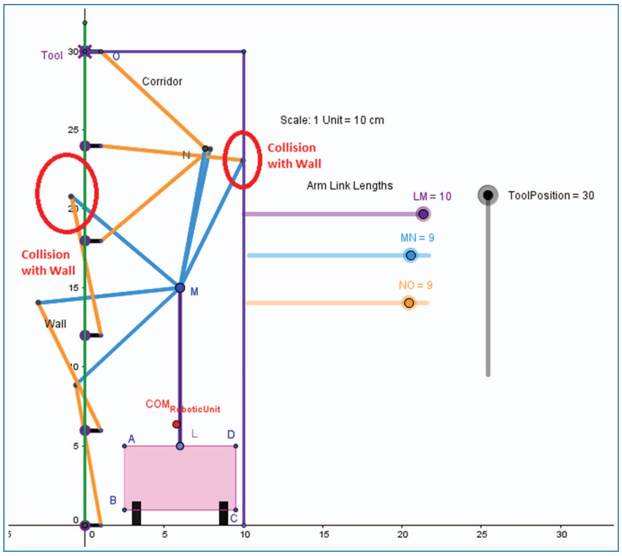


Figure 4.65. Cleaning in a narrow corridor with architecture A leads to arm collision with the wall.

Architecture B) 6 DoF redundant solution with a 5 DoF arm mounted on a vertical slider

This redundant architecture is considered in order to avoid the use of two long link lengths required in case of a classical anthropomorphic arm, as discussed for architecture A. To study the planar case, the architecture B is reduced to a 3R serial arm with a vertical slider at the base. A Geogebra model of this architecture was built to determine link lengths and study static stability. The link lengths obtained in this

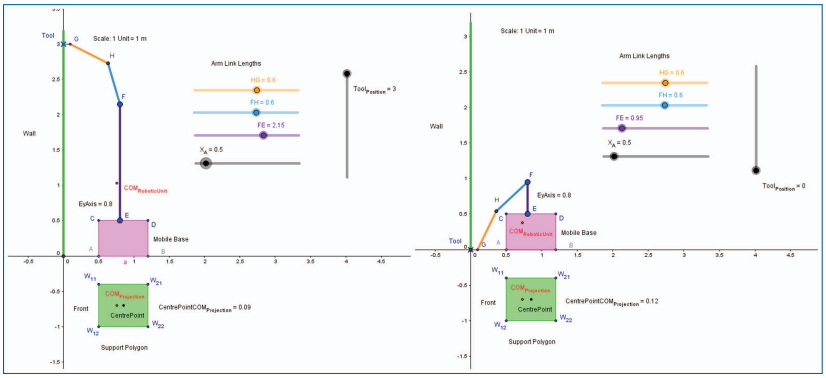


Figure 4.66. 2-R architecture with slider: Wall cleaning scenario. (a) Topmost point. (b) Ground point.

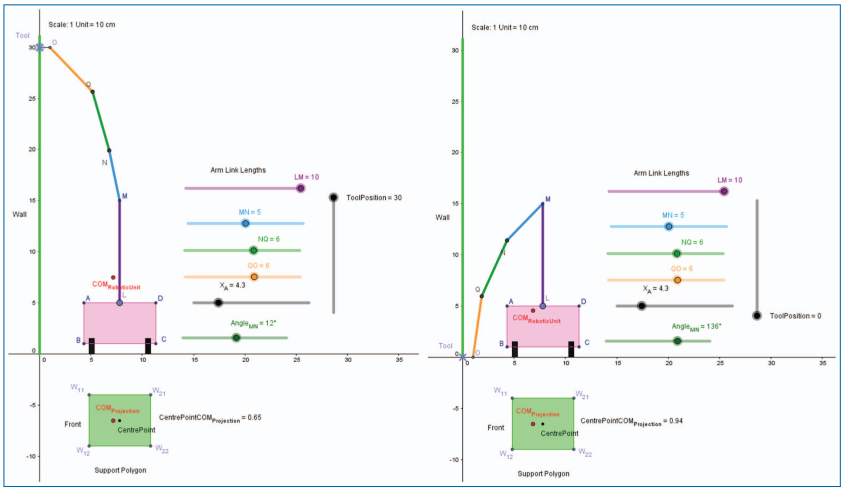


Figure 4.67. Wall cleaning scenario with architecture C. (a) Topmost point. (b) Ground point.

case are around 60 cm each. Also, the structure shows a stable behaviour during cleaning trajectory and the longitudinal stability margin P_{Stab} for the configurations shown in Figure 4.66(a) and 4.66(b) are 74% and 66% respectively.

Architecture C) 6 DoF redundant solution with a 6 DoF arm having three shorter moving segments

Ignoring the spherical wrist and the base revolute joint, this architecture can be reduced to a 4R plane serial arm. A Geogebra model of architecture C was built to determine link lengths and study its static stability. One important advantage of architecture C is that it can clean in narrow corridors, whereas architecture A fails because its arm collides with walls of the corridor (Figure 4.65).

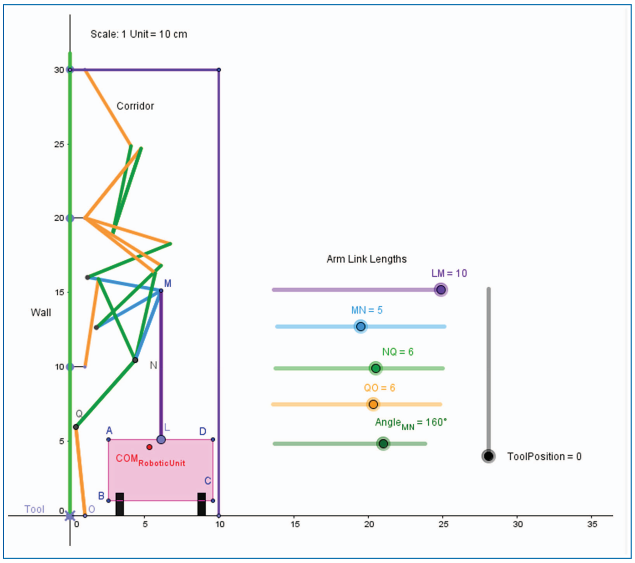


Figure 4.68. Cleaning in a narrow corridor with architecture C.

The static stability of this architecture was studied (Figure 4.67) and the values of P_{Stab} in (a) and (b) configurations are 81% and 73% respectively. This means **architecture C seems to be more stable than the two others** in these two reference positions.

Figure 4.67 also illustrates the notion of redundancy. The idea is to adjust the angle of the revolute joint M so that the NQO double-segment arm can move vertically without crossing a singularity. The optimal angle of joint M could be chosen by maximizing the distance to singularity during the sweeping motion. Three positions of joint M can be seen in Figure 4.106 during the path planning of the arm for a vertical cleaning motion in a narrow corridor. It can also be seen that no collision occurs with the left or right walls.

In order to demonstrate the potential interest of architecture C, a CAD model was also built, showing a possible implementation (Figure 4.107).

An interesting advantage of architecture C over architecture B can be seen in Figure 4.107(a) and Figure 4.108: the first segment can be made horizontal so that the tool can reach all the periphery of the mobile base, even in tight corridors.

Two simulations of path planning for a straight line can be seen in Figure 4.109. A vertical motion on a wall can be seen in Figure 4.109(a). When the end-effector is ascending, the first segment is progressively lifted. Then, a singularity is avoided by deliberately bringing the first segment downwards so that the two remaining segments can find their natural position. In a second time, a horizontal motion on a ceiling at 3 m can be seen in Figure 4.109(b). Thanks to the lateral shifting of the arm, no collision can occur between the arm and the central stabilizer.

Figure 4.109(b) also shows a possible issue during the arm control: two possible configurations of the wrist joints J5 and J6 allow pointing the tool axis towards the ceiling. The first one, represented in Figure 4.107(b), brings the tool actuator towards the right side of the picture. This wrist configuration was not possible in Figure 4.109(b) because of a collision risk between the tool actuator and the terminal wall of the corridor for the right part of the trajectory. Collision was avoided by wrist reversal to the second possible configuration, which points the tool actuator towards the left of the figure. Passing from the first to the second configuration is achieved by adding a rotation angle of 180° to joints J5 and J6. The robot should be always capable to switch from one configuration to the other, according to collision risks of the tool actuator with the environment. This collision avoidance should be considered for control.

As a conclusion on architecture C, it has four advantages over the two others:

- Three shorter segments instead of two long segments avoid any collision with the walls in narrow corridors
- Its first segment can be oriented horizontally and allows the tool to reach the complete periphery of the mobile base
- Its stability at reference poses seems to be the best of the three architectures
- Its redundant additional mobility can be used to keep the arm far from singularities

4.6.2.6 Force analysis of the actuators

Model A1

The first arm model is shown in Figure 4.110, where **J1** and **J2** are two passive rotational joints. The arm is similar to the architecture A presented in section 3.3. The joints could be actuated either by a rotational or translational actuator. In this model A1 was considered the second case, using for instance two screw jacks **P1** and **P2**. This architecture is inspired from the digging machines or earth movers which provide high stiffness and big load. The mass of the arm is 80 kg, based on the project specifications divided on two links of 0.95m each. The arm is simulated in the planar case while cleaning the wall and then cleaning the ceiling. The interaction forces with the wall along x and y directions are defined as **Rx** = -100 N and **Ry**= -100 N respectively while in the ceiling case only **Ry** = -200 N is considered (100 N of normal force + 100 N of tool mass).

Case 1: Wall cleaning

The arm is simulated during tangential wall cleaning from top to bottom with a speed of 3 m/min which is realistic for such kind of work and enough to keep the

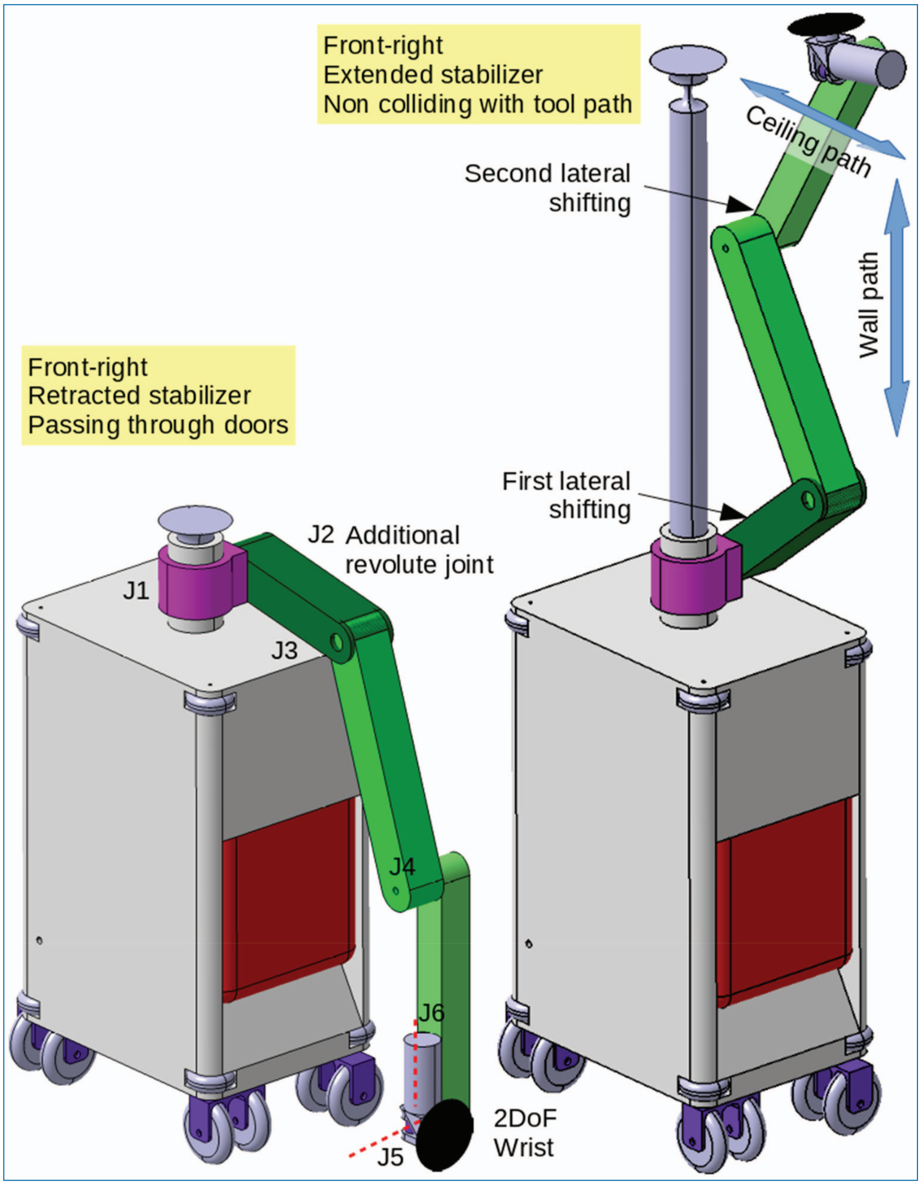


Figure 4.69. CAD model and possible implementation for an arm with architecture C. (a) Tool cleaning the lower part of a wall. (b) Tool cleaning the ceiling above the mobile platform.

stability of the platform to avoid falling down because of dynamics (quasi-static model). The results of the simulation are analysed and shown in the figures below. Figure 4.111 shows that the maximal lengths needed for **P1** and **P2** prismatic joints are around 0.9m. Figures 4.112 and 4.113 show that the maximal torques needed on the rotational joints **J1** and **J2** are 543 and 259 Nm respectively.

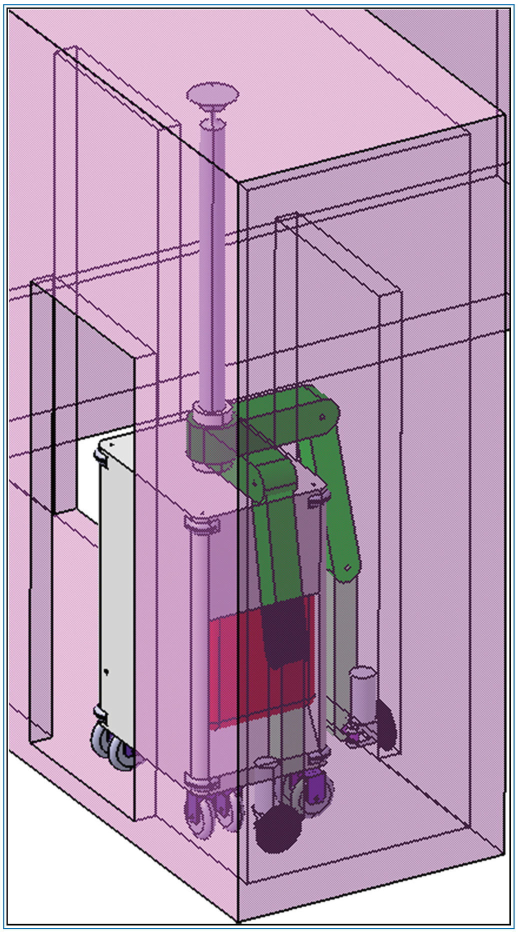


Figure 4.70. The first segment can be made horizontal so that the tool can reach all the periphery of the mobile base, even in tight corridors.

The maximal forces needed on the translational actuators **P1** and **P2** are 11606 N and 819 N respectively as shown in Figures 4.76 and 4.77. Hence, the smaller the angle b between the force direction and the radial direction from the joint, the maximum the torque, based on what Figure 4.78 shows. Thus, working near a fully extended arm leads to the high actuator forces shown in the graphs. These singular positions should be absolutely avoided as the actuation forces tend towards infinite in this case.

Case 2: Ceiling cleaning

The robot arm moves tangentially to the ceiling with a speed of 3m/min. The results of the simulation are analysed and shown in the figures below. Figures 4.79 and 4.80 show that the maximal torques needed on the rotational joints **J1** and **J2**

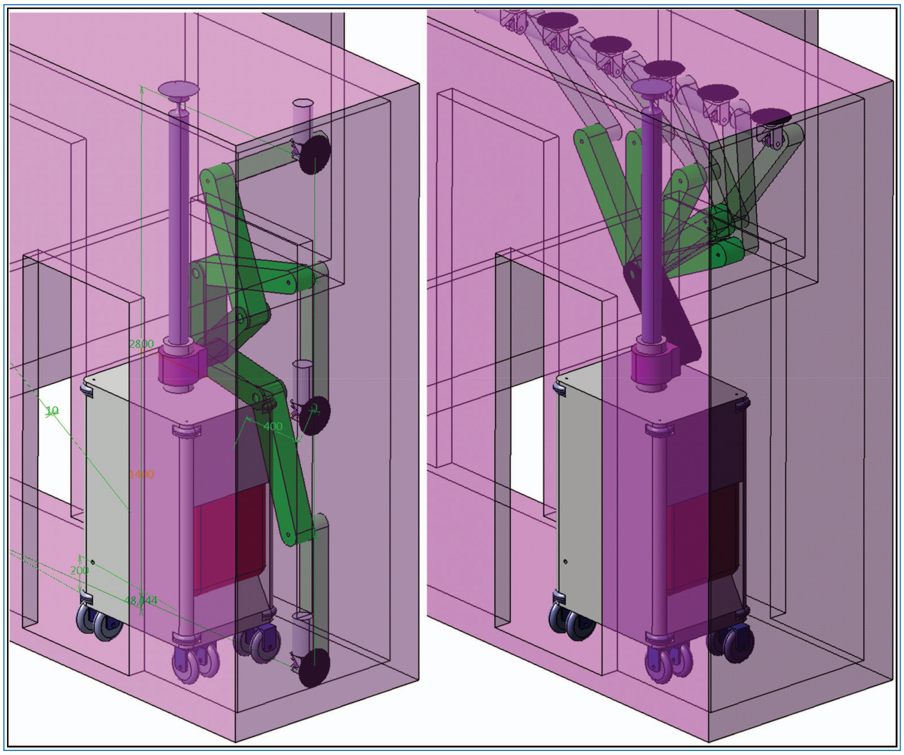


Figure 4.71. Path planning of a straight line. (a) Vertical line on a wall with singularity avoidance thanks to J3 reversal. (b) Horizontal line on the ceiling with no collision arm-stabilizer thanks to the lateral shifting of the arm. Collision avoidance between the tool actuator and the terminal wall required switching to a second wrist configuration with respect to Figure 4.107(b).

are 550 and 380 Nm respectively. The maximal forces needed on the translational actuators **P1** and **P2** are 1980 N and 1545 N respectively as shown in Figure 4.81 and 4.82.

Summary about model A1:

The model A1 is a special implementation of architecture A with translational actuators. The interest of using, for instance, electric screw jacks as actuators would be their non-reversible behaviour and compatibility with the electric power supply chosen for the robotic unit. The drawback of this implementation is that it has a limited workspace and that the actuation forces tend to become very high at full arm extension.

Model A2

A second actuation model as shown in Figure 4.83 is studied to reduce the limitations of the first model. It is still based on architecture A with two revolute joints

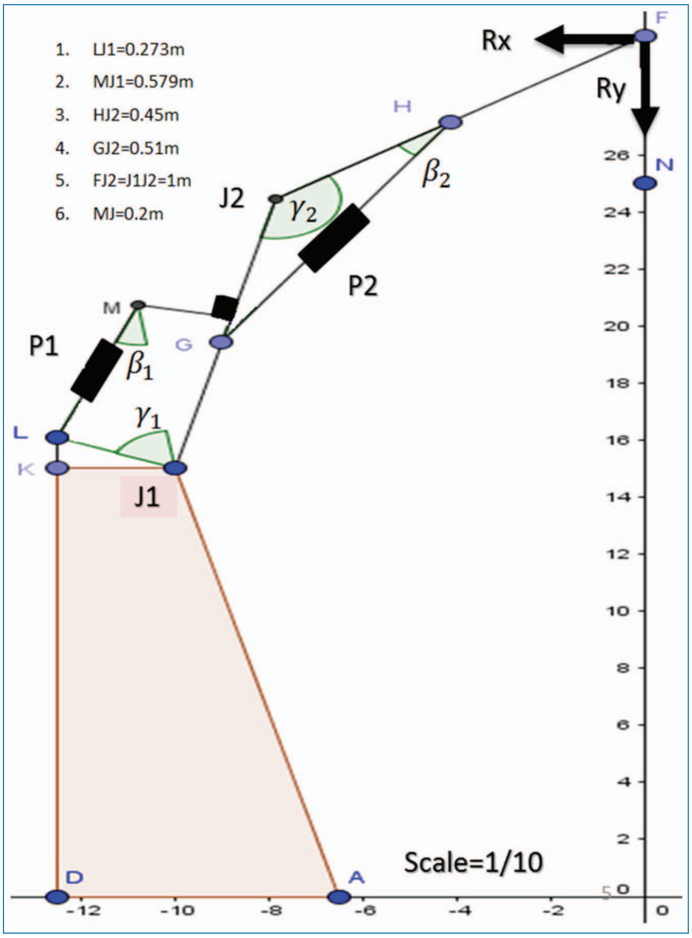


Figure 4.72. model A1 for force analysis, based on architecture A motorized by two translational actuators P1 and P2.

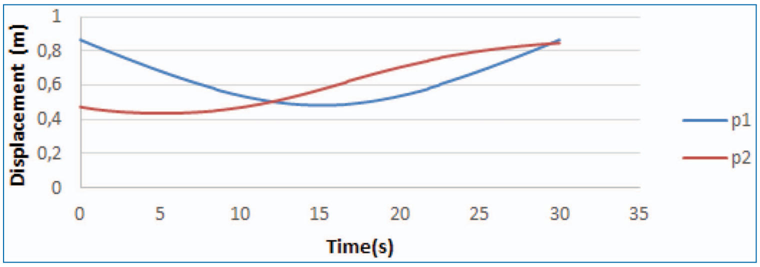


Figure 4.73. Evolution of translational actuators P1 & P2 positions for model A1.

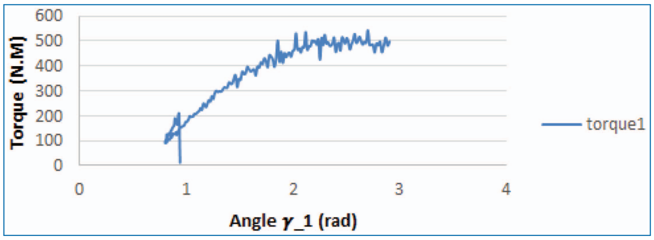


Figure 4.74. Torque on joint J1 vs. angle g_1 for model A1.

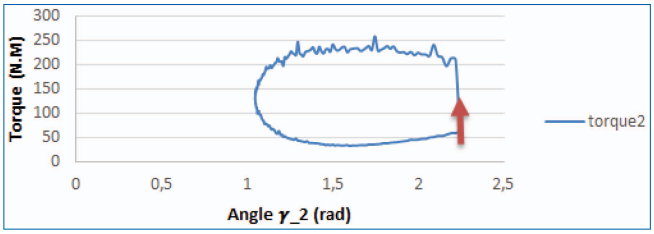


Figure 4.75. Torque on joint J2 vs. angle g_2 for model A1.

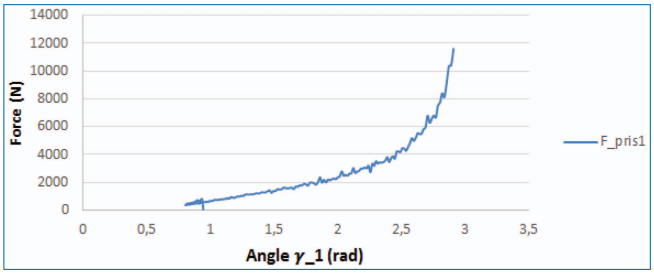


Figure 4.76. Force applied by translational actuator P1 vs. angle g_1 for model A1.

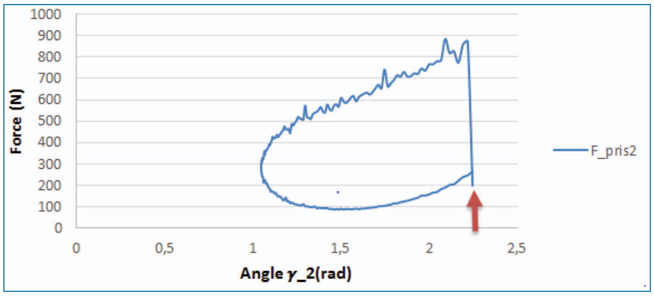


Figure 4.77. Force applied by translational actuator P2 vs. angle g_2 for model A1.

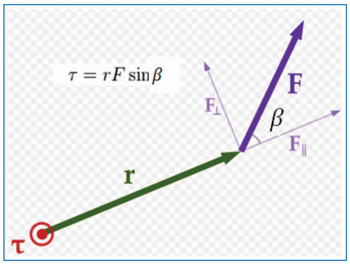


Figure 4.78. Relation between torque t and force F.

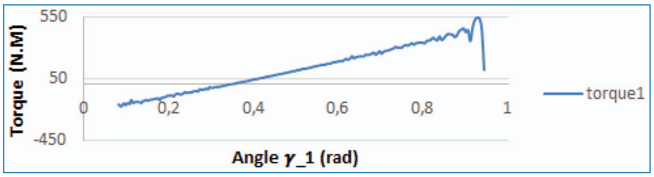


Figure 4.79. Torque on J1 vs. g₁for model A1.

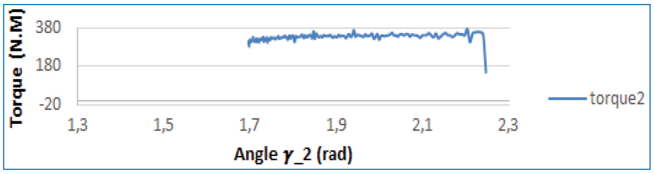


Figure 4.80. Torque on J2 vs. g₂ for model A1.

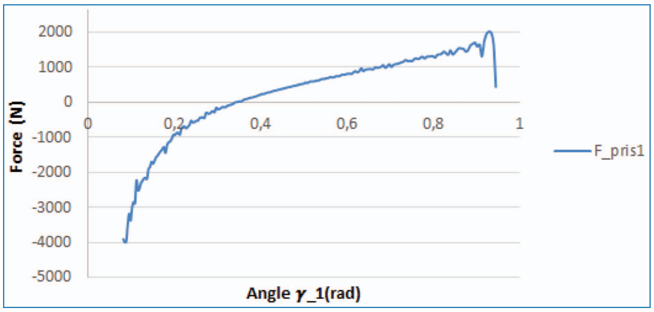


Figure 4.81. Actuator force of P1 vs. angle g₁for model A1.

J1 and J2. This time, J1 is actuated by a rotational actuator whereas J2 is controlled by an indirect linkage connecting the fore-arm to the base. Joint **J2**, **J3** and **J4** are passive rotational joints, whereas **P1** is a translational actuator. The interest of this structure is that all the actuators are supported by the base, which could bring to lighter links and a higher dynamics. This model is similar to the well-known 5-bar mechanism which is famous for its stiffness and high acceleration performance,

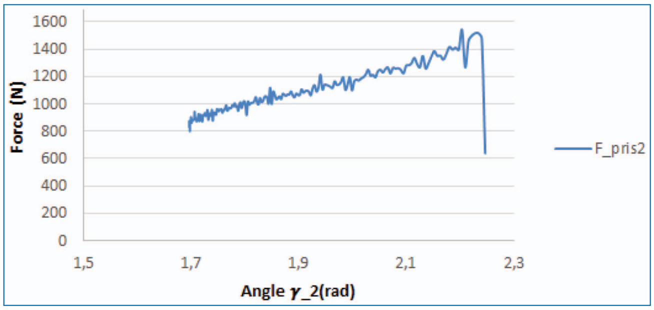


Figure 4.82. Actuator force of P2 vs. angle g_1 for model A1.

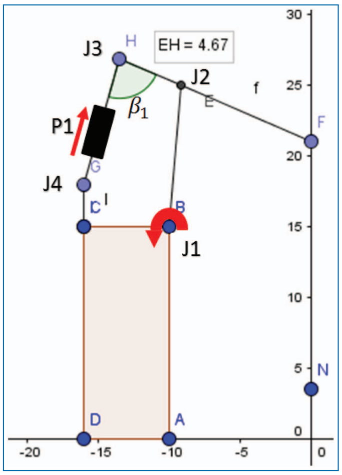


Figure 4.83. Model A2 for force analysis, based on architecture A motorized by a rotational actuator J1 and a translational actuator P1. The overall setting has a parallel architecture, suitable for high dynamics.

typical of parallel architectures. The arm weighs 80 kg and is simulated for wall and ceiling cleaning as previously for model A1, with the same interaction forces.

The conclusion about the simulation results showed that, in order to reach point **A** (bottom right of the robot in Figure 4.83) the required length of the prismatic joint **P1** is 1.7 m. Taking into account the torques estimated before on **J1** and **J2** (Figures 4.76 and 4.77), if **J1** and **P1** are actuated only, then the maximal force to be applied by **P1** will be roughly 2858 N.

Summary about model A2:

The forces and torques needed to actuate the arm for the asbestos removal task are feasible although they are high. But the main drawback of this arm is its big size for our specifications and the singularity when J4, J3 and J2 are collinear which limits operation on the ceiling.

Summary about force analysis of actuators:

Overall, the recommended solution for actuation is to keep rotational actuators on all the joints and not use translational actuators, as it is less limiting in terms of workspace and as forces in the jacks may become very high at full arm extension. Electric screw jacks could have the interest of non-reversibility and high stiffness, but cannot be used because of the limitations shown above. The designers should be careful to find a rotational actuator that can be locked or braked for better stiffness.

4.6.2.7 Workspace of the manipulator

This section investigates the workspace for architecture A. The intersection between the arm workspace and the wall or ceiling is shown in Figure 4.84 and represents the attainable area in the wall/ceiling for a fixed position of the arm base. The workspace was constructed by discretizing the joint space and using the direct geometric model (DGM) of a 3R arm with architecture A and two links of 0.95 m as discussed before. The inputs of the DGM are the joint interval values. Iterating the joint value between the maximum and minimum joint limits gives a 3D mesh, that can be intersected with the wall and ceiling to obtain two 2D meshes shown in Figure 4.84.

The workspace, and consequently the arm, must be dimensioned to reach the top and bottom of the wall, the ceiling and the furthest points discussed before. For the walls, the usable workspace is the biggest rectangle included into the 2D projected workspace on the wall (Figure 4.84(a)) with a Z height covering the complete wall height (3 m). Its width LY is the maximal length that can fit in the 2D projected workspace. This allows minimizing the displacements of the mobile platform in Y, as the idea is to cover the wall surface by jointly tiling several rectangles separated by mobile platform displacements of length LY.

The approach is a bit different for the ceiling, where there is no limitation to 3m for one side of the rectangle. The two dimensions LX and LY of the rectangle included into the 2D projected workspace on the ceiling (Figure 4.84(b)) result from a compromise and should be optimized, depending on the motion capacities of the mobile platform.

4.6.2.8 Preliminary dynamic model of the robotic unit

A parametrized 3-D model of the robotic unit based on arm with architecture A was built with ADAMS to study the effect of dynamics during a vertical trajectory in the plane of wall. This model will be adjusted to the chosen arm architecture in the future. Thus, it should be considered as preliminary and useful for demonstrating the model possibilities. The masses of the components of the robotic unit were adjusted according to the latest assumptions.

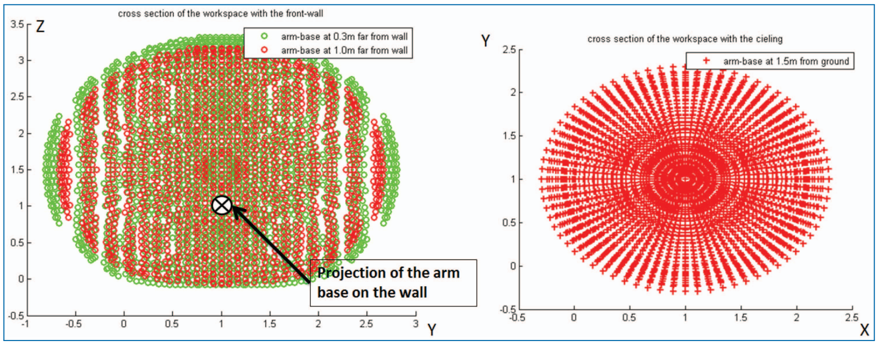


Figure 4.84. Arm workspace of architecture A.

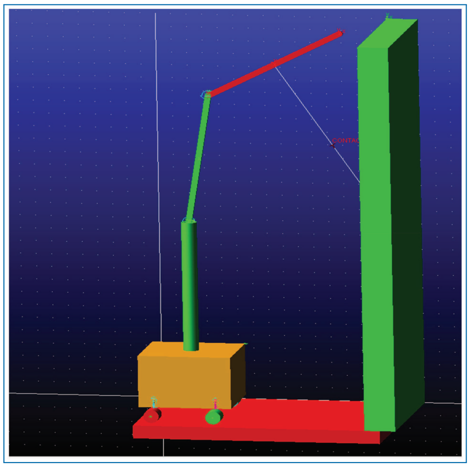


Figure 4.85. ADAMS model of the robotic unit with arm architecture A.

The inverse geometric model of the 2R robot was programmed in Matlab to get the joint angles corresponding to a vertical trajectory. These values were then imported to the ADAMS model as a spline to generate the trajectory.

A contact between link 2 (red) and the wall was created. Each of the four wheels was also constrained to move in the plane of the floor by creating contacts. This model showed a stable behaviour throughout a vertical trajectory.

This preliminary dynamic analysis shows a stable behaviour of the robotic unit for the given speed of the end-effector. However, this model must be further developed by using the chosen arm architecture and introducing realistic external tool forces in order to have a better understanding of the dynamics.

4.6.2.9 Productivity of the robotic unit

It is important to evaluate the productivity of the robotic unit and to bring it as high as possible with respect to the productivity of a human operator.

The current productivity of a human worker has been measured around 6.4 square meters/hour (source: Bouygues). The approximate productivity of a robot with a tool of 150 mm diameter moving at 3 m/min is $0.15mx3m/minx60min/h = 27m^2/h$. However, the productivity is related to the tool size and its translational speed. Even at a speed of 3 m/min, which is relatively low for machining, the expected productivity of a single robot could be four times the productivity of a human operator, provided the cleaning can be made in a single pass. But this has to be confirmed by removal tests. The path planning strategy has also to be defined. This important figure of productivity should be assessed as precisely as possible all along the project.

References

[Geogebra, 2016] Geogebra, the graphing calculator for functions, geometry, algebra, calculus, statistics and 3D maths. https://www.geogebra.org. Retrieved online in 2016.

[McGhee and Frank, 1968] R. B. McGhee and A. Frank, "On the stability properties of quadruped creeping gaits," Mathematical Biosciences, vol. 3, pp. 331–351, 1968.

[Prause *et al.*, 2015] Isabel Prause, Jean-Christophe Fauroux, Mathias Hüsing and Burkhard Corves. "Using Geometry Sketchers and CAD Tools for Mechanism Synthesis", paper OS3-032, in Proc. of IFToMM 2015, The 14th World Congress in Mechanism and Machine Science, October 25–30, 2015, Taipei International Convention Center, Taiwan, 11p.

Dynamic Modelling of the Robotic Unit for Stability Evaluation

Process of asbestos removal involves dynamic interaction between cleaning environment and robotic unit. This interaction occurs at two levels. Firstly, grinding tool mounted on the robotic arm interacts with the cleaning surface to generate reaction forces. Secondly, wheels of the mobile base are subjected to ground reaction forces due to:

- weight of the robotic unit
- inertia forces generated due to the motion of robotic arm.

In Figure 1 a typical interaction model existing within the process of asbestos removal is detailed. Reaction forces generated by tool-surface interaction (ground,

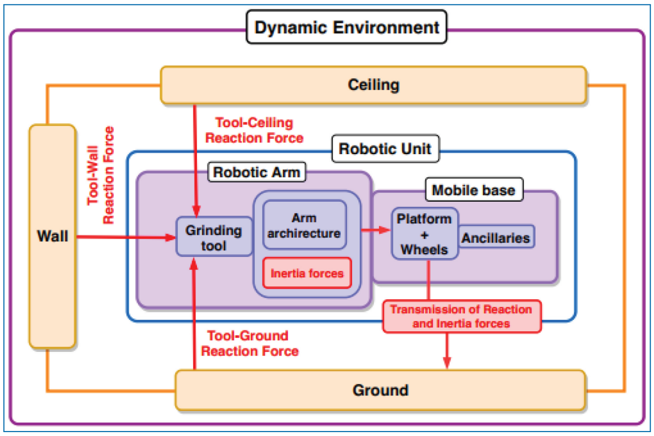


Figure 4.86. Interaction model of asbestos removal use case.

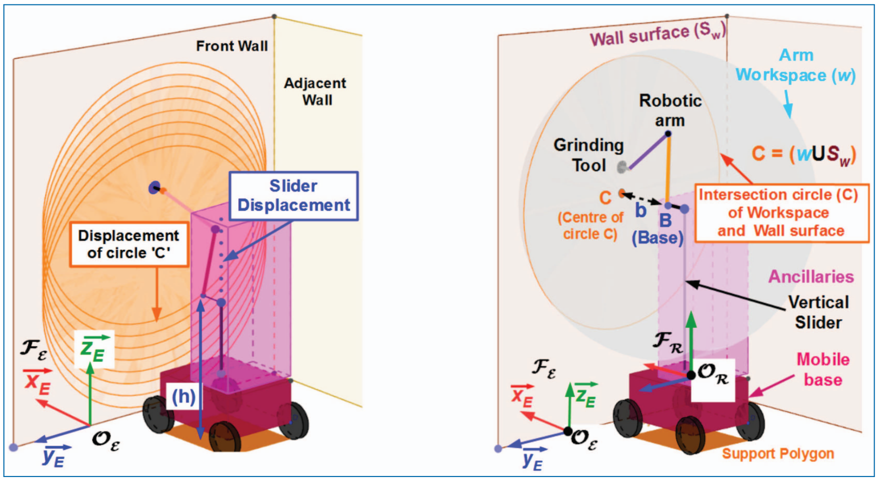


Figure 4.87. Workspace (left) and GeoGebra representation (right).

wall and ceiling) are transmitted to the arm architecture through tool-arm connection. Since, arm performs accelerated motions, varying inertial forces are generated in addition to grinding reaction forces. Connection of arm to the mobile base transmits these forces to the mobile base which are further passed on to the ground surface through wheels. Due to these transmissions, stability of the robotic unit is significantly affected. Therefore, in order to study effects of arm motion and grinding reaction forces on the stability of robotic unit, a dynamic simulation model capable of simulating realistic asbestos removal scenario should be constructed.

Representation of asbestos removal use case

Figure 2 shows robotic unit placed at a distance 'b' from wall surface (Sw). At this distance, the 3D workspace (w) of the robotic arm intersects with surface (Sw) in circle C. The area of C is the area available for cleaning at a given base placement such that collision free (arm-cleaning environment) continuous trajectories are feasible. However, since the robotic arm is mounted on a vertical slider (figure 3), circle C can be translated along a vertical axis to a resulting workspace with a geometric shape called 'stadium' (rectangle with semicircles on either of the opposite sides). Thus, even at fixed base position, workspace of the robotic arm is extended due to presence of architectural redundancy (P-joint).

Stability Evaluation

Two methods of stability evaluation are adapted during this study.

a) Numerical Method

Cartesian trajectory input: Targeted operational surfaces in construction applications can be 3-dimensional in nature (curved walls, ceiling or ground). However, in this work, focus is limited to non-curved surfaces. Thus, 2-D Cartesian trajectories are defined while performing simulations. Cartesian pose of the end-effector frame corresponding to trajectory coordinates serves as an input to the algorithm.

IKS: Inverse kinematics solution is needed to generate arm motion. Since arm is kinematically redundant, IKS is determined using redundancy resolution method that uses pseudo-inverse of the kinematic Jacobian matrix.

Recursive computation of joint velocities: Angular velocities and accelerations of moving links contribute in generating dynamic effects on the robotic unit. Newton-Euler recursive algorithm is used to compute these entities.

b) Stability Evaluation: Zero moment point for the instantaneous dynamic state of the robotic unit is computed using equations 3.2a and 3.2b. Stability is then evaluated as per equation 3.3

Method of co-simulation

In this section, a procedure for constructing ADAMS-MATLAB co-simulation is described in detail (Fig. above).

1. Parameterized multibody ADAMS model of the robotic unit
   (a) Direct geometric model (DGM) of the robotic arm is found using SYMORO. DGM provides pose of the each link frame in Cartesian space coordinates

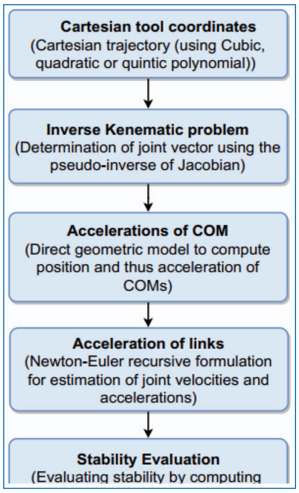

Cartesian tool coordinates
(Cartesian trajectory (using Cubic, quadratic or quintic polynomial))
Inverse Kenematic problem
(Determination of joint vector using the pseudo-inverse of Jacobian)
Accelerations of COM
(Direct geometric model to compute position and thus acceleration of COMs)
Acceleration of links
(Newton-Euler recursive formulation for estimation of joint velocities and accelerations)
Stability Evaluation
(Evaluating stability by computing

Figure 4.88. Numerical evaluation of stability.

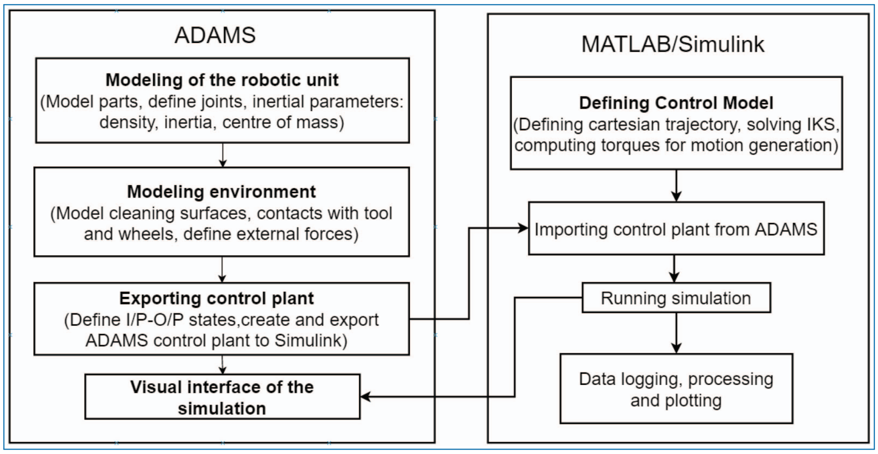

ADAMS
Modeling of the robotic unit
(Model parts, define joints, inertial parameters: density, inertia, centre of mass)
Modeling environment
(Model cleaning surfaces, contacts with tool and wheels, define external forces)
Exporting control plant
(Define I/P-O/P states,create and export ADAMS control plant to Simulink)
Visual interface of the simulation
MATLAB/Simulink
Defining Control Model
(Defining cartesian trajectory, solving IKS, computing torques for motion generation)
Importing control plant from ADAMS
Running simulation
Data logging, processing and plotting

Figure 4.89. Flow-chart of ADAMS-Matlab co-simulation.

(b) Using these co-ordinates, construction points are created in ADAMS environment

(c) Individual links are then built with the help of these construction points

2. Input-Output control plant

   (a) State variables are assigned to each joint angle as well as C.o.M of each link to track position, velocity and acceleration. These are output variables of the control scheme.

   (b) Input variables consist of torques given to the joints of the robotic arm

   (c) ADAMS control plant is created, then exported to Matlab-Simulink environment co-simulation purpose.

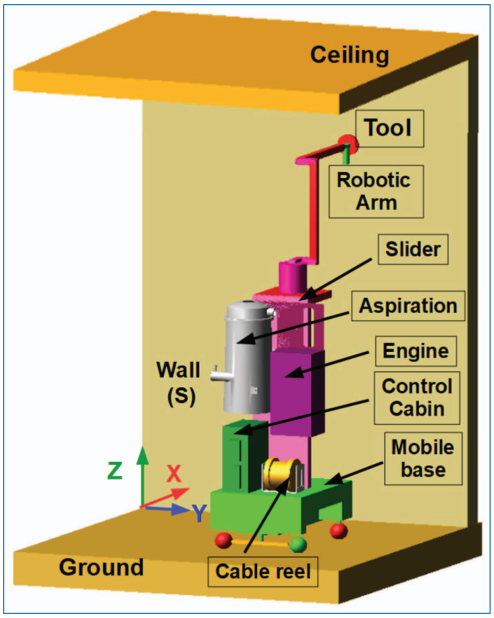


Figure 4.90. Multibody dynamic model of the Robotic unit – V1.

3. Control Scheme in Simulink environment
   (a) PID controllers are implemented at each joint angle to compute required input torques for desired motion
   (b) Gravity compensation torques are added to PID generated torques to account for the effect of gravity
   (c) Output coming from ADAMS i.e. joint velocities and accelerations are used for PID feedback
4. Estimation of ZMP
   (a) All the necessary components to calculate ZMP are obtained in run-time through ADAMS output whereas angular moments can be calculated from available information
   (b) ZMP criteria is then implemented to compute stability at each pose while performing desired motion

Results of stability evaluation

The results of the stability analysis by this approach is shown in the following section.

Acceleration in +ZE, h = 1.675 m: Here, tangential reaction forces act along -ZE. Moment responsible for longitudinal stability decreases from top to bottom

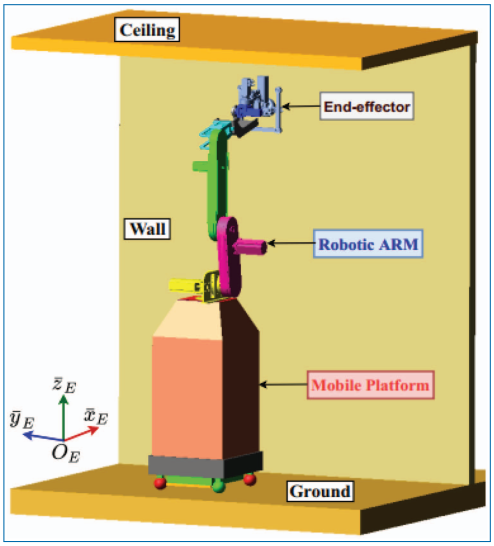


Figure 4.91. Multibody dynamic model of the Robotic unit – V2.

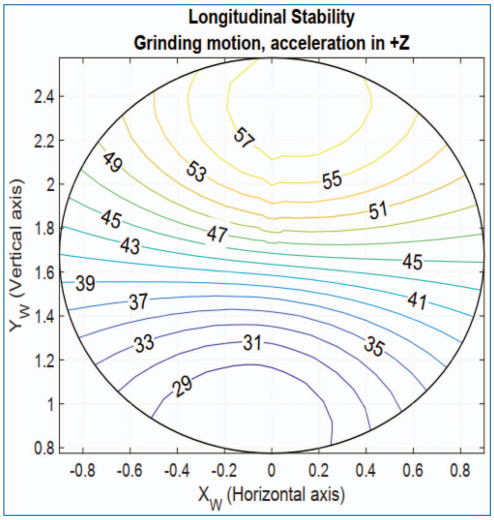


Figure 4.92. V1-Longitudinal stability contour for acceleration in +ZE.

due to decreasing distance along ZE. This is reflected in the longitudinal stability maps shown in figure 9.

Acceleration in −ZE, h = 1.675m: Here, tangential reaction forces act along −ZE and moment now pushes ZMP along −XE. Thus, longitudinal stability at respective points for −ZE motion is greater than that for +ZE motion (Fig. 10).

Acceleration in +YE, h = 1.675m: Tangential forces act along -YE pushing ZMP towards lateral edge of the polygon. Area on the right of red line shown in figure

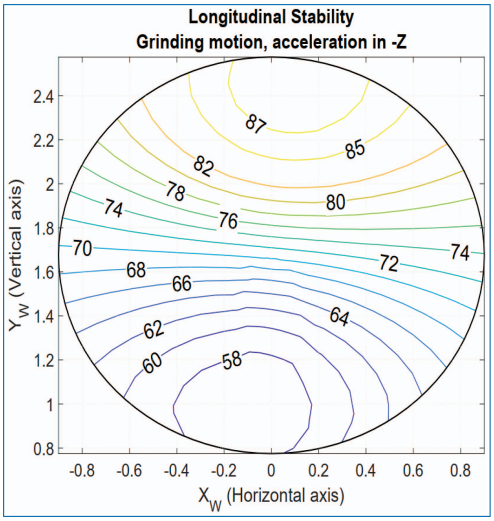


Figure 4.93. V1-Longitudinal stability contour or acceleration in −ZE.

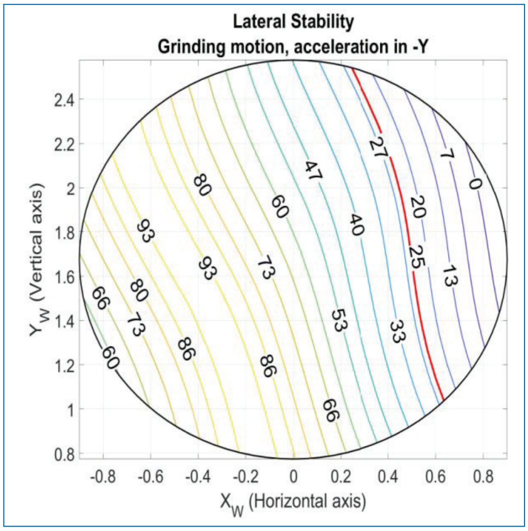


Figure 4.94. V1-Lateral stability contour for acceleration in −YE.

10 is stable. However, area on the left side of the line has stability is below 25% as shown in figure 11.

Acceleration in -YE, h = 1.675m: Tangential forces act along +YE direction creating moment that tends to move ZMP towards other lateral edge of the support polygon. In general, the following important conclusions can be made from stability maps:

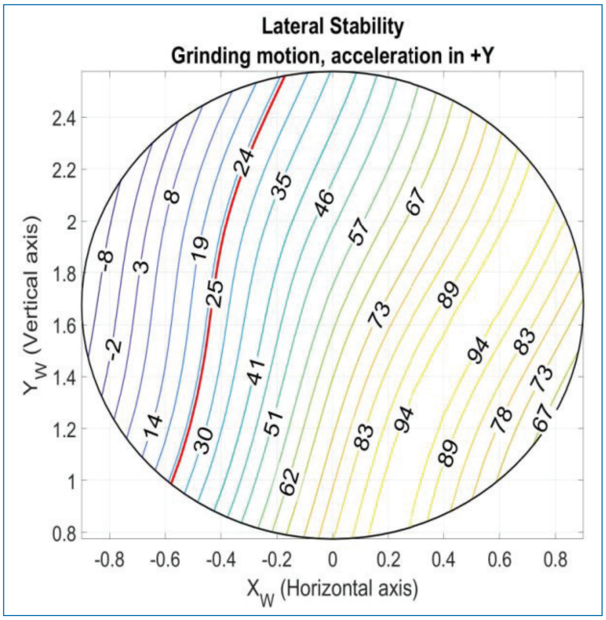


Figure 4.95. V1-Lateral stability contour contour for acceleration in +YE.

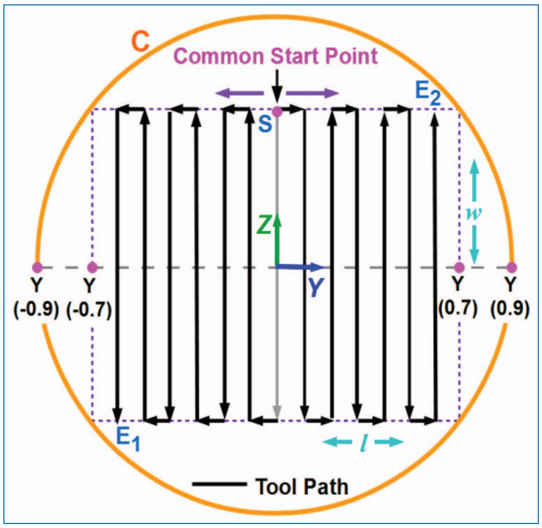


Figure 4.96. Path traced by tool on the wall.

- Longitudinal stability is always above critical limit of 25% for motion along Z direction.
- Lateral stability is affected based on the positive or negative direction of the cleaning along Y.

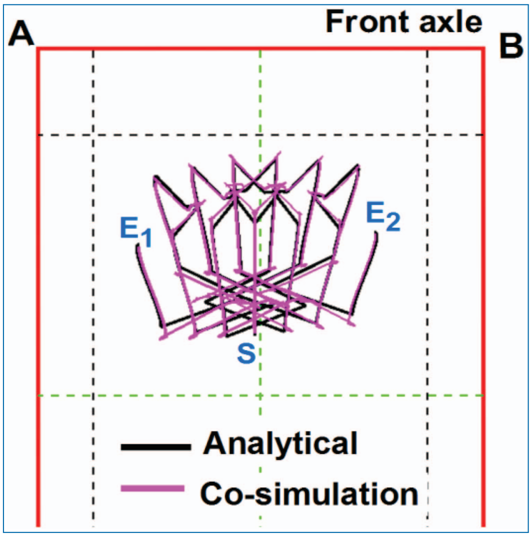


Figure 4.97. ZMP estimation through numerical and co-simulation.

Stability based trajectories

Asbestos removal grinding trajectories are illustrated based on the conclusions of stability analysis (Fig. 13). Since, most cleaning walls are rectangular, we assume a rectangle of dimensions (1.38 × 1.13) m inside C. A boustrophedon path is tracked separately from point S to E1 and S to E2 instead of one single path from E1 to E2. The tool motion along -Y while cleaning inside $Y< 0$ zone and along +Y while cleaning inside $Y> 0$ zone results in a stable manipulator operation. Fig. 14 shows variation of ZMP obtained by MATLAB simulation and co-simulation during the two grinding trajectories. It can be safely concluded that analytical model can be trusted for simulating grinding trajectories in future.

Conclusions of stability-based computations

Important conclusions of the work are listed below:

- Dynamic interaction model of the asbestos removal scenario was studied
- Dynamic simulation of the use case was created using two methods namely-numerical and analytical.
- Dynamic stability of the robotic unit was studied for identifying reaction wrench during cleaning scenario
- Stable directions for cleaning, starting from the centre of the workspace and moving laterally towards the outside were identified, and a global boustrophedon cleaning trajectory in two parts was proposed to maximize stability.

References

Cooperation

[1] Zine Elabidine CHEBAB, “Conception et Commande Collaborative de Manipulateurs Mobiles Modulaires (C3M3)”, PhD dissertation. December 08th, 2019

[2] Zine Elabidine CHEBAB, Jean-Christophe FAUROUX, Grigore GOGU, Laurent SABOURIN, Nicolas BOUTON and Youcef MEZOUAR. “Structural Analysis of Mobile Manipulators”, Session 3-Robotics, in Proc of 15th IFToMM World Congress, June 30-July 4, 2019. Krakow, Poland. 10 p.

[3] Siddharth MARAJE, Jean-Christophe Fauroux FAUROUX, Chedli-Belhassen BOUZGARROU and Lounis ADOUANE. “Stability analysis of an asbestos removal mobile manipulator for safe grinding trajectories”, Session 3-Robotics, in Proc of 15th IFToMM World Congress, June 30-July 4, 2019. Krakow, Poland. 10 p.

Grinding

[1] QUENTIN, Trebot, PHILIPPE, Vaslin, JEAN-CHRISTOPHE, Fauroux, et al. Robotized grinding experiments of construction materials for asbestos removal operation. In : IFToMM World Congress on Mechanism and Machine Science. Springer, Cham, 2019. p. 2621-2630.

[2] https://www.numatic.fr/project/hzd900/

References

[Geogebra, 2016] Geogebra, the graphing calculator for functions, geometry, algebra, calculus, statistics and 3D maths. https://www.geogebra.org. Retrieved online in 2016.

[McGhee and Frank, 1968] R. B. McGhee and A. Frank, “On the stability properties of quadruped creeping gaits,” Mathematical Biosciences, vol. 3, pp. 331–351, 1968.

[Prause *et al.*, 2015] Isabel Prause, Jean-Christophe Fauroux, Mathias Hüsing and Burkhard Corves. “Using Geometry Sketchers and CAD Tools for Mechanism Synthesis”, paper OS3-032, in Proc. of IFToMM 2015, The 14th World Congress in Mechanism and Machine Science, October 25–30, 2015, Taipei International Convention Center, Taiwan, 11p.

Deliverable 2.2
D2.2 Simulation environment and model of the representative asbestos removal use-case with achievable precision

Authors and contributors:
MARAJE Siddharth, SIGMA Clermont
FAUROUX Jean-Christophe, SIGMA Clermont

4.6.3 Deliverable in the Context of the Work Program

Deliverable **D2.2**, simulation environment and model of the representative asbestos removal use-case with achievable precision is a part of task **T 2.3** – Virtual scenario of a representative asbestos removal use case. In deliverable **D2.1**, three arm architectures were analyzed for evaluating static stability. The second of the three solutions – 6 DoF redundant solution with a 5 DoF arm mounted on a vertical slider was selected for version 1 prototype while the third one – a 6 DoF redundant solution with a 6 DoF arm having three shorter moving segments was kept for consideration for version 2 prototype. After assuming a 3 DoF wrist for each of the architectures, the second one becomes a redundant 7 DoF (***P-6R***) architecture while the third one becomes redundant ***7-R*** architecture. In this deliverable a dynamic simulation environment was built in order to create a platform for overall system layout and analysis of the two robotic units resulting from the integration of the two redundant arm architectures, Version 1 – robotic unit integrated with P-6R arm and version 2 – robotic unit integrated with 7R arm. Preliminary investigation of the two robotic units is carried out to test the multibody dynamic models as well as preliminarily investigate static and dynamic stability. The objectives of this deliverable are:

[1] **Preliminary investigation of robotic units**: This investigation involves simulating simple cleaning trajectories e.g vertical or horizontal cleaning motions to test the models as well as to preliminarily investigate the dynamic stability by recording ground reaction forces.

[2] **Characterization of Workspace**: Evaluation of the intersection of the 3D workspace of the robotic arm on the cleaning surfaces is necessary to anticipate the movement of the mobile platform. Thus, the 3D workspaces are visualized to ensure the accessibility to the cleaning surfaces while performing simulation.

This task takes input from tasks 1.1 (use-case specifications), 3.2 (technical requirements), 1.4 (operation process parameters), 2.1 (results from preliminary simulation of the robotic unit) and 2.2 (preliminary proposition on mechanical stabilization).

4.6.4 Methodology

This section explains a methodology adapted for building and testing a Simulation environment consisting of realistic dynamic model of the robotic unit and the cleaning environment.

[1] **Multibody Dynamic Simulation (ADAMS Software)**:

- a. **Modelling of the Robotic unit:** A parameterized multibody dynamic model of the two robotic units will be built. The models will serve for simulation of trajectories and estimation of stability. The typical characteristics of this model are:

[2] Dimensions of the mobile manipulator:

- a. Realistic dimensions of the mobile platform and stabilizers
- b. Parameterization of the robotic arm based on the DH-model providing ease of accommodating changes in link lengths, joint offsets and wrist design.
- c. Pose of the arm base mounting on the mobile platform

[3] Pose of the mobile manipulator within environment:

- a. Distance of the mobile platform from the cleaning wall
- b. Different types of cleaning environments like large rooms, corridors, rooms with obstacles like bath tub or kitchen sink
- c. Different ceilings with variable heights and structural resistance

[4] Contacts of the mobile manipulator with the environment:

- a. Tool-surface
- b. Wheels-ground
- c. Stabilizer-environment(ground)

[5] Trajectory generation: Inverse geometric model of the robotic arm is solved in matlab to get the joint variables. These variables are then imported to ADAMS to feed to each of the joint.

- a. **Locomotion into environment** to cover all surfaces inside the flat. A retracted configuration of the arm has to be found for optimal stability during locomotion. The optimal trajectory in constrained environments such as narrow corridors has to be planned.

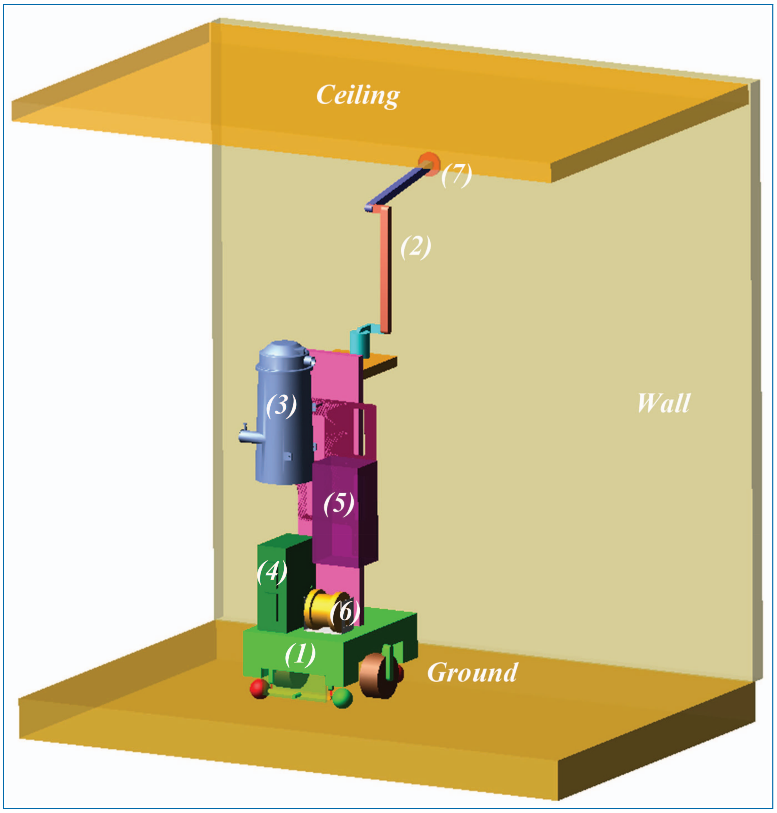


Figure 4.98. ADAMS model of the Robotic unit *Version 1*.

b. **Simulation of precise tool trajectories for fixed pose of the mobile base:** The attempt is to identify trajectories for cleaning wall, ground and ceiling. Cleaning productivity is directly related to joint accelerations; however, it is limited by process quality. Thus, the simulation environment will be used to identify maximum joint acceleration in future.

4.6.5 Multibody Dynamic simulation

To evaluate the dynamic behavior of the robotic unit inside cleaning environment, ADAMS rigid body model of the two robotic units is built. Some sample cleaning trajectories are generated to assess the dynamic performance of the robotic unit. The description of the model goes below.

[1] **Description of the version 1 ADAMS Model**:

The main components of the robotic unit indicated in figure 1 are: mobile platform (1), robotic arm (2), aspiration shell (3), cabinet (4), engine (5), cable reel

Table 4.3. Modified DH parameters of the *P-6R* architecture.

j	*σ*	*μ*	*γ*	A	*d*	*θ*	*R*
1	0	1	0	0	0	th_1	r1
2	0	1	0	*pi*/2	d2	th_2	r2
3	0	1	0	0	d3	th_3	r3
4	0	1	0	-*pi*/2	d4	th_4	0
5	0	1	0	*pi*/2	0	th_5	0
6	0	1	0	-*pi*/2	0	th_6	0

(6) and the tool (7). Masses of all the components are adjusted according to the real masses of the components. Ground wall and ceiling constitute the cleaning environment.

a. **Robotic Arm (*P-6R* architecture)**

Modified DH model of the architecture is formulated using SYMORO-software [1]. The model defines set of parameters, joint variables and constants, which decide the pose of the arm and its geometry respectively. Table 1 lists all the parameters while figure 2 shows the resulting architecture. These parameters are incorporated into ADAMS model so as to allow quick modifications in the pose as well as dimensions of the lengths to evaluate changes in dynamic performance. Here, parameters d_3 and d_4 determine two link lengths of the arm and values of these parameters are 0.6m each. Parameters r_2 and r_3 take equal values so that the offset between joints th_1 and th_3 along axis Z_3 stays zero.

b. **Mobile Platform:**

Model of the platform with dimensions equivalent to the original dimensions is built in ADAMS. Platform supports the mounting of Cabinet, Cable reel, Vertical slider and aspiration unit. Platform has six wheels as shown in the figure below. Three axles are named 1, 2 and 3 respectively from front to rear. The wheel naming respects the W_{as} Convention with a an axle number (1 for front, 2, for middle, 3 for rear) and s a side number (1 for right, 2 for left).

c. **Constraints:**

To model the interaction between the robotic unit and the cleaning environment, contacts are added into the model. These 3D contacts evaluate contact forces based on the inter-penetration of meshed parts. Four wheels of the platform plus the two additional center wheels have contacts with the ground.

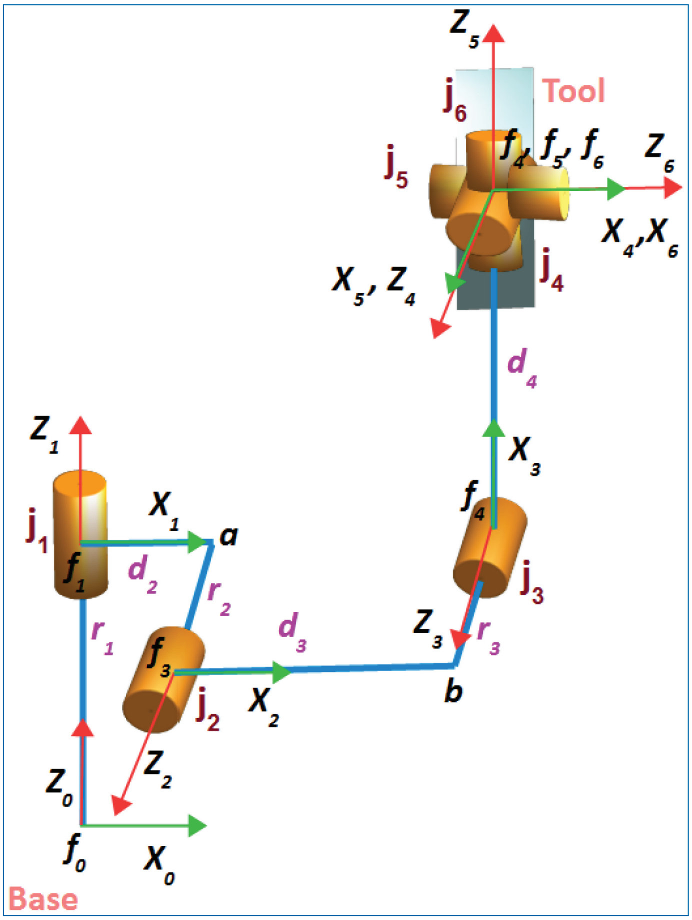


Figure 4.99. Visualization of the *P-6R* architecture.

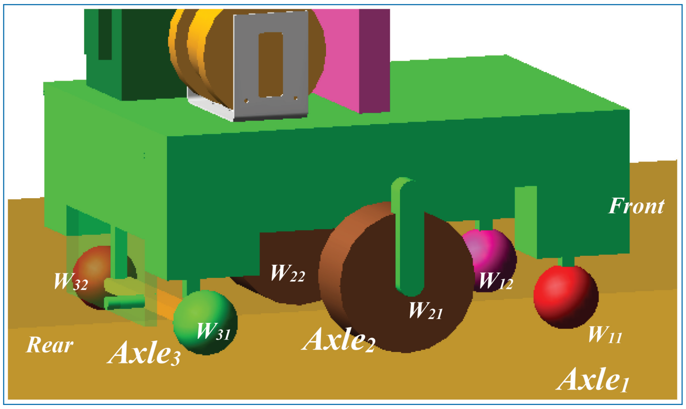
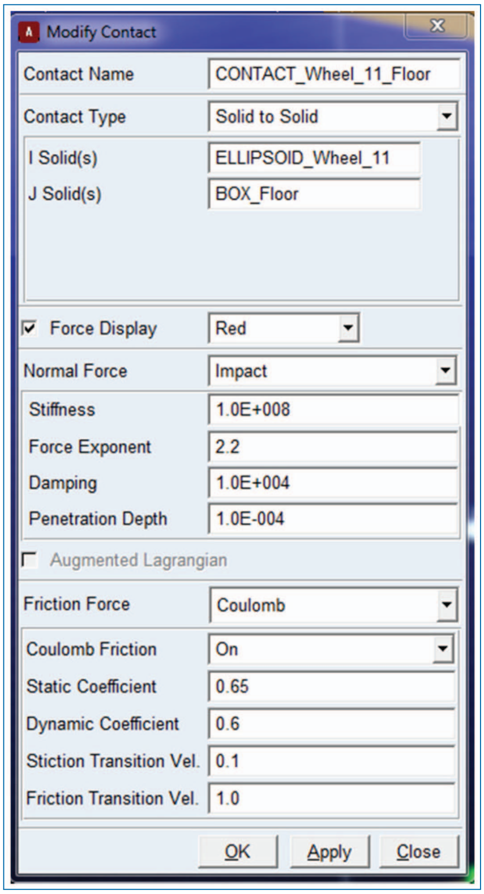


Figure 4.100. Nomenclature of the wheels.


Modify Contact

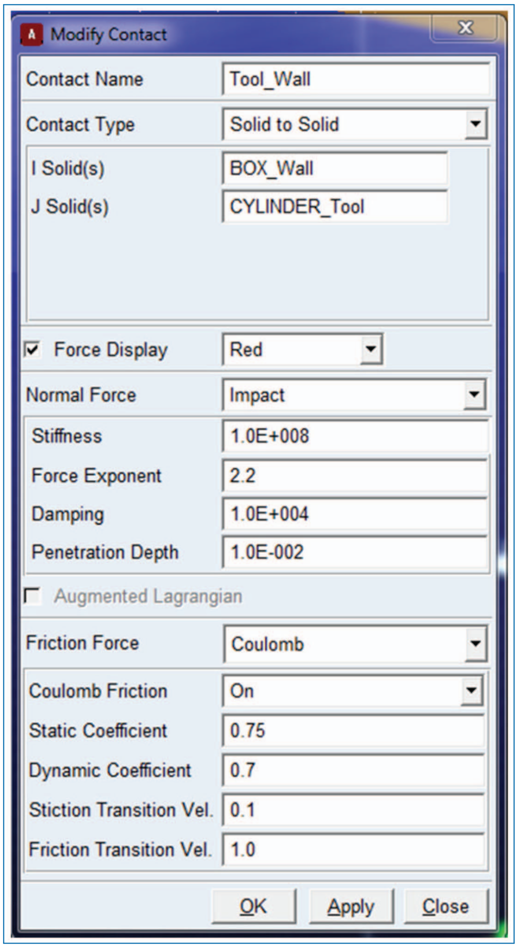
Contact Name	CONTACT_Wheel_11_Floor
Contact Type	Solid to Solid
I Solid(s)	ELLIPSOID_Wheel_11
J Solid(s)	BOX_Floor
☑ Force Display	Red
Normal Force	Impact
Stiffness	1.0E+008
Force Exponent	2.2
Damping	1.0E+004
Penetration Depth	1.0E-004
☐ Augmented Lagrangian	
Friction Force	Coulomb
Coulomb Friction	On
Static Coefficient	0.65
Dynamic Coefficient	0.6
Stiction Transition Vel.	0.1
Friction Transition Vel.	1.0

Figure 4.101. Wheel-ground contact in ADAMS.

Figure 4.101 shows contact parameters for wheel W_{11} with the ground. Values of parameters – stiffness, force exponent, damping, and penetration depth are kept unchanged (ADAMS default values). Coefficient of friction values are approximated as 0.65 and 0.6 for static and dynamic friction respectively. Also, a contact is provided between the tool and the wall to be cleaned (Figure 4.102) so as to apply desired cleaning force and to experience reaction forces.

d. **Workspace of the Robotic Arm**

In order to imagine the workspace of the robotic arm at a given configuration of the robotic unit, 3D geogebra model is built. Since ADAMS can't construct exact geometrical volume of the workspace, geogebra is preferred over ADAMS. Pink box represents the space occupied by ancillaries (cabinet, engine, aspiration unit, cable reel etc.)



Modify Contact

Contact Name	Tool_Wall
Contact Type	Solid to Solid
I Solid(s)	BOX_Wall
J Solid(s)	CYLINDER_Tool
☑ Force Display	Red
Normal Force	Impact
Stiffness	1.0E+008
Force Exponent	2.2
Damping	1.0E+004
Penetration Depth	1.0E-002
☐ Augmented Lagrangian	
Friction Force	Coulomb
Coulomb Friction	On
Static Coefficient	0.75
Dynamic Coefficient	0.7
Stiction Transition Vel.	0.1
Friction Transition Vel.	1.0

Figure 4.102. Tool-wall contact in ADAMS.

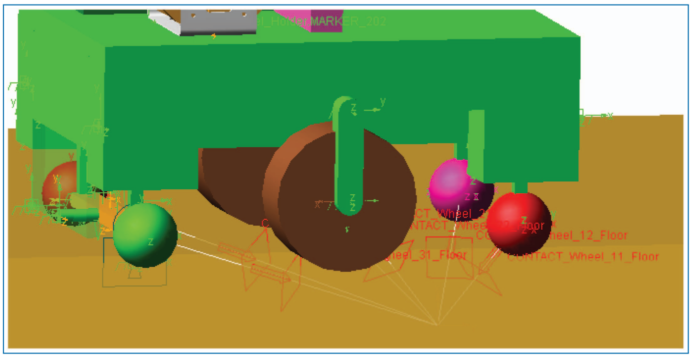


Figure 4.103. Contacts between wheels and ground.

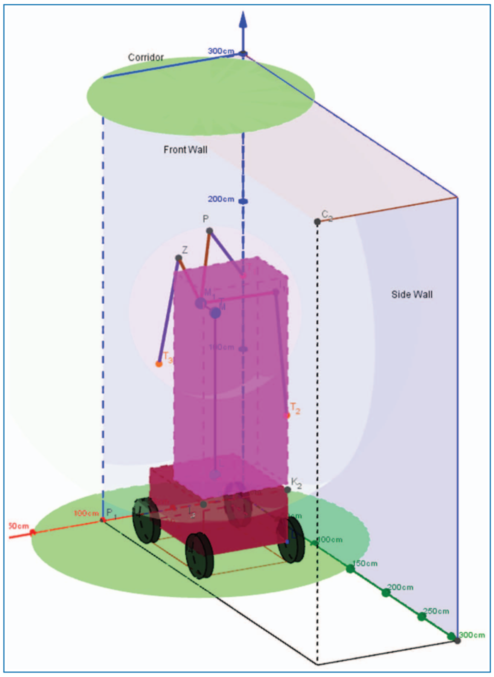


Figure 4.104. Intersection of the workspace with ground and ceiling.

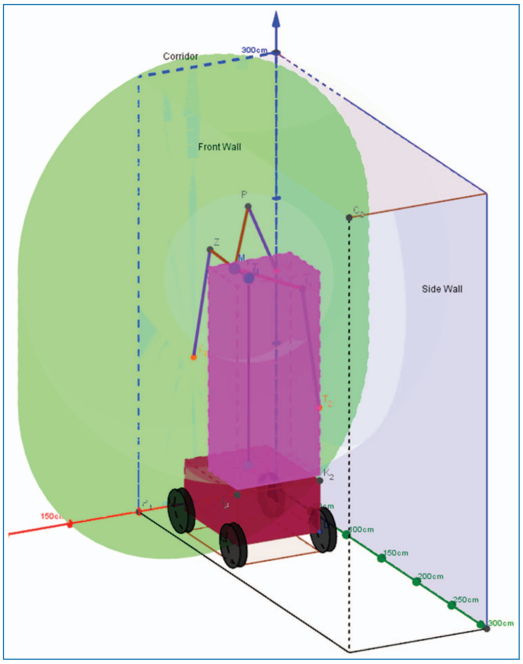
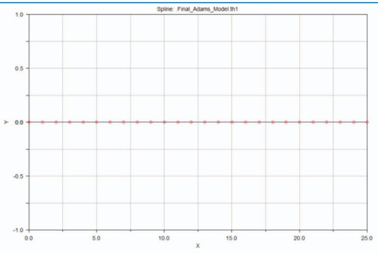
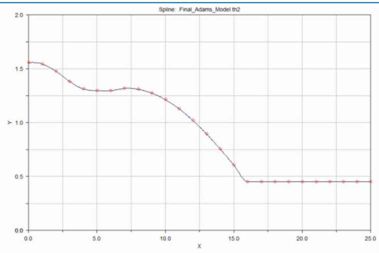


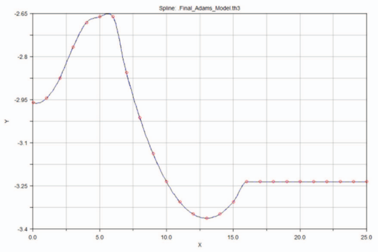
Figure 4.105. Intersection of the workspace with side and front all.



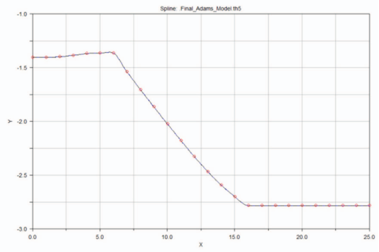
(a) Joint (j_1) angle - th_1



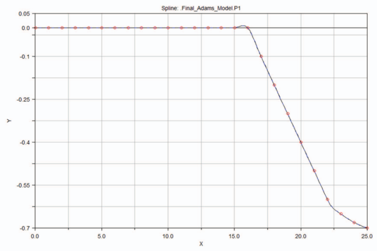
(b) Joint (j_2) angle - th_2



(c) Joint (j_3) angle - th_3



(d) Joint (j_3) angle - th_4



(e) Slider displacement P_1

Figure 4.106. Joint angle curves of the joints actuated during vertical trajectory.

e. Dynamic Simulation

The aim of this simulation is to test the functioning of the model by simulating a simple vertical cleaning trajectory. Ground reaction forces also have to be evaluated in order to assess the stability of the robotic unit.

i. Vertical wall cleaning trajectory

As already described in the section 2.c, an inverse geometric model of the robotic arm is built in matlab. A vertical cleaning trajectory is generated to get the joint angles for the corresponding end-effector positions. These joint angles are imported to ADAMS to feed the respective joints of the dynamic model.

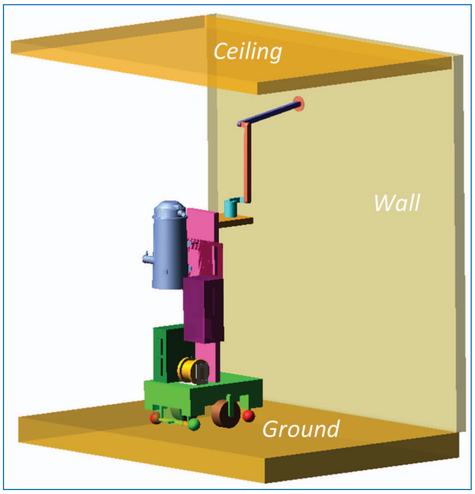


Figure 4.107. Initial position of the robotic unit.

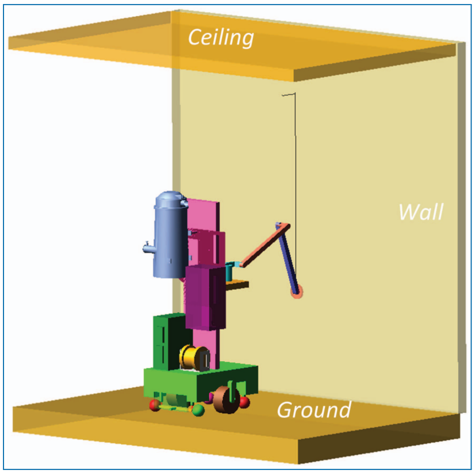


Figure 4.108. Final position of the robotic unit.

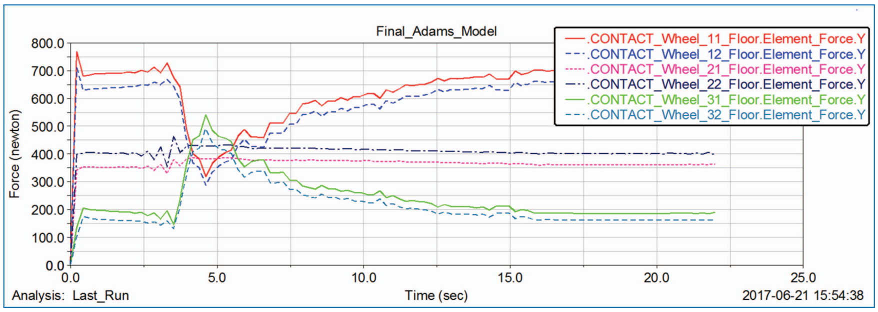


Figure 4.109. Ground reaction forces while performing vertical trajectory.

Ground reactions obtained during this simulation are shown in the figure below. Form Figure 4.109, it can be seen that none of the ground reaction forces approach to zero during the complete cleaning motion. Thus, the robotic unit has a stable behavior. Ground reactions on front wheels (W_{11} and W_{12}) initially have higher values since the robotic arm operates in the frontwards. Once the robotic arm starts pushing on the wall, ground reactions on the front walls decline whereas, those on the rear wheels increase.

ii. Conclusions on Version 1

a. Parameterized multibody model of the robotic unit was built in ADAMS. Parameterzation of the robotic arm was achieved through DH parameterization
b. 3D geogebra model helped in evaluating intersection of the workspace of the robotic arm with cleaning surfaces which is dependent on the configuration of the mobile base (distance of the robotic unit from the wall).
c. The model was successfully tested by implementing a vertical cleaning motion. Model showed a realistic behavior while performing the trajectory. Static and dynamic stability was observed that gives us the confidence for actual testing of the prototype.
d. Limited workspace: Figure shows the collision of the Link M_1I_1 with the pink box. Thus, the workspace of the robotic arm is limited only frontwards. Limited workspace implies restriction on the workspace to be cleaned.
e. Issues due to vertical slider: To enable the movement of the vertical slider, plenty of space is consumed (shown with the green box in figure 11). This space can't be used for mounting components and hence the big pink box sized space is needed at the back side to allow all the mountings.

However, a robotic unit integrated with a 6 DoF arm having three shorter moving segments takes eliminates these limitations and allows all around cleaning. In deliverable **D2.1** the idea of such a robotic unit has been already elaborated. In the present deliverable the attempt will be to build the multibody model of the robotic unit and evaluate stability. This robotic unit will be referred as version 2 in this draft.

2. **Description of the Version 2 ADAMS Model:**

The components of the version 2 robotic unit are: mobile platform, 7R redundant robotic arm, aspiration shell, cabinet, engine and the tool and stabilizers. Masses of all the components are adjusted according to the real masses of the components.

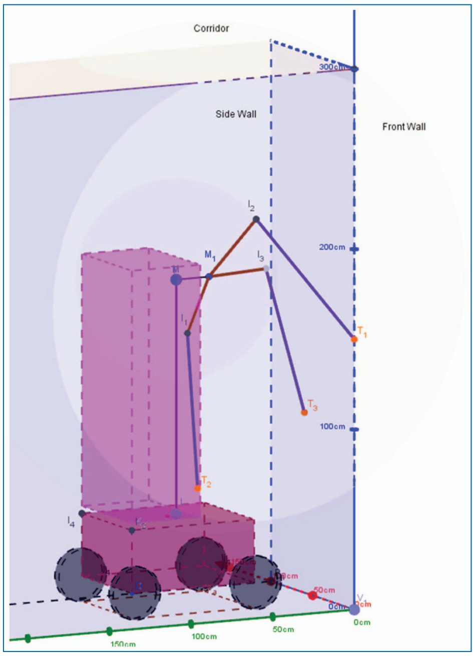


Figure 4.110. Collision of the link with the slider.

a. **Robotic Arm**:

Similar to version 1, robotic arm is parameterized according to the DH parameters. Table 2 lists the modified DH of the *7R* redundant architecture. Parameters d_3, d_4 and r_5 determine the three link lengths of the architecture. In ADAMS model, d_3 is 0.4m while d_4 and r_5 are 0.7m each. These values are obtained from the CAD design provided by TLabs based on the idea proposed by SIGMA. Offsets r_3 and r_4 collectively compensate for the offset r_2 such that the offset between joints j_1 and j_4 along z_4 is zero.

A tree diagram explaining the architecture of the robotic arm is presented in Figure 2. The figure presents joints, frames attached to each joint as well as the parameters that determine the link lengths and joint offsets. DH-parameters define the architecture of the arm. These parameters are incorporated in modelling to modify the pose and dimensions of the robotic arm and study the effect of their variation.

b. **Wheels Nomenclature**

Wheel nomenclature follows the same methodology as that of version 1. In addition to wheels, stabilizers are also added to the mobile platforms that are named using same nomenclature.

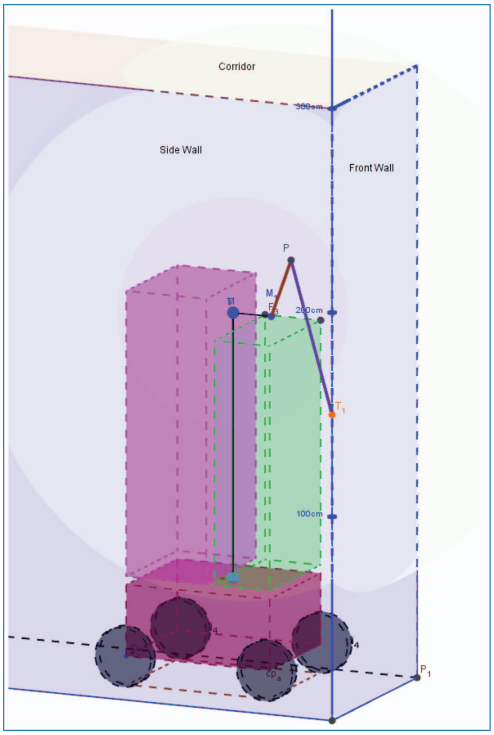


Figure 4.111. Space consumption in Version 1.

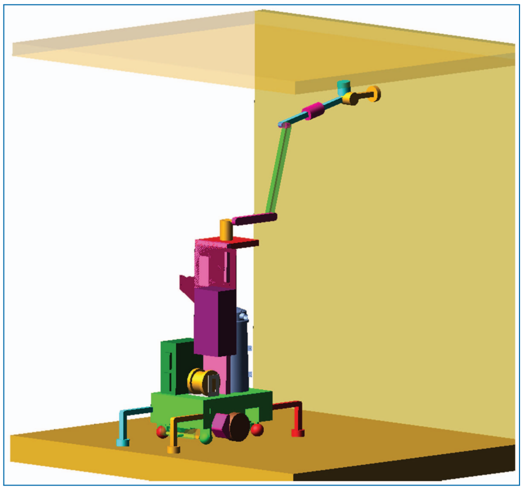


Figure 4.112. ADAMS model of the Robotic unit.

Table 4.4. DH parameters of the robotic arm.

j	σ	μ	γ	α	d	θ	r
1	0	1	0	0	0	th_1	r1
2	0	1	0	*pi*/2	d2	th_2	r2
3	0	1	0	0	d3	th_3	r3
4	0	1	0	0	d4	th_4	r4
5	0	1	0	-*pi*/2	0	th_5	r5
6	0	1	0	*pi*/2	0	th_6	0
7	0	1	0	-*pi*/2	0	th_7	0
8	2	0	0	0	d8	- *pi*/2	0

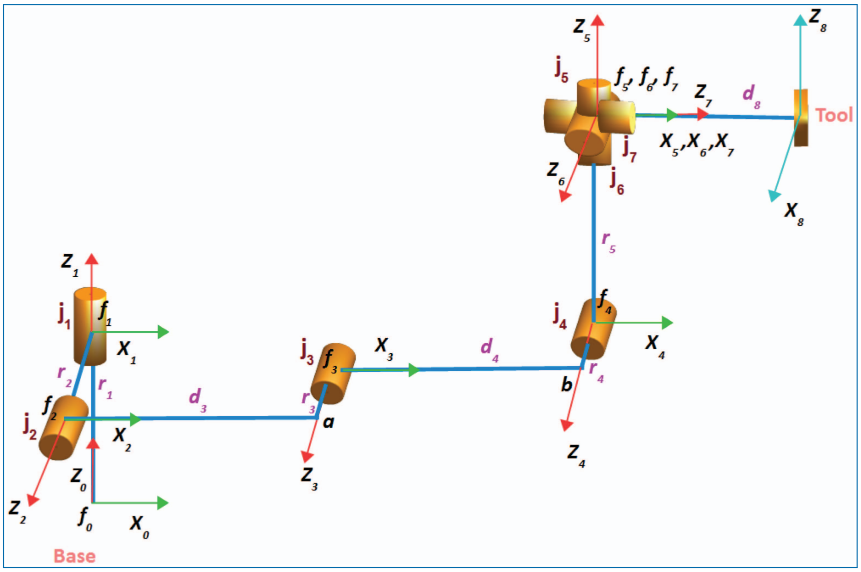


Figure 4.113. Visualization of the *7R* architecture.

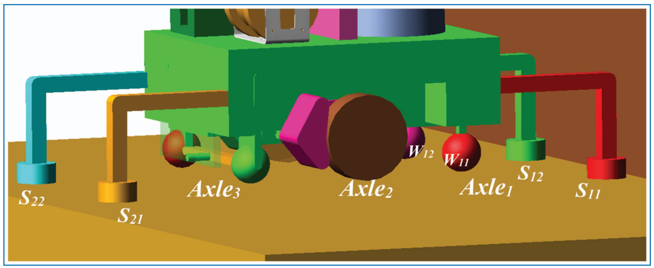


Figure 4.114. Wheels Nomenclature.

c. **Workspace of the robotic arm**

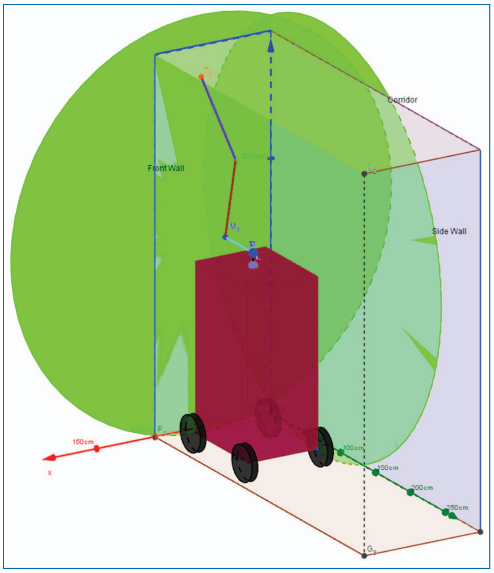


Figure 4.115. Intersection of the workspace with side and front wall.

d. **Dynamic Simulation:**

In case of presence of obstacles, robotic arm has an overhanging configuration and hence the stability can be a critical issue. Currently, to investigate a worst case scenario dynamic stability of the robotic unit is tested for two cases – 1) Without stabilizers 2) With stabilizers. Top to bottom vertical trajectory is simulated to study the dynamic response of the system. The objective of this simulation is also to study the stability in worst case and evaluate the need of stabilizers.

1. Vertical Trajectory without stabilizers:

Robotic unit is places as close to the obstacle as possible. Width of this obstacle is 1m. A vertical trajectory is traversed to study the stability of the robot. Figure 4.114 shows initial and the final position of the trajectory performed by the robotic unit. Figure 4.118 shows ground reaction forces in Y direction on all the wheel contact are presented.

Front wheels (W_{11} and W_{12}) experience higher reaction forces as the robotic arm is operational on the front side. Middle wheels (W_{21} and W_{22}) have moderate reaction forces while the rear wheels (W_{31} and W_{32}) are subjected to low reaction forces.

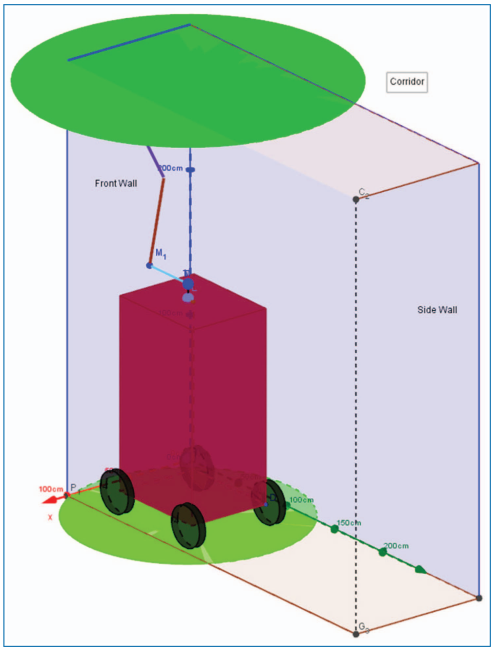


Figure 4.116. Intersection of the workspace with ground and ceiling.

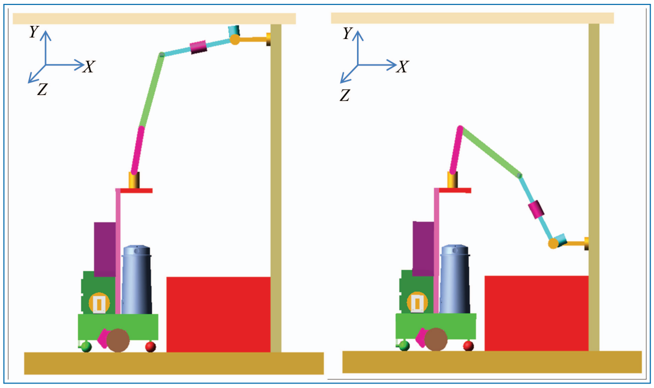


Figure 4.117. Initial and final positions of Robotic unit for vertical trajectory.

In Figure 4.116, a graph of lateral stability margins against time for the vertical trajectory is shown. Stability margin is calculated by a formula:

$$\frac{(W_{11} + W_{21} + W_{31}) - (W_{12} + W_{22} + W_{32})}{(W_{11} + W_{21} + W_{31}) + (W_{12} + W_{22} + W_{32})}$$

A good stability can be seen during the entire operation. Longitudinal stability margin is difficult to calculate here as we have six wheels. However, the rear wheels

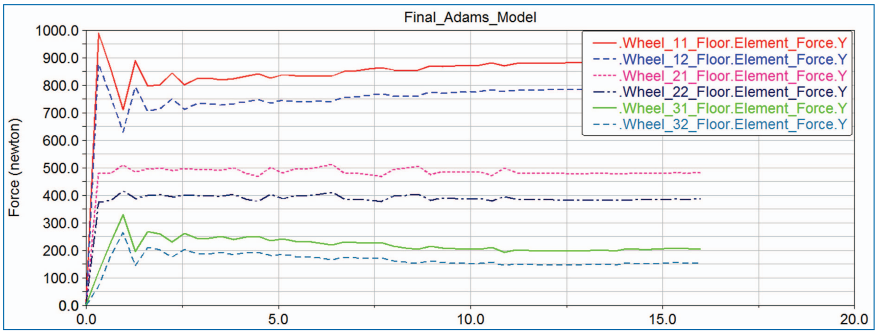


Figure 4.118. Ground reactions on wheel contact points.

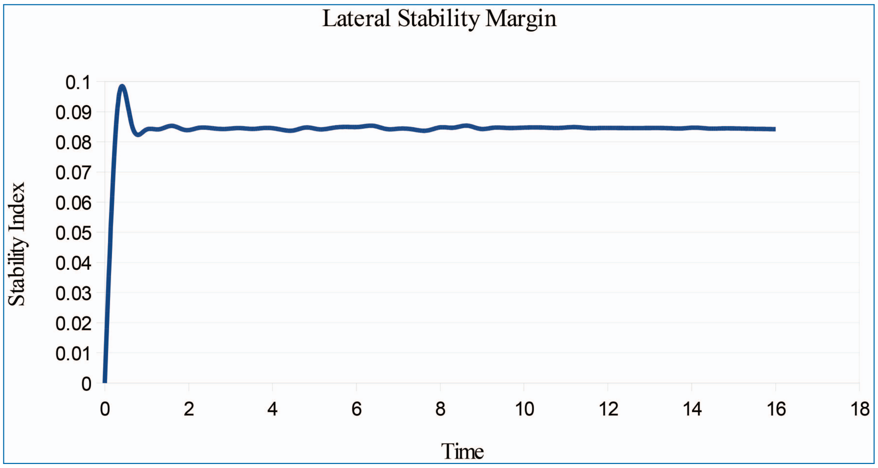


Figure 4.119. Lateral stability margin.

show a reaction force in the range of 200–250 N and hence we can say that the robotic unit is not close to instability.

2. Vertical Trajectory with Stabilizers:
   a. Stabilizers aligned in front

During this simulation, robotic unit was lifted up by 4 – stabilizing legs, hence no wheels are in contact. Stabilizers deployed on the ground carry the entire load of the robotic unit and a larger support polygon is capable of imposing higher stability to the robotic unit. Figure 4.121 shows ground reaction forces on 4-stabilizing legs.

The longitudinal stability margin is calculated as:

$$\frac{(S_{11} + S_{21}) - (S_{12} + S_{22})}{(S_{11} + S_{21}) + (S_{12} + S_{22})}$$

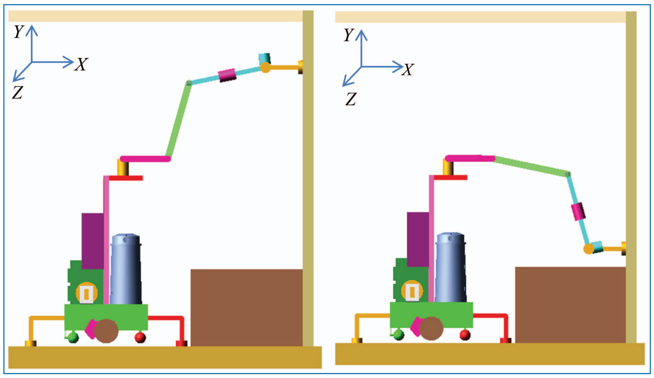


Figure 4.120. Initial and final positions of Robotic unit for vertical trajectory.

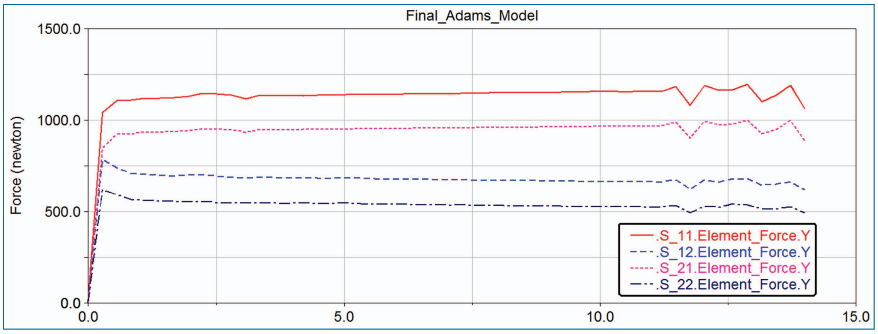


Figure 4.121. Ground reactions on stabilizer contact points.

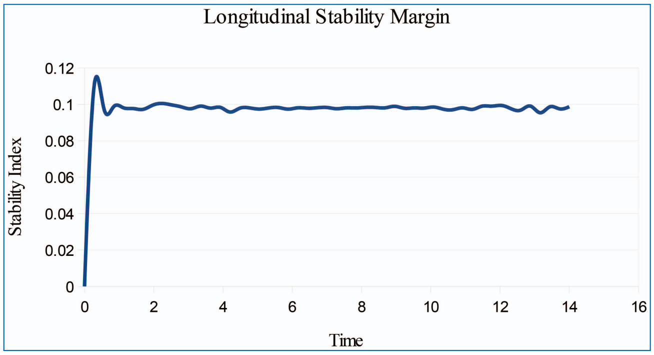


Figure 4.122. Longitudinal stability margin for vertical trajectory.

Figures 4.119 and 4.120 show lateral and longitudinal stability margins for the vertical trajectory with stabilizers extended in the front of the robot. Using stabilizers improvement in the lateral stability can be seen by comparing graphs presented in Figures 4.116 and 4.120.

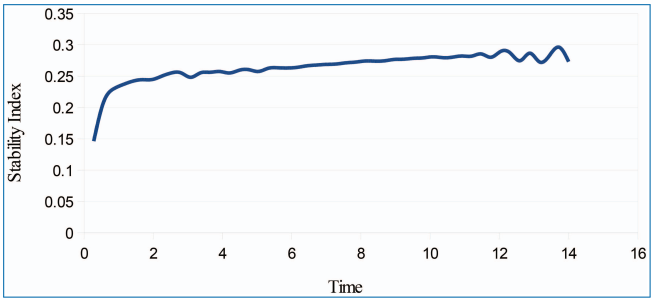


Figure 4.123. Lateral stability margin for vertical trajectory.

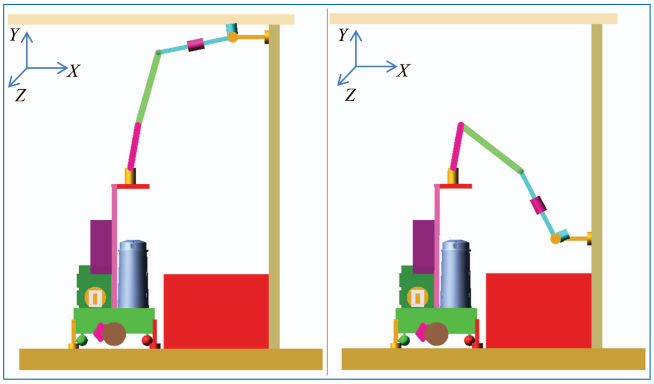


Figure 4.124. Initial and final positions of Robotic unit for vertical trajectory.

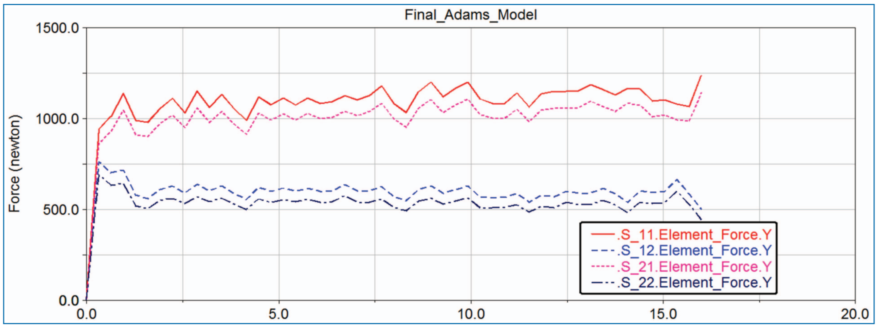


Figure 4.125. Ground reactions on stabilizer contact points.

b. Stabilizers aligned laterally:

In this scenario, stabilizers are aligned laterally, hence increasing the support polygon in lateral direction.

The ground reactions are similar to the one obtained in case of stabilizers aligned frontwards (Figure 4.122). However, the ground reaction curves in this case are not as smooth as they are in Figure 4.118 indicating more vibrations. Thus, a strategy of putting stabilizers in front is recommended.

4.6.6 Outlook and Future Work to be Carried Out

The simulation environment developed in during task 2.3 will now serve for rigorous investigation of efficient cleaning trajectories. As version 2 is identified to be a better solution, future investigation will mostly concentrate this version.

The detailed Workflow of the simulation goes as belows:

[1] Wall case:

- 1.1 Vertical Trajectory and front-rear stability
  - 1.1.1 Bottom Stabilizers
    - 1.1.1.1 Ascending
    - 1.1.1.2 Descending
    - 1.1.1.3 Comparison
    - 1.1.1.4 Vertical dynamic efforts vs gravity – influence on front-rear stability margin
    - 1.1.1.5 Removal of front stabilizers for improving stability margin
    - 1.1.1.6 Bottom-top stabilizers
      - 1.1.1.6.1 Ascending
      - 1.1.1.6.2 Descending
      - 1.1.1.6.3 Comparison
      - 1.1.1.6.4 Vertical dynamic efforts vs gravity – influence on front-rear stability margin
      - 1.1.1.6.5 Interest of bottom-top co pushing
  - 1.1.2 Horizontal Trajectory and lateral stability
    - 1.1.2.1 Low altitude of end effector, low slider
    - 1.1.2.2 Mid-altitude of end effector, high slider
    - 1.1.2.3 High altitude of end effector, extended arm
    - 1.1.2.4 Collision avoidance with stabilizer feet
  - 1.1.3 Influence of the distance from the wall
    - 1.1.3.1 Collisions
    - 1.1.3.2 Singularities
- 1.2 Ground
  - 1.2.1 Hollow workspace
  - 1.2.2 Preferable trajectory

- 1.2.3 Collision avoidance with the base and stabilizer feet
- 1.2.4 Path planning of the mobile platform
- 1.3 Ceiling
  - 1.3.1 Limited plane workspace
  - 1.3.2 Worst case scenarios for stability
  - 1.3.3 Collision avoidance with stabilizer feet
- 1.4 Productivity evaluation of the mono-robot system
- 1.7 Interest on Productivity of using multi-robot system

Summary and Conclusion

Expected results: It is expected that through these realistic simulations, precise and efficient trajectories could be obtained. This task will further provide input to task **T2.4 for further improving stability through collaboration**. The results from this section would also be helpful for motion planning and task planning.

References

[1] Wisama Khalil, et al. OpenSYMORO: An open-source software package for Symbolic Modelling of Robots. IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Jul 2014, Besan¸con, France. pp.1206-1211, 2014.

4.7 Hybrid Control

Deliverable 6.3
Semi-autonomous execution of a of a single asbestos-removal-task implemented and process control system documented

Authors: Tobias Haschke, Oriol Orra, Diego Fogliacco, Tim Detert, Claudia Cornely

4.7.1 Deliverable in the Context of the Work Program

The deliverable 6.4 is marked as the forth milestone of the Bots2ReC project. As written in the deliverable name the project plan scheduled a semi-autonomous execution of a single asbestos removal task. This means in form of robot actions that the asbestos removal robot can move towards the desired execution point, can bring the grinding disk towards the wall and start to clean a plain wall element.

In this case, the definition of "semi-autonomous" execution is chosen as the user can always interact with the robot through tele-operation via the user interface. As an example the operator can drive the robot via joystick to the desired operational point.

According to these functional parts, this deliverable strictly depends on different pre steps that are mentioned below. The first important input for this deliverable are the 3D environment data, which are delivered by the **Task 6.1**. The **Task 6.2** is in a quite close connection to the environment data as it uses the environment model for localization and mapping during the task execution. As this deliverable is meant to be "semi-autonomous", the connection to **Task 6.2** is not as close as to **Task 6.1**. Beside these tasks, this deliverable depends on the implemented control approaches for the mobile platform and especially for the robotic arm. These control approaches are implemented within the **Task 4.3** and **Task 5.3** and require robust components due to the aspired use case. As the process of asbestos can vary due to changing environment properties (e.g. wall thickness or subsoil) some data sets of process parameters need to be extracted from the removal tests of **Task 1.4**.

After this deliverable the next steps are to widen up the size and the number of asbestos removal tasks. According to this the current solution needs to be adapted to the second version of the prototype which will also bring the side-effect adding multiple inhomogeneous robots to the robot team that should execute the removal tasks in parallel. These steps are going to be solved in **Task 6.5** and **Task 2.4**.

4.7.2 Methods and Work Carried Out

The Central Process Control System (CPCS) is fully implemented in C++ within the Robot Operating System (ROS) framework. Due to this, the main elements as e.g. the task planner or task_manager of the CPCS are written as node-based units, which can be distributed over the whole ROS network. Beside the ROS-based implementation, the control algorithms are extended to use Linux real-time kernels to take care of the necessary control frequencies. According to Bots2ReC, the RT Preempt Kernel is used for the self-developed control algorithms. The complete development process is related to the overall asbestos removal process.

4.7.3 Overall Asbestos Process

The overall asbestos removal process can be separated into main processes and their sub-processes that automatically create an easy-understandable workflow for the user and for the programmer as well. All in all the five main processes *Construction* *site preparation*, *Data Mining*, *Task Planning*, *Task Execution* and *Construction site completion* are defined and individual sub-processes are listed within the table below.

#	Main Processes	#	Sub-Process	Part of CPCS
1	Construction site preparation	a	Measurement of asbestos contamination	No
		b	Seclusion of removal area	No
		c	Robotic setup on construction site	Yes/No
2	Data Mining	a	Create environment model	Yes
		b	Specify removal jobs	Yes
3	Task Planning	a	Create complete removal data set	Yes
		b	Generate robot tasks	Yes
4	Task Execution	a	Reactive task management	Yes
		b	Navigate to execution position	Yes
		c.i	Asbestos removal (Open Loop Control)	Yes
		c.ii	Asbestos removal (Closed Loop Control)	Yes
5	Construction site completion	a	Removal of robotic setup	Yes/No
		b	Removing seclusion	No
		c	Official acceptance	No

The first and fifth main processes and their sub-processes are mostly done by external experts and therefore do not need to be described in detail for this deliverable. The only robotic relevant element of these main processes is the setup and the removal of the robotic infrastructure. This contains the CPCS with its computers and displays for the User Input and the local network setup with Ethernet switches and Wi-Fi routers.

The second main process is *Data Mining* that is an all-important step within the robotic process chain. Due to the non-existing 3D environment sensors during the removal process, a correct and exact model of the refurbishment site has a significant impact on the removal efficiency. If the environment model is too inaccurate, the CPCS might not find the necessary overlaps between model and current sensor data. This can have the effect that the robot will also need tele-operated input by the user for simple geometries as plain walls. A part of the environment model is the task allocation towards geometrical elements as walls, floors and ceilings.

Task Planning is the next main process after mining the environment data. It contains the conversion from environment and process data into a complete data set that is called the *removal problem*. Within the planning step, this problem is solved in an optimal way to generate robot-specific commands that can be executed directly.

During the *Task Execution* process, the commands are executed to perform the desired job in the real world. Therefore, it is necessary to implement a task management system that can actuate the mobile base for the navigation towards the operational points. In addition, the task management system also needs the ability to control the robotic arm for the asbestos removal. Especially the multi-modal control of the robotic arm is a key step within the removal process for handling all the different sensor inputs for the optimal task execution.

4.7.4 Realization of sub-processes

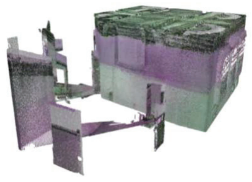
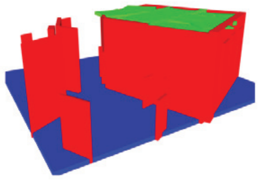
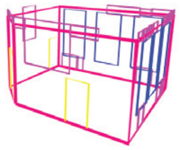
This chapter describes the sub-processes in detail and can point out the technical details for each step. Some elements are already documented in other deliverables and therefore these explanations are skimped.

4.7.4.1 Data Mining

Data mining is a typical pre-process that need to run before the asbestos removal robot is running on site. As it is not possible to run these steps in the necessary accuracy and resolution as there are no sensors planned for this task.

a) Create environment model

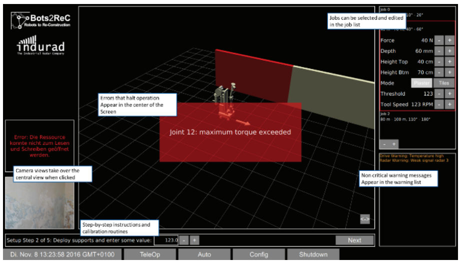
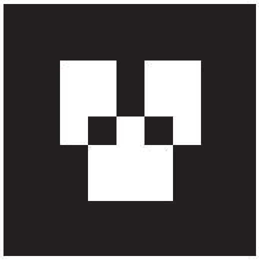
The root of the environment data set is a point cloud, which can be created via two different ways. The first one is to use the so-called BIM files, which are often used for modern refurbishment sites to document all the details about the removal job. However, as BIM is not used for every removal site another solution for the point cloud generation is necessary. For these cases, it is planned to use a low-cost 3D mapping robot that can scan the already secluded rooms beforehand. The point cloud is processed in two consecutive steps from bounding boxes into polygons. The complete workflow and the geometric mechanisms are explained in detail in the **Deliverable 6.2**. In the following table the three processing steps are summarized with three figures taken from the translating algorithm.

Point Cloud	Bounding Boxes	Polygons
		

The outcome of polygons is the pre-defined interface for the task planning routine as all of the polygons contain more information than just the points for their geometry. Out of the shape it is possible to find elements like doors or windows and after a short computation the normal vector can distinguish polygons into walls, ceilings and floors. This property comes in handy for the planning phase as the polygons can be connected to pre-defined task parameters as e.g. maximal lateral speed or nominal tool pressure efficiently.

b) Specify removal jobs

In addition to the properties extracted from the polygons the information about the removal process are not granted as they are related to the working surface and its quality. There are obvious process differences as plaster and tiles which automatically need different tools. But there are also non-obvious differences that can occur between the grinding process on painted and plastered walls.

User Interface	AR Tag
	

Bots2ReC
Robots to Re-Construction
indurad
The Industrial Radar Company
Jobs can be selected and edited in the job list
Errors that halt operation Appear in the center of the Screen
Camera views take over the central view when clicked
Non critical warning messages Appear in the warning list
Step-by-step instructions and calibration routines

Joint 12: maximum torque exceeded

Error: Die Ressource konnte nicht zum Lesen und Schreiben geöffnet werden.

Job 0
10° - 20°
40° - 60°
Force 40 N
Depth 60 mm
Height Top 40 cm
Height Btm 70 cm
Mode Plaster Tiles
Threshold 123
Tool Speed 123 RPM
Job 2
80 m - 100 m, 110° - 180°

Drive Warning: Temperature high
Radar Warning: Weak signal radar 3

Setup Step 2 of 5: Deploy supports and enter some value: 123.0

For handling this input about the surface quality, two options are taken into account for the Bots2ReC prototypes. On the one hand, the user can select wall elements within the user interface and setup pre-defined values for grinding and tiles removal processes. The other option is the automatic detection of the surface with AR tags that can be used during the robotic 3D mapping routine mentioned in 4.1.a. This means that the experts, which are doing the asbestos analysis in the run-up to the removal process, will mark the different wall with corresponding AR tags. The information that is given by an AR Tag can contain information such as “Plaster – contaminated” or “Not contaminated” to ease the task definition for the user. The definition of tasks via AR tags is tested at the IGMR at Aachen University together with the 3D mapping approach.

c) Create complete removal data set

Finally, both information, the polygons and the jobs, are combined together in a so-called semantic map object in C++. This one is the final step of the data mining process and the basis for the *Task Planning*.

4.7.4.2 Task Planning

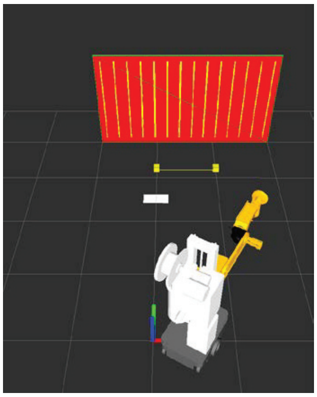
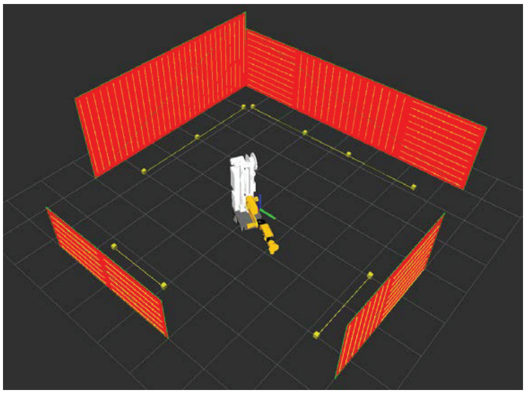
With this complete definition of the asbestos removal problem, the plan for the robotic-based removal process can be generated. Due to the complexity of the problem an optimal way for the *Task Execution* is aspired. The solution should answer questions such as:

- Where does each robot start its removal job?
- How does the user manage problems or unforeseen delays in the best way?
- How many robots are best for this removal problem?
- Which tools should be mounted for this removal problem to finish it as fast as possible?

The details how these questions are answered are worked out in **Task 2.4**. Within this deliverable report, only a brief description shall be granted.

b) Generate robot tasks

The main idea of the *Task Planning* step is driven by the benchmarking process for being as fast as possible for reaching a profitable task fulfillment. Because the robots are wired, the flexibility of each robot is limited by the position of the other robots. Having this in mind, saturation effects exist for the number of robots per room and flat which of cause are related to condition of the removal flat. As an example, two flats with the same base area can be processed with different number of robots if one contains many small rooms and the other one only a few big ones. Beside this question of robots quantity the processing way for one wall by one robot can be optimized as well. As the robot‘s reach is limited due to the size of the arm and the stability margin, walls can be spitted up into small sub-segments that can be processed by the robot from one fix operational point.

Single Wall Task Planning	Room-wide task planning
	

As it is shown in the table above for a *Single Wall Task Planning* situation in this case two operational points (small yellow boxes in front of wall) are necessary to process the complete wall. Combining multiple walls for a *room-wide task planning* will create solutions, as they are shown in the table in the right figure. Within the CPCS software, the solution is separated into some understandable elements for monitoring the systems state and to be executed by the robots:

- *removal_problem*
  One semantic map plus the available robot_team
- *robot_team*
  Collection of available robots and their capabilities
- *removal_plan*
  Solution of the removal problem that contain all removal_jobs for each robot
- *removal_job*
  Collection of all removal_tasks that one robot need to fulfill
- *removal_task*

Removal area of a wall, ceiling or floor that can be removed by one robot from one operational point

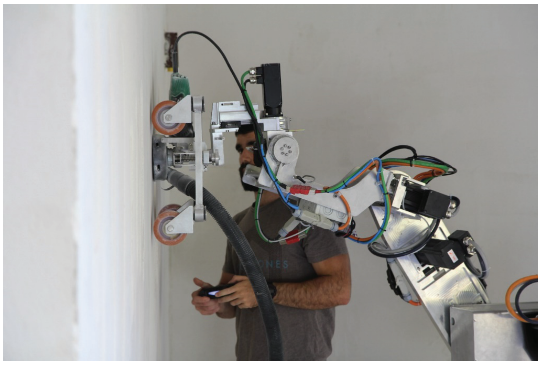
Only the removal_job with the embedded removal_tasks are forwarded to the robot for the execution. The data within these objects can be executed directly by the robots hardware without costly computational effort.

b) Navigation to execution task

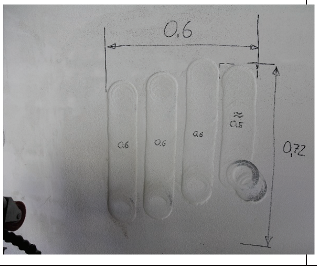
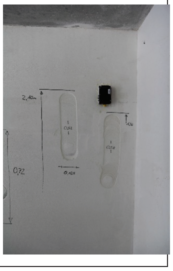
The navigation towards the operational point given by the *robot_job* is done with typical SLAM approaches based on Lidar sensors and new developed algorithms for the radar-based SLAM if the air is or the sensor surface is polluted with dust. The planning of local paths and the subsequent execution are interwoven into the reactive control architecture that was explained above. Further details about the operating modes of the radar-based SLAM are explained within the **Deliverable 6.3**.

c.i) Asbestos removal (Open Loop Control)

According to the progress of the project, it was necessary to run the first grinding tests in an open control loop. This means, that the robotic arm was fully controlled via the joint encoders without additional information from the process. The planned input by the process are the distance measurements by the iLDRs from Indurad and the Force/Torque feedback of the tool. The Force/Torque feedback is available over two different ways where the one way is to measure it directly with a force/torque sensor at the end effector. The other way is to use the compliant tool and to compute the forces out of the spring stiffness and the measured travelling distance of the compliant tool. With this setup the user finds the initial position of a grinding movement via tele operation. The situation of aligning the tool to the wall is shown in the figure below.

Moving towards the processing wall with a PS3 Controller (Open Loop)


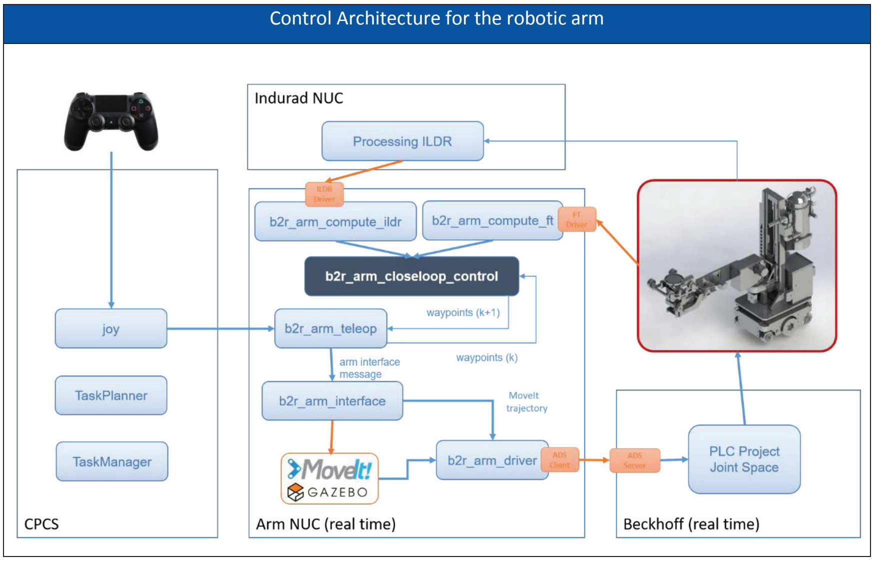
After aligning the tool, some pre-defined trajectories can be executed. In the figure below some rudimentary vertical grinding movements are shown. According to the open loop control the lack of process information, cause some problems for the grinding execution. For example, it is quite difficult for the robot to keep touch to the wall during turns or other non-linear movements with the compliant tool. Often the turning motion of the castor wheels initiate shear forces, which oblige the grinding disk to dive deeper into the walls surface. During this non-desired motion, the force limits of some motors are exceeded and the robotic arm throws an emergency stop.

Single Grinding Operation	Multi-Grinding Operations	Grinding near walls and objects
		

Overall, the open loop control approach is just an intermediate solution for gaining first test data of the grinding process. The first tests have shown that adding the 1D radar sensors and the force/torque feedback is inevitable for a robust removal process.

c.ii) Asbestos Removal (Closed Loop Control)

In parallel to the open loop control tests the closed loop control approach was developed and implemented. As the control approach represents a multi-modal control approach which is not build-in into the Beckhoff Controller of the robotic arm, the self-written controller is running on a real-time preempted Linux NUC. This configuration allows high control frequencies because of the guaranteed computation of the implemented control algorithm by the operating system. Basically there are five building blocks of this control architecture that are shown in the figure below. While the *CPCS* is playing a tangential role the main elements are the *robot*, the *beckhoff_controller*, the *indurad_nuc* and the *arm_nuc*. All the interfaces marked in orange are typical ROS drivers which connects the different elements through pre- or self-defined ROS messages. The only exception is the connection towards the *beckhoff_controller* which is implemented with an ADS Client and Server structure.



The compensation of process influences through motions of the robotic arm are computed in an interwoven setup of MoveIt, an analytical inverse kinematic solver and the self-written multi-modal controller *b2r_arm_closeloop_control*. A special feature of this setup is the independency of position and force/torque control. This means that there are two different controllers written by the Bots2ReC consortium where the first one is controlling the motors of the arm by position and velocity and the other controller is only using forces and torques. For changing between the different controllers the developer just need to change the PLC Project on the *beckhoff_controller* to address the motor drivers in the correct way. These two approaches are implemented in parallel for running comparable tests for defining the better option for the asbestos removal process.

4.8 Removal Control Approaches

Task 5.3 Control and sensors system of the robotic arm

For the first implementation of the control of the arm while performing the grinding task, a smooth position-force hybrid controller was proposed. This controller changed smoothly between free space (while approaching the wall to be grinded) and contact (while beginning the grinding) modes, thus reducing the impact force of previous similar approaches. The controller was tested not only in a simulation based on Matlab+Adams but also with the Kuka LWR robot at Sigma Clermont by using a camera (for wall distance measurement) and a wrist force-torque sensor. This controller presented good position and force tracking performances and the impact factor was small. Nevertheless, the lateral forces had an important effect on the controller because of the large offset between the disc tool center (in contact with the wall) and the wrist center (where the force-torque sensor is installed), which generated large tilting wrenches that involved tool contact loss when the surface of the wall was irregular. In addition, the controller was dependent on the wheels installed around the tool and these wheels also lost contact. Finally, the cutting parameters (i.e. feeding velocity and grinding force) were not estimated accurately, hence leading to inaccurate results in the grinding quality.

Thereby, a second version of the arm controller is proposed for overcoming all these shortcomings of the first version. The grinding model is identified and included in the controller and the wheels around the tool are removed. In addition, the controller is unified by commanding the arm and the wrist as one unit (and not separately, as in version 1). This controller employs a hybrid hybrid velocity-force control strategy expressed in a desired frame in the end-effector. Addition to that, the proposed controller makes the disc of the grinding tool adapt to the curvity change of the wall surface and corrects the desired directions of force and velocity in real-time. The controller is tested again on the KUKA LWR 7-dof robotic arm programmed to grind different planar and curved walls with unknown geometry. The KUKA arm is again equipped with a six-axes ATI Gamma 3 force-toque sensor and a grinding tool to treat a piece of wall covered with resurfacing concrete. As shown in Figure 4.126, there are no wheels supporting the tool.

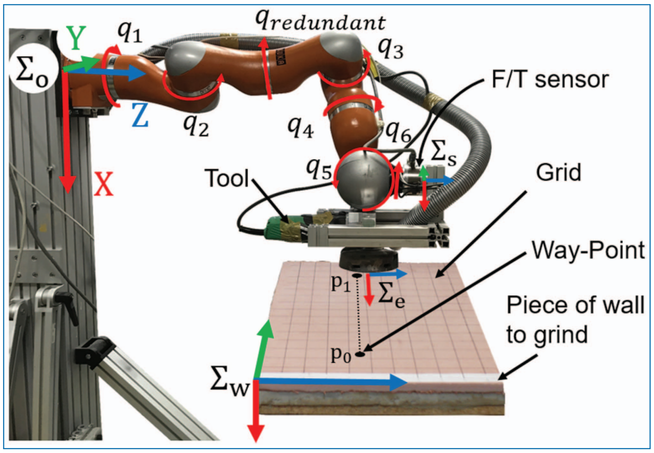


Figure 4.126. The robotic system equipped with grinding tool and force/torque sensor to perform grinding on a piece of wall. The grid is to evaluate the wall before and after grinding.

The three main new features of this new controller strategy are going to be detailed:

- The definition of new compliant frames with regard to the tool, for more precise velocity and force control while contact
- The modeling of the lateral grinding forces, that provoked the tool contact loss with the previous controller
- The non-contact forces elimination from the force-torque sensor readings, so that a more precise contact force control can improve the grinding quality

Choice of the compliant frames

The choice of the velocity and force control frames in the grinding controller implemented here is crucial. The chosen frame for force control is $\{cf\}$ on the center of the disc, it coincides with the end-effector frame origin $\{e\}$ and it is rotated by $\pi/2 - \varphi_v$ around e X, while the control frame for velocity and motion control is $\{cv\}$ fixed at an offset $-R$ along the z direction of $\{cf\}$. The frames are depicted on the disc in Figure 4.127.

There are several reasons why these frames are chosen like this. The force control frame must be fixed to the center of the grinding disc to have good measurements of the lateral forces acting on the disc, and good force control of the normal force at the wall along the direction ${}^{cf}x$, hence there is no velocity control along this direction while the disc is in contact with the wall. This released direction allows the robot to move along the surface of the wall while maintaining the axis of force control normal to the surface. Hence, a desired force can be controlled between the

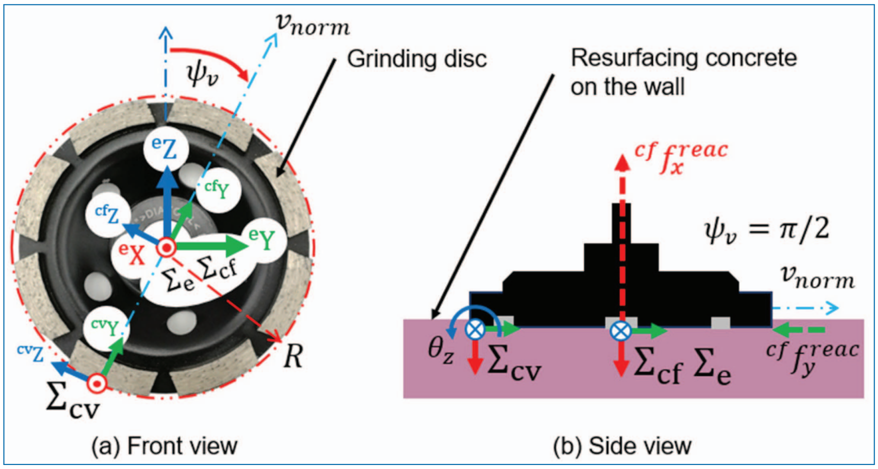


Figure 4.127. The choice of the frames for simultaneous force and motion control.

disc and the wall along this direction. Velocity control is applied in the frame $\{cv\}$ along the remaining linear directions, namely ^{cv}y and ^{cv}z. Hence, position control is applied in the same frame to the remaining axes which are not controlled in force or velocity: the three orientations around ^{cv}x, ^{cv}y and ^{cv}z. The reason why $\{cv\}$ is chosen as in Figure ??, is that the proposed controller needs to adapt automatically the surface based on special orientation controller that will be explained in the next sections. Hence, if the disc is moving along the wall with velocity v_{norm} along ^{cv}y and it needs to orient, applying rotation θ_z around ^{cv}z will generate no reaction force along ^{cv}x which is natural and important for the grinding quality. However, if $\{cv\}$ was chosen to coincide with $\{cf\}$, applying the rotation θ_z around ^{cv}z will generate reaction force along ^{cv}x and the wall can be damaged.

Modeling of the lateral grinding forces

The key idea of this controller is to adapt the orientation of the disc to the wall surface based on keeping the lateral reaction force $^{cf}F^{lateral}$ around a desired value that should be appropriate for grinding. However, for the case of wall grinding in the construction industry, there has been no published model that identify the nominal grinding lateral force $f^{lateral\_nom}$. Moreover, the desired normal grinding force $^{cf}f^{reac}$ and the translational velocity of the disc on the wall $^{cv}v_{norm}$ are not known as well.

Hence, in order to explore the relation between $^{cf}f^{reac\_x}$, $^{cv}v_{norm}$ and the nominal grinding lateral force $f^{lateral\_nom}$, it is necessary to firstly have an idea about the values of $^{cf}f^{reac}$ and $^{cv}v_{norm}$ that should be used as input desired values for the controller. Thus, assisted manual grinding tests are carried out with the experimental setup shown in Figure 4.128. These tests were done by setting the

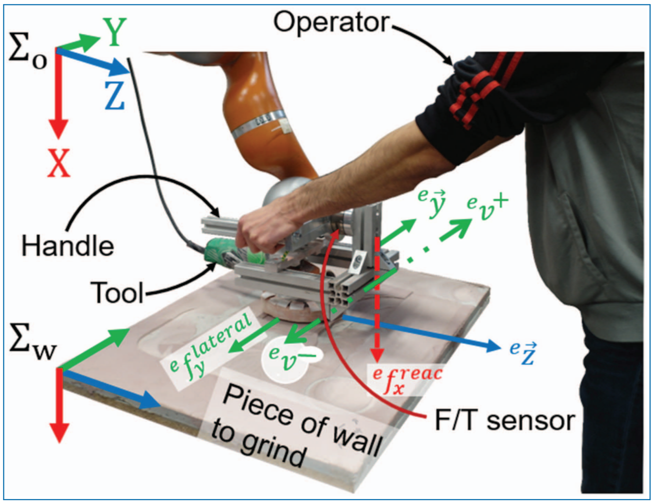


Figure 4.128. The manual grinding setup used to determine the normal grinding force f_{reac} and the translational velocity of the disc on the wall v norm. It is equipped with grinding tool, force/torque sensor and a handle.

robot controller to Cartesian impedance mode, this mode allows to constraint the motion in some directions and release it for others. The wall used for the data collection is assumed planar and its frame $\{w\}$ is set parallel to the frame $\{o\}$. To this end, the robot is set to move freely by hand along the ^ox and ^oy directions while the direction along ^oz is locked. The three orientations around ^ex, ^ey and ^ez are fixed such that the frame $\{e\}$ is parallel to $\{w\}$ in order to keep the disc parallel to the wall. Then, the operator performed three grinding tests by moving the wrist manually along ^ey and pressing along ^ez to grind the piece of wall.

The average value of the normal forces in the three tests is 26.21 N. Concerning the lateral velocities, they are obtained from the robot kinematics and their mean square average is 0.0243 m/s. This speed is considerably high for the KUKA LWR to execute the grinding task on the wall. For this issue, the lateral velocity is scaled down to 0.015 m/s and consequently the force to 15 N by respecting the approximate proportional value ($f^{reac}/v_{norm} \approx 1000$). It is assumed here that the normal force is proportional to the translational velocity because of the collision between the disc and the wall. The grinding tool used in the experiment is made of an electric spindle rotating in a high speed with the abrasive disc attached to its end. The description of the grinding parameters that are identified with these experiments are detailed in Table 4.5.

The disc used in the tests has the outer radius $R = 0.0625$ m and the inner radius $r = 0.0575$ m. The spindle rotates 11000 rounds per min (rpm), this is equivalent to $\omega = 1.1519 \times 10^3$ rad/s. F^{reac} and v_{norm} can be obtained from the

Table 4.5. Parameters of the grinding model.

Variable	Units	Explanation
μ_e	(W/s)/m^3	Energy gain
ω	rad/s	Rotational speed of the disc
f^{reac}	N	Applied force
t	m	Thickness of the cut ahead
R, r	m	Contact radii on the disc
d	m	Depth of the cut
b	m	Width of the cut
v_{norm}	m/s	Translational velocity of the disc on the wall
v_{tan}	m/s	Tangential velocity of the disc
μ_f		Friction constant

force-torque sensor and the robot kinematics respectively. In order to determine μ_e, μ_f and d, two automatic grinding tests were performed using the setup in Figure 4.129. Similar to the manual setup, the wall used is planar and its frame $\{w\}$ is set parallel to the frame $\{o\}$. Also, the orientation of the end-effector is fixed such that the disc is parallel to the wall. Then, the robot is commanded using hybrid control in the operational frame $\{o\}$ to execute normal grinding force f^{reac} along $^o x$ and translational velocity v_{norm} on the wall along $^o y$. The values of f^{reac} and v_{norm} are the ones deduced previously from the manual grinding tests as 15 N and 0.015 m/s respectively. The lateral and normal forces (f^{reac} and $f^{lateral}$) are recorded from the force-torque sensor measurements, the value of v_{norm} is obtained from the robot kinematics and the depth of the cut d is measured using a coordinate measurement machine and found to be 0.002 m. The surface's profile of the automatic grinding tests is evaluated using the profilometer (as shown in Figure 4.129), their average roughness is 16.7 μm and they were smooth enough by tactile sensing as well. Accordingly, the values of μ_e and μ_f are obtained by linear least square method.

Non-contact forces elimination from the force-torque sensor readings

Non-contact forces compensation in force-torque sensor readings is important in the grinding task. It is necessary to decouple the forces caused by the weight, inertia, Coriolis and centrifugal effects of the grinding tool from the pure contact forces with the environment. The force-torque sensor provides the readings of wrench W acting on the sensor frame $\{s\}$ with respect to the frame itself. This wrench not only contains the pure contact wrench $^s W_c$ but also the previously described

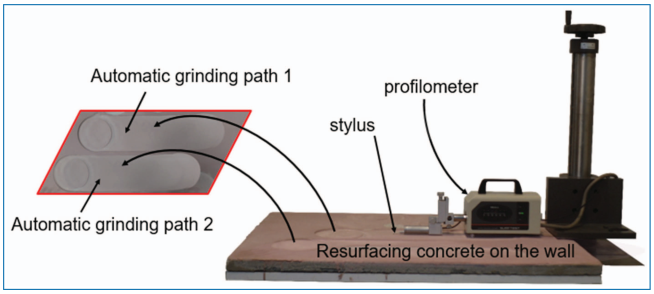

Automatic grinding path 1
profilometer
stylus
Automatic grinding path 2
Resurfacing concrete on the wall

Figure 4.129. Surface roughness measurement setup.

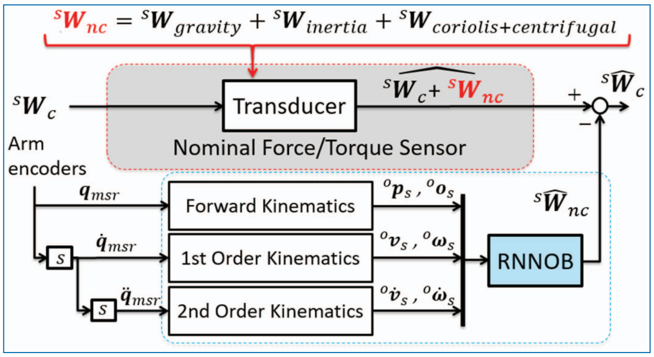

$^{s}W_{nc} = {}^{s}W_{gravity} + {}^{s}W_{inertia} + {}^{s}W_{coriolis+centrifugal}$
$^{s}W_c$
Transducer
$\widehat{^{s}W_c + {}^{s}W_{nc}}$
$^{s}\widehat{W}_c$
Nominal Force/Torque Sensor
Arm encoders
q_{msr}
$\dot{q}_{msr}$
$\ddot{q}_{msr}$
s
Forward Kinematics
1st Order Kinematics
2nd Order Kinematics
$^{o}p_s, {}^{o}o_s$
$^{o}v_s, {}^{o}\omega_s$
$^{o}\dot{v}_s, {}^{o}\dot{\omega}_s$
$^{s}\widehat{W}_{nc}$
RNNOB

Figure 4.130. Recurrent neural network observer to estimate non-contact forces.

non-contact wrenches $^{s}W_{nc}$ (gravity, inertia and Coriolis-centrifugal). The sensor is activated when the robot is not moving and the disc is facing down, that is, the sensor frame $\{s\}$ is parallel to the operational frame $\{o\}$. Thus, it is necessary to first estimate the non-contact forces and then subtract them from the force-torque sensor output in order to obtain the pure contact forces.

In order to estimate these non-contact forces ($^{s}W_{nc}$), a Recurrent Neural Network observer (RNNOB) is used since it outperforms analytical based methods. The recurrent neural network model is trained using the force-torque sensor pose ($^{s}p_o$, $^{s}o_o$), twist ($^{s}v_o$, $^{s}\omega_o$),and its accelerations ($^{s}v?_o$, $^{s}\omega?_o$), obtained from direct kinematics, to estimate $^{s}W_{nc}$. Fig. 5 depicts the block diagram of the RNNOB used here to estimate the contact forces.

The data was collected by *manual*, *automatic* and *rotational* trajectories with the grinding tool attached to the force-torque sensor as shown in Fig.3. The *manual* data was collected by setting the robot controller to gravity compensation mode and then moving the wrist manually to various poses in the workspace with random velocities and accelerations. The *automatic* data was generated by turning the

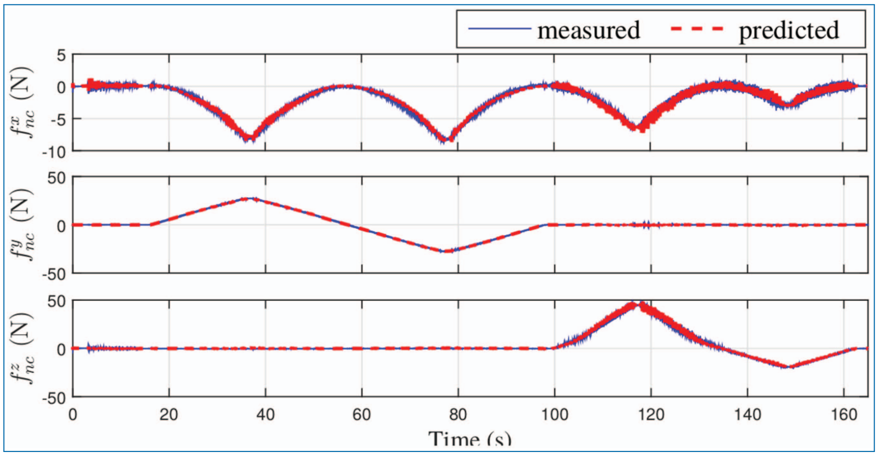


Figure 4.131. Non-contact forces (up) and torques (down) estimated by the RNNOB against the measured ones.

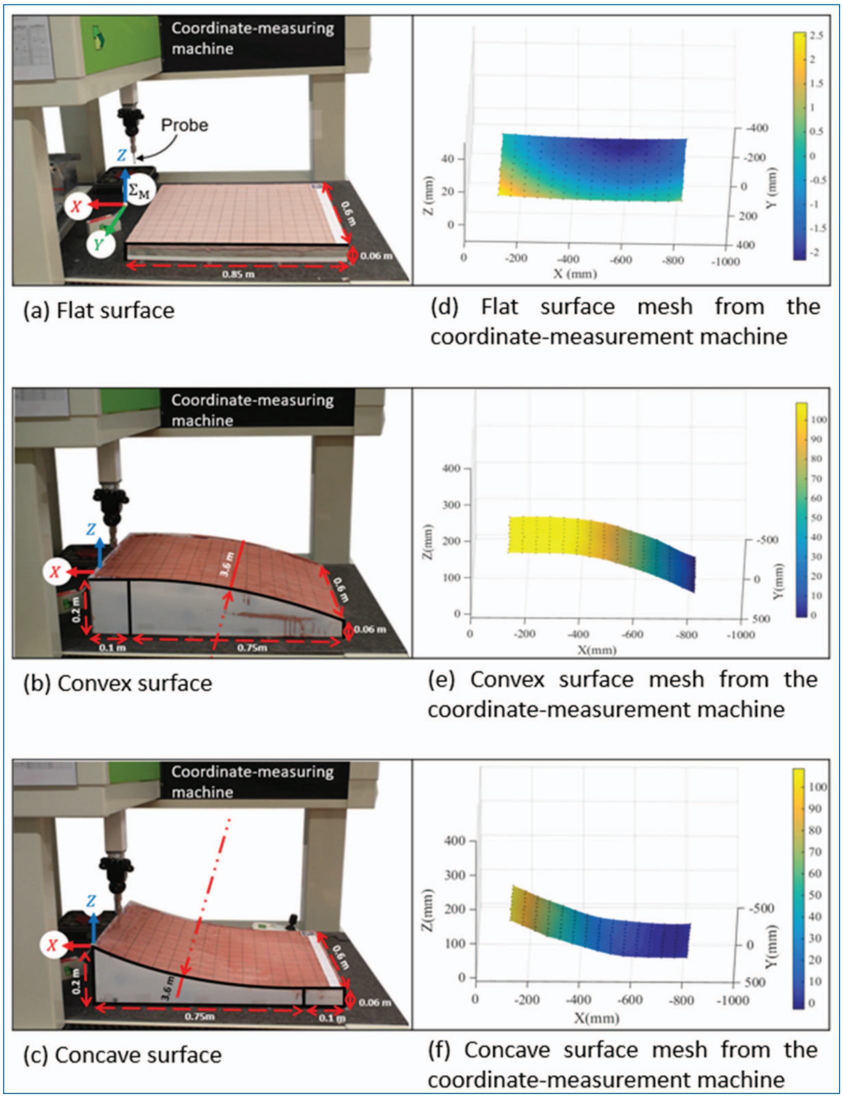
grinder on and moving the robot between random points in its workspace using various trapezoidal velocity profiles. The *rotational* trajectory was generated by rotating the sensor frame *{s}* and the compliant frame *{cv}* around each of their axis sequentially. The *manual* data was collected in 2 trials, each about three minutes long; similarly the *automatic* data was collected in 3 trials with an average time of 1.5 minutes each as well, and the *rotational* data was collected in two trials (in the frames *{s}* and *{cv}*) about 2 minutes long each. These datasets were then combined into one training dataset, where one trial of the automatic data and the rotational data around *{cv}* were separated to create a test dataset. The *manual* data was used entirely for training. The RNN model is trained for 20 epochs using an architecture with two hidden layers, with 15 and 10 Long Short-Term Memory (LSTM) units respectively, and a sequence length for the input layer of 15 time steps. Stochastic Gradient Descent was applied in the output layer with 0.01 learning rate to minimize the mean square error of the regression problem. A hyperbolic tangent sigmoid function was used as the activation function between the hidden layers and a linear activation function was used for the output layer. The efficiency of the RNNOB in estimating gravitational forces was evaluated by the rotational test around {cv} and is shown in Figure 4.131. Non-contact forces (up) and torques (down) estimated by the RNNOB against the measured ones.

Experimental evaluation

In order to validate the effectiveness of the proposed hybrid controller, wall grinding experiments are carried out on three different cases: (1) on flat surface; (2) on convex surface; (3) on concave surface. The curvature radius in the cases (2) and (3) is chosen such that the full contact between the disc and the surface is on its limits (e.g. if the radius of curvature is too big, the contact between the disc and the surface will be a disc, else the contact will be small segments at the contact points). The wall samples can be seen in Figure 4.132. In order to later evaluate the grinding quality, the initial state of the surface of each of the wall samples is evaluated using the coordinate-measuring machine (CMM). The CMM measures the surface geometry of the walls by sensing the discrete points of the mesh projected on their surface using the probe shown on the left side of Figure 4.132. Thus, the initial surface geometry of the walls is shown in Figure 4.132. The experimental grinding setup is shown in Figure 4.126. The KUKA lightweight arm is controlled by an external computer. The control framework is implemented on he computer using Robot Operating System (ROS) that communicates with the KUKA Fast Research interface (FRI) at a rate of 500 Hz. The six-axes ATI force-torque sensor is attached to the face of the six joint. Then, on the other end of the force-torque sensor, the grinding tool with a speed of 11000 rpm is attached. The tool rotates a disc with abrasive grains of 125 mm diameter. The overall end-effector (tool + disc + tool-frame) is around 7.3 Kg; its weight, inertia, Coriolis and centrifugal effects on the force-torque sensor readings are accurately estimated and compensated using the RNNOB previously presented. The force sensor publishes data at a rate of 1000 Hz and the RNNOB runs at 500 Hz. The force sensor output is filtered using a moving average window of 25 samples. The robot exerts on the wall the instantaneous normal force required for grinding. previously obtained and equal to 15 N. Along with exerting normal force on the wall, the disc moves on the wall with the identified desired velocity v norm = 0.015 m/s while it adapts automatically the surface based on the adaptive orientation controller.

In order to evaluate the grinding quality, the ground surface of the flat wall is evaluated in Figure 4.133. Figure 4.133a shows the superposition of the meshes measured before and after grinding. The two meshes are subtracted and the depth of the cut along the lines L1 and L2 (see Figure 4.133b) is plotted in Figure 4.133c. Figure 4.133c shows that the grinding depth varies between 1 and 2.67 mm along the ground path. The average depth of the cut under L1 and L2 is 1.79 and 1.86 mm respectively.

The grinding quality on the convex surface is evaluated as well in Figure 4.134. Figure 4.134a shows the superposition of the meshes measured before and after grinding. The depth of the cut can be obtained by calculating the Euclidean distances between the points of the meshes measured before and after grinding. Addition to that Figure 4.134c shows the depth of the cut evaluated under the curves C1 to C7 (see Figure 4.134b). The Figure shows that the grinding depth varies between 0 and 3.5 mm in the ground area. The average depth of the cut under each Curve Ci is also shown in Figure 4.134c and the overall average depth of the cut in the ground area is 1.64 mm.



(a) Flat surface

(d) Flat surface mesh from the coordinate-measurement machine

(b) Convex surface

(e) Convex surface mesh from the coordinate-measurement machine

(c) Concave surface

(f) Concave surface mesh from the coordinate-measurement machine

Figure 4.132. Different wall geometries. On the left side of the figure, the walls are placed on coordinate-measuring machine (CMM) to evaluate their surface before grinding. On the right side is the mesh obtained for each wall, it is measured in the machine frame.

Finally, the grinding quality on the concave surface is evaluated in Figure above. The superposition of the meshes before and after grinding is shown in Figure 4.135a. Similar to the previous case, the depth of the cut is obtained by calculating the Euclidean distances between the points of the meshes measured before and after grinding. Figure b shows the depth of the cut evaluated under the curves C2 to C5 (see Figure b). The Figure c shows that the grinding depth varies between 0 and 2.5 mm in the ground area. The average depth of the cut under each Curve

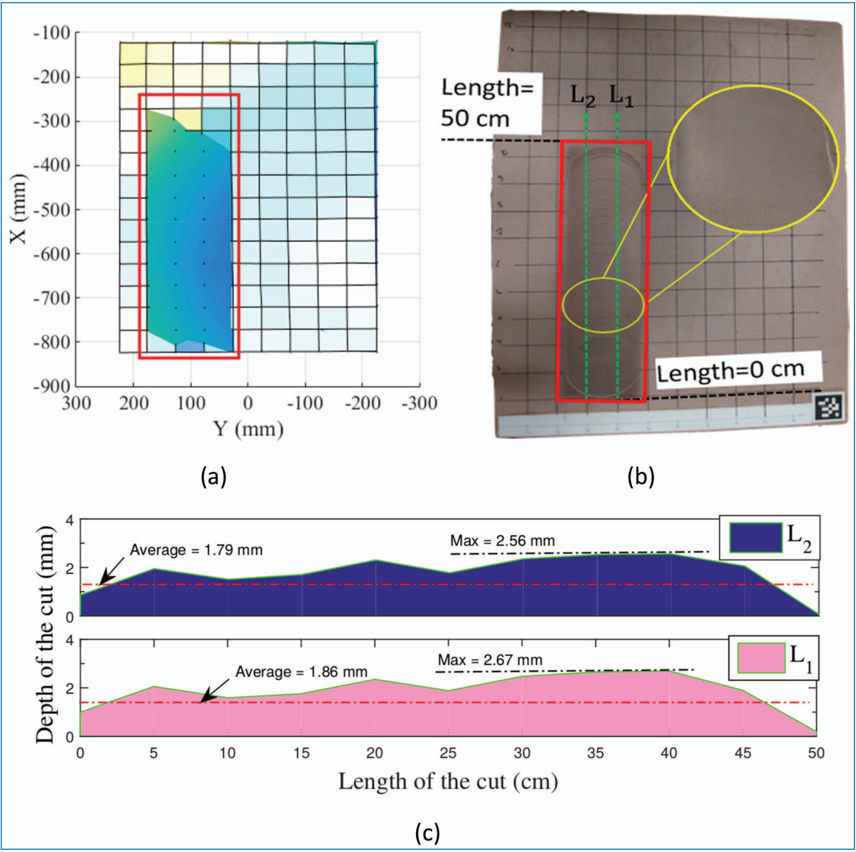


Figure 4.133. The surface state of the flat wall evaluated before and after grinding: (a) Top view of the superposed meshes describing the flat wall surface before and after grinding, (b) Flat wall after grinding, the depth of the cut is evaluated for 50cm under two lines (L1 and L2). (c) Grinding path depth evaluated under lines L1 and L2.

Ci is also shown in Figure 4.135c and the overall average depth of the cut in the ground area is 1.43 mm.

To sum up, although the geometries of the three surfaces were not known before hand, the grinding task was achieved with the disc exerting a normal force of 15 N on the wall surface while tracking its curvature. The experimental results showed good velocity, force and orientation tracking performances. Concerning the grinding quality, it was evaluated for the three test cases and good grinding was shown. The depth of the grinding cut in the flat wall case is around 1.8 mm (Figure 4.133c). For the convex wall case, the depth of the cut varies between 0 and 3.5mm (Figure 4.134c). And for the concave wall case the depth of the cut is around 2 mm (Figure 4.135c). The main reason why the depth of the cut in the tests is not constant is that it is not controlled directly, i.e., the depth of the cut should be around 2mm (the depth of the automatic grinding tests used for identifying the nominal

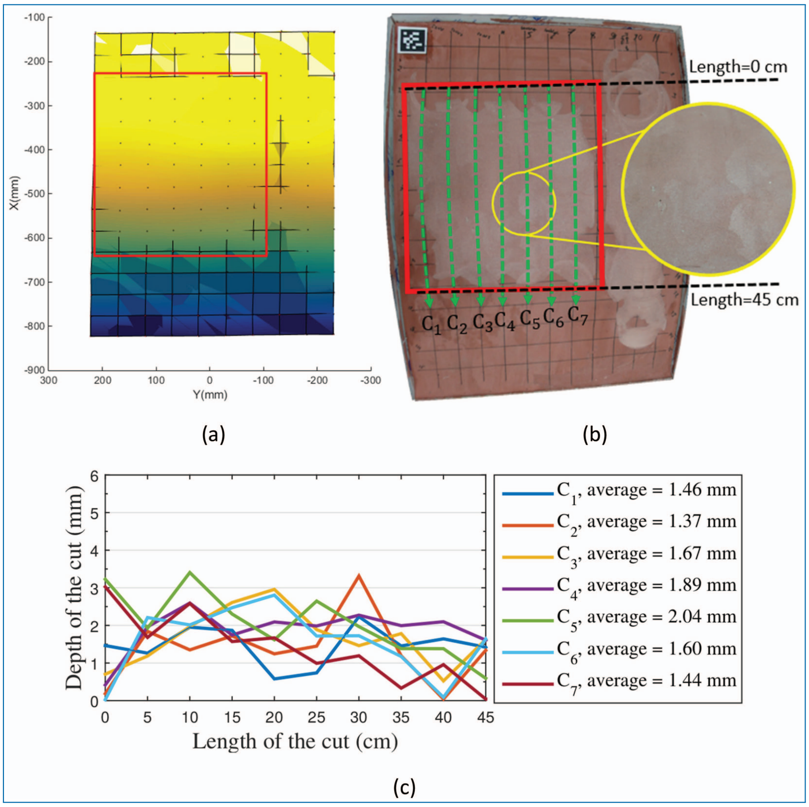


Figure 4.134. The surface state of the convex wall evaluated before and after grinding: (a) Top view of the superposed meshes describing the convex wall surface before and after grinding, (b) Convex wall after grinding, the depth of the cut is evaluated in a area with length of 45cm and under 7 curves (from C1 and C7). (c) Grinding path depth evaluated under the curves C1 to C7.

grinding lateral force) if the robot executes the identified normal force and lateral velocity on the wall. Addition to that, the walls surface was not smooth before grinding (it has some bumps). Moreover, the lateral grinding force model used is valid when the disc is in full contact with the wall. In case the radius of the curvature of the surface to be ground is small, the disc will not be in full contact with the wall, and then the lateral force model will have some error which in turn will lead to some variations in the depth of the cut.

Grinding test campaign

Because of lack of such studies and data in the public sphere a test campaign of thirty grinding tests was made in controlled conditions. The main goal is to understand with a first global practical vision the consequences of this construction materials machining operation on a robotic arm [1].

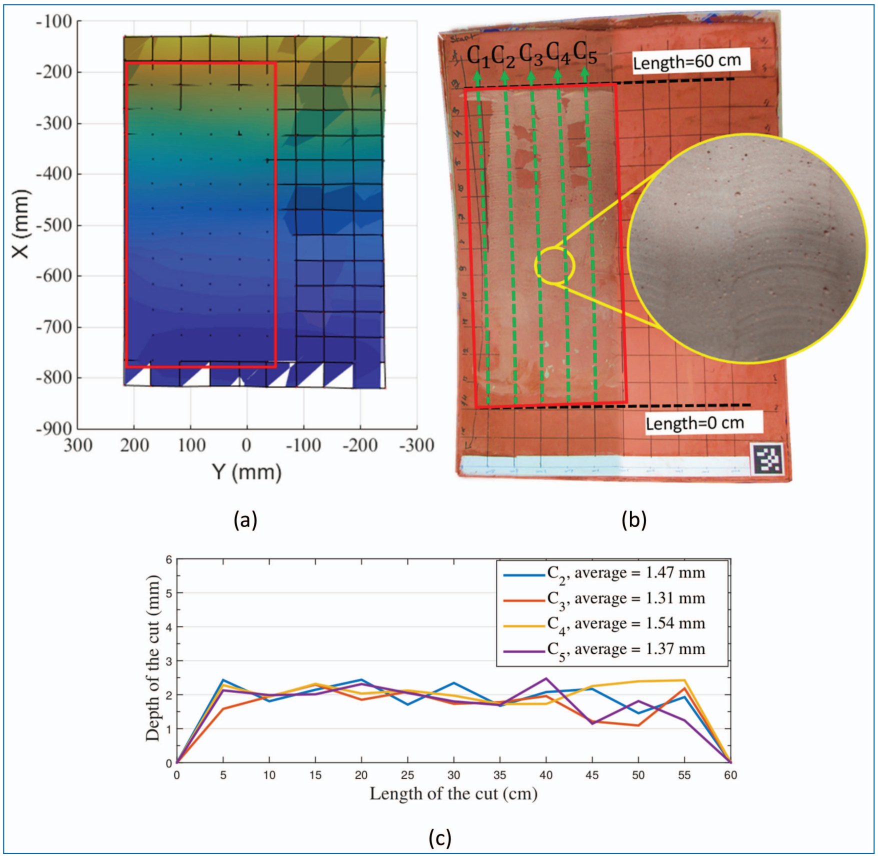


Figure 4.135. The surface state of the concave wall evaluated before and after grinding: (a) Top view of the superposed meshes describing the concave wall surface before and after grinding, (b) Concave wall after grinding, the depth of the cut is evaluated in a area with length of 60cm and under 5 curves (from C1 and C5). (c) Grinding path depth evaluated under the curves C1 to C5.

Tests care about the removal of a material layer considered as “contaminated by asbestos” from a material support with a grinding operation. This last disintegrates in dust the concerned layer by abrasion mechanism. It separates definitively the two materials sealed together. A cell test was prepared and a protocol established in order to record physical parameters and output physical variables issued of the tests done. A major part of the test set up and their course are explained in the proceeding paper published for IFToMM 2019 [1].

Parameters like feed speed, depth of cut, grinding wheel rotation speed and material machined were considered as of these constant and imposed. Physical output variables took care were in a first time the grinding forces and torques and in a second time the electrical power absorbed by the grinder.

Test made are:

- Tests I
  - Transverse grinding (figure a)
  - Constant feed-speed
  - Full resurfacing concrete layer machined
  - 9 tests were done
- Tests II
  - Transverse grinding (figure a)
  - Constant feed-speed
  - Full plaster layer machined
  - 6 tests were done
- Tests III
  - Transverse grinding (figure a)
  - Constant feed-speed
  - Intermediary layer (including resurfacing concrete, raw concrete and their border surface) machined
  - 7 tests were done
- Tests IV
  - Plunge grinding and transverse grinding (figure b)
  - Non-constant feed speed (acceleration after plunging)
  - Full resurfacing concrete layer machined
  - 4 tests were done
- Tests V
  - Plunge grinding and transverse grinding (figure b)
  - Non-constant feed speed
  - Full plaster layer machined
  - 4 tests were done

All needed parameters and all physical variables output for tests I and II were correctly recorded.

Considerations

- Two main construction material were studied: one "soft" (plaster) and one "hard" (resurfacing concrete).
- Material mechanical characteristics are unknown in the literature.
- Particular cases of construction material grinding are analyzed: recent materials without asbestos with adapted but particular tool. None are presented in literature.

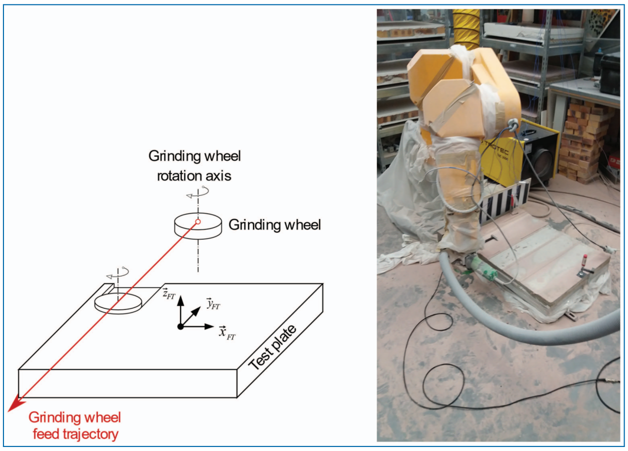


Figure 4.136. Transverse grinding operation scheme and Staübli RX130b robotic arm equipped with grinder set up in initial configuration to grind for vibration test picture.

- Measurement and machining devices used for the tests are the ones that where available at Sigma Clermont.

Data exploitation limitations

For tests IV and tests V (figure b) It was not possible to record easily the robotic arm position in space and time for tests due to the old communication interface of this machine. So, speed and accurate robotic arm segment positions are missing and not rebuildable from others data in our study. In this case it would not wise to exploit other data with this lack to analyse the tests IV and V data.

Another limitation appeared during the tests progress: Lack of knowledge about the real machined material layer. The layer has a many form errors due to material support layer form errors and due to its building technics used. As consequences the 3D volume of the studied layers are not perfect parallelepipeds. They have random forms errors making it difficult to characterize their real geometry and attribute them a precise geometric marker which locate them accurately in space. When the grinding operation only machine the contaminated material layer, and not the support material, no problem appears (figure c). However, the thickness of the contaminated layer has to be pretty significant in relations to the dimensions of the grinding tool. Otherwise a layer of the contaminated material plus a layer of the support material (interface materials layer) have to be removed (figure d).

In this case, we did not know the geometric characteristics of the frontier surface between both materials, in order to remove all the contaminated layer and the minimum volume of the support layer.

In fact, it necessary to remove the minimum of the support material to prevent from mechanical weakening of the whole concerned structured including this

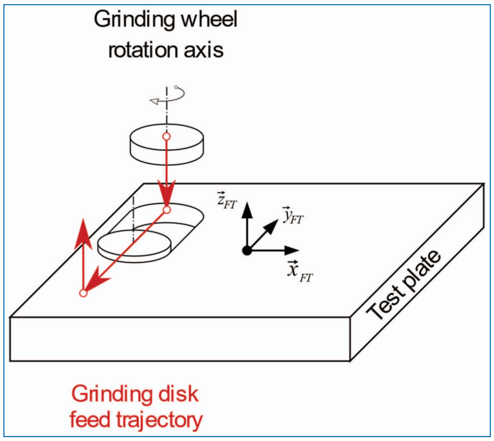


Figure 4.137. Plunge + transverse grinding operations scheme.

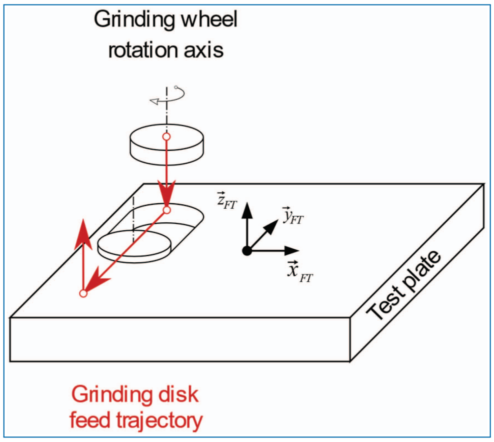


Figure 4.138. Grinding of contaminated layer.

material. At this point a new problematic appears: how can we determine the frontier surface? State of art seems leads at the uses of scanning devices designed for inspection of material layers after their construction in the building area. It would permit, without damaging the construction, to rebuild the geometry of the contaminated material layer about its material support.

In the grinding tests several successive transverse grinding operations were done until a support material surface appears free of every traces of contaminated material. But in these cases, the proportion of each machined material were unknown and finally the output physical variables could not be attributed correctly to define the machining operations. So, data of test III are not analyzable. Finally, the principal work of data treatment and analyses is concentrated in the tests I and II (Figure I).

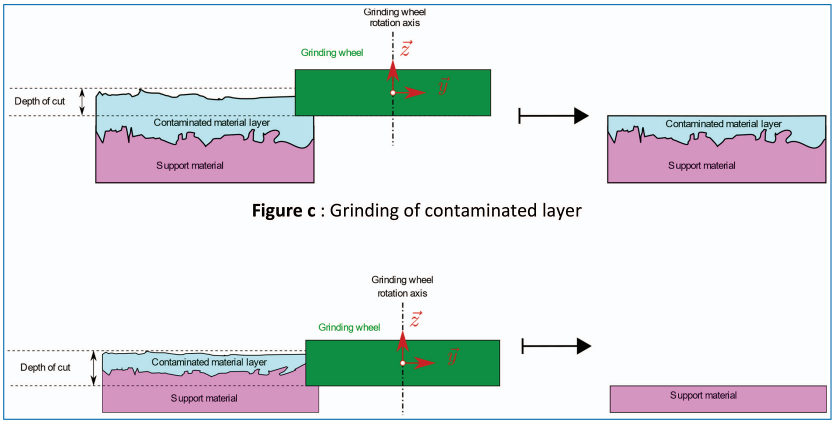

Grinding wheel rotation axis
Grinding wheel
$\vec{z}$
$\vec{y}$
Depth of cut
Contaminated material layer
Support material
Contaminated material layer
Support material
Figure c : Grinding of contaminated layer
Grinding wheel rotation axis
Grinding wheel
$\vec{z}$
$\vec{y}$
Depth of cut
Contaminated material layer
Support material
Support material

Figure 4.139. Grinding of layer of interface layer (contaminated material layer + support material layer).

Technical observations during tests

- Dust and chip production are very important and may participate in significant way to increase at the consumption electrical power when they are evacuated by the aspiration device because of rubbing phenomenon (with the grinding wheel and the dust protection cover).
- Dust protection cover does not adapt its orientation with efficiency to the surface of the layer machine when the grinder is attached to a robotic arm. The cover is too stiff and apparently lacks two rotations motions with respect to the grinder head for perfectly surrounding the grinding disk and creating an efficient depression chamber on the surface. The aspiration of the construction materials dust and chips (Numatic HZD900 [2], Power: 900W, Double filters of class H, different device than the proto V1) were not efficient enough and that is why a great) quantity of dust was disseminated in the whole test cell.
- A strong filtering device must be installed to remove the very fine particles from the cell test air and protect the operator's lungs.
- Moreover, each time dust protection cover had touched the grinding wheel because of an excessive deformation, the grinding operation was stopped.
- Mechanical vibrations, created during the grinding operations and propagated in all the test cell, felt by operators are very important in frequency and amplitude.
- The grinding wheel was damaged when grinding the steel armatures embedded into the concrete support. After this, an important decrease of its cutting performances was noticed although no observable damages were observed on the grinding wheel. We presume that a major metallurgy change in the bond material of the grinding wheel has happened.

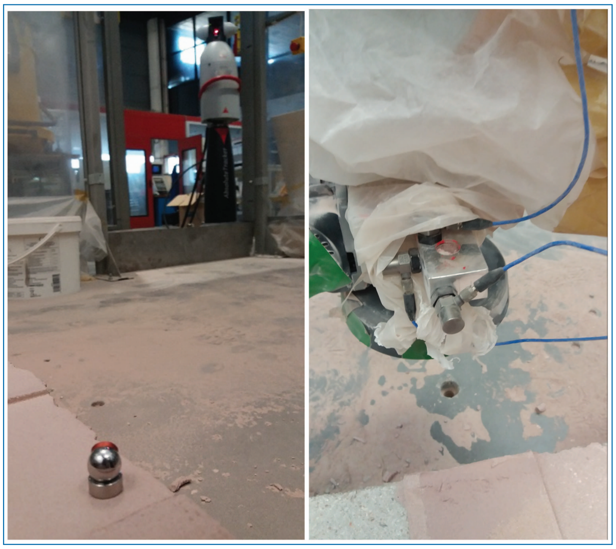


Figure 4.140. Measurement with laser tracker of one surface point from resurfacing concrete layer, Accelerometers glued on the terminal organ of the robot along its tree principal axis.

- The frontier surface position and 3D geometry between contaminated material layer and support material are unknown in spite of the care taken in the production of the test plates.
- A pinch angle of about one degree was observed between the robotic terminal segment and the grinding wheel due to the assembly of the grinder on the robotic arm.
- The assembly between the grinding wheel and the grinder spindle was not adapted for an automatic use. The constraints generated during the tests lead to a more important tightening constraint than manual use. In consequence, the grinder had to be dismounted from the robotic arm in order to change, with a lot of difficulties, the grinding wheels from the grinder.

Data treatment

Test I and II have shown that two parameters considered as constant were not: depth of cut (DOC) and rotation speed of the grinding wheel. For each studied test, they were computed with indirect data.

First, the lack of stiffness of the robotic arm coupled of the lack of relative position data between the terminal organ of the robot and the contaminated layer, completed by the lack of position loop control use for the robot, lead to a non-constant thickness of the material layer removed.

Before and after each grinding test, about twenty points of the considered layer surface were measured with a laser tracker. With an interpolation method, the

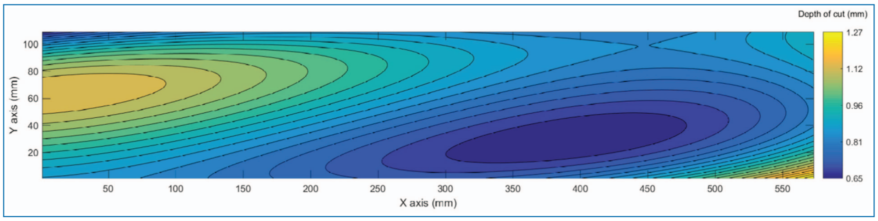


Figure 4.141. "Real DoC" computed along the tool trajectory for the RC grinding test with" imposed DoC (1 mm)" and feed-speed of 31.25 mm/s.

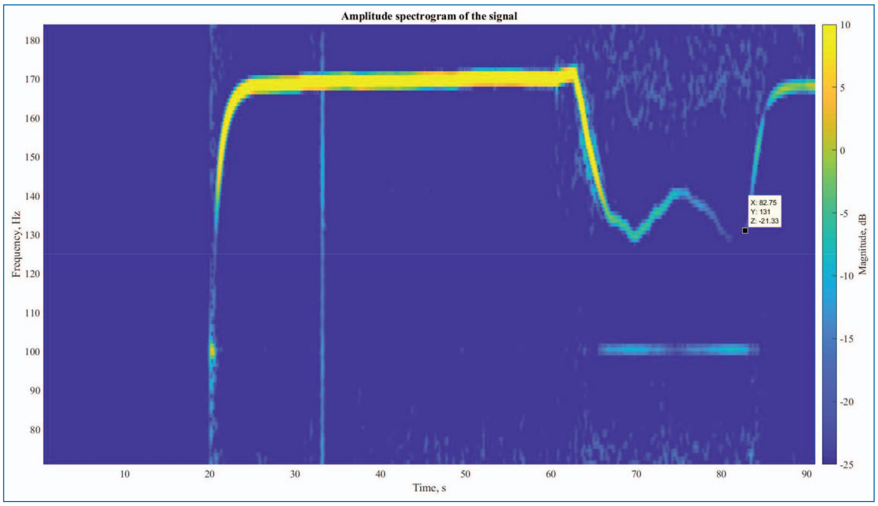


Figure 4.142. Amplitude spectrogram of the mechanical acceleration data of a grinding test, the no deep blue curve corresponds to the rotation frequency of the grinding tool during the test.

surface of the test plate was rebuilt before and after the grinding test for finally computing the thickness (or DOC) of the removed layer (figure f).

Due to an absence of speed rotation loop control implemented physically in the grinder, the grinding wheel rotation speed was not constant for each grinding test. After discussion with vibration tool specialist Dr. Vincent Gagnol, we have considered that the fundamental frequency measured by the accelerometers in the robot terminal organ was corresponding to the frequency of the grinding wheel rotation. Initially the accelerometers had to be used to characterize the vibration phenomena of the test assembly. Unfortunately, the used grinder had very poor assembly clearance and the setup of the vibration measurement chain was not precise enough (measured frequency range: 0-10 kHz), which made this complex study unfeasible. But the acceleration data gave access to the grinding wheel speed rotation with a Short Time Fourier Transformation of the measured raw acceleration data (figure g).

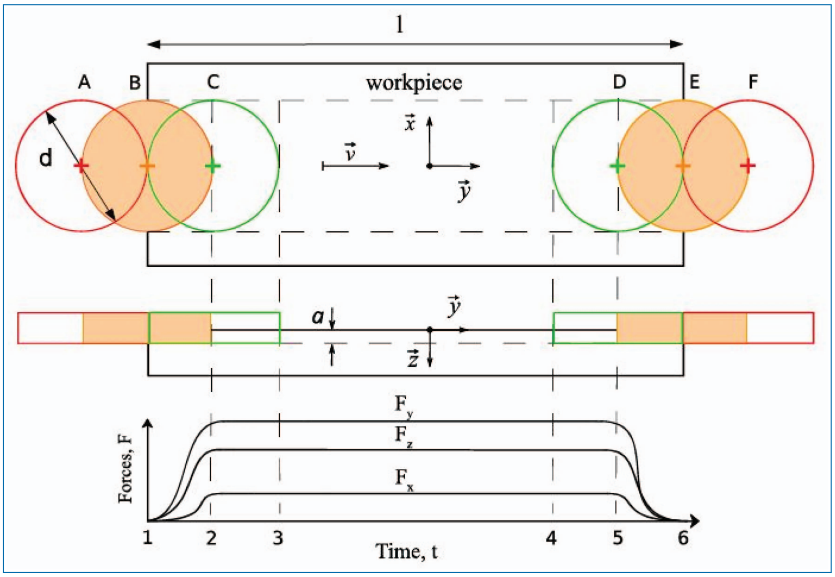


Figure 4.143. Transient cut-in and cut-out stages for vertical spindle surfacing during a Transverse Grinding Operation (TGO).

As the different measures had different temporal bases, they required a treatment for proper synchronization. Grinding torques were computed at the intersection of the grinding wheel axis and the machined material layer surface thanks to the torques measure by the force table in its reference marker. However, it appears that the grinding torques strongly depend on the arm pose, that was not recorded in these tests. So the recorded grinding torques were not analyzed because of the lacking data.

For each test of category I and II, the grinding forces, the electrical power absorbed by the grinder, the material flow rate and the grinder spindle rotation rate were computed from the raw measurements. The diagram made between both material grinding shown different behaviours for the grinding operation. Figure i reminds the different cut-in and cut-out phases of the operation.

Figure i, j, k present the curves parameters and the force grinding measured for a transverse grinding operation. Full resurfacing concrete layer is machined in this test with imposed parameters like DOF of 1 mm and feed speed (vt = 29.5 mm/s).

It makes sense to analyze the curves when the grinding disk is completely inside the test plate limit (figure h, configuration C) and omit the data when it is not. So, in the figures i, j, k, only the curves from 4,4s to 20,5s (stage from C limit to D limit or from time 3 to time 4 in the time graph of figure h).

The constant parameters normally expected for Material Flow Rate (MFR = Depth Of Cut/t in mm^3/ms) and Grinder Spindle Rotation Rate (rad/s) are

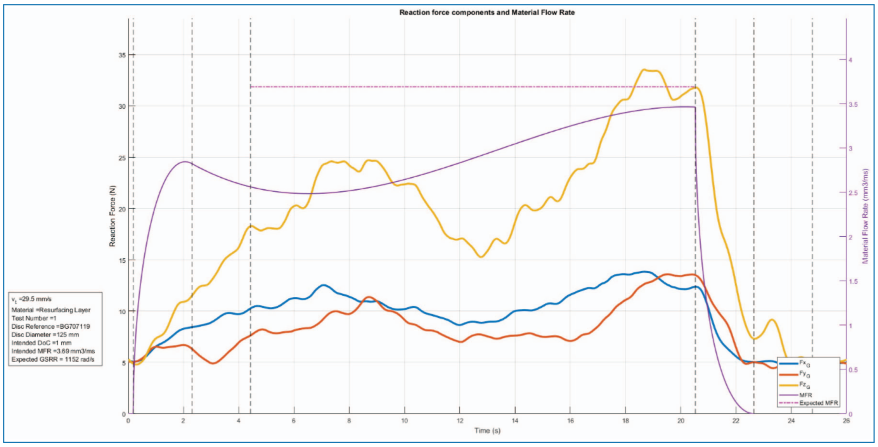


Figure 4.144. Grinding force and material flow rate for transverse grinding operation of resurfacing concrete layer. DOC targeted = 1 mm and feed speed targeted vt= 29.5 mm/s.

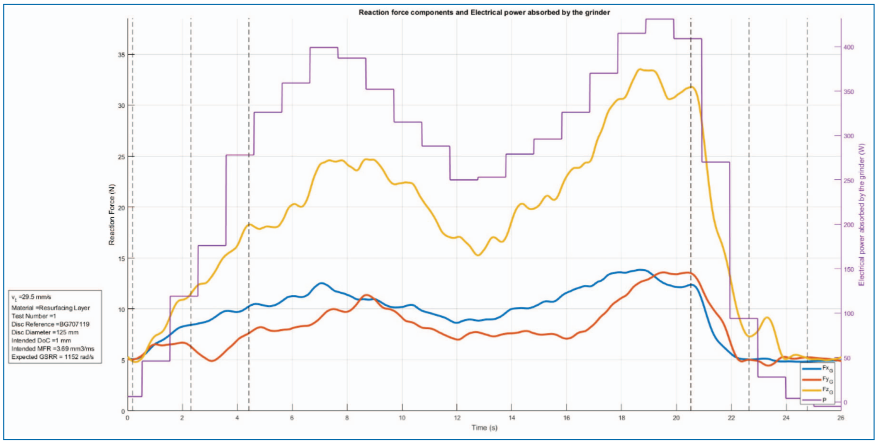


Figure 4.145. Grinding force and absorbed electrical power by the grinder for transverse grinding operation of resurfacing concrete layer. DOC targeted = 1 mm and feed speed targeted vt= 29.5 mm/s.

respectively displayed in figure j and k. As explained above and as shown in these last figures both of these parameters are non-constant and non-linear. The MFR increases from 2,5 to 3,5 mm^3/ms. The GSRR varies from 800 to 900 rad/s.

A general trend appears about the grinding force behavior. For the full resurfacing concrete material grinding tests the normal grinding force (Fzg) (Figure 4.144, 4.145, 4.146), dominates the tangential grinding forces (Fxg and Fyg). This trend is exactly inverse for plaster material (Figures 4.147, 4.148, 4.149) where the Fyg force against the toot advancement dominates. At the considered slow feed speed of 29.5 mm/s, the normal force Fzg is always under 35N for resurfacing concrete,

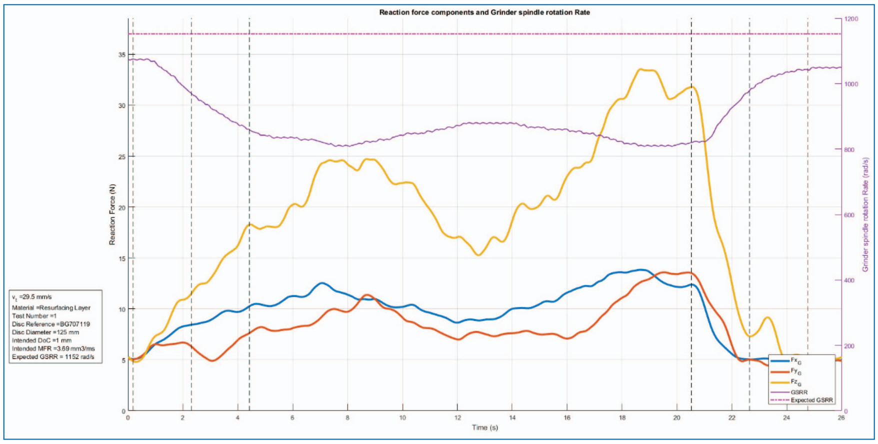


Figure 4.146. Grinding force and grinder spindle rotation rate for transverse grinding operation of resurfacing concrete layer. DOC targeted = 1 mm and feed speed targeted vt= 29.5 mm/s.

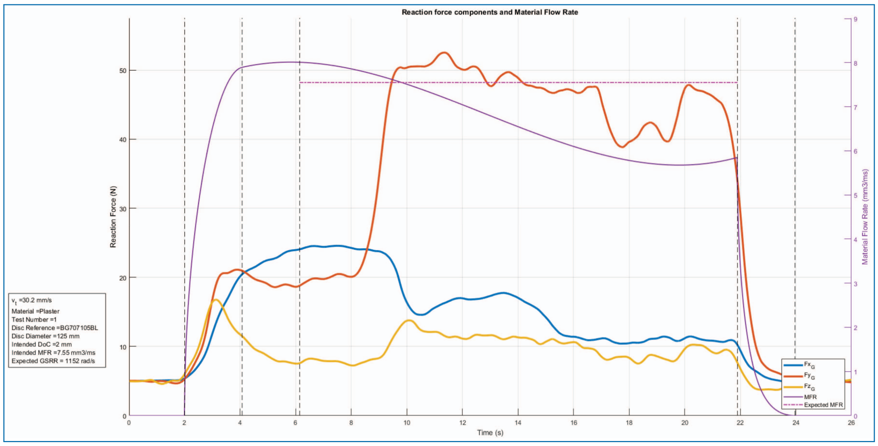


Figure 4.147. Grinding force and material flow rate for transverse grinding operation of plaster layer. DOC targeted = 1 mm and feed speed targeted vt= 29.5 mm/s.

whereas Fxg and Fyg are below 15N. For plaster and at the same feed speed, Fyg never exceeds 50N, Fxg is under 25N and Fzg under 15N. Obviously, these forces can vary a lot according to machining conditions, specifically feeding speed, that was limited on our robot. But these orders of magnitude remain interesting.

The electrical power consumed by the grinder follows the trend of the grinding normal force (Fzg) in figures j and m.

For only Fzg, a behavior asymmetry in time is observed and which find in the electrical power absorbed by the grinder. This asymmetry does not exist for the MFR (following an increase trend) and GSRR (symmetric behavior).

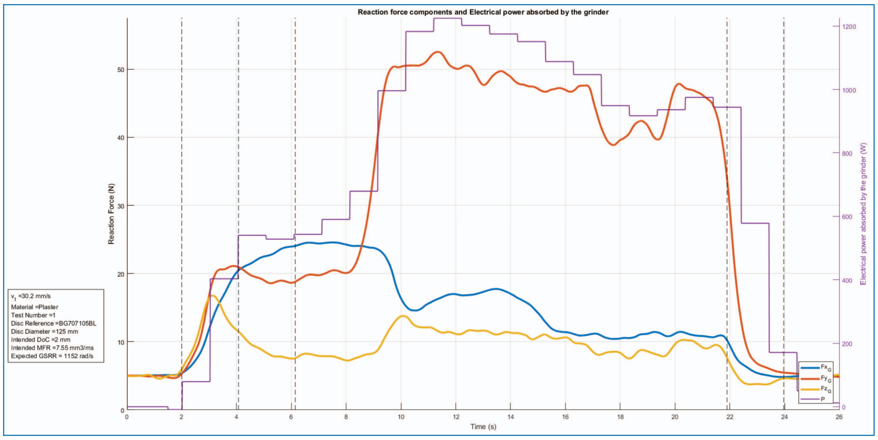


Figure 4.148. Grinding force and absorbed electrical power by the grinder for transverse grinding operation of plaster layer. DOC targeted = 1 mm and feed speed targeted vt= 29.5 mm/s.

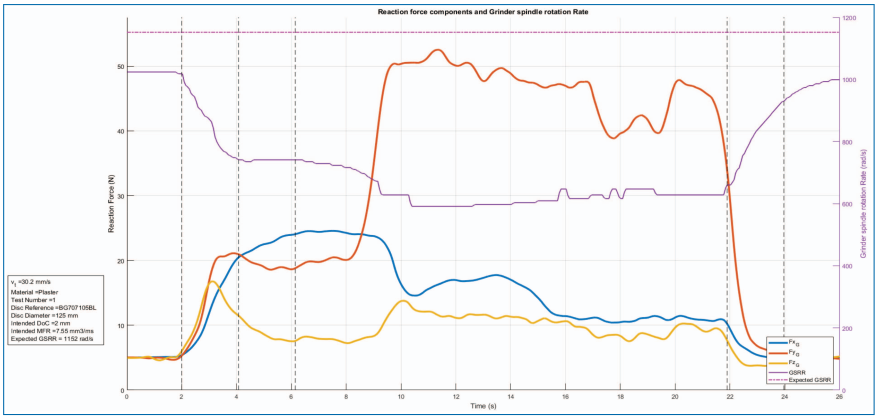


Figure 4.149. Grinding force and grinder spindle rotation rate for transverse grinding operation of plaster layer. DOC targeted = 1 mm and feed speed targeted vt= 29.5 mm/s.

The non-constant MFR shows non-constant thickness of the removed layer. We can assume the grinding torque asked on the grinding wheel by the machining operation is higher at the end of the machining operation due to the increase of material volume to remove and the GSSR low level (not controlled) for this test.

This example of test shows it is difficult to characterize the connection between the input parameters and the grinding force if the parameters are non-constant. According to the material, the dominating force components are not the same: the normal force Fgz dominates for resurfacing concrete, whereas the feeding force Fgy dominates for plaster.

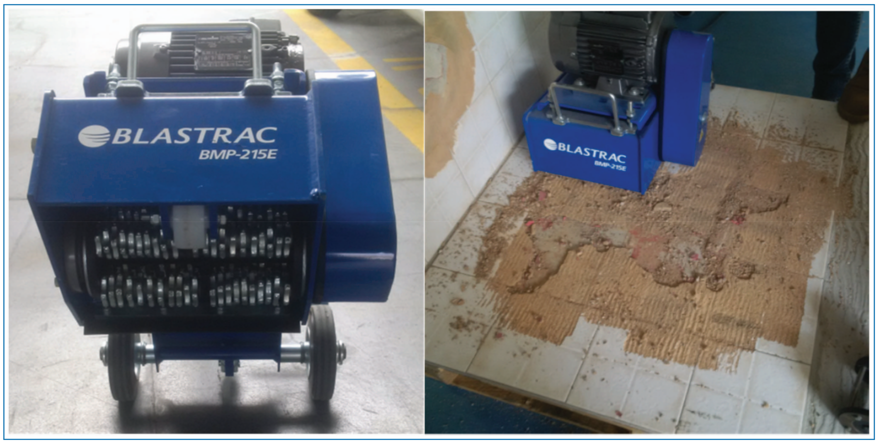


Figure 4.150. Preliminary test on standard machine.

Two points are highlighted by our tests:

Firstly, to master efficiently the grinding force, the parameters (GSSR and DOC) have to be controlled. This is not simple and has to be considered for future work.

Secondly, a grinding force model (in relation with the parameters observed above) can be extracted with a realization of the experimental protocol designed here and following the different recommendations done above.

The model will be valid for one specific couple tool-material and cannot be general. However, it will give knowledge about what can happen to our prototype during grinding operation.

The model will require a non-linear regression with multiple parameters, based on our experimental results.

Tiles tool development and testing

A specific tool for tiles removal was developed and tested in parallel to main grinding tool integrated with the arm. Not to overlap with other ongoing testing activities on the integrated robotic system the tool was tested and interaction force evaluated on a independent simulated mock-up. The following section describes the solution developed for tiles removal and collect results of the different tests performed with the tool in the stand alone configuration.

The first goal of this activity was to develop and test a scaled model of a scarifier tool, broadly used in surface preparation on concrete. As first investigation a standard Blastrac 215 was tested on a tiles mock up.

Test performed with the support of Blastrac Italy shown the potential of the scarifier tool to beak tiles in a controlled manner on small collectable debris. As

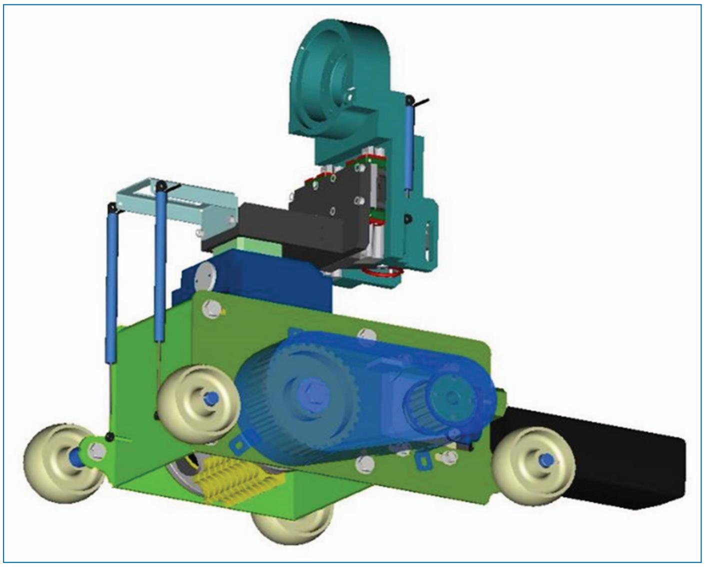


Figure 4.151. 100 mm drum scarifier machine.

in previous similar cases TLABS developed a "scaled" version of the Blastrac tool using Blastract standard consumables.

The first version of this tool shown in the CAD view above was driven with a WurthEWS14-125 1400W unit. Initial trials performed with this motor shown the potential performance of the tool but the lack of control of a Wurth motor limited significantly overall tool reliability. At the same time the motor unit, forseen for manual operation, was not enough robust for continuous automated operation.

The tool was hence updated using the same driver/motor solution adopted for V2 robot arm: a Merkes MN3-350-30-320 3Nm@3000 rpm (920 W) AC brushless motor controlled in torque look with an Infranor Ethercat PAC-ET-230/11 driver. With this configuration, both torque and speed control are possible allowing full control of the tool performance. The following picture shows a view of the tool integrated with the final motor and some "traces" performed as first qualitative trials.

A second set of tests was then performed to define the interaction forces of the tool and the required torque at motor. Tool was positioned on a dedicated horizontal test surface with a lower layer of concrete and an upper layer of tiles. The tool own weight in test configuration is about 15 kg. Test were performed with different ballast on the tool and rotating speed to identify the minimum contact force required to keep stable the tool on the surface in working conditions. The following pictures shows the tool with 25,15 and 5 kg of added ballast (total mass is reported in each picture).

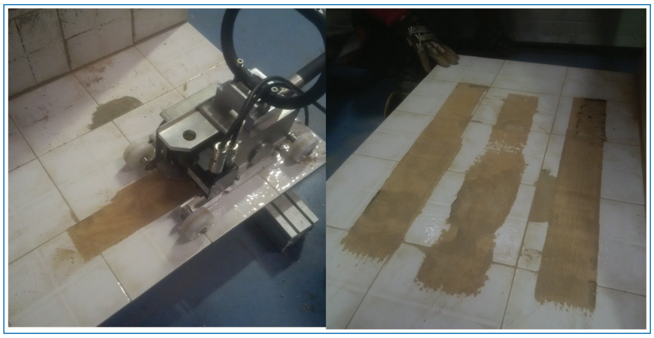


Figure 4.152. Tool in final configuration and qualitative tests.

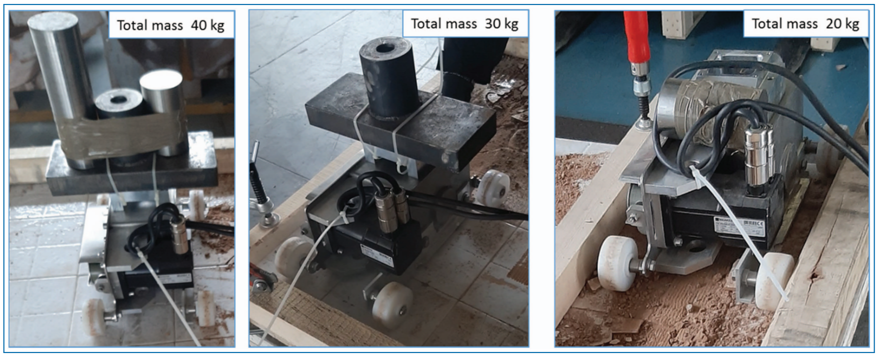


Figure 4.153. Different ballast configuration.

The following plots show the torque reading at tool motor shaft in different configuration tested:

The first two plot are torque reading for test performed with 15 kg ballast and motor set at 3000 and 2000 rpm. As expected torque increase with rotating speed as the energy fed to the tool increases.

The second plot compares torque reading at the same tool speed with different ballast mass (5kg vs 15 kg).

In this case, the two plots have roughly the same maximum torque values so ballast mass increase over the minimum set value of 5 kg has no impact on tool performance.

Tool developed for tiles removal as scaled version of a standard scarifier tool was tested in stand alone configuration with positive results. Motor and belt transmission sizing are in line with measured tool process torque (Merkes motor nominal torque is 3 Nm with peak torque of 10.5 Nm). Tool required contact force with the wall is estimated about 200 N where tool feed force is approximately 20:50 N.

The following last two pictures shows tool/robot flange forces as reference values for robot payload sizing. As discussed with tool laying on the floor an extra force of

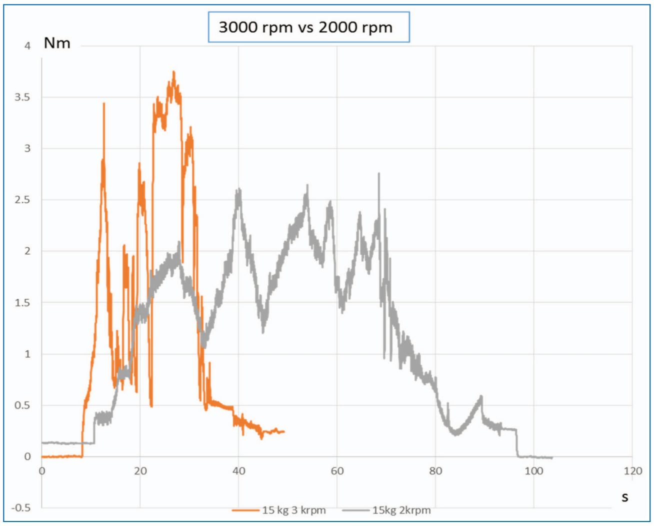


Figure 4.154. Tool speed test.

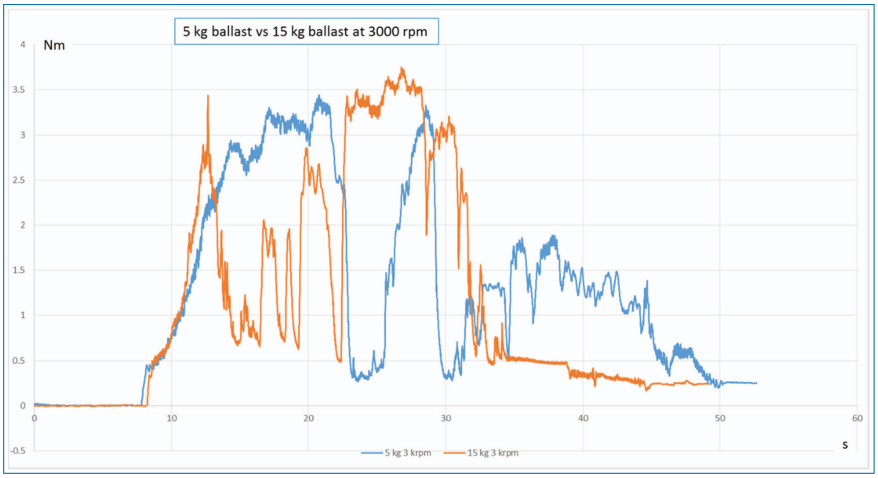


Figure 4.155. Tool ballast test.

50 N is recommended to reach a total contact force (dead weight+ added force) of 200 N. Tool feed force is 20:50 N typical. Total robot arm required process force is between 60 and 80 N while 150 N of lifting capability are required to handle the tool.

The same consideration for tool working on vertical wall gives different results: The robot arm must in this case support tool dead weight and push with the tool on the wall for 200 N. Robot payload in this case is in the range of 250:300 N. Version 2 prototype arm is not compatible with vertical wall tool processing but could be efficiently used on floor tiles removal.

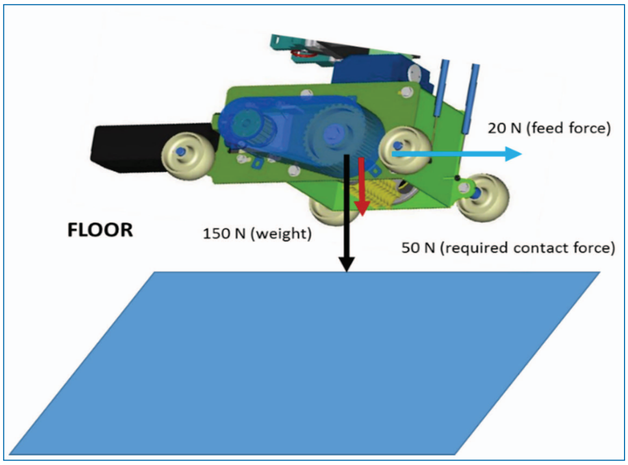


Figure 4.156. Vertical wall tool forces.

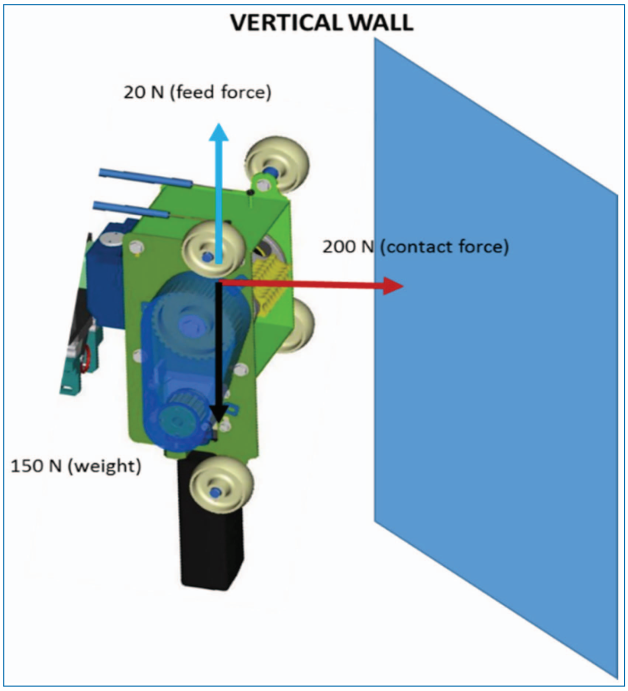


Figure 4.157. Floor tool forces.

Updates on grinding tool design

Grinding tool integration on the robotic arm was futher reviewed in the last period.

The first two prototypes built were both based on a sensorized fully compliant interface between the tool and the robot. The following picture took from the first periodic report shows the original concept.

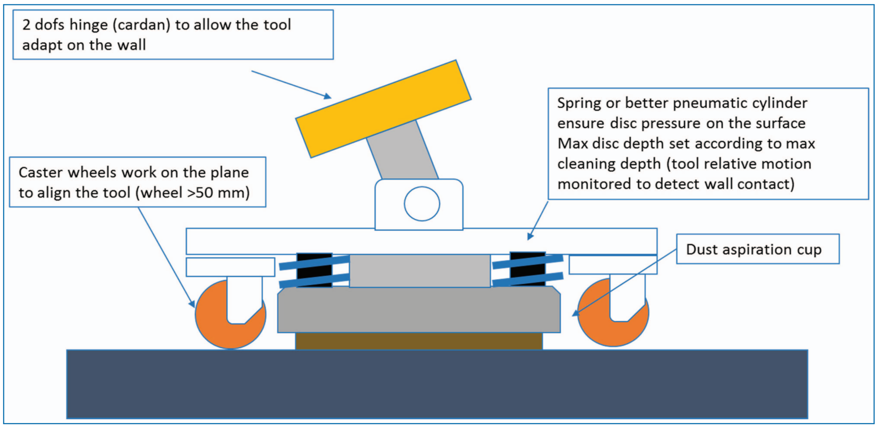

2 dofs hinge (cardan) to allow the tool adapt on the wall
Spring or better pneumatic cylinder ensure disc pressure on the surface Max disc depth set according to max cleaning depth (tool relative motion monitored to detect wall contact)
Caster wheels work on the plane to align the tool (wheel >50 mm)
Dust aspiration cup

Figure 4.158. Original grinding tool concept.

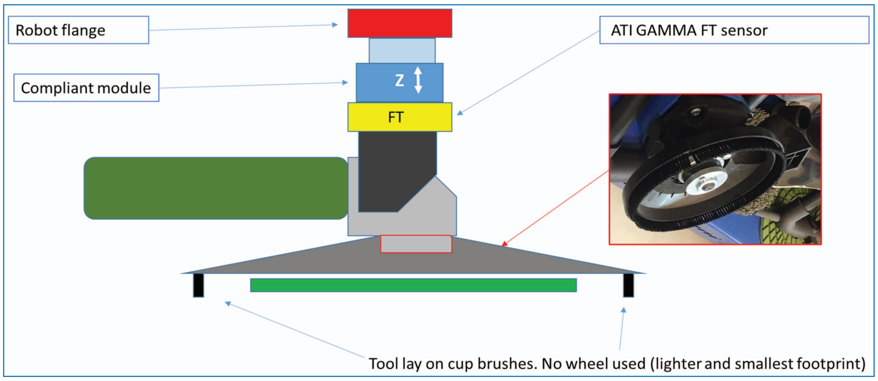

Robot flange
ATI GAMMA FT sensor
Compliant module
Z
FT
Tool lay on cup brushes. No wheel used (lighter and smallest footprint)

Figure 4.159. Updated Grinding Tool Concept.

This solution was implemented in two different release, first integrating caster wheels as in the sketch then with omni-wheel with a lighter and smaller tool footprint. A different, simpler approach was then developed and implemented on V2 prototype: The reference idea is to exploit, as human worker will do, the dust collection brush that surrounds the grinding disc as reference support and guide.

An active tool orientation control based on the FT sensor integrated on the tool can be used to ensure the even contact of the disc with the wall.

For smoother control, a compliant module is the integrated on the tool support so to compensate small wall/robot misalignment that may otherwise lead to a change in the grinding depth. The following picture shows a detail of the Z compliance module as integrated in the existing tool interface.

The next picture shows the new tool integrated in the V2 prototype. Tool control embed FT sensing and Indurad sensors.

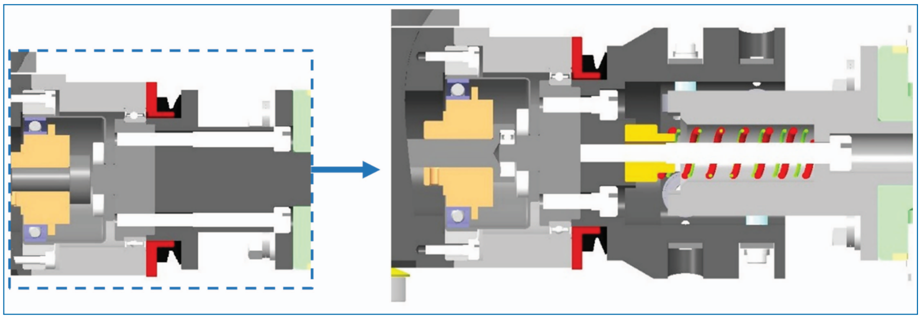


Figure 4.160. Z compliant module integration ref. design.

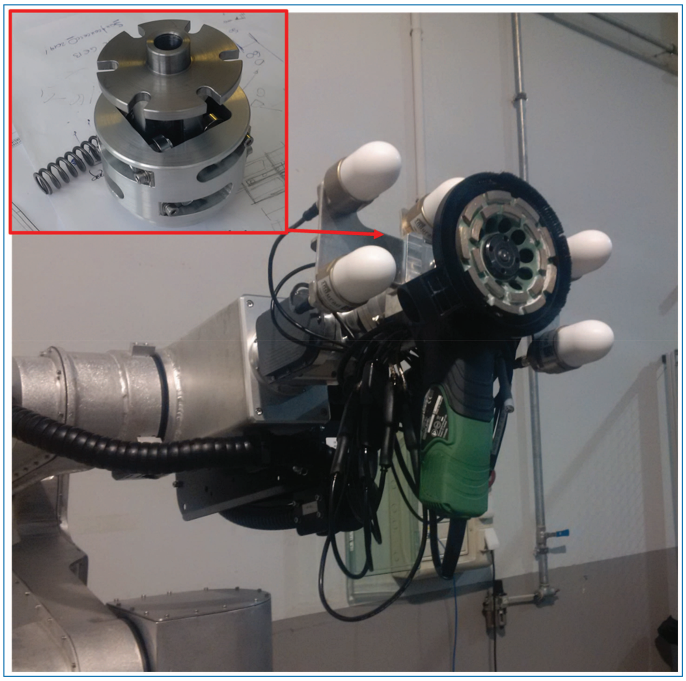


Figure 4.161. New tool integrated on the V2 robot arm.

The arm joints can be actuated independently, through the respective joint drive, or all together in a coordinated manner. In this latter case, a trajectory planner and controller is needed. For those cases where the arm moves freely in space without touching the wall, moving to a certain arm configuration or tool pose, the MoveIt! interface is used, which takes care of the calculation of the collision-free joint trajectory and its execution. For grinding, a pre-defined Cartesian path is used, which covers the wall area to be treated with an S-shaped path. In this case, both the pose of the tool and the contact pressure must be controlled within the grinding process. Hence, a simultaneous position and force control structure was implemented using

Inverse Differential Kinematics. Given the compliant property of the tool, the contact pressure can be controlled by slightly moving the robot wrist in the direction normal to the wall (without losing tool-wall contact). In this way, the error in the contact force is converted to a Cartesian speed command that is added to the one corresponding to the tool's position error. The Cartesian command is converted to a joint velocity command by the Jacobian pseudo-inverse method. In addition, use is made of the null space of the arm (it has 7 DOFs) to avoid singularities and reaching the joint limits. The joint velocity is sent to the robot, closing the loop.