The EU Framework Programme for Research and Innovation H2020
Research and Innovation Action

CENTAURO

Deliverable D7.3 First Integrated CENTAURO System

Dissemination Level: Public

Project acronym:	CENTAURO
Project full title:	Robust Mobility and Dexterous Manipulation in Disaster Response by Fullbody Telepresence in a Centaur-like Robot
Grant agreement no.:	644839
Lead beneficiary:	IIT – Fondazione Istituto Italiano di Tecnologia
Authors:	T. Klamt, M. Schwarz, C. Lenz, L. Baccelliere, D. Buongiorno, T. Cichon, X. Chen, A. Di Guardo, D. Droeschel, M. Gabardi, K. Holmquist, F. Järemo-Lawin, M. Kamedula, N. Kashiri, A. Laurenzi, D. Leonardis, L. Muratore, D. Pavlichenko, A. Selvam Periyasamy, A. Robinson, D. Rodriguez, F. Schilling, M. Solazzi, M. Felsberg, J. Folkesson, A. Frisoli, M. Gustmann, P. Jensfelt, K. Nordberg, J. Roßmann, U. Süss, N. G. Tsagarakis, S. Behnke
Work package:	WP7 Integration
Date of preparation:	2018-08-02
Type:	Demonstrator
Version number:	1.0

Document History

Version	Date	Author	Description
0.1	2018-08-02	Tobias Klamt	First draft
0.2	2018-08-26	Luca Muratore	Added Soft. Arch.
0.3	2018-09-10	Luca Muratore	Revised Sec. 2
0.4	2018-09-23	Nikos Tsagarakis	Add Summary, revised Introduction
0.5	2018-10-16	Nikos Tsagarakis	Revised Sec. 2
0.6	2018-10-19	Nikos Tsagarakis	Added Sec. 9
0.7	2018-10-23	Luca Muratore	Minor revisions in overall document
0.8	2018-11-20	Max Schwarz	Integration Overview section
0.9	2018-11-20	Nikos Tsagarakis	Integration section revision and integration
0.95	2018-11-20	Luca Muratore	Integration Overview revision
0.99	2018-11-20	Max Schwarz	Internal review & revision
1.0			Submitted version

Executive Summary

This deliverable describes the first release of the integrated CENTAURO system. The report first covers the integration efforts undertaken and successes achieved by the CENTAURO Consortium and then details the CENTAURO disaster-response system at the current level of integration. The Centauro robot is developed for solving mobile manipulation tasks in affected man-made environment that are considered as key challenges in the field of search and rescue robotics. The first integrated CENTAURO system consists of the actual Centauro robot hardware, its control station and several software modules that permit the robot to navigate in a wide variety of terrains, perform dexterous manipulation, perceive the environment and provide to users intuitive interfaces that enable fast and safe operation of the robot to execute known or partially known tasks. The first version of the integrated CENTAURO disaster-response system was evaluated in several experiments performed in the First Evaluation Camp at the premises of our application partner KHG.

Contents

1 Introduction

Mobile manipulation robots are promising platforms that can serve in many applications. Particularly for search and rescue tasks in situations where humans cannot be deployed due to risks such as radiation or collapsing structures, loco-manipulation capable mobile robots can be effective solutions to explore. Looking at respective environments, for instance the damaged nuclear plant in Fukushima, it can be observed that they are mostly man-made but cluttered with debris and unpredictable. Hence, a suitable platform needs to provide a wide range of capabilities to address possible tasks with such unstructured environments. Concerning the locomotion skills, exemplary tasks are to overcome a variety of obstacles such as ramps, gaps, cluttered terrain, stairs, or to open and pass doors. Regarding manipulation, tasks may be to use power tools, to physically connect and disconnect objects such as electrical plugs, or to scan surfaces, e.g., for radiation. Furthermore, since maintenance is not possible during missions, a high hardware and software reliability is necessary. Finally, suitable operator interfaces are key to enable the control of a system that must solve such a large variety of tasks.

With the above requirement in mind the first integrated Centauro platform was developed and integrates a number core system components including the actual robot body hardware with all mechatronic subsystems, various operator interfaces including a telepresence pilot station, autonomous, and semi-autonomous locomotion and manipulation functions, and several additional modules for communication purposes tools for the robot simulation.

The first version of the integrated Centauro platform is presented in Fig. 3. The robot has a centaur-like body configuration with four articulated legs and an anthropomorphic upper body. Its joints are powered by torque-controlled series-elastic actuators. Each leg has five degrees of freedom (DoF) and ends in a directly driven, 360° steerable wheel. This leg design allows for both omnidirectional driving and stepping locomotion and combines their advantages. This hybrid locomotion feature further enables motions that are neither possible for pure driving nor for pure walking robots, such as changing the foot print under load.

Centauro's anthropomorphic upper body consists of a yaw joint in the spine and two 7 DoF arms. In combination with the adjustable base height and orientation, this results in a workspace equal to an adult person that can reach down and manipulate in the ground or as high as 1.9m if required. The arms end in two different anthropomorphic five-finger hands to provide a wide range of manipulation capabilities. An under-actuated SoftHand, driven by a single actuator, is mounted on the left arm. Through its compliant design, it allows for robust power grasping. On the other side the right arm is equipped with a 9 DoF Schunk hand with 20 joints, capable of dexterous, human-like manipulation.

The perception system of the robot integrates a variety of sensors. A continuously 3D rotating scanner with a spherical field-of-view provides range measurements. Three wide-angle RGB cameras capture images from the robot head perspective. An additional RGB-D sensor is mounted on a pan-tilt unit below these cameras. Finally, for some tasks, the robot can be equipped with additional RGB cameras at suitable positions such as under the robot base. Situation awareness for the Centauro operators is realized through a simulation-based environment, generated from sensor data and enriched with semantic information. A digital twin of the robot is placed in this representation. This allows for flexible views on the current scene through a *HTC Vive* head-mounted display or additional monitors.

To enable the operation of the robot the integrated system includes effective teleoperation interfaces that provider intuitive to enable fast and safe operation and flexible to address unknown tasks are provided. A telepresence station allows an operator to intuitively control the whole robot. Key component is an upper-body exoskeleton, which transfers the operator's arm,

wrist and hand movements to the robot. It provides force feedback and, thus, allows the operator to solve a large variety of manipulation tasks.

To support the pilot in the operation of the robot, core autonomous locomotion and manipulation functions are integrated. In particular, a locomotion planner provides hybrid driving-stepping paths and a respective controller executes them given the input from the pilot that is the desired robot goal pose. An autonomous manipulation interface detects and categorizes objects, plans and executes optimized arm trajectories towards these objects and finally provides grasping movements. The latter even accounts for unknown objects, since suitable grasping poses are derived from known instances of the same object class.

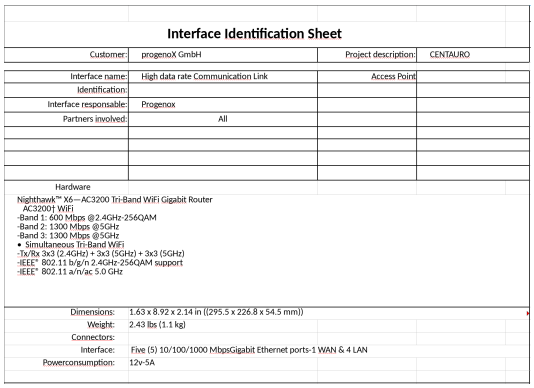
All developed components were integrated to form the first release of the Centauro platform. Most software components and the communication between them are realized in the *Robot Operating System (ROS)*. Those components requiring deterministic real time performance are running in dedicated real time threads implemented inside the XBotCore software framework that was used to implements the low-level system software architecture.

The functionality and capabilities of the first integrated Centauro platform was evaluated during an intensive testing period at the facilities of Kerntechnische Hilfsdienst GmbH (KHG) partner in Karlsruhe, Germany. Locomotion capabilities were evaluated in tasks like driving up a ramp, overcoming a gap, moving through an irregular step field and climbing a flight of stairs. Semi-autonomous and autonomous manipulation was evaluated in tasks like drilling a hole with a power drill, screwing a screw with an electrical screw driver, cutting a cable with an electrical cutting tool, mounting a snap-hook, opening and closing lever-type and gate-type valves, connecting and disconnecting a 230 V power plug and a fire hose, scanning a surface with a radiation measurement device, and grasping and operating an electrical screw driver. Most of the tasks were performed successfully and without previous training during this evaluation period which demonstrates the wide range of capabilities the integrated Centauro platform. In Section 2, an overview of the integration tools, activities and status is given. Sections 3 to 8 then report on the individual components in more detail and draw a complete picture of the first integrated CENTAURO system. Finally, Sections 9 and 10 draw conclusions from the integration work.

2 Integration Overview and Status

During this project period, one of the major activities of the project within workpackage WP7 was related to the integration of core components and results generated by the other research and development workpackages into the first release of the CENTAURO disaster-response system. To reach this target, several meetings and integration weeks were organized by the Consortium and took place at the premises of the project partners focusing on the integration of core components as soon as they became available. Some of these integration camps were aligned with the regular internal project progress meetings where all partners participated, while other bilateral meetings between two or more project partners were also arranged to execute integration activities on specific core components of two or more WPs.

This section provides an overview of the integration activities and associated achievements in terms of integrated components and establishment of interfaces to reach milestone MS3 of the first integrated CENTAURO disaster-response system.



Interface Identification Sheet

Customer:	progenoX GmbH	Project description:	CENTAURO
Interface name:	High data rate Communication Link	Access Point	
Identification:			
Interface responsable:	Progenox		
Partners involved:	All		

Hardware

Nighthawk™ X6—AC3200 Tri-Band WiFi Gigabit Router
AC32001 WiFi
-Band 1: 600 Mbps @2.4GHz-256QAM
-Band 2: 1300 Mbps @5GHz
-Band 3: 1300 Mbps @5GHz
• Simultaneous Tri-Band WiFi
-Tx/Rx 3x3 (2.4GHz) + 3x3 (5GHz) + 3x3 (5GHz)
-IEEE® 802.11 b/g/n 2.4GHz-256QAM support
-IEEE® 802.11 a/n/ac 5.0 GHz

Dimensions:	1.63 x 8.92 x 2.14 in ((295.5 x 226.8 x 54.5 mm))
Weight:	2.43 lbs (1.1 kg)
Connectors:	
Interface:	Five (5) 10/100/1000 MbpsGigabit Ethernet ports-1 WAN & 4 LAN
Powerconsumption:	12v-5A

Source / Sink		1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	3D Simulation (RWTH)	X	IIII	-	-	-	III	II	-	-	-	II	-	-	I
2	Central World Model (RWTH)	II	X	IIII	IIII	IIII	IIII	IIII	IIII	-	IIII	II	-	-	I
3	Terrain Perception (KTH)	-	III	X	-	-	-	-	IIII	-	-	-	-	-	-
4	Workspace Perception (LIU)	-	III	-	X	-	-	-	-	-	III	-	-	-	-
5	Communication Link (PGX)	-	IIII	-	-	X	-	-	-	-	-	-	-	-	-
6	Main Operator Station (SSSA)	III	II	-	-	-	X	-	II	-	II	-	IIII	II	-
7	Support Operator Station (RWTH)	IIII	IIII	-	-	-	-	X	-	-	-	-	-	-	-
8	Locomotion Planning (KTH)	-	III	-	-	-	-	-	X	II	-	-	-	-	-
9	Whole-body Balance (IIT)	-	II	-	-	-	-	-	-	X	-	II	-	-	-
10	Manipulation Planning (UBO)	-	II	-	-	-	-	-	-	-	X	IIII	-	-	-
11	Robot Control (IIT)	IIII	IIII	-	-	-	II	-	IIII	IIII	-	X	-	-	-
12	Robot Upper Body (IIT)	IIII	-	-	-	-	-	-	-	-	IIII	-	X	-	II
13	Robot Lower Body (IIT)	IIII	-	-	-	-	-	-	-	-	-	-	-	X	I
14	Robot System Software (IIT)	I	-	-	-	-	-	-	-	III	-	III	III	III	X

Figure 1: Integration tools. Left: Interface Identification Sheet. Right: Interface Status Table (prior to integration meeting in Pisa).

Table 1: Integration meetings.

PM	Date	Partners	Host	Duration (days)	Topics
13	04/16	All	RWTH	3	VEROSIM Workshop
21	12/16	IIT, SSSA	SSSA	4	WP2 + WP3 interface
21	12/16	All	UBO	8	Integration, review meeting
27	06/17	All	IIT	3	XBotCore Workshop
31	10/17	RWTH, UBO	UBO	2	WP4 + WP5/6 interface
31	10/17	IIT, SSSA	SSSA	4	WP2 + WP3 interface
31	10/17	All	SSSA	5	General integration meeting
32	11/17	IIT, UBO	IIT	5	WP2 + WP5/WP6 interface
32	11/17	All	KHG	5	Evaluation camp

2.1 Integration Tools

The Consortium made use of several tools to facilitate, guide, and monitor integration. As a first example we chose, very early in the project, to use ROS as the underlying framework and data exchange method between the different software components. This effectively removes the concern about interface implementation details and shifts the focus towards interface definition.

Especially in the beginning, additional tools were used for identification and tracking of interface status. An attempt was made to fully specify the interfaces using *Interface Identification Sheets* (see Fig. 1). However, this was too early in the project, and the actual interface content was hard to define. A less fine-grained method was required. The *Interface Status Table* (see Fig. 1) allows identification of required interfaces and tracking of their status on a core component level. This method was used up to the evaluation camp in Karlsruhe.

For the individual components unit tests were written and regularly executed on a continuous integration server maintained by UBO. Using this service, breaking changes could be identified quickly.

The most important integration tool, however, were physical meetings. These meetings brought the hardware together, which is necessary for hardware integration, but also proved to be very important for software integration, as interface issues can be solved in close loop with quick iterations. Table 1 lists the integration meetings performed before the evaluation camp.

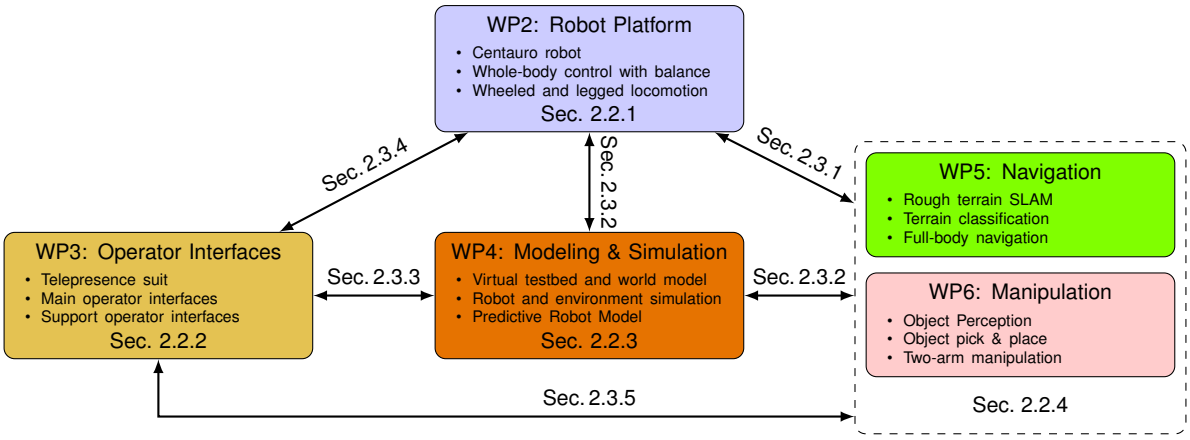


Figure 2: CENTAURO workpackages and inter-workpackage interfaces as described in this report.

2.2 Integration of Individual Workpackages

Figure 2 gives an overview of the CENTAURO workpackages and the inter-package dependencies. Concerning the hardware components of the first integrated CENTAURO disaster-response system, the integration activities executed towards the first release of the CENTAURO platform include the integration of the mechatronics components of the CENTAURO robot and the mechatronics components of the telepresence operator station. At the same time, the modules inside each software workpackage were integrated into larger modules in preparation for usage in the first CENTAURO system.

2.2.1 WP2: Integration of Robot Hardware

Starting from the robot status at the end of the first period the integration effort on the robot side concentrated on the the integration of the upper body (available at the end of the first period) with the lower body (pelvis and legs) of the robot as soon as they became available. The integration of the two subsystems of the robot body involved the establishment of electromechanical interfaces between the two body parts as well as the communication interfaces between the decentralized low-level control modules and the on-board computers. A second hardware integration on the robot side is associated with the mounting of the head module on the upper body of the robot and the incorporation of the physical interfaces between the various head perception sensors with the on board computers. In addition to the realization of the first physically integrated CENTAURO robot hardware, the integration effort on the robot also produced the following integration results related to robot models and physical interfaces:

- The model of the robot was updated incorporating the introduction of all hardware components including the lower body and the head of the robot.
- The robot configuration files were updated and tuned to incorporated the latest hardware changes in terms of kinematics, joint limits, low-level control parameters etc.
- The physical electrical and communication buses were established among the different body parts, the low-level control units, the board computers, the robot power management unit controller and the perception devices.

- The interfaces and protocols between the on-board centralized real time control computer and the low-level control units were established allowing the monitoring of the robot status as well as the communication with low-level control, permitting tuning of the control parameters and the selection of the operation mode.
- Finally, a set of robot initialization modules and postures were defined and implemented to assist the starting, homing and shutting down procedures of the robotic platform.

The above models and interfaces were provided to permit integration of robot with the rest of the core components introduced in the next sections of this deliverable report.

2.2.2 WP3: Integration of Telepresence Station Hardware

Starting from the state of the operator station at the end of the first period, the integration activity on this core component focused on the integration of the mechatronic components of the tele-presence station including the integration of the second arm exoskeleton system (four DoF) to realize the dual arm exoskeleton master. Furthermore, the integration of the wrist exoskeleton submodule was performed enabling the full human arm motion monitoring and feedback control. The last mechatronic component that was integrated towards the first version of the integrated CENTAURO disaster-response system was the first prototype of the hand exoskeleton device that was incorporated in one of the two arm exoskeletons to permit the motion monitoring and force feedback at the level of the operator hand. To make the operator station hardware operational, the integration activity on the operator station also focused and established the following interfaces:

- The kinematic models of the exoskeleton devices were updated to incorporate the extensions of the wrist and hand modules.
- The physical electrical and communication interfaces were established among the different exoskeleton components, the motor control units and the centralized computer of the tele-presence station.
- An application programming interface (API) was developed permitting the reading and writing to the exoskeleton devices allowing the interfacing with the rest of the core components including the robot platform, the simulation environment and the manipulation control components.

2.2.3 WP4: Modeling & Simulation

The VEROSIM simulator is a central component in the CENTAURO system. For connection to the other ROS-based modules, a ROS interface was added to VEROSIM. It allows the loading of robot models specified in the ROS URDF format, and implements a client for the ROS publisher/subscriber message passing protocol, thus allowing input of robot state and sensory data for visualization, input of robot commands for simulation, and output of simulation results.

2.2.4 WP5/6: Autonomous Locomotion and Manipulation

The modules for terrain perception, classification, localization, and mapping were integrated into a pipeline for autonomous navigation. The pipeline is described in Sections 5 and 7. Analogously, the components for autonomous grasping, namely object segmentation, pose estimation, non-rigid registration, and trajectory optimization were integrated as described in Section 8. Interfaces to permit the use of the modules by other workpackages were realized.

2.3 Inter-Workpackage Software Components and Interfaces

With the hardware and physical interfaces established for the two major mechatronic core components the integration effort was concentrated on the realization of the interfaces to permit the interconnection of the various software, control and simulation tools developed in the different workpackages required for carrying out the experimental evaluation of the first integrated CENTAURO disaster-response system. This subsection summarizes the interfaces developed and established among the different workpackage core components until M30 and towards the evaluation camp of the first integrated platform.

Figure 2 gives an overview of the inter-workpackage interfaces that need to be established in the project. The following sections describe the achieved progress for each of these interfaces on workpackage topic basis. Note that WP5 and WP6 are grouped together, since they are built upon the same underlying framework, namely UBO's ROS-based software framework, and thus use the same interfaces to communicate with other workpackages.

2.3.1 WP2 and WP5/6: Primary Loco-manipulation and Perception to Robot Interfaces

To enable the execution of manipulation and locomotion trials with the CENTAURO robot the establishment of the interfaces between the robot and the associated locomotion and manipulation modules of WP5 and WP6 was necessary. For this purpose the XBotCore framework was setup in the on board computers and established / exposed a number of initial interfaces allowing the connection to the robot. These interfaces between the robot low-level control (WP2) and higher-level control modules (WP5 and WP6) are bidirectional and are defined on the joint level. These primary interfaces permit to read the robot status and to command the robot through the position, velocity or torque interfaces and to execute basic motions by imposing motion trajectories through the provided interfaces.

These primary connection interfaces to the robot were extended by the integration activities towards the first integrated CENTAURO disaster-response system. In particular, the following interfaces and features required to execute the first evaluation camp trials were realized.

- The primary interfaces to the robot low-level state were extended to incorporate other robot sensors including the FT sensors on the arms and the Inertial measurement unit (IMU) installed on the robot pelvis.
- The interfaces to permit the communication of the software stack developed at UBO and the on board XBotCore framework control were realized enabling the UBO software stack to command the robot using the position and velocities interfaces provided by the XBotCore framework. Furthermore, this allowed the UBO Software stack to read the state of the robot joints including the joint position, velocity and torque. Position, velocity, acceleration, and auxiliary information like joint motor current and temperatures are published by the IIT *XBotCore* control framework as custom ROS messages. UBO's *robotcontrol* software was extended with a custom hardware interface plugin to understand these ROS messages.
- In the inverse direction, the higher-level control modules in WP5 and WP6 generate motion requests, which are handled inside *robotcontrol*. After keyframe interpolation and trajectory optimization steps the resulting trajectory is played back and target joint positions are streamed to *XBotCore*, again via custom ROS messages published by the hardware interface plugin.

- As a particular benefit, this interface allows to use UBO's kinematic control software, which was developed for the UBO's Momaro robot. It allows kinematic control of the high-DoF arms and base, including omnidirectional wheeled or legged locomotion. This allowed testing and development to continue while the corresponding components in WP2 were under development. When these components were ready, the interfaces were extended to allow higher-level goals like foot placements to be transmitted towards WP2.

- Independently from the rest of the robot an interface was established to enable the control of the right arm end-effector of the robot (Schunk hand) using the available ROS drivers,

- Finally, the interfaces between the robot sensors, especially the sensors mounted on the torso, and the perception components of WP5 and WP6 were also established by using standard ROS drivers for the sensor components.

  Further to the above software integration and interfaces to enable the full use of the integrated robot the execution of activities related to the calibration of robot perception system (PointGrey cameras, Kinect, Laser scanner) were executed producing the required calibration procedures and transformations for the perception system data that were necessary for the execution of semi-autonomous of fully autonomous loco-manipulation trials.

2.3.2 WP2, WP4 and WP5/6: Robot Simulation

To enable the effective simulation of the CENTAURO robot inside the VEROSIM simulation environment a set of interfaces between VEROSIM and the XBotCore framework and ROS ecosystem were realized to enable the simulation of the robotic platform and ensure the consistency between the simulated model and the real robotic platform. The updated Universal Robot Description File (URDF) model was utilized to realize the simulation of the full robot model within the VEROSIM environment reflecting the most recent updates on the robot hardware.

Since ROS is used as the interface to VEROSIM, the simulator can be easily put in place of the real robot. While the canonical way is to emulate the XBotCore interface, a second interface mimicking the Momaro robot was also added. Using this second interface, the simulator can be tested against existing control modules for Momaro.

In addition, visualization methods and interfaces related to the perception system of the robot were established allowing the display of the perception data of the robot through the VEROSIM environment using a head mounted display. In total, we established displays for laser data, images, Kinect v2 point clouds. An interface to display low-resolution depth (i.e. from the 3D laser or Kinect) with high-resolution texture (from Kinect or PointGrey cameras) is the focus of ongoing work.

For testing actions in simulation of a real scene, a snapshotting procedure is considered where a copy of the current scene as measured by the robot is used as the background scene in simulation.

2.3.3 WP3 and WP4: Immersive Teleoperation

For immersive teleoperation, WP4 needs to display the robot state to the operator interfaces established in WP3. Towards this goal, VEROSIM was extended to be able to render its visualizations inside a head-mounted display for the main operator.

2.3.4 WP2 and WP3: Teleoperation Station to Robot Interfaces

For latency and control reasons, there is a direct low-level interface between the exoskeleton of the main operator and the robot itself. Here, a UDP-based bidirectional protocol was established for transmitting measured forces and torques to the operator and the commanded end-effector positions to the robot. In particular the following integration and interfaces were realized:

- For the tele-manipulation control through the tele-presence station the input and feedback interfaces were realized allowing the command of the robot arm motion at the operational space through references produced by the operator wearing the arm exoskeleton system. In addition the tele-manipulation control of the robot arms can be also performed through alternative interfaces like 6D input devices that can be used to command the wrist of the robot arm.
- Similarly the interface to control the Schunk end-effector through the the hand exoskeleton master device was established permitting to control the Schunk hand fingers through the motions of the human operator finger.
- The rendering of the forces was enabled through the same low latency interface that directly conveys the measured interaction forces from the robot side to the telepresence station.

2.3.5 WP2, WP3 and WP5/6: High-level teleoperation

The execution of the loco-manipulation trials during the evaluation of the first integrated platform required the integration of a number of modules and interfaces for the loco-manipulation control. These resulted to the following integration results for the robot loco-manipulation control.

- For the omnidirectional control of the robotic platform the UBO kinematic control was incorporated with the realization of the associated interfaces allowing the wheeled and legged driving control of the CENTAURO robot through steering commands and wheel velocities or leg stepping references calculated for the different robot driving and stepping configurations. A number of operator interfaces were incorporated to assist the execution of wheeled and legged mobility of the CENTUARO robot including joystick and pedal interfaces as well as other interfaces like the key frame editor tool and related GUIs that enables effective control of the leg and foot motions through interactive markers and mouse interfaces.
- For the manipulation control of the robot upper body interfaces permitting the commanding of the two robot arms at the operational space were established utilizing the on board inverse kinematics modules. In addition interfaces to control the posture of the robot pelvis/height were realized permitting the regulation of the robot height for the purpose of loco-manipulation tasks.
- The modules for autonomous locomotion and manipulation offer interfaces for operator input. For the first integrated CENTAURO system these inputs can be made using secondary operator interfaces.

The details of the above components and interfaces established in the first phase of the project integration to reach milestone MS3 and the first integrated CENTAURO disaster-response system are presented in the following sections of this deliverable report.

3 Robot Mechatronics and Software Architecture

Mobile manipulation robots have been developed for a variety of fields such as search and rescue, planetary exploration, or personal assistance. Those robots vary in their locomotion strategy which can be realized by driving with wheels or tracks, by walking with multiple legs, or by a combination of both. The Centauro robot was designed for disaster-response missions. Since the tasks in such missions are in general unknown in advance and depend on the nature of the disaster, the system has to cover a wide variety of navigation and manipulation capabilities. Concerning the locomotion skills, the target environments are mostly man-made and thus, contain flat surfaces, ramps, and stairs. Those environments may, however, be affected, e.g. by cluttered debris. As far as the manipulation ability is concerned, addressed workspaces are also man-made and contain tools and objects that are designed for human usage. A summary of the robot hardware core component is introduced here while the details of the robot hardware can be found in in Section 3.1.

3.1 Robot Design

According to requirements of common disaster-response scenarios, the Centauro robot (see Fig. 3) is designed as a centaur-like platform embodying an anthropomorphic upper-body and a quadrupedal lower-body—bringing together Momaro's kinematics and Walk-Man's compliant actuation.

The lower-body comprises four wheeled legs whose first joints are accommodated in the four corners of the base structure. Fig. 4 illustrates the kinematic scheme and joint positioning. To allow for versatile locomotion, each leg consists of five DoF in a spider-like configuration, which can be more beneficial in terms of stability required for the manipulation of powerful tools, as shown in [15]. Furthermore, in this configuration, the first leg joint has to deliver substantially lower effort and power compared to a mammal-like configuration. According to the chosen spider-like configuration, each hip module consists of a yaw and a pitch joint, followed by another pitch joint in the knee. Each ankle consists of a pitch and a yaw joint which allow for steering the wheel and adjusting its steering axis to the ground. Finally, each leg ends in an actively drivable wheel. The described configuration allows for omnidirectional driving as well as for articulated stepping locomotion.

Since no posture change is needed to switch between the two, it is even possible to perform motions which are unique for that design such as moving a foot relative to the base while under load. Thus, a wide range of locomotion capabilities is provided. To permit versatile leg articulation in difficult terrains, the ranges of the leg joints were maximized while taking into account the mechanical and electrical interfacing constraints. The robot torso incorporates two 7 DoF arms and an additional rotational joint in the waist, to endow the upper-body with yaw rotation. The kinematics of the two arms closely resembles an anthropomorphic arrangement to provide a large workspace, to enable dexterous manipulation, and to simplify teleoperation. Each arm comprises of three DoFs at the shoulder, one DoF at the elbow and another three DoFs at the wrist.

The degree of redundancy helps to overcome possible constraints that may be introduced in the task space by the surrounding environment. Even though this is a traditional design that aims at replicating the anthropomorphic structure of the human arm with seven DoF, it is only approximately equivalent for the human arm kinematic structure. To extend the range of motion of the elbow joint, an off-center elbow configuration was chosen. Similarly, for the wrist, a non-anthropomorphic configuration with non-intersecting axes was considered to

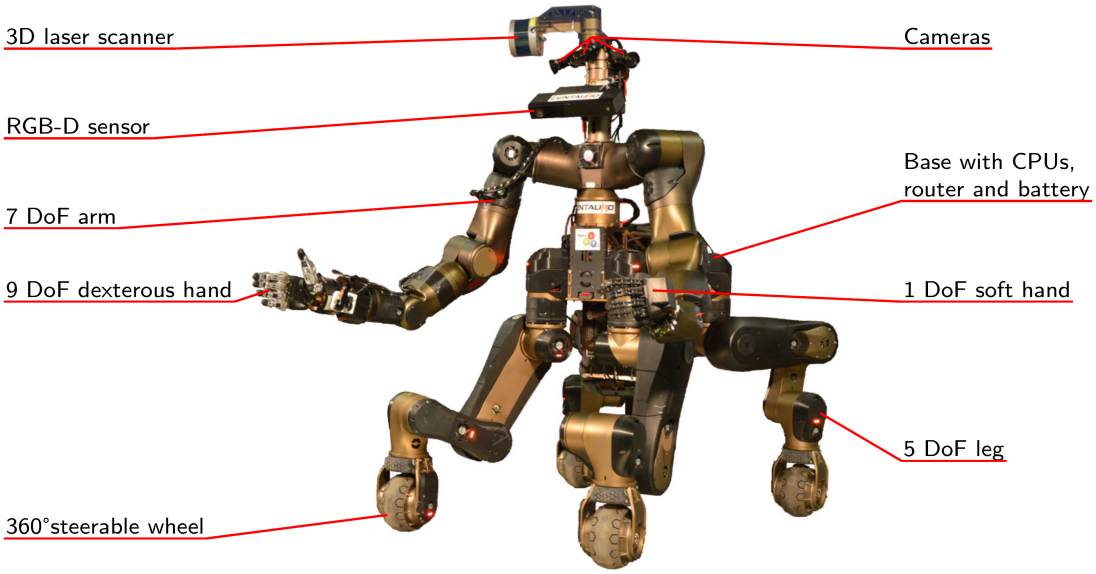


Figure 3: The Centauro robot.

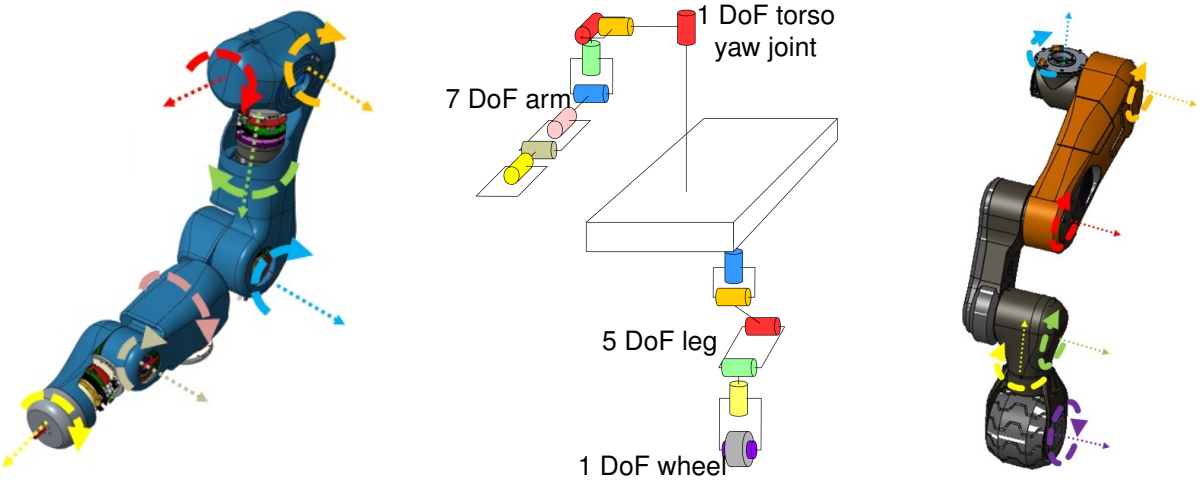


Figure 4: Kinematic layout. Left: The right arm. Joint axes are marked with colored lines. Center: Kinematic tree. For clarity, only one arm and one leg are pictured. The hands are neglected. Proportions are not to scale. Right: The front left leg.

maximize the range of motion of the wrist flexion and abduction motions. Finally, humans have the ability to elevate (upward/downward) and to incline (forward/backward) the shoulder joint, utilizing supplementary kinematic redundancy of the arm to achieve certain goals in task coordinates. This, however, would require the addition of two more DoF to each arm, increasing the complexity/weight and dimensions of the overall platform.

To avoid this, while at the same time obtain, to some extent, the benefits provided by the elevation (upward/downward) and inclination (forward/backward) of the shoulder, a fixed elevated and inclined shoulder arrangement was selected based on the optimization study in which important manipulation indices were considered and evaluated in a prioritized order [3].

The two arms end in different end-effectors with complementary properties to provide an overall wide range of manipulation capabilities. Both hands are shown in Fig. 5. On the left side, a 1 DoF SoftHand provides compliant and robust manipulation. The right arm utilizes an

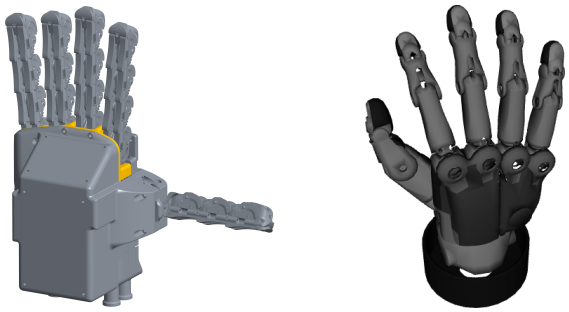


Figure 5: Centauro manipulation end-effectors: 1 DoF SoftHand (l.) and anthropomorphic 9 DoF Schunk hand (r.).

anthropomorphic 9 DoF Schunk hand for dexterous manipulation tasks. A customized force-torque sensor between the right arm wrist and the Schunk hand measures 6D forces/torques (F-T), which are applied to the end-effector and can be used for force feedback by the exoskeleton. An upgrade of the F-T sensors local regulator and the installation of decoupling capacitors between the two isolated grounds and the load cell structure were performed, to avoid EtherCAT errors possibly caused by a noisy power supply.

The perception system of the robot includes a head module that integrates a set of cameras and sensors. It encompasses a *Microsoft Kinect V2* RGB-D sensor [10], an array of three *PointGrey BlackFly BFLY-U3-23S6C* wide angle color cameras, and a rotating *Velodyne Puck* 3D laser scanner with a spherical field-of-view. A *VectorNav VN-100* inertial measurement unit (IMU) is mounted in the torso. Two additional RGB cameras are mounted under the robot base to get a view on the feet.

On board computation is provided with the incorporation of three computing units. One is responsible for the Real-Time control of the robot and runs on a XENOMAI RT development kit[1], while the other two used for perception and high level robot control, with the specification reported in Table 2 and 3.

Table 2: CENTAURO RT on-board computational unit hardware specifications.

COM Express Type 6	Conga-TS170
CPU	Intel Core i7-6820EQ 2.80GHz up to 3.50GHz 4 cores (2 logical cores per physical) TDP: 45 W
RAM	16GB

3.1.1 Actuation System

The series-elastic actuation (SEA) technology is utilized to protect the reduction gear against impacts—improving the system sturdiness, while at the same time being used for measuring the actuator torque through the monitoring of the elastic element deflection. Fig. 6 introduces the manufactured actuation units used in the robot while Table 4 shows the respective specifications. Considering the influence of different joints' stiffness levels on the natural dynamics and control

[1] `https://xenomai.org/`

Table 3: CENTAURO on-board perception and high level robot control unit hardware specifications.

ZOTAC MAGNUS	ZBOX-EN1070K
CPU	Intel Core i5-7500T 2.7 GHz, up to 3.3 GHz 4 cores (2 logical cores per physical) TDP: 35 W
GPU	GeForce® GTX 1070 8GB GDDR5 256-bit
RAM	32GB

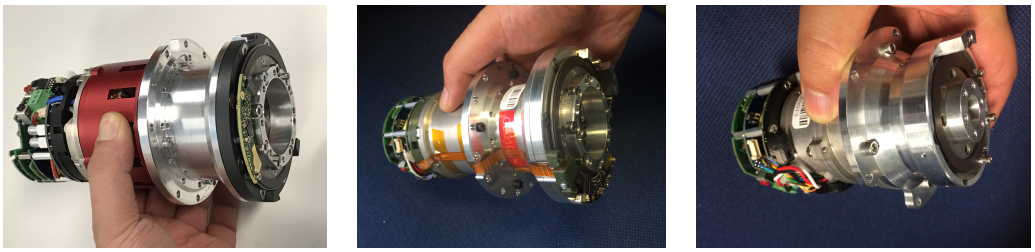


Figure 6: Manufactured actuation units of the identified classes (f.l.t.r.): Large, Medium, Small.

Table 4: Actuator specifications.

Type	Gear ratio	Joints	max. velocity [rad/s]	Torque [Nm] peak - continuous	Power [W] peak - continuous	Torque sensing resolution [Nm]	Mass [kg]
Large	120	Hip, Knee	8.8	268 - 92	2096 - 778	0.2	1.73
Medium A	160	Ankle Pitch, Elbow, Shoulder yaw	6.1	147 - 46	820 - 259	0.07	1.28
Medium B	160	Torso, Shoulder pitch, Shoulder roll	3.9	147 - 81	1014 - 295	0.07	1.45
Small A	100	Forearm yaw, Forearm pitch	11.7	55 - 17	556 - 179	0.07	1.0
Small B	100	Wheel, Ankle yaw, Wrist yaw	20.4	28 - 9	518 - 167	0.07	0.87

of the robot, as discussed by [16] and [31], and taking into account the available space for the different actuators, two technologies were utilized for joint torque measurement based on strain-gauge and deflection-encoder principles. The stiffness of the SEA deflection-encoder-based sensor is defined with respect to the required torque measurement resolution across the different joints.

The decentralized controller of the actuators is developed based on an impedance control

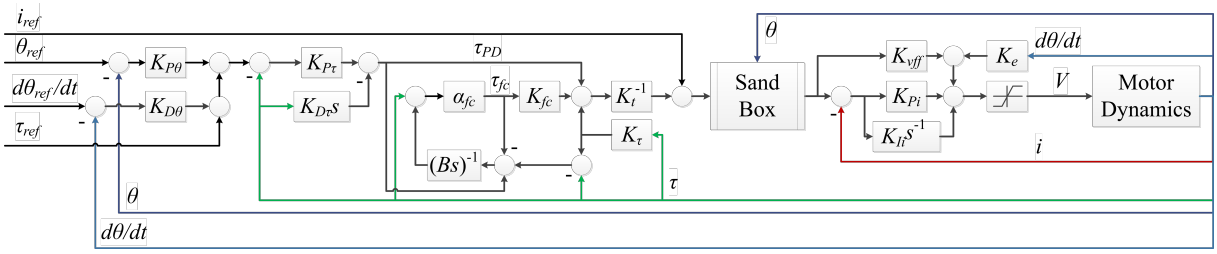


Figure 7. Block diagram scheme of the joint controllers: current feedback in red, torque feedback in green, position and velocity feedbacks in blue.

scheme utilising motor positions θ and velocities $\dot{\theta}$, and measured joint torques τ, displayed in Fig. 7. The inner most loop carries out the control of measured current i using a Proportional-Integral (PI) controller, with P and I gains of K_{Pi} and K_{Ii}. The current control execution is complemented by compensation of back-electromagnetic force (back-emf) effects with K_e denoting the back-EMF constant, and by addition of a voltage feed-forward term with a gain of K_{vff} on the current reference i_{ref}. The reference value i_{ref} is set by the torque controller when converted to current using the motor torque constant reflected to the rotor side, K_t, and by a centralized offset value i_{off}, provided that the joint position is within the admissible range. To respect the mechanical position limit of joints, we implement the "sand box" module. It includes a one-directional stiffening PD position controller that is activated when the joint position is approaching the end limit, and prevents the joint position from meeting the mechanical stop. The torque controller is based on a Proportional-Derivative (PD) regulator, with P and D gains of $K_{P\tau}$ and $K_{D\tau}$, with a torque state feedback with a gain of K_τ. Furthermore, a friction compensation scheme [14] [39] is implemented for the cancellation of motor friction/damping, and a portion of estimated friction τ_{fc}, to be set by $K_{fc} \in [0, 1]$, is added to the torque controller output.

3.1.2 Low-Level Software Architecture

For the low-level control of the CENTAURO platform, we developed *XBotCore* (*Cross-Bot-Core*), a light-weight, real-time (RT) software platform for robotics [23]. *XBotCore* is designed to be both an RT robot control framework and a software middleware. It satisfies hard RT requirements, while ensuring a 1 kHz control loop even in complex multi-DoF systems. The *XBotCore* Application Programming Interface (API) enables an easy transfer of developed software components to multiple robot platforms (cross-robot feature), inside any robotic framework or with any kinematics/dynamics library as a back-end. Out-of-the-box implementations are available for the *YARP* and *ROS* software frameworks and for the *RBDL* and *iDynTree* dynamics libraries.

A Robot Hardware Abstraction Layer (R-HAL), introduced in [29, 28] that permits to seamlessly program and control any robotic platform powered by *XBotCore*, is also provided by the framework. Moreover, a simple and easy-to-use middleware API, for both RT and non-RT control frameworks is available. The *XBotCore* API is completely flexible with respect to the external control framework the user wants to utilize. Inside the Centauro robot, an RT Cartesian control plugin based on the OpenSoT [22], a hierarchical whole-body control library and the built-in *ROS* support are used. As shown in Fig. 8, *XBotCore* spawns three threads in the Linux Xenomai, recently upgraded to version 3.0 RTOS:

- The R-HAL RT thread is running at 1 kHz and is responsible to manage and synchronize the EtherCAT slaves in the robot, i.e., the electronic boards responsible for motor control

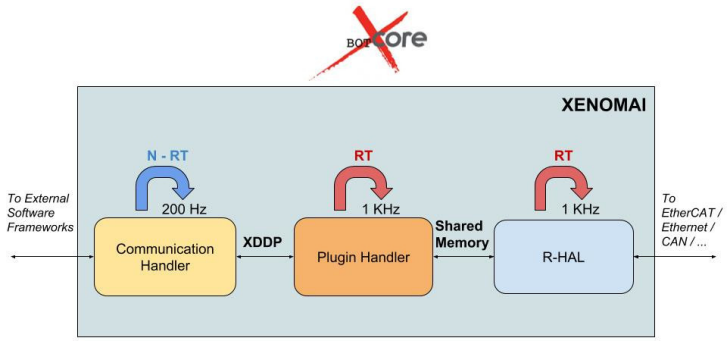


Figure 8: *XBotCore* threads and communication architecture.

and sensor data acquisition.

- The Plugin Handler RT thread is running at 1 kHz and is responsible to start all the loaded plugins, execute them sequentially and close them before unload. It is possible to dynamically load and unload one or more plugins in the Plugin Handler. As an example, the above mentioned RT Cartesian control plugin is running inside the Plugin Handler. A shared memory communication mechanism is used to share data between this component and the R-HAL at 1 kHz.
- The Communication Handler non-RT thread is running at 200 Hz and is responsible for the communication with external frameworks. This component provides the option to send the desired robot state from the non-RT API to the chosen communication framework and to receive the reference, respectively. The Communication Handler uses XDDP (Cross Domain Datagram Protocol) for the asynchronous communication between RT and non-RT threads, guaranteeing a lock-free IPC (Inter-Process Communication). The run loop of this component is quite simple: it updates the internal robot state using the XDDP pipe with the non-RT robot API, sends the robot state to all the communication frameworks, receives the new reference from the requested ”master” (we avoid to have multiple external frameworks commanding the robot) and finally, sends the received reference to the robot using the XDDP non-RT robot API.

4 Telepresence Pilot Station

In this section we introduce the main features of the telepresence pilot station while the complete description of the design of a telepresence station for the operator can be found in Section 4.

An intuitive teleoperation interface is key to control a robot, as complex as Centauro, with all its capabilities. A full-body telepresence suit (see Fig. 9) allows for immersive control of the robot, especially for manipulation tasks. A high degree of intuition is provided by giving the operator the feeling of being present in the robot by perceiving the scene from the robot head perspective and moving the robot arms and hands as own limbs. This is realized through a head-mounted display for visual and acoustic situation awareness, pedals for basic locomotion control, and—most important—an upper-body exoskeleton. The latter enables the operator to transfer his upper body movements to the robot upper body providing intuitive telemanipulation control.

The upper body exoskeleton permits about 90% of the human’s arm workspace without singularities or other constraints and provides force feedback to the operator by displaying the interaction forces between the robot and its environment. Thus, it allows the operator to intuitively transfer his or her experience and capability of situation assessment into the scene.

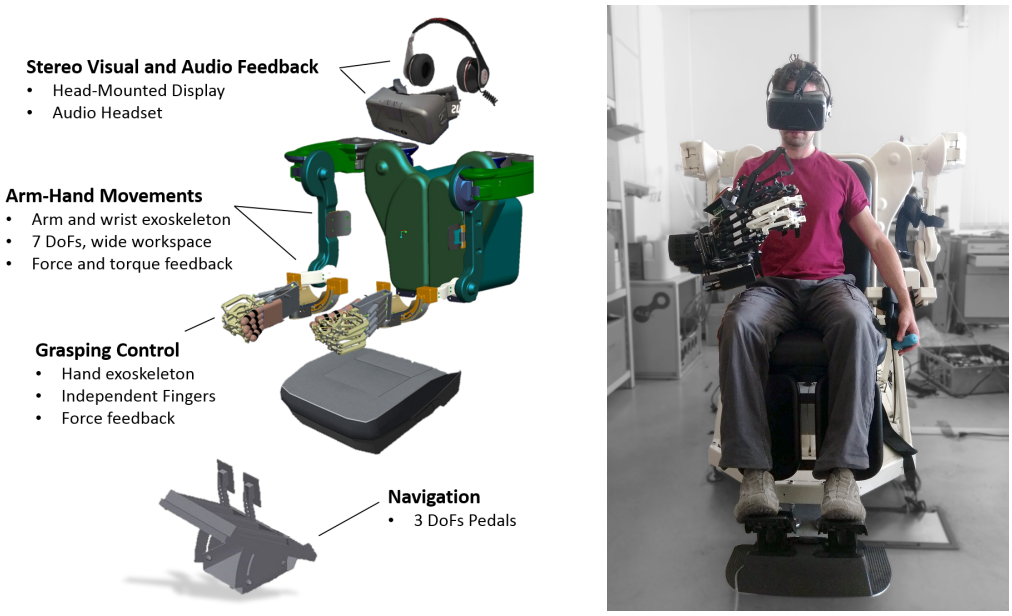


Figure 9: The full-body telepresence suit, components and implementation.

The upper-body exoskeleton consists of

- two upper limb active exoskeletons for the arms with four DoF each (see Section 4.1),
- two active wrist exoskeletons with three DoF each (see Section 4.2),
- and an underactuated active hand orthosis with one DoF for each finger (see Section 4.3).

4.1 Arm Exoskeleton

The dual arm exoskeleton is a mechanically compliant robotic manipulator that provides full support for shoulder and elbow joints [26]. It features an innovative and lightweight backdrivable transmission with electric actuators that are located behind the operator's back. The torque transmission from the actuators to the joints is achieved through idle pulleys and in-tension metallic tendons. This design allows for light moving parts which achieve a high dynamic performance due to low inertia. The total weight of the moving parts is only 3 kg, of which about 2 kg belong to the first two proximal links.

As the utilized metallic tendons show some compliant behavior, which would result in inaccurate motion tracking or force transmission two position sensors are used: one at the motor shaft, the other at the joint shaft, after the tendon transmission. The difference of the measured relative position is an indirect measurement of the tension of the tendon. Such method allows a higher degree of robustness regarding the force control stability to the variability of the human limb mechanical impedance.

Fig. 10 shows the manufactured arm exoskeleton and its kinematic design.

The arm exoskeleton kinematic is isomorphic to that of the human arm, i.e., the device's axes of rotation are aligned with those of the operator's physiological articulations. It has the important property of presenting no singularities in the natural workspace of the upper limb. To accommodate enough space for the user's shoulder without having any interference with the mechanical structure of the device, the implementation of the 2nd DoF utilizes a remote center of rotation mechanism (RCRM) based on a four-link pantograph.

The major part of the control electronics has been custom built to be embedded into the mechanics of the device, in order to minimize the electrical wiring effort at the motors and sensors. The non-moving base link of the device (named the backpack) hosts the electric actuators

(a) (b) (c)

Figure 10: (a),(b) The arm exoskeleton features four actuated DoFs (shoulder and elbow) in a wide operative workspace, (c) kinematic scheme.

(brushless DC motors) and the embedded control electronics and is connected to a rigid support structure that holds the power supply unit and the main control computing unit. A chair, having a manual hydraulic height regulation of the seat, allows users of different size to operate the device from a comfortable seating position.

4.2 Wrist Exoskeleton

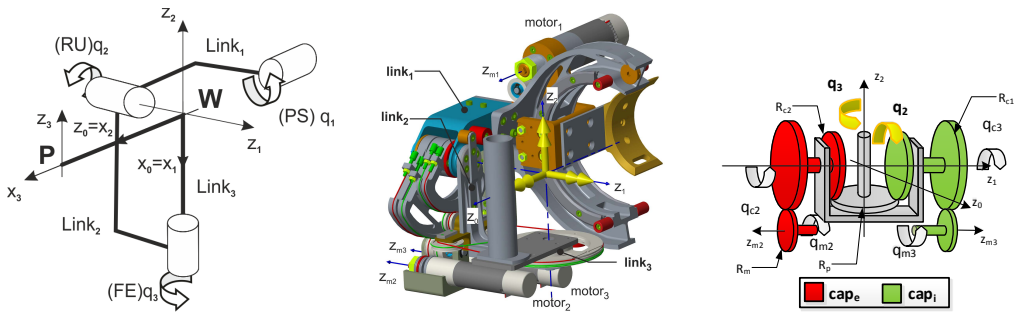
For successful manipulation, the wrist movement is important, as it determines the orientation of the hand. Consequently, wrist motions are realized by an active wrist exoskeleton [4]. It covers the three rotational DoF of the wrist and provides respective torque feedback. The covered articulations are: forearm pronation/supination (PS), wrist flexion/extension (FE), and radial/ulnar deviation (RU).

The design is based on serial kinematics as shown in Fig. 11. Again, actuators are located remotely at the non-moving parts and forces are transmitted via metallic tendons to obtain good dynamic behaviors. Main design requirements concern the lightness of the device, its easiness to be worn, and the need to have an inwards open structure to avoid collision with other parts of the telepresence interface during bimanual teleoperation tasks. The PS joint has been designed to improve the wearability of the wrist device by using an open curvilinear rail and rolling slider solution. FE and RU motions are jointly transmitted by two parallel actuators using a differential transmission as shown in Fig. 11c. A passive regulation of the handle position along the PS axis allows to adapt the last link length to the user's hand size. The total weight of the wrist exoskeleton is 2.9 kg. Fig. 12 shows the wrist exoskeleton worn by an operator.

4.3 Hand Exoskeleton

Finally, a lightweight, underactuated hand exoskeleton was designed for the teleoperation of hand motions, providing independent finger motion tracking with force feedback [33] [11]. Each finger is connected to an individual underactuated parallel kinematic in two points, which can adapt to the individual finger length as shown in Fig. 13. Force feedback is applied to each of the five fingers by an individual actuator.

The five parallel kinematic chains are mounted on a ground link which is rigidly connected with the back of the users hand, as shown in Fig. 13. The ground link shape is modeled to follow the shape of the back of a human hand while a layer of foam between the hand surface and the ground link improves the adaptability of the system to different hands and increases



(a) (b) (c)

Figure 11: Wrist exoskeleton design: (a) kinematic scheme, (b) CAD model, (c) schematic representation of the FE/RU differential transmission.

Figure 12: The wrist exoskeleton worn by an operator. Left: Isolated test. Right: Mounted on the arm exoskeleton.

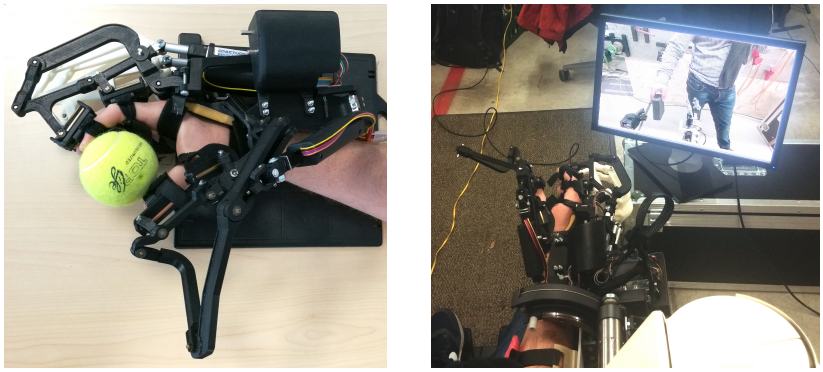


Figure 13: The hand exoskeleton worn by an operator.

comfort. The hand exoskeleton has been realized with 3D printed parts, resulting in a low-cost and lightweight device which only weighs about 350 g. For the actuation, *Firgelli L16* linear motors with 50 mm stroke have been used. Moreover, the lack of magnetic hardware, the small case and the low weight of the actuators allow the motors to be placed close to each other to fit all the actuators on top of the hand. For sensing, potentiometers of the type *Bourns 3382G-1-103G* are utilized in addition to the potentiometers that are integrated in the actuators.

The utilized linear motors include a mechanical gearbox which prohibits backdriveability. However, backdriveability is fundamental to allow the operator to move its fingers. Consequently, force sensors are positioned between the ground link and the actuators. Those sensors measure the intentional forces applied by the user's fingers and a control algorithm is utilized to let the actuators follow the intended finger motion.

5 Environment Perception

The chosen sensor setup generates data of several types: while some sensor measurements, such as camera images, can be directly shown to the operators, other data must be processed. The results serve as more intuitive visualizations or as input for some of the autonomous control functions.

5.1 Ground Contact Estimation

When navigating in challenging terrain, it is helpful to detect if a foot has ground contact. Such ground contact estimation enables reliable semi-autonomous stepping with few additional required environment knowledge. Inputs are joint torques of the legs which are provided by the actuators. Via forward dynamics, a 6D force vector is computed for each foot. If, after gravity compensation, the foot exerts a force onto the ground, contact is detected.

5.2 Camera Calibration

On a mobile robot with limited power and computational resources, it is desirable to have a minimal number of cameras and minimal viewing overlap between adjacent cameras to avoid redundant image processing. However, a small or non-existent overlap makes the calibration of relative (camera-to-camera) extrinsic transforms difficult. A simple solution is to temporarily insert additional cameras during the calibration procedure, with the aim to increase the camera-to-camera overlap in the field-of-view. Due to the transitivity of the relative camera poses, the intermediary cameras can be removed once the calibration is complete. Specifically, let P_{ji}, denote the relative transform from the coordinate frame of camera i to that of camera j. Given a pair of transforms P_{ji} and P_{kj} between three cameras i,j, and k, it is trivial to factor out camera j, as $P_{ki} = P_{kj}P_{ji}$.

5.3 Laser-based 3D Mapping and Localization

Laser range measurements from the *Velodyne PUCK* 3D rotating laser scanner with spherical field-of-view are aggregated to a dense 3D map of the environment. We use our local multiresolution surfel grid approach for this [9]. The laser scanner provides about 300,000 range measurements per second with a maximum range of 100 m. It is rotated at 0.1 rotations per second, resulting in a dense omnidirectional 3D scan per halve rotation. Slower rotation is possible if a higher angular resolution is desired. For our current setup, we acquire one full 3D

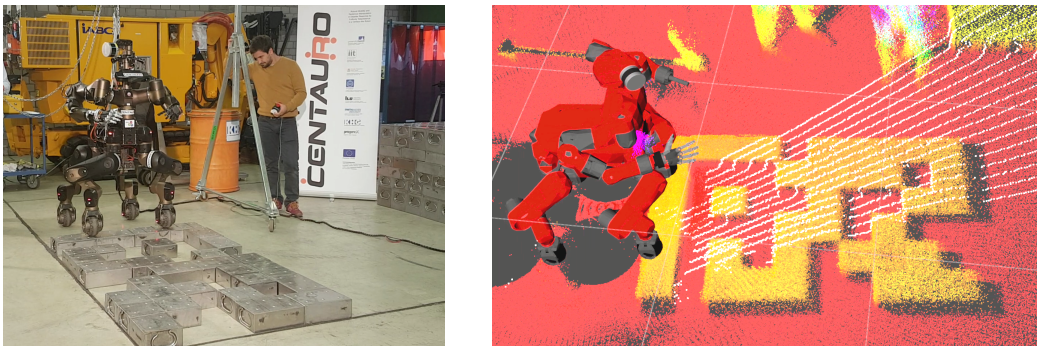


Figure 14: Centauro overcoming a step field: Scenario (left), localized robot and registered point cloud color coded by height (right).

scan every 5 seconds and compensate for sensor motion during acquisition by incorporating measurements of the IMU.

3D scans are aggregated in a robot-centric local multiresolution map by registering consecutive scans to a denser egocentric map. The resulting egocentric maps from different view poses form nodes in a pose graph to allow for allocentric mapping of the environment. They are connected by edges representing spatial constraints, which result from aligning these maps with each other. The global registration error is minimized using graph optimization. The resulting 3D map allows for localizing the robot in an allocentric frame. Fig. 14 shows an example of a generated point cloud and a localized robot.

5.4 Terrain Classification

In order to provide a general description of the terrain traversability considering both semantic and geometric characteristics, a 2D terrain class map with the labels *safe*, *risky*, and *obstacle* is generated from RGB images and registered laser scanner point clouds. A patch of terrain is considered *safe* if the robot can traverse it easily without encountering problems such as wheels that are getting stuck. An *obstacle* label is reserved for areas that pose a security threat to the robot or where the robot must take care of its surroundings, for example tall vegetation or humans, respectively. A *risky* patch is defined as neither of the other two and typically corresponds to areas where caution is needed like low vegetation, snow, or small debris.

The terrain classification algorithm relies on two complementary processing steps, as illustrated in Fig. 15 and Fig. 16. Geometric features such as height, slope and roughness are extracted from the registered point clouds. In parallel, a fully convolutional network is trained to perform pixel-wise semantic segmentation from RGB images.

Finally, the geometric features and pixelwise semantic predictions are fused into the three terrain classes using a random forest classifier. Some examples for classified terrains can be seen in Fig. 17. Please refer to [34] for further details.

5.5 Object Segmentation

To enable autonomous or semi-autonomous manipulation of the workspace, the Centauro robot has to detect and determine the pose of useful objects in its environment, for example tools. For the purpose of object detection and semantic segmentation, we train CNN-based models on a dataset of tools. In the CENTAURO project, two alternative detection and segmentation pipelines have been developed, with focus on either speed or precision. Both approaches use

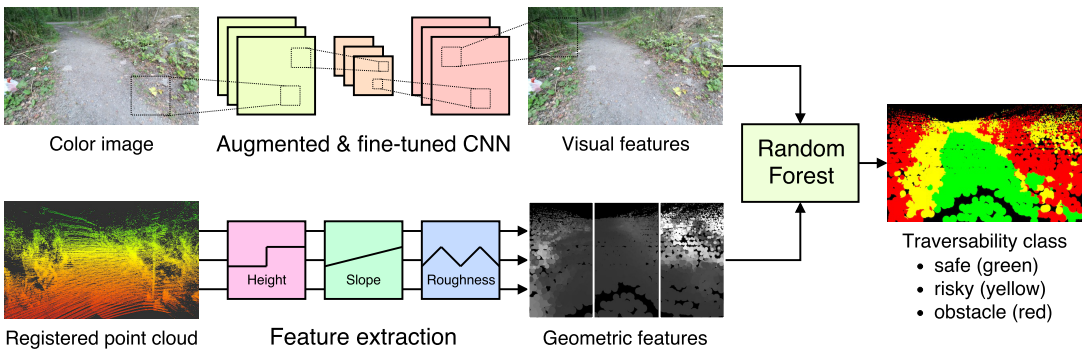


Figure 15: The terrain classification algorithm consists of two complementary processing pipelines. Visual features are extracted from color images using a fully convolutional network trained with data augmentation and fine-tuned to our target environment. Geometric features are computed efficiently from registered point clouds. The system generates fast and accurate traversability estimates in novel environments by training a random forest on small amounts of labeled data.

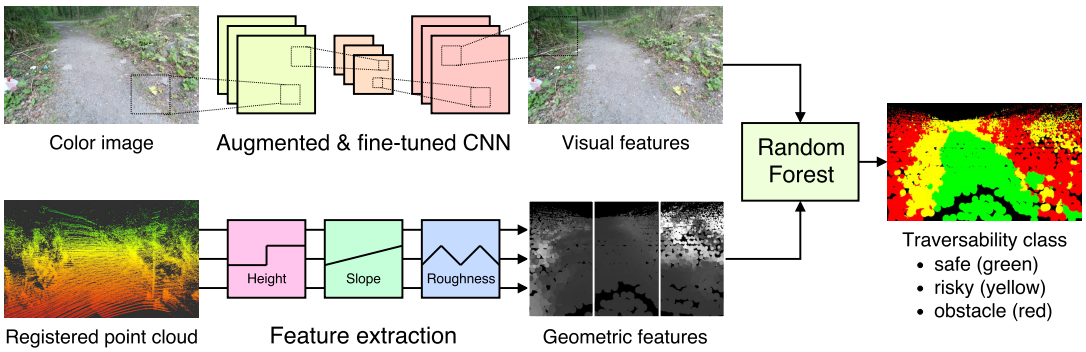


Figure 16: The terrain classification algorithm consists of two complementary processing pipelines. Visual features are extracted from color images using a fully convolutional network trained with data augmentation and fine-tuned to our target environment. Geometric features are computed efficiently from registered point clouds. The system generates fast and accurate traversability estimates in novel environments by training a random forest on small amounts of labeled data.

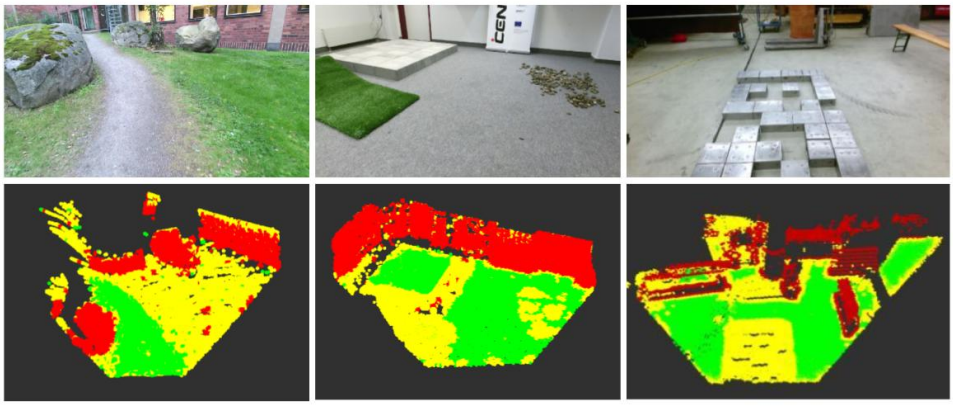


Figure 17: Examples for classified terrain: outdoor scene (left), indoor scene with gras, rubble and an elevated platform (center), step field (right). green = *safe*, yellow = *risky*, red = *obstacle*.

Figure 18: Turntable capture and scene synthesis. Left: Different drills on the turntable as captured by a DSLR camera. Right: Synthetic training scene generated by inserting new objects. The top image shows the resulting color image, the bottom shows generated ground truth for training the segmentation.

Kinect V2 RGB-D data as input. An multi-frequency phase unwrapping algorithm improves the depth estimates [19].

YOLO with Tabletop Segmentation The first alternative is based on the YOLO object detector [27]. It outputs rectangles which also contain background pixels. If we assume that the objects are placed on planar surfaces, which constitutes most of the background, most of it can be removed. This can be done by first estimating planes in the depth image and then remove all points belonging to the significantly supported planes. After this step, in the ideal case, the remaining points belong to the objects.

Semantic Segmentation A more integrated pipeline was developed using semantic segmentation, which directly produces pixel- (or point-)wise class labels. Utilizing the recent RefineNet [21] architecture, we create a representation of the input image with both highly semantic information and high spatial resolution. Deep learning methods require large amounts of training data. We address this problem by generating new training scenes using data captured from a turntable setup. Automatically extracted object segments are inserted into precaptured scenes, as shown in Fig. 18. For details on the capturing and scene synthesis pipeline, we refer to [36].

5.6 Pose Estimation

To facilitate robust autonomous grasping of objects, we need their 6D pose. To this end, we developed a 5D pose predictor network for efficient pose estimation on a rough scale, and a 6D registration method for fine-grained pose estimation. Both methods can be used separately or in conjunction, depending on the scenario. Again, RGB-D data from the *Kinect V2* serves as input.

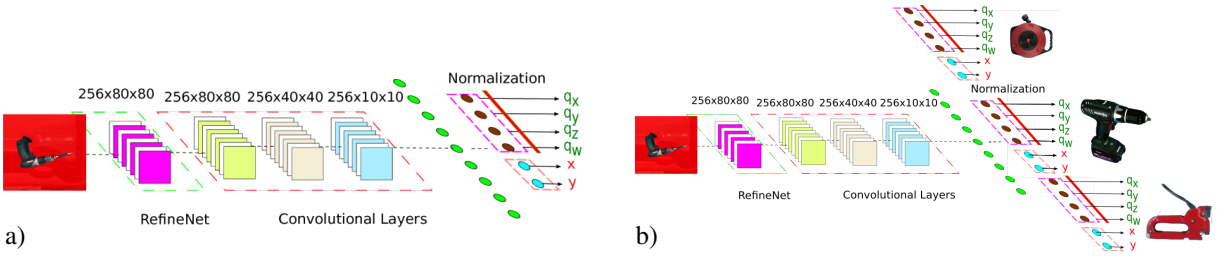


Figure 19: Pose estimation network architecture. a) Single-block output variant; b) Multi-block output.

5.6.1 Pose Estimation Network

For efficient pose predicting, we augmented the semantic segmentation pipeline (Section 5.5) with an additional CNN to estimate the 5D object pose (rotational, and x and y of translation) from the RGB crops from the scene. Those crops are extracted from the bounding boxes of detected object contours. To encode the segmentation results, pixels classified as non-object are pushed towards red (see Fig. 19). This representation allows the network to focus on the specific object of which the pose should be determined. The pretrained RefineNet network from Section 5.5 is used to extract features. To generate the ground truth poses for network training, the data acquisition pipeline described in [36] is extended to record turntable poses automatically and fuse captures with different object poses or different objects (see Fig. 18) with minimal user input.

We developed two different types of CNN architectures shown in Fig. 19. The single-block output variant predicts six values (rotation represented as a unit quaternion and x and y of translation), whereas the multi-block output variant predicts these for each object category. Our evaluation showed that the single-block variant performed slightly better in the presence of occlusion, and was thus chosen for use in the Centauro system. The predicted 5D pose can be projected into 6D using the mean depth measurement d in a window around the object center.

5.6.2 Feature-based Point Cloud Registration

For fine-grained pose estimation, we employ the work of [7] to register detected object point clouds to models of objects with a known relative pose to the robot sensor head. Descriptive high-dimensional features are integrated to a probabilistic framework for point cloud registration. We use the Fast Point Feature Histograms [FPFH, 32], due to its discriminative power and invariance to rigid transformations. However, other types of invariant features can also be employed in the framework. In order to incorporate the high-dimensional histogram representation into the probabilistic framework, the features are clustered using K-means. In this way, each histogram is labeled by the corresponding cluster index, and the observed feature of a specific point is given as the index of its histogram vector. Finally, this representation is used in an Expectation Maximization framework, where the likelihood of a categorical distribution over all observed features and a Gaussian mixture model of the spatial distribution are maximized. Further, EM also optimizes the unknown rigid transformation parameters, which we use to determine the world frame 6D pose of the detected objects.

6 Operator Interfaces

Although the considered disaster-response environments are too dangerous for humans to work in, the human capabilities of situation assessment, mission planning, and task experience are key to a successful search and rescue mission. The operator interfaces enable the operators to transfer these desired capabilities into the scene by giving them an awareness of the situation and enabling them to control the robot. Both require a communication infrastructure, since visual contact is not available.

Regarding locomotion and manipulation control, we aim at enabling the operators to solve as many previously known and unknown typical disaster tasks as possible. Hence, we propose a set of control functions with advantages in different task classes to perform successful teleoperation. A key requirement is to utilize the whole range of the robot's kinematic capabilities while keeping the control itself intuitive and flexible to adapt to unforeseen situations. Different degrees of autonomy are used to fulfill this requirement.

6.1 Communication

For data transmission between the operator station and the robot, we use a wired Ethernet connection or a standard IEEE 802.11ac 5 GHz WiFi link. Depending on the type of data, different protocols are used. The low-latency control and feedback messages for the exoskeleton use raw UDP packets. All other communication takes place using *ROS* topics, which are either directly accessed using *ROS* network transparency, or encoded with FEC for robustness using the `nimbro_network` developed for Momaro [37]. For extending the coverage, a WiFi repeater can be carried by the Centauro robot and dropped at an appropriate location.

6.2 Situation Awareness

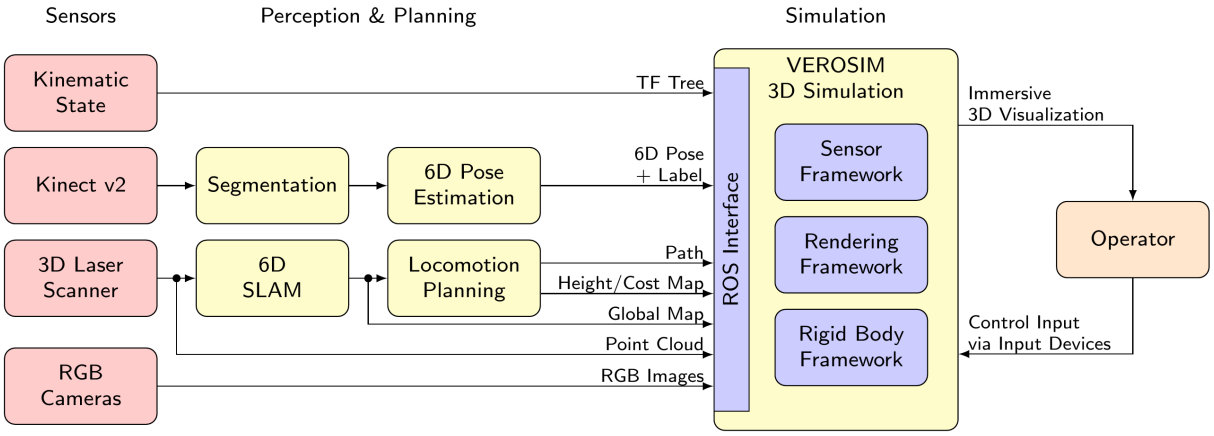


Figure 20: Pipeline for 3D simulation. Sensors are colored red and pipeline components yellow.

A simulation-based approach is used to generate intuitive user interfaces which can be displayed in an *HTC Vive* head-mounted display or on arbitrary regular monitors. Thus, we aim at providing an intuitive, semantically enriched visualization of data for the teleoperation of real robots in dynamic scenarios. Based on a Digital Twin of the real system, we use 3D simulation in-the-loop to visualize the current state of the robot in its environment and add semantic information for the operators, to intuitively interact with the real robotic system. This involves

- the (planar) visualization of incoming data, raw and preprocessed,

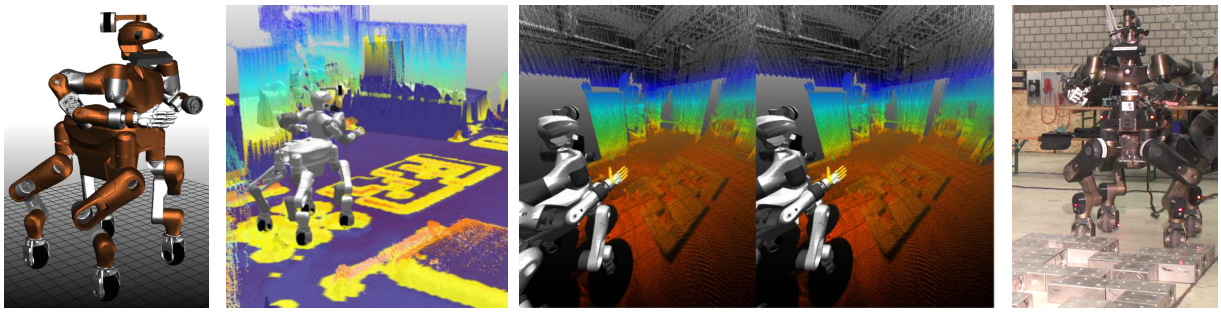


Figure 21: l.: *VEROSIM* provides a Digital Twin of the Centauro robot encompassing all links, joints, sensors etc., c.: third person view (standard and stereoscopic) on the current scene with a rigid body height map overlaid by a cost map accompanied by color-coded point cloud data, r.: image of the visualized scene, Centauro standing in front of a stepping field.

- the projection of data onto the place of occurrence,
- the stereoscopic rendering and thus immersion of the operator, and
- a modular combination and selection of all modalities suited for the given application or situation.

For a holistic incorporation of central modules, as well as interfaces to all system soft- and hardware components, we use *VEROSIM* (Virtual Environments and Robotics Simulation System) with its integrated rendering, sensor, and rigid body frameworks as well as its *ROS* interface (see Fig. 20).

Based on the raw sensor data (robot pose, point clouds and images), the Digital Twin of the robot, as well as perceived environment information can be visualized in simulation (see Fig. 21 l.,c.). Besides the possibility to visualize raw data streams, the 3D simulation can also use preprocessed data from all software modules to generate dynamic environments based on the robot's perception. Based on generated point clouds, rigid body height maps can be created and cost map visualizations can be overlaid (see Fig. 21 c.). Relying on the semantic detection of objects with a given name and pose, the simulation can add the dimension of space to the data by projecting it onto the place of occurrence. Consequently, a template-based model insertion can be used to insert a 3D model of the recognized object into the 3D scene which overcomes point cloud gaps (see Fig. 22 b.l.). Head-up display visualization of data can help each operators to create unique, personalized and individually optimized user interfaces [see Fig. 22, 6].

Stereoscopic rendering encompasses all of the aforementioned aspects. Incorporating the *SteamVR* API into *VEROSIM* enables the utilization of software modalities such as using stereoscopic headsets (see Fig. 21 t.r.). This enables an operator wearing a head-mounted display to see through the robots eyes, increasing the immersion and the embodiment in simulation. In addition, it is also possible to optimize the operator's view by either physically walking around in the scene or by moving the camera pose through the 3D scene via computer mouse or to predefined poses in the simulator (see Fig. 22 r.). This leads to a wide set of opportunities to individually change and reposition the view on the scene for each task and to generally mediate and support robotic teleoperation via 3D simulation (see Fig. 22 c.). Further details can be found in [5].

Although this optimized user interface in terms of visualization is the main focus in this contribution, the use of a 3D simulation backend comes along with additional opportunities. During the development process of such complex systems, the simulator is used for most of the system components. The implemented *ROS* interface allows for the combination of internal and external frameworks and the use of the simulation framework as the central integration tool for

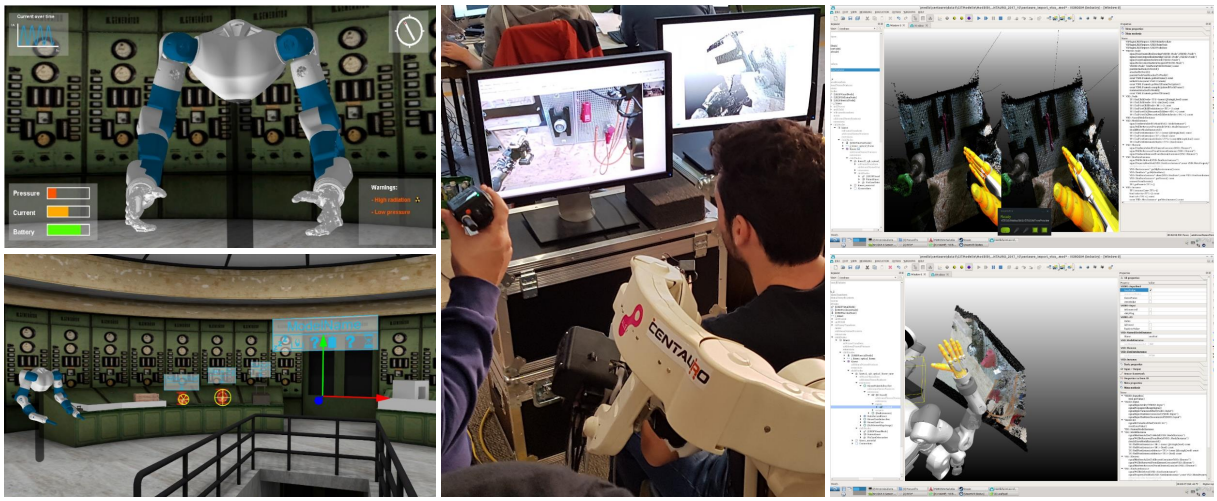


Figure 22: Overview of visualization possibilities: t.l.: head-up display visualization of internal parameters, b.l.: projection of semantic object information via billboard visualizations, c.: first person operator using the exoskeleton with direct camera view and point cloud visualizations, t.r.: stereoscopic and b.r.: third person view on the scene during a manipulation task.

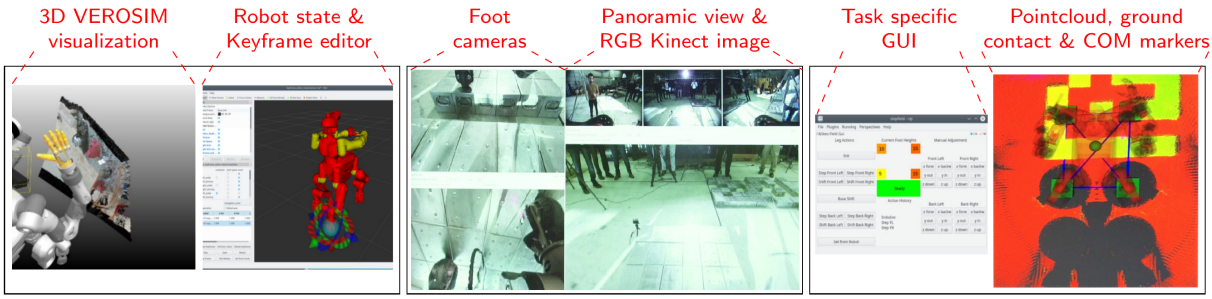


Figure 23: Environment and robot state visualization for the support operators.

all system components. Moreover, it is possible to use the Digital Twin to test actions before they are executed by the real system and to predict action results. Additional optimization, prediction and logging can either be accessed during or post operation to analyze the overall system performance [2, 1].

Besides the main operator sitting in the telepresence suit, there are additional operators who

- can control the robot via control interfaces other than the telepresence suit,
- provide the required visualizations for the operator in the telepresence suit, and
- are responsible for keeping an overview over the whole situation while the operator in the telepresence suit might focus on task details.

Hence, those operators need a wide range of visualizations. We provide these by displaying processed and unprocessed data from several sensors on multiple monitors, as shown in Fig. 23. A panoramic view from the robot head perspective is helpful for general scene understanding. In addition, images from the two RGB cameras under the robot base are arranged to give a detailed assessment for the terrain under the robot base which was key to a save stepping locomotion operation (Fig. 23). Moreover, ground contact for each foot is visualized by correspondent markers. Together with the visualized robot center of mass (CoM), this helped assessing robot stability. 3D *VEROSIM* visualizations further increased the scene understanding, especially in manipulation tasks where occluded areas are compensated by the simulation-based approach. Several control GUIs were developed for task-specific robot control.

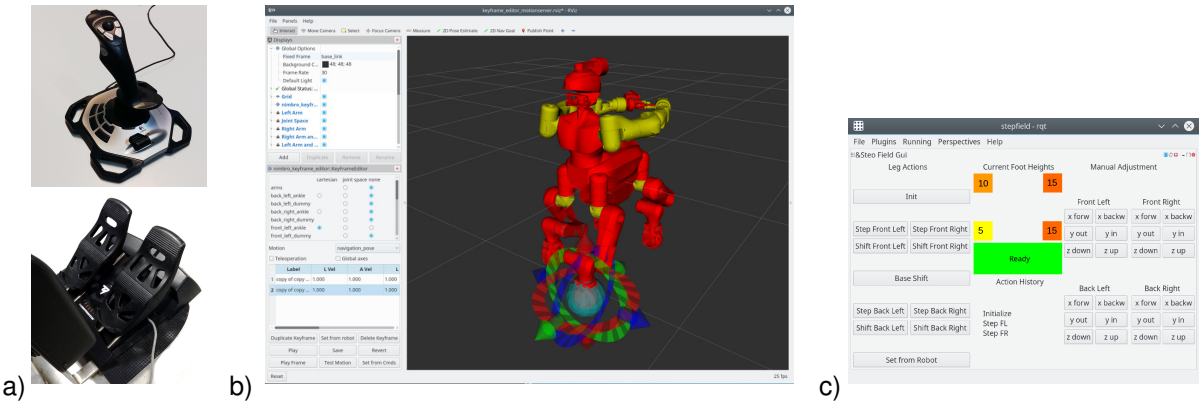


Figure 24: Locomotion control interfaces: a) 4D joystick and 3D pedal controller, b) keyframe editor, c) semi-autonomous stepping controller GUI.

6.3 Locomotion Control

The Centauro lower body provides a wide range of locomotion capabilities which require suitable interfaces to be controlled efficiently. Intuitive interfaces for omnidirectional driving control are a joystick and a pedal controller (Section 6.3.1). Leg movements can be controlled by a keyframe editor (Section 6.3.2). A higher degree of autonomy is reached by utilizing a semi-autonomous stepping controller (Section 6.3.3). Finally, we developed a hybrid driving-stepping locomotion planner which autonomously plans and executes locomotion to a goal pose specified by an operator using *VEROSIM*, as described in Section 7.

6.3.1 4D Joystick & 3D Pedal Controller

Omnidirectional driving can be controlled by a joystick with four axis. Robot base velocity components v_x, v_y and v_θ are mapped to the three corresponding joystick axis. Foot specific velocities and orientations are derived from this robot base velocity as described in Section 7.4. The joystick throttle controller jointly scales all three velocity components.

The same three robot base velocity components are also mapped to a 3-axis *Thrustmaster TFRP Rudder* pedal controller. In addition to individual throttles for each of the two pedals, it also allows for changing the pedal's longitudinal position relative to each other which results in a third DoF. The pedal controller is less intuitive than the joystick but can be used by the teleoperator in the telepresence suit. Both devices are shown in Fig. 24a.

6.3.2 Keyframe Editor

Leg motions can be controlled by a keyframe editor [35]. It allows for the control of joint groups (e.g., the front left leg) in joint space or Cartesian end-effector space online during the mission or through predefined keyframes. Keyframes can also be sequenced to motions. The GUI provides a graphical interface in which joints and end-effectors can be configured using interactive markers and the computer mouse (Fig. 24b). A numerical configuration is also possible. Finally, robot motions are generated by interpolating between given keyframes.

6.3.3 Semi-autonomous Stepping Controller

To efficiently control stepping locomotion in irregular terrain, we developed a semi-autonomous controller. It provides a set of stepping and stepping-related motions that can be triggered by

the operator. The available motions are: step with a chosen foot, drive a chosen foot forward, and shift the robot base forward. If a stepping motion is triggered, the robot shifts its base longitudinally and laterally and rolls around its longitudinal axis to establish a stable stepping pose. The stepping foot is then lifted, extended by a given length and lowered. The latter stops as soon as ground contact is detected (see Section 5.1). Hence, the robot automatically adapts to the ground structure. Motions are represented as sequences of keyframes, as described for Section 6.3.2.

The controller is operated through an intuitive GUI which provides buttons to trigger the described motions for an individual foot (Fig. 24 c). Additional buttons allow the operator manual foot movement in Cartesian space, which is helpful to apply minor corrections. Furthermore, the GUI visualizes detected terrain heights under the individual feet and a history of the last triggered motions which helps to follow certain motion orders.

6.3.4 Motion Execution

Motions from any source need to be transformed to joint space trajectories to be executable by the robot. We use a keyframe interpolation method which was originally developed for Momaro [37]. The interpolation system generates smooth joint trajectories obeying velocity and acceleration constraints set per keyframe. Input are keyframes consisting of joint space or 6D Euclidean space poses for each of the robot limbs.

6.4 Manipulation Control

The Centauro system possesses several degrees of autonomy to control manipulation. The upper body can be controlled in joint or Cartesian space or by executing keyframe motions with the keyframe editor which is described in Section 6.3.2. Furthermore, manipulation can be controlled via the upper-body exoskeleton which is highly intuitive since it mimics the operator's behavior and provides force feedback (Section 6.4.1). Moreover, precise control of the wrist pose is provided by a 6D input device (Section 6.4.2). Finally, we propose an autonomous grasping functionality which is described in Section 8.

6.4.1 Telemanipulation by the Upper-body Exoskeleton

The full-body telepresence station can be used to control the robot through an immersive teleoperation system with force feedback. By means of the bilateral and full upper limb exoskeleton, the operator directly controls the pose and the applied forces of the upper limbs of the robot. Different teleoperation control architectures have been implemented for the arm-wrist segments of the exoskeleton and for the hand grasping.

Regarding teleoperation of the proximal segments of the upper limb, the operator's arm and wrist pose are measured by the correspondent parts of the exoskeleton and a 6D end-effector pose is computed. This pose is then transferred to the robot and applied to the arm using inverse kinematics based on the *OpenSoT* framework [12]. A direct mapping of joint configurations from the teleoperator to the robot is not helpful since their kinematic structures are not identical. Force feedback is generated by transferring measured 6D force-torque vectors from the force-torque sensor between right arm and Schunk hand to the exoskeleton and mapping them to joint torques using inverse dynamics. Fig. 25 shows an operator using the upper-body exoskeleton to open a door.

For the teleoperation of grasping motions, the right hand exoskeleton is connected with the anthropomorphic Schunk hand mounted at the right robot arm. Again, a direct joint mapping

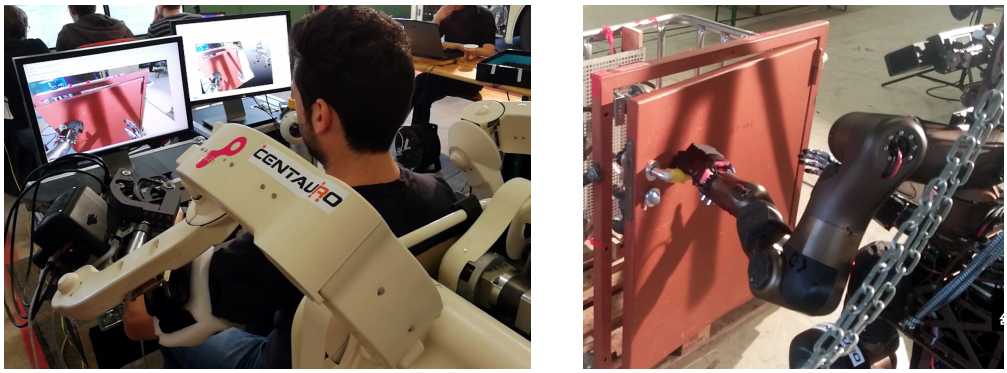


Figure 25: An operator opening a door teleoperating Centauro with the upper-body exoskeleton.

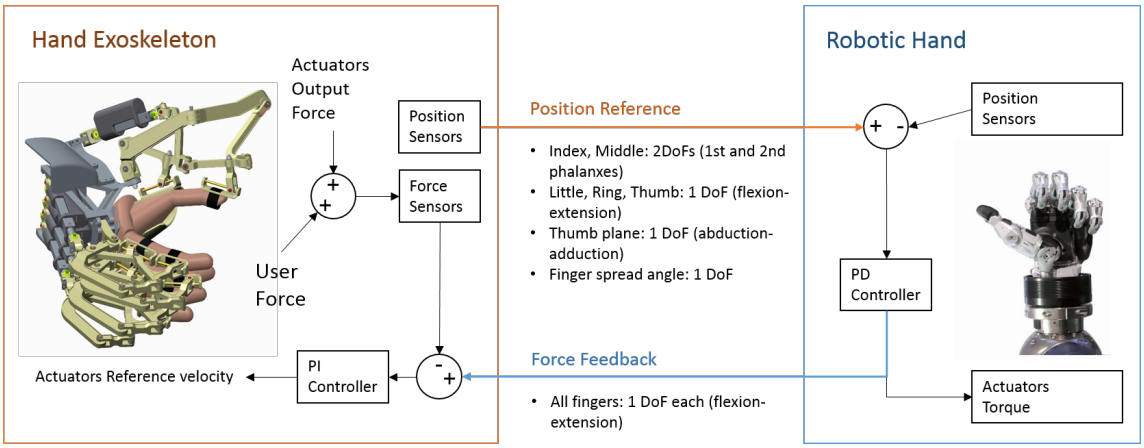


Figure 26: Teleoperation scheme between the Hand Exoskeleton and the Schunk robotic hand.

of control commands and force feedback is not possible due to differences in the kinematic concept, variable operator hand sizes and underactuation for both the hand exoskeleton and the Schunk hand. To transfer finger and hand motions and applied forces from the operator's hand to the robot hand, we utilize the local admittance control of the hand exoskeleton and the local impedance control embedded in the Schunk hand. The hand exoskeleton estimates angular position references of the operators hand. Those are sent to the proportional-derivative (PD) position control system of the Schunk hand. The embedded PD control converts position references into actuator torques proportional to the position error. Hence, once in contact with an object, the operator can increase grasping forces by further closing his or her fingers.

The mapping of positions and forces differs between fingers. Thumb, ring finger and pinkie are operated on the basis of a single DoF each on the operator and robot side, estimating the overall closing/opening of the finger. Index and middle finger are position controlled by the operator using two DoF, matching the rotation of the first and second phalanxes of each finger. The Schunk hand provides this additional DoF for those two fingers which results in a higher grasping precision. In addition, the spread between the four long fingers and the rotation of the thumb's opening/closing plane (abduction/adduction) is controlled.

To provide force feedback, torques applied by the Schunk hand are sent to the local admittance control of the hand exoskeleton and applied to the operator's hand as force feedback. The mapping in this direction is different since the exoskeleton provides less actuators than sensors. Each finger receives a single DoF feedback force which is applied through the adaptive and underactuated mechanism of the hand exoskeleton. For the finger spread, force feedback is not applicable since no correspondent actuator exists in the hand exoskeleton. Fig. 26 visualizes the

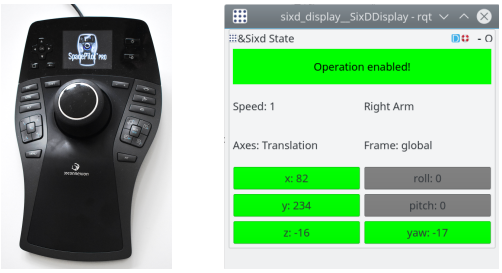


Figure 27: 6D input device (l.) and corresponding GUI (r.) for dexterous wrist control.

control architecture between hand exoskeleton and Schunk hand.

To control the 1-DoF SoftHand at the left robot arm, the left side of the exoskeleton is equipped with a lever-shaped 1-DoF grasping controller instead of a hand exoskeleton. Its position is mapped to the SoftHand actuator and the measured force at the robot hand is transferred to the controller as force feedback.

6.4.2 6D Input Device for Wrist Control

A *3DConnexion SpacePilot Pro* 6D input device and a corresponding GUI provide a teleoperation interface for dexterous wrist control (see Fig. 27). The developed interface establishes the connection between the device and the motion player mentioned in Section 6.3.4. The *SpacePilot* movement is streamed as a desired 6D end-effector pose to the motion player which interpolates from the current to the desired pose and executes the motion. The GUI can be used to easily adjust the following control parameters: *End-effector* (a wrist for arm control or an ankle for leg control), *Reference frame* (end-effector frame, robot base frame, or a custom frame), *Enabled axes* (each translational and rotational axis can be enabled/disabled so the user input on this axis is considered/ignored), and *End-effector speed*. All control parameter can also be changed using buttons on the input device. This interface is well suited for manipulation tasks where very precise arm movement along certain axes is required (e.g., moving the arm along a plane surface, or turning an object around a specified axis).

7 Hybrid Locomotion

Autonomous locomotion planning and execution is a promising approach to increase locomotion speed and safety, to lower the operator's cognitive load, and to keep the flexibility to control the robot in unknown tasks and unforeseen situations. The only required operator input is a desired robot goal state. The locomotion planning pipeline is visualized in Fig. 28. Laser scanner point clouds and terrain class maps are processed to cost maps that represent the environment (Section 7.1). A search-based planning approach uses this environment representation to generate plans (Section 7.2). Due to the unique hardware design and the many DoF of the robot platform, this planning approach is novel since it merges planning approaches for both driving-based and stepping-based locomotion and is capable of handling the respective complexity. To handle planning queries for large environments, the approach is extended to plan on multiple levels of abstraction (Section 7.3). Finally, a controller executes these paths (Section 7.4).

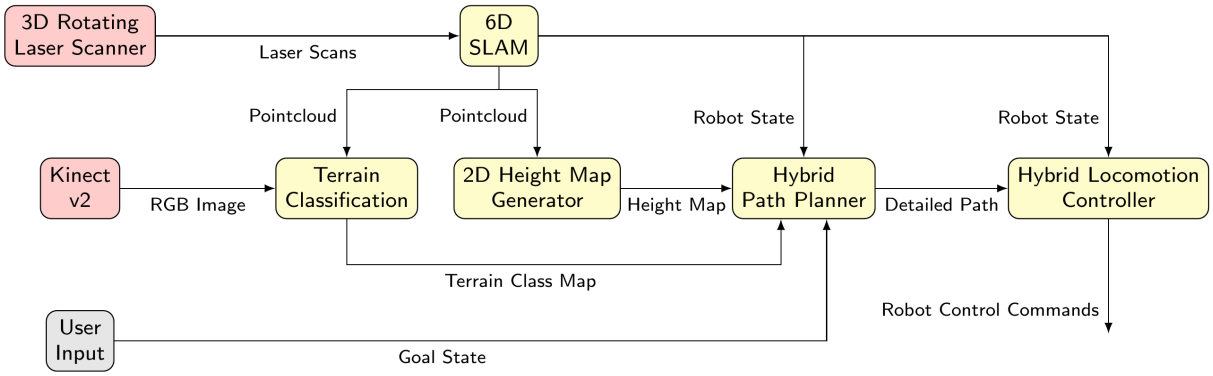


Figure 28: Overview of the pipeline for locomotion planning. Sensors are colored red, pipeline components yellow, and other inputs are colored grey.

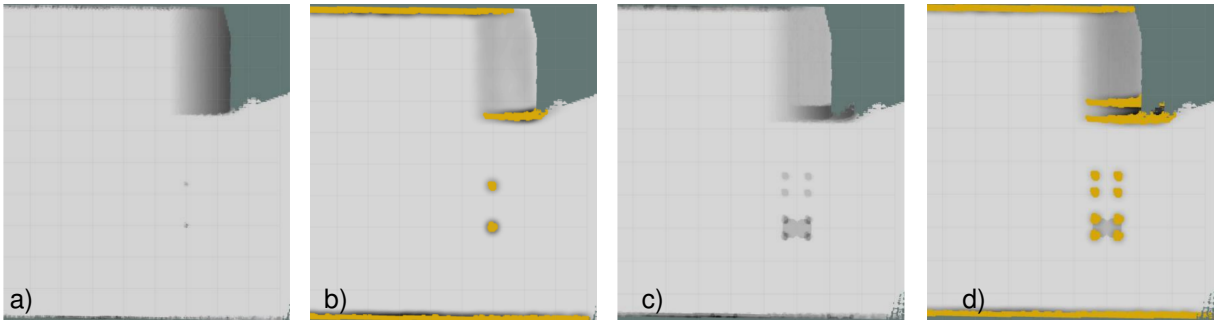


Figure 29: Environment representation: The map shows two walls, a ramp and two poles of different height. a) Heights, b) Foot costs, c) Body costs, d) Pose costs.

7.1 Environment Representation

A 2D height map is generated from the registered point clouds and serves as the environment representation. Height values are processed to foot and body costs. Foot costs describe the cost to place an individual foot at a given position in the map. They include information about the terrain surface and obstacles in the vicinity. Besides the costs based on height values, foot costs also include the provided terrain classes. Body costs describe the costs to place the robot base in a given configuration in the map. They include information about obstacles and the terrain slope under the robot. Foot costs of all four feet and body costs are combined to pose costs which describe the costs to place the whole robot in a given configuration on the map (see Fig. 29).

7.2 Search-based Planning

Path planning is done by a search-based approach on these pose costs. The used algorithm is Anytime Repairing A* [20]. Neighbor states are generated online during the planning. They include omnidirectional driving (see Fig. 30 l.) and stepping motions (see Fig. 30 r.). Since driving shall be the preferred locomotion mode, stepping motions are only considered if several criteria are fulfilled. During path search, steps are represented as abstract manoeuvres—the direct transition from a pre-stepping pose to an after-stepping pose. Robot stability and detailed motion sequences are explicitly not considered in this planning layer.

The resulting path is expanded to a motion sequence which can be executed by the robot. During this path expansion, abstract steps are transformed into stable motion sequences. Stable stepping poses are established through roll motions, foot shifts and longitudinal base shifts.

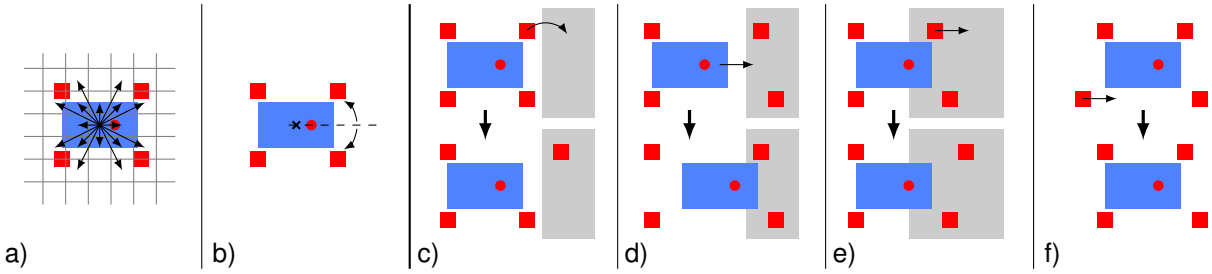


Figure 30: Neighbor states can be reached by driving or stepping related motions: a) Omnidirectional driving with fixed orientation, b) Turning on the spot to the next discrete orientation, c) Abstract step, d) Longitudinal base shift, e) Shift a single foot forward, f) Shift a foot back to its neutral position.

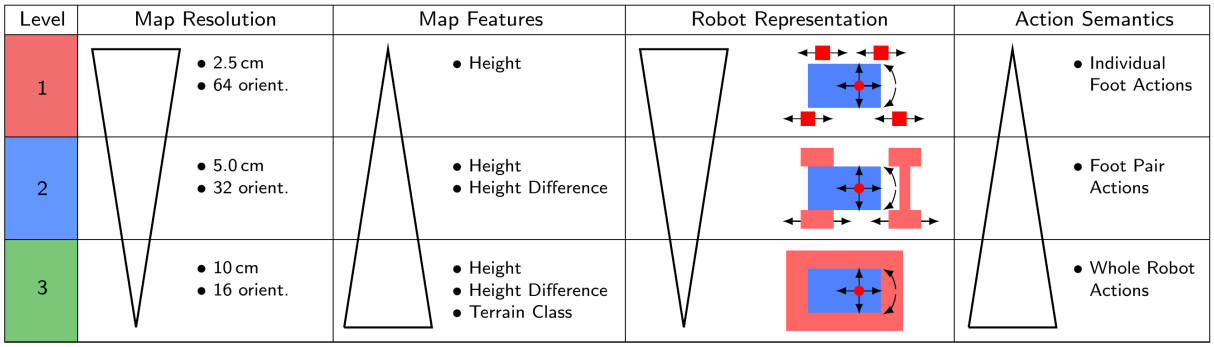


Figure 31: The planning representation is split into three levels of abstraction. Coarser representations are enriched by additional semantics to compensate the loss of information, caused by abstraction.

Stability computation is limited to static stability since motion execution is sufficiently slow and thus, dynamic effects can be neglected. In addition, leg length information is generated for each pose in the resulting path. Further details about the approach can be found in [17].

7.3 Planning on Multiple Levels of Abstraction

The search-based planning approach is extended to plan on multiple levels of abstraction, as illustrated in Fig. 31, which allows for planning for significantly longer path queries while the result quality stays comparable A detailed planning representation is only utilized in the vicinity of the robot. This representation is called *Level 1* representation and is described in Section 7.2. With increasing distance from the robot, the planning representation becomes more abstract. This is achieved by representing the environment and possible actions in a coarser resolution and by using a robot representation with less DoF. The loss of information that comes along with such abstraction is compensated by enriching those representations with additional semantics. The environment is represented with additional features and robot action generation accesses these features. All three representation levels are unified in a single planner which is able to handle transitions between these levels.

Transition between the different representation levels is realized in a simple manner. As soon as a robot action in a given representation level leaves the represented area, planning of this specific action is repeated in the next higher representation level. Moreover, path segments can be refined from a coarse representation to a finer representation, if this is available. This

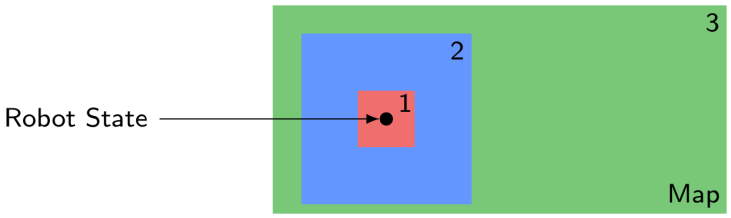


Figure 32: Representation level positioning. A fine planning representation is only provided in the vicinity of the robot. With increasing distance from the robot, the representation becomes more abstract.

allows for continuous refinement during path execution: As the robot moves, the represented area of each representation level moves with the robot. Consequently, respective path segments can be refined. This allows for the availability of a detailed path representation in the vicinity of the robot without the need of continuously replanning the whole path. Finally, we utilized a heuristic for the planner which is based on precomputed costs in the *Level 3* representation. Further detail can be found in [18].

7.4 Path Execution

Path segments that are represented in Level 1 and which are expanded to detailed motion sequences are executed by a driving-stepping controller. This is split into the control of omnidirectional driving and the control of leg movements.

To control path segments that require omnidirectional driving, a three-dimensional (x, y, θ) B-Spline [8] is laid through the next five robot poses and a pose on this B-Spline is chosen as the controller set value $\mathbf{r}_{\text{sv}} = (r_{x,\text{sv}}, r_{y,\text{sv}}, r_{\theta,\text{sv}})$. Extracting this set value from the B-Spline in some distance from the current robot pose $\mathbf{r} = (r_x, r_y, r_\theta)$ leads to a smoother controller output. A distance of half the B-Spline length yields the desired behavior. The velocity command $\mathbf{v} = (v_x, v_y, v_\theta)$ is computed with

$$\mathbf{v} = (\mathbf{r}_{\text{sv}} - \mathbf{r}) \times k, \tag{1}$$

where k is chosen in a way that the linear velocity component norm $\left\|(v_x, v_y)^{\text{T}}\right\|$ is equal to v_{des}, the desired velocity. v_{des} is 0.1 m/s close to obstacles and 0.25 m/s otherwise. With the given relative foot position $\mathbf{f}^{(i)}$, the individual velocity command

$$\begin{pmatrix} v_x^{(i)} \\ v_y^{(i)} \\ v_z^{(i)} \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ v_\theta \end{pmatrix} \times \mathbf{f}^{(i)} \tag{2}$$

can be computed for each of the four wheels. Before moving with the linear velocity $||(v_x^{(i)}, v_y^{(i)})^{\text{T}}||$ in the desired direction, each wheel needs to rotate to the respective yaw angle $\alpha^{(i)} = \text{atan2}(v_y^{(i)}, v_x^{(i)})$. While driving, the robot continuously adjusts the ankle orientations.

For path segments that require leg movement, actions are expanded to sequences of linear end-effector movements. It is distinguished between movements with wheel rotation (e.g., shift an individual foot on the ground relative to the robot base) and actions without wheel rotation (e.g., longitudinal robot base shift). The transformation from the Cartesian space to joint space is described in Section 6.3.4.

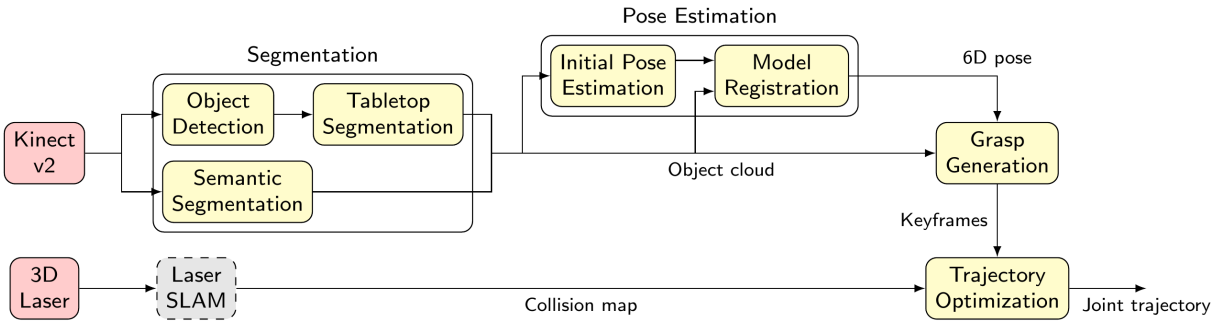


Figure 33: Pipeline for autonomous manipulation. Sensors are colored red, pipeline components yellow, and external modules from other workpackages are colored grey.

8 Autonomous Manipulation

Autonomous manipulation capabilities can help reducing the load on the main operator. For example, repetitive motions such as grasping a tool, opening a door, and placing an object may be automated, allowing the operator to focus on the current higher-level task rather than direct teleoperation of the manipulation actions. The CENTAURO system allows commanding the robot's manipulation capabilities on several levels of autonomy: Starting at low-level direct joint control; over inverse kinematics control with end-effector poses from the exoskeleton, a 6D input device, or 6D markers on the screen; keyframe motions with collision avoidance; and finally autonomous pick-and-place actions triggered by the operator. To realize autonomous manipulation, several components were integrated (see Fig. 33). In addition to the perception components described in Sections 5.5 and 5.6, we will discuss the grasp generation and trajectory optimization modules here.

8.1 Grasp Planning

Objects belonging to a category often exhibit several similarities in their extrinsic geometry. Based on this observation, we transfer grasping skills from known instances to novel instances belonging the same category such as drills, hammers, or screwdrivers. Our approach has two stages: a learning stage and an inference stage.

During learning, we build a category-specific linear model of the deformations that a category of objects can undergo. For that, we firstly define a single canonical instance of the category, and then we calculate the deformations fields relating the canonical instance to all other instances of the category using Coherent Point Drift (CPD) [24]. Next, we find a linear subspace of these deformations fields, which defines the deformation model for the category (Fig. 34).

In the inference stage, we formulate the problem as: given a newly observed instance, search this subspace of deformation fields to find the deformation field which best relates the canonical instance to the novel one. Associated control poses used for grasping defined for the canonical model are also transformed to the observed instance and used for the final grasping motion.

A category is defined as a group of objects with similar extrinsic shape. From the set of objects, we select one to be the canonical model of the class. For each training sample, we find the deformation field that the canonical model has to undergo to transform itself into the training sample. As explained in [30], this transformation can be expressed as:

$$\mathcal{T}_i = \mathbf{C} + \mathbf{G}\mathbf{W}_i, \tag{3}$$

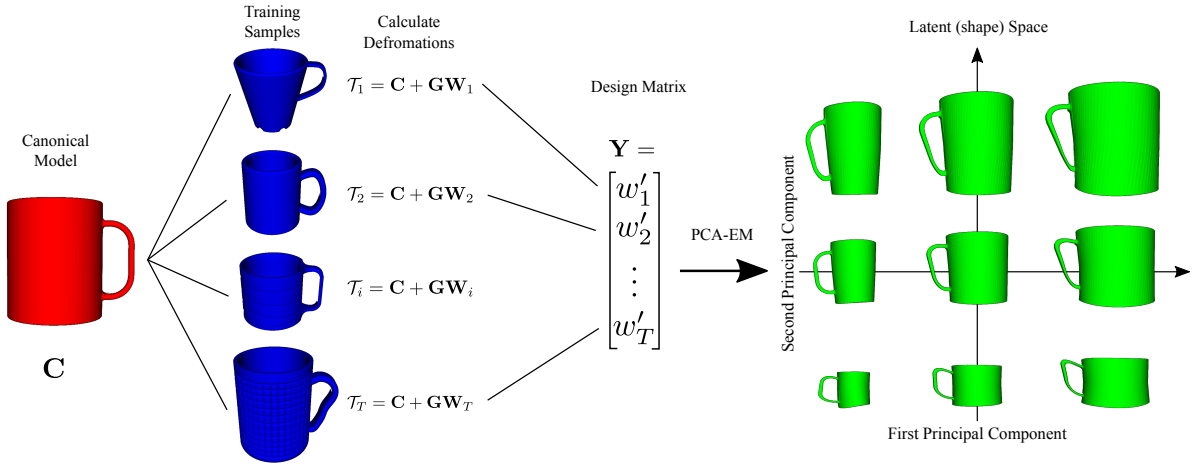


Figure 34: Training phase. Deformations between the canonical model and each instance are calculated by using CPD. The deformations are then assembled into the design matrix $\mathbf{Y}$. Finally, the principal components (latent space) are found by using PCA-EM.

where $\mathbf{C}$ refers to the canonical model, $\mathbf{G}$ is a Gaussian kernel matrix, and $\mathbf{W}_i$ is a matrix of kernel weights.

Because $\mathbf{C}$ and $\mathbf{G}$ remain constant across all training samples, the uniqueness of the deformation field is captured only by $\mathbf{W}_i$. Each of the $\mathbf{W}_i$ matrices contains the same number of elements. This allows us to assemble a $\mathbf{Y}$ design matrix containing all deformation fields as column vectors. Finally, we apply Principle Component Analysis Expectation Maximization (PCA-EM) on the design matrix $\mathbf{Y}$ to find a lower-dimensional manifold of deformation fields for this category.

The transformation between a novel instance and the canonical model is defined by its latent vector plus an additional rigid transformation. The function of the rigid transformation is to reduce the impact of minor misalignments in the pose between the canonical shape and the target shape. In the inference phase, we use gradient descent to simultaneously optimize for pose and shape. In general, we aim for an aligned dense deformation field that when exerted to the canonical shape $\mathbf{C}$ minimizes the following energy function:

$$E(\mathbf{x}, \theta) = -\sum_{m=1}^{M} \log \sum_{n=1}^{N} e^{\frac{1}{2\sigma^2}\|\mathbf{O}_n - \Theta(\mathcal{T}(\mathbf{C}_m, \mathbf{W}_m(\mathbf{x})), \theta)\|^2} \tag{4}$$

with M number of points of the canonical model, N number of points of the observed instance $\mathbf{O}$, $\mathbf{x}$ being the latent vector, and Θ as a function that applies a rigid transformation with parameters θ.

A grasping action is composed of a set of parametrized motion primitives. Poses expressed in the same coordinate system of the shape of the object serve as the parameters of these motion primitives. These poses are defined only for the canonical model. For novel instances, the poses are calculated by warping the poses of the canonical model to the novel instance. We orthonormalize the warped poses because the warping process can violate the orthogonality of the orientation. Fig. 35 shows how the canonical model of a *Drill* category is warped to fit to the observed point cloud (leftmost), the warped grasping poses are also shown. For a complete analysis and discussion about this method, please refer to [30].

The output of the grasp generation pipeline are endeffector poses on the object. These are converted into joint-space trajectories using inverse kinematics and our keyframe-based interpolation system (see Section 6.3.4).

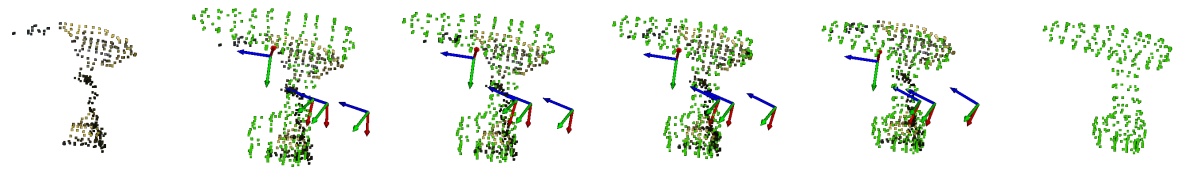


Figure 35: Transferring grasping knowledge to the presented novel instance. The input point cloud is at the leftmost while the inferred shape is at the rightmost.

8.2 Trajectory Optimization

In an unstructured environment there may be obstacles which obstruct a direct way to the final pose, which would render the output of the keyframe interpolation step unusable. An additional optimization step can repair the trajectory. Moreover, by performing optimization of trajectory duration and actuator loads, it is possible to decrease the power consumption, which is of high importance during extended missions. In order to reduce the time of the task completion, it is necessary to perform the planning fast. In this subsection, we describe our approach to trajectory optimization which satisfies these criteria.

Our approach [25] extends Stochastic Trajectory Optimization for Motion Planning (STOMP) [13]. An initial trajectory Θ is the input to this method. This trajectory may be very naïve, for example a straight interpolation between the start and the goal. The method outputs a trajectory, optimized with respect to a cost function. Both trajectories are represented as a set of N keyframes $\boldsymbol{\theta}_i \in \mathbb{R}^J$ in joint space. The initial and goal configurations, as well as the number of keyframes N are fixed during the optimization. In order to gradually minimize the cost, the optimization is performed in iterative manner. The method reliably finds a feasible solution despite the fact that the initial trajectory is far from a valid one.

For collision avoidance, we consider the robot and the unstructured environment. We assume that the robot base does not move during the trajectory execution and that the environment is static. Two signed Euclidean Distance Transforms (EDT) are used for static objects. One of them is used to represent the environment and is precomputed before each optimization task. The other EDT represents the static part of the robot and is computed only if the robot base moves. The moving parts of the robot are approximated by a set of spheres, which allows for performing fast collision checking against EDTs.

In contrast to the original STOMP, our cost function is defined as a sum of costs of *transitions* between the consequent keyframes instead of the keyframes themselves. The cost estimated by each component is normalized to be within $[0, 1]$ interval. This allows to attach a weight $\lambda_j \in [0, 1]$ to each cost component $q_j(.,.)$ in order to introduce a prioritized optimization. An example of two qualitatively different trajectories obtained with two different values of weight λ_{obst} for obstacle costs is shown in Fig. 36.

In order to accelerate motion planning, the optimization is split into two phases. During the first phase, we use a simplified cost function, which only includes obstacle, joint limit, and constraint costs. Optimization with the simplified cost function continues until a valid solution is found. In the second phase, we use the original cost function. This approach allows reducing the overall optimization time by eliminating duration and torque costs as long as no valid solution is found due to obstacles. From our observations, the two-phased approach in most cases did not yield qualitatively different trajectories compared to direct full optimization.

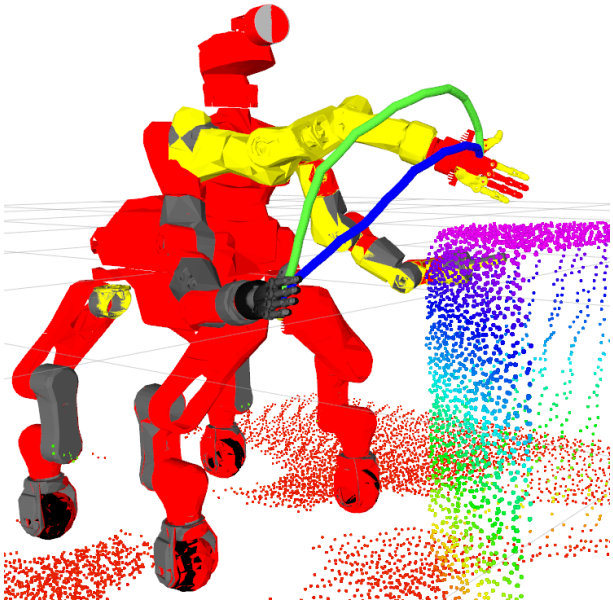


Figure 36: Two qualitatively different trajectories generated by our trajectory optimization: priority on obstacle avoidance (green) and priority on minimizing trajectory duration (blue).

9 Evaluation

We evaluated the Centauro system with dedicated tests at facilities of the Kerntechnische Hilfsdienst GmbH in Karlsruhe, Germany (KHG). Those tests cover a wide range of locomotion and manipulation tasks as occurring in typical disaster-response missions. In addition, we performed multiple tests on component level for different modules. All test documentations and results can be found in detail in *Deliverable D8.3 First Centauro System Evaluation* [38].

10 Lessons Learned & Conclusion

This deliverable report discussed the first integrated Centauro platform introducing the details of the individual core components developed in the project and their interfaces and integration to realize the first version of the Centauro robot. The performance and capabilities of this first integration version were validated during an intensive evaluation week carried out at the premises of Kerntechnische Hilfsdienst GmbH (KHG), Karlsruhe, Germany, which is part of the German nuclear disaster-response organization. KHG operates a variety of remote controlled manipulator vehicles and has deep knowledge of the application domain.

The development, testing and debugging of such advanced and complex new system is a continuous and iterative process that requires significant effort, time and several trials to rigorously evaluate and debug the system capabilities and robustness of the individual components. In this section, we will take a step back and discuss the resulting insights and issues we faced during the first evaluation period.

Starting from the preparation of the integrated system it is evident that such integration is a process that requires continuous interaction between partners and being physically in the same working space for adequate periods is vital requirement to have a successful result. Although this happened to a certain extent and as allowed by the resources available in the project it is obvious that having additional physical integration meeting among the groups of the consortium would have been very beneficial in terms of general robustness as well as functionality.

Concerning the actual hardware platform, the delay on the fabrication of the lower body of the robot generated a handicap on the overall hardware integration that did not leave enough time to perform intensive low level tests and debugging of the hardware and the low-level software and firmware components of the platforms. Consequently, issues related to not adequate cooling for the power unit board of the robot and individual joints resulted in thermal shutdowns that caused delays in the execution of the evaluation tasks.

In addition it was evident that a component that could detect system abnormal system conditions and notify the operator about potential system warnings and faults and their type could assist and make the identification of system issues more efficient.

Another limitation observed was the grasping functionality of the first prototype hand used on the left arm of the robot, making the grasping of some the tools used in the tasks challenging. The left hand design will be changed for the second iteration of the system.

However, apart from the above limitations that caused some delays during the evaluation process, the hardware and low-level software components demonstrated good overall performance permitting the execution of most tasks during the evaluation week, with the exception of the stairs climbing task where the cooling system limitations did not allow to fully execute and achieve this task due to joint thermal issues.

Concerning the system interfaces, it was also very evident that a precise definition of interfaces and interactions between the partner modules was of key importance, but it was difficult to achieve early in the project. In particular, inter-dependencies might not be immediately obvious. For example, a slightly bigger battery than expected resulted in less movement range for the legs.

These inter-dependencies were becoming more obvious as components were becoming more mature, however many of them may not become evident even at the time of the first trials or even require several trials to be investigated. An example for this was the integration of the upper body of the robot and the tele-presence interfaces, especially the tuning of the workspace mapping as well as the rendering of the generated forces and the calibration and testing of the force-torque sensors on the robot arm, which were essential for integration with the exoskeleton.

We also observed that a detailed simulation-based and specification-driven hardware design may not be sufficient if the simulations or specifications are imprecise. In our case, the actuators failed in some cases, such as in complex robot configurations on the staircase. A detailed prediction of occurring forces and moments was not possible since the manoeuvres were not known during the hardware development but were developed in parallel as part of the autonomous planning components.

Finally, testing autonomous functionality in the integrated system has many dependencies. E.g. to test autonomous hybrid driving-stepping locomotion, laser range measurements, point-cloud generation, IMU, inverse kinematics, joint control, and RGB cameras have to work reliably to achieve successful execution.

All the above observations and limitations provided us with inspirations and important recommendations on what we need to improve and consider in the revision and further development activities of the system towards the final integrated Centauro system, which will be evaluated towards the end of the project.

References

[1] L Atorf, T Cichon, and J Roßmann. Flexible data logging, management, and analysis of simulation results of complex systems for eRobotics applications. In *European Simulation and Modelling Conference (ESM)*, 2015.

[2] Linus Atorf, Michael Schluse, and Juergen Rossmann. Simulation-based optimization, reasoning, and control: The eRobotics approach towards intelligent robots. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, 2014.

[3] Lorenzo Baccelliere, Navvab Kashiri, Luca Muratore, Arturo Laurenzi, Małgorzata Kamedula, Alessio Margan, Stefano Cordasco, Jörn Malzahn, and Nikos G Tsagarakis. Development of a human size and strength compliant bi-manual platform for realistic heavy manipulation tasks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5594–5601, 2017.

[4] Domenico Buongiorno, Edoardo Sotgiu, Daniele Leonardis, Simone Marcheschi, Massimiliano Solazzi, and Antonio Frisoli. WRES: a novel 3DoF WRist ExoSkeleton with tendon-driven differential transmission for neuro-rehabilitation and teleoperation. 2018. Accepted for IEEE Robotics and Automation Letters (RAL), `http://www.ais.uni-bonn.de/papers/JFR_Centauro_Buongiorno.pdf`.

[5] Torben Cichon and Jürgen Roßmann. Robotic teleoperation: Mediated and supported by virtual testbeds. In *International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2017.

[6] Torben Cichon and Jürgen Roßmann. Simulation-based user interfaces for digital twins: Pre-, in-, or post-operational analysis and exploration of virtual testbeds. In *European Simulation and Modelling Conference (ESM)*, 2017.

[7] Martin Danelljan, Giulia Meneghetti, Fahad Khan, and Michael Felsberg. Aligning the dissimilar: A probabilistic feature-based point set registration approach. In *International Conference on Pattern Recognition (ICPR)*, 2016.

[8] Carl de Boor. *A Practical Guide to Splines*. Springer, 1978.

[9] David Droeschel, Max Schwarz, and Sven Behnke. Continuous mapping and localization for autonomous navigation in rough terrain using a 3D laser scanner. *Robotics and Autonomous Systems*, 88:104 – 115, 2017.

[10] P. Fankhauser, M. Bloesch, D. Rodriguez, R. Kaestner, M. Hutter, and R. Siegwart. Kinect v2 for mobile robot navigation: Evaluation and modeling. In *International Conference on Advanced Robotics (ICAR)*, pages 388–394, July 2015.

[11] Massimiliano Gabardi, Massimiliano Solazzi, Daniele Leonardis, and Antonio Frisoli. Design and evaluation of a novel 5 DoF underactuated thumb-exoskeleton. 2018. Accepted for IEEE Robotics and Automation Letters (RAL), `http://www.ais.uni-bonn.de/papers/JFR_Centauro_Gabardi.pdf`.

[12] Enrico Mingo Hoffman, Alessio Rocchi, Arturo Laurenzi, and Nikos G Tsagarakis. Robot control for dummies: Insights and examples using opensot. In *Humanoid Robotics, IEEE-RAS Int. Conf. on*, pages 736–741, 2017.

[13] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. STOMP: Stochastic trajectory optimization for motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[14] Kenji Kaneko, Shin'ichi Kondo, and Kouhei Ohnishi. A motion control of flexible joint based on velocity estimation. In *Annual Conference of IEEE Industrial Electronics Society*, pages 279–284, 1990.

[15] Navvab Kashiri, Arash Ajoudani, Darwin G. Caldwell, and Nikos G. Tsagarakis. Evaluation of hip kinematics influence on the performance of a quadrupedal robot leg. In *International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 1, pages 205–212, 2016.

[16] Navvab Kashiri, Nikos G Tsagarakis, Matteo Laffranchi, and Darwin G Caldwell. On the stiffness design of intrinsic compliant manipulators. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1306–1311, 2013.

[17] Tobias Klamt and Sven Behnke. Anytime hybrid driving-stepping locomotion planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[18] Tobias Klamt and Sven Behnke. Planning hybrid driving-stepping locomotion on multiple levels of abstraction. 2018. Accepted for IEEE International Conference on Robotics and Automation (ICRA), `http://www.ais.uni-bonn.de/papers/ICRA_2018_Klamt.pdf`.

[19] Felix Järemo Lawin, Per-Erik Forssén, and Hannes Ovrén. Efficient multi-frequency phase unwrapping using kernel density estimation. In *European Conference on Computer Vision (ECCV)*, pages 170–185, 2016.

[20] Maxim Likhachev, Geoffrey J Gordon, and Sebastian Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[21] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[22] Enrico Mingo Hoffman, Alessio Rocchi, Arturo Laurenzi, and Nikos G. Tsagarakis. Robot control for dummies: Insights and examples using opensot. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 736–741, 2017.

[23] Luca Muratore, Arturo Laurenzi, Enrico Mingo Hoffman, Alessio Rocchi, Darwin G. Caldwell, and Nikos G. Tsagarakis. XBotCore: A real-time cross-robot software platform. In *IEEE International Conference on Robotic Computing (IRC)*, 2017.

[24] Andriy Myronenko and Xubo Song. Point set registration: Coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(12):2262–2275, 2010.

[25] Dmytro Pavlichenko and Sven Behnke. Efficient stochastic multicriteria arm trajectory optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[26] Elvira Pirondini, Martina Coscia, Simone Marcheschi, Gianluca Roas, Fabio Salsedo, Antonio Frisoli, Massimo Bergamasco, and Silvestro Micera. Evaluation of a new exoskeleton for upper limb post-stroke neuro-rehabilitation: Preliminary results. In *Replace, Repair, Restore, Relieve–Bridging Clinical and Engineering Solutions in Neurorehabilitation*, pages 637–645. Springer, 2014.

[27] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[28] Giuseppe F. Rigano, Luca Muratore, Arturo Laurenzi, Enrico M. Hoffman, and Nikos G. Tsagarakis. A mixed real-time robot hardware abstraction layer (r-hal). *Encyclopedia with Semantic Computing and Robotic Intelligence*, 02(01):1850010, 2018.

[29] Giuseppe F Rigano, Luca Muratore, Arturo Laurenzi, Enrico Mingo Hoffman, and Nikos G. Tsagarakis. Towards a robot hardware abstraction layer (R-HAL) leveraging the xbot software framework. In *IEEE International Conference on Robotic Computing (IRC)*, 2018.

[30] Diego Rodriguez, Corbin Cogswell, Seongyong Koo, and Behnke Sven. Transferring grasping skills to novel instances by latent space non-rigid registration. In *IEEE International Conference on Robotics and Automation (ICRA)*, May 2018. `http://www.ais.uni-bonn.de/papers/ICRA_2018_Rodriguez.pdf`.

[31] Wesley Roozing, Jörn Malzahn, Navvab Kashiri, Darwin G Caldwell, and Nikos G Tsagarakis. On the stiffness selection for torque-controlled series-elastic actuators. *IEEE Robotics and Automation Letters (RAL)*, 2(4):2255–2262, 2017.

[32] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (FPFH) for 3D registration. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3212–3217, 2009.

[33] Mine Sarac, Massimiliano Solazzi, Edoardo Sotgiu, Massimo Bergamasco, and Antonio Frisoli. Design and kinematic optimization of a novel underactuated robotic hand exoskeleton. *Meccanica—International Journal of Theoretical and Applied Mechanics*, 52(3):749–761, 2017.

[34] Fabian Schilling, Xi Chen, John Folkesson, and Patric Jensfelt. Geometric and visual terrain classification for autonomous mobile navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[35] Max Schwarz, Marius Beul, David Droeschel, Sebastian Schüller, Arul Selvam Periyasamy, Christian Lenz, Michael Schreiber, and Sven Behnke. Supervised autonomy for exploration and mobile manipulation in rough terrain with a centaur-like robot. *Frontiers in Robotics and AI*, 3:57, 2016.

[36] Max Schwarz, Christian Lenz, Germán Martín García, Seongyong Koo, Arul Selvam Periyasamy, Michael Schreiber, and Sven Behnke. Fast object learning and dual-arm coordination for cluttered stowing, picking, and packing, 2018. Accepted for International Conference on Robotics and Automation (ICRA), `http://www.ais.uni-bonn.de/papers/ICRA_2018_Schwarz.pdf`.

[37] Max Schwarz, Tobias Rodehutskors, David Droeschel, Marius Beul, Michael Schreiber, Nikita Araslanov, Ivan Ivanov, Christian Lenz, Jan Razlaw, Sebastian Schüller, et al. NimbRo Rescue: Solving disaster-response tasks with the mobile manipulation robot Momaro. *Journal of Field Robotics*, 34(2):400–425, 2017.

[38] Uwe Süss, Klas Nordberg, Navvab Kashiri, Luca Muratore, Nikos Tsagarakis, Tobias Klamt, and Sven Behnke. Deliverable D8.3 First CENTAURO System Evaluation.

[39] L. Le Tien, A. Albu-Schaffer, A. De Luca, and G. Hirzinger. Friction observer and compensation for control of robots with joint torque measurement. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3789–3795, Sept 2008.