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

Deliverable D7.4 Final 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:	L. Muratore, P. Guria, Z. Ren, N. G. Tsagarakis, T. Klamt, T. Cichon, K. Holmquist, M. Felsberg, H. Karaoguz, D. Leonardis, M. Solazzi, A. Frisoli, S. Behnke
Work package:	WP7 Integration
Date of preparation:	2018-11-01
Type:	Demonstrator
Version number:	1.0

Document History

Version	Date	Author	Description
0.1	2018-11-04	Luca Muratore	First draft
0.2	2018-11-07	Paolo Guria	Firmware and actuation control
0.3	2018-11-10	Luca Muratore	Integration of partners' contributions
0.4	2018-11-13	Luca Muratore	Revised Section 2
0.5	2018-11-16	Nikos Tsagarakis	Lessons learned, conclusions added
0.6	2018-11-20	Luca Muratore	Integration overview and Introduction
0.7	2018-11-21	Nikos Tsagarakis	Executive summary revision
0.8	2018-11-21	Luca Muratore	Adding material to Section 2
0.9	2018-11-21	Luca Muratore	Integration Overview
0.91	2018-11-22	Nikos Tsagarakis	light revisions in sec. 3,4,5
0.92	2018-11-23	Nikos Tsagarakis	Introduction added, revision of Lessons learned
0.93	2018-11-23	Max Schwarz	More material for individual workpackages, overall revision
1.0			Submitted version

Executive Summary

This report describes the final version of the integrated CENTAURO disaster-response system. It details the revisions and extensions performed in the system mechatronics and software components to improve and extend the functionality of the first release of the system. The report provides an overview of the final system integration presenting the revisions of the existing core components and any additional components added as well the updated and newly introduced interfaces and interconnections incorporated in the final CENTAURO system. The upgraded final CENTAURO system was evaluated during the final evaluation camp that was organized at the premises of the KHG partner. The report concludes with a discussion of the experience gained from these final validation trials and the observations made concerning the performance of various components and capabilities.

Contents

Figure 1: Snapshots of the final Centauro robot while executing validation trials at the evaluation camp.

1 Introduction

This document describes the integration work performed in the final version of the CENTAURO disaster-response system (see Fig. 1). The final system integrates the latest results from the research and development workpackages as well as the upgrades of components that have been integrated in the first release of the CENTAURO system.

The integration activities towards the final version of the CENTAURO system were carried out in several integration meetings that took place in the premises of the project partners after the first evaluation camp and until the end of the project. Some of these integration activities were synchronized with the internal project progress meeting involving all partners while additional bi-lateral integration meetings were also organized between two or more partners focusing on specific components and integration of results between two or more workpackages

With regards to the robot hardware, the final version of Centauro has been upgraded to incorporate an on board power system that consists of a 1.6 kW Li-Po battery and a wireless router that permits the communication with the robot through a WiFi link. In addition, a new more effective left end-effector robotic hand has been installed in the final version of the robot to improve its manipulation capabilities. The new hand is based on a modular design and allows better manipulation control through the independent control of each finger. In addition to these upgrades, another revision was performed on the actuation electronic cooling systems which were updated enabling Centauro to run continuously without reaching critical thermal conditions. Finally, several updates on the low-level software and firmware components as well as new tools for online system monitoring and debugging were integrated in the final system.

As far the telepresence station is concerned, the final version integrates the wrist and the hand exoskeleton modules in both exoskeleton arms allowing to track the full arm and hand / fingers of the human operator. The final version of the prototype consists of the bilateral upper arm module (covering full shoulder and elbow rotation), two wrist modules (covering wrist abduction, flexion and pronosupination), and two hand exoskeleton modules (providing underactuated, individual force feedback at each of the fingers of the hand). Finally, interfaces between the hand exoskeleton and the Schunk and HERI robotic hands were developed and integrated allowing the full bi-manual teleoperation control of the Centauro upper body.

Concerning the simulation tools, their final version incorporates the latest body models of the Centauro robot and integrates the final interfaces of the operator components as well those of the locomotion and manipulation modules.

Finally, the updated versions of the navigation and manipulation components, in particular bi-manual grasping of larger objects, and associated revisions in their interfaces have been integrated in the final CENTAURO system, including components that permit the autonomous

execution of navigation and manipulation tasks.

The performance of the revised integrated CENTAURO system was validated in the final evaluation camp, demonstrating that the final system is able to execute challenging locomotion and manipulation tasks. The Centauro robot demonstrated capabilities in terms of navigation, manipulation with autonomous grasping, environment perception, tele-manipulation using tools for solving bi-manual tasks, wheeled mobility and legged locomotion in uneven terrains and on stairs.

We follow the structure of Deliverable D7.3 First Integrated CENTAURO System [11] and first give a short overview over the integration *actions* in Section 2 and then provide a more detailed *description* of the final integration CENTAURO system.

2 Integration Overview and Status

During the last year of the project, the activities within workpackage WP7 were related to the further integration of core components and results generated by the other research and development workpackages into the second and final 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 issues and lesson learned after the first evaluation camp. Bilateral meetings between two or more project partners were arranged to execute integration activities on specific core components of two or more WPs (see Table 1). In addition, an SSSA researcher worked on his master thesis while visiting UBO for two months, which resulted in tight integration of the hand exoskeleton with the right hand of the robot.

This section provides an overview of the integration activities and associated achievements in terms of integrated components and establishment of interfaces to reach milestone MS4 of the final integrated CENTAURO disaster-response system. Figure 2 gives an overview of the involved workpackages and the inter-package dependencies.

Table 1: Integration meetings.

Month	Date	Partners	Host	Days	Topics
32	11/17	All	KHG	5	Evaluation Camp
34-36	1/18 - 3/18	SSSA, UBO	UBO	2 months	Master thesis on the hand exoskeleton
39	6/18	KTH, UBO	UBO	4	Terrain segmentation
39	6/18	IIT, UBO	IIT	5	Autonomous navigation
40	7/18	RWTH, UBO	UBO	2	“Digital Twin” simulation
42	9/18	RWTH, SSSA, UBO	UBO	3	HMD visualization
42	9/18	IIT, UBO	IIT	5	Autonomous grasping autonomous navigation
42	9/18	All	IIT	4	General integration
43	10/18	All	KHG	5	Final Evaluation

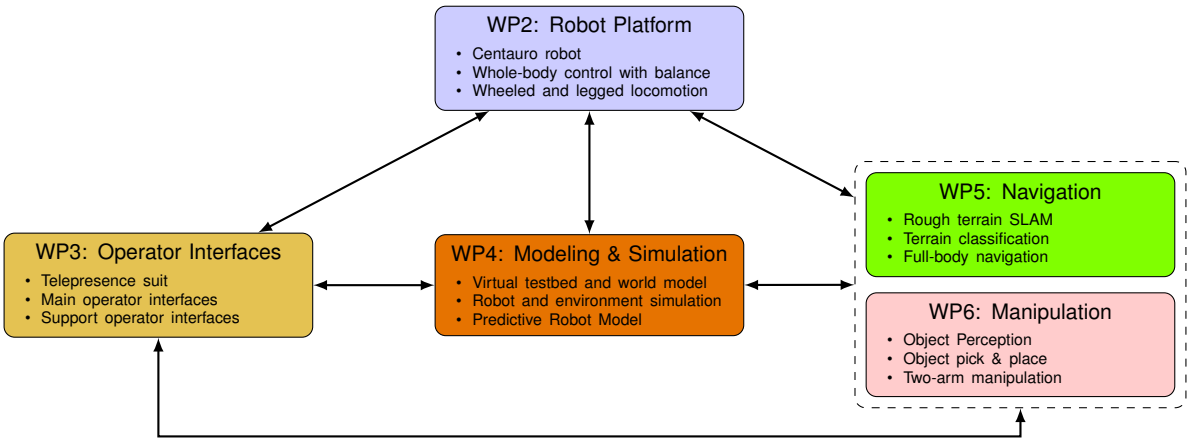

Figure 2: CENTAURO workpackages and inter-workpackage interfaces as described in this report.

2.1 WP2: Integration of Robot Hardware

Concerning the hardware components of the final integrated CENTAURO disaster-response system, the integration activities executed towards the second and final release of the Centauro robot include the integration of the battery, the on-board wireless router, the HERI Hand and the rearrangement of the fans for the cooling of the robot joints. At the same time, the firmware of each motor controller was upgraded to have a better fault handling and the software middleware of the robot was extended to have a shared control between the RT controller and the ROS controller.

In addition to the realization of the final 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 again updated incorporating the introduction of all hardware components including the battery and the wireless routers.
- 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 HERI Hand end-effector, its low level control units and the robot power management unit controller to handle both the control from power supply or battery.
- The interfaces and protocols between the on-board centralized real time control computer and the high level robot WebGUI were established allowing the monitoring of the robot status and eventually to debug fault conditions.

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

2.2 WP3: Integration of Telepresence Station Hardware

Starting from the state of the operator station at the end of the first evaluation camp, the integration activity focused and established the following interfaces:

- A ROS-based communication protocol was designed and implemented to control the HERI hand end-effector from the hand exoskeleton.
- The left hand exoskeleton was fully integrated.
- The force feedback channel was fully established and tested.

2.3 WP4: Modeling & Simulation

The VEROSIM simulator is a central component in the Centauro system: the previously developed ROS interface to VEROSIM was upgraded and maintained based on the latest communication protocol defined in the Centauro software middleware. This allows input of robot state and sensory data for visualization, input of robot commands for simulation, and output of simulation results.

In particular, a high-performance method for display of low-resolution depth data with high-resolution texture information was realized. Additional displays were integrated on request of the operators, such as 2D image views and previews of navigation plans. In some cases, these displays also act as input devices, for example to select an alternative navigation plan.

The modeling pipeline, which processes ROS URDF models for usage with VEROSIM, was extended and further automated in order to react quicker to model changes. Finally, VEROSIM integration was improved to the point where it can be used to simulate actions on a snapshot of the world perceived by the robot, before the actions are then executed using the real robotic platform.

2.4 WP5: Autonomous Locomotion

For the autonomous locomotion pipeline (WP5), main components were already integrated as reported in D7.3 [11]. In the current reporting period, terrain classification was extended by the classification of stairs. Respective modules for geometric and visual classification have been developed and integrated into the pipeline. Regarding the locomotion planner, the interface with *VEROSIM* has been established. The whole control process, including environment visualization, goal pose definition, and selection between different path alternatives can now be performed via *VEROSIM* which also has been tested.

2.5 WP6: Autonomous Manipulation

Regarding WP6: Autonomous Manipulation, the established pipeline that was tested in the First Evaluation was further improved. We aimed towards tighter integration of fewer components, i.e. some unused alternatives were dropped from the pipeline. The interfaces to other workpackages established up to M30 proved to be stable and sufficient for manipulation. New functionality was added for bi-manual grasping of larger objects. Further details of the improvements to the individual components can be found in Section 7 and Deliverable D6.4 [15].

3 Final Centauro Robot

The CENTAURO system was developed to operate in disaster-response missions. The tasks in such missions do in general possess high uncertainty and are partially unknown in advance depending on the nature of the disaster. To be effective in these situations, the disaster-response robotic platform should demonstrate a range of capabilities including navigation and manipulation skills that can address the requirements in operating in such environments. Regarding navigation capabilities, addressed environments are mostly man-made and thus, contain flat surfaces, ramps, and stairs. Those environments may, however, be affected, e.g. by cluttered debris that the platform should be able to transverse or navigate around. Concerning manipulation, addressed workspaces are also man-made and contain tools and objects that are designed for human usage. Therefore the robotic platform should be able to execute manipulation tasks involving common tools and interfaces.

The Centauro robot has been realized with the above requirements in mind and the latest upgrades and integration effort towards the final system aimed to improve the system capabilities towards these directions and the enhancement of the loco-manipulation skills as well as reliability of the CENTAURO disaster-response system.

Figure 3: Final version of the Centauro robot.

3.1 The Centauro Robot

The Centauro robot (see Fig. 3) is designed as a centaur-like platform embodying an anthropomorphic upper-body and a quadrupedal lower-body. The legs are equipped with active wheels at their lower end and the two arms possess two hands with complementary capabilities as end-effectors.

In the final version of the robot a rearrangement of the pelvis and the torso was carried out for two main reasons. The first was to eliminate the interference of the legs with the battery system and increase the range of the hip yaw joints. To achieve this, the perception PC was moved to the torso back and the on-board power location was changed as show in Figure 4. The installed battery make use of the latest high energy density battery technology (220 Wh/Kg) enabling the delivery of 1.6 kW within a compact volume, and low weight (less than 8 Kg) battery pack that provides power autonomy of approximately two hours.

The second reason is related to the integration of the wireless communication link on the torso back allowing tether free communication that in combination with the battery availability permits the full tether-free operation of the final Centauro robot.

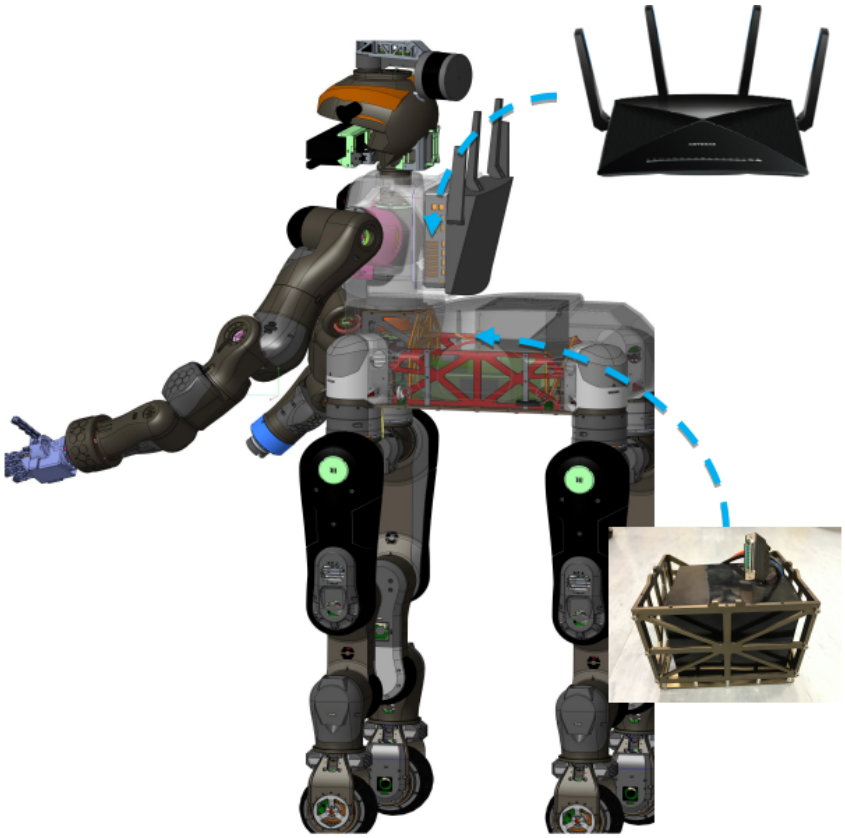

Figure 4: Revisions of the hardware integration: wireless communication link and battery location.

Furthermore, as a response to the thermal issues observed on the first version, an upgrade of the robot cooling system was performed to increase the air flow and direction which was routed through the intermediate motor driver stack layers as shown in Figure 5. This involved the relocation of the fans in the robot joints placing them perpendicularly to the motor driver stacks. In addition, at the firmware level efficient thermal strategies were implemented with the active control of the fans based on the thermal state of the joint actuation system (motor + driver electronics). In addition enhancements of the cooling system of the on board computers and power management electronics were applied by providing increase airflow in the related pelvis areas.

3.1.1 HERI Hand Integration and Control

Given the grasping limitation of the first end-effector prototype used for the Centauro left arm, we decided to design and develop the HERI (Hardware Embedded Reduced Intricacy) Hand, an under-actuated hand based on highly modular finger units. As shown in Figure 6, we adopted a four-finger configuration for the HERI Hand mounted on CENTAURO left arm.

Each finger module is powered by a single actuator through an under-actuated transmission and equipped with a sensory system for delicate and precise grasping, which includes absolute position measurements, contact pressure sensing at finger phalanxes and motor current readings.

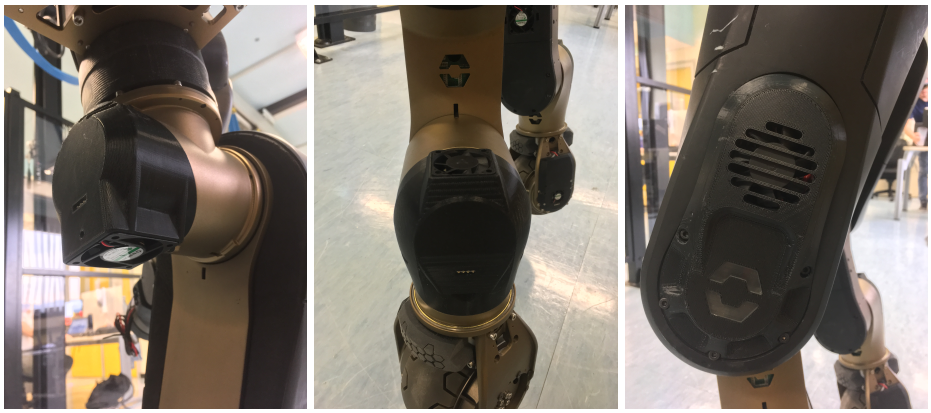

Figure 5: Joint cooling system upgrades at the leg joints of CENTAURO.

Finally, intrinsic elasticity integrated in the transmission system makes the hand robust and adaptive to impacts when interacting with the objects and environment.

The HERI Hand was successfully integrated in the Centauro thanks to its design as described in [18].

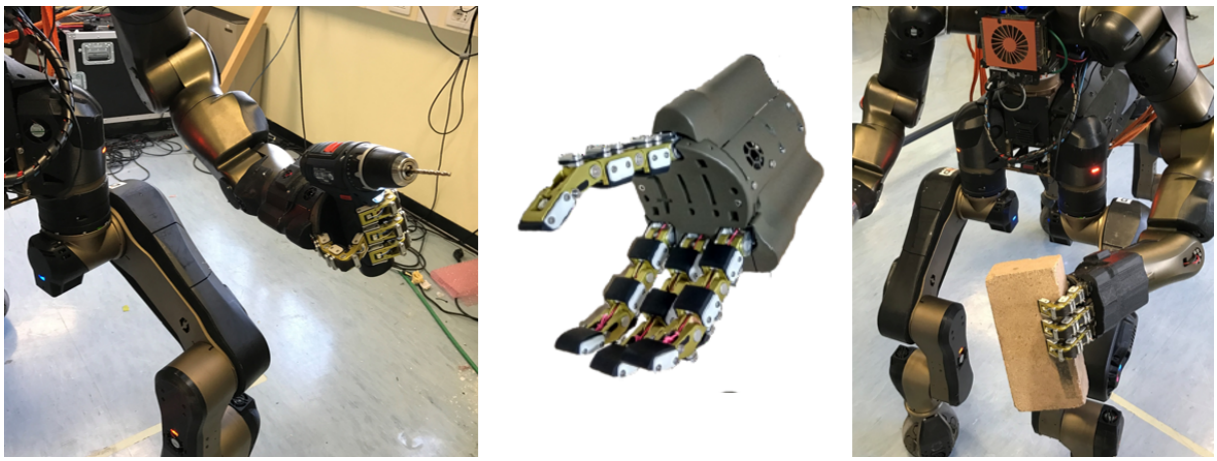

Figure 6: HERI hand with a four-finger configuration, successfully integrated as Centauro left arm end-effector.

3.1.2 Firmware and Actuation Control

After the experience in the first evaluation camp in KHG, hardware and firmware were both modified. We identified two main weaknesses in the installed electronics hardware components: Motor driver, interface board failure at high current, and Encoder Line Drivers (FLEX printed circuit board (PCB)) failure due to low mechanical robustness.

The BLDC Controller was redesigned. The issue was the underestimated outlet inductor from the 15 V DC/DC regulator, located on the Interface board of the motor driver electronics. This board of the motor driver stack was revised, according to the new electronics constraints (double MOSFET stage). With an improved air flow (see Subsection 3.1) the electronics now is able to work for several hours without facing any thermal or overloading issue. The Encoder Line Drivers flexible PCBs were too fragile. To fix this issue it has been added a layer of Bakelite beneath the soldered components, as well the PCB tracks were enlarged in thickness

and in width. The flexibility in the connection has been kept, increasing the reliability of the component.

On the firmware side, all the effort was focused on the identification of the sporadic joint failure causes. To enable more efficient debugging, a new Fault management has been implemented reporting in real time the FAULT condition as introduced in Figure 7.

BIT	Meaning	Trigger Condition
0	Position Reference not Allowed	Position reference beyond the position limits set in the firmware
1	Velocity Reference not Allowed	Velocity refence beyond the maximum speed set in the firmware
2	Torque Reference not Allowed	Torque refence beyond the maximum torque set in the firmware
3	Hardware Fault (BLDC Controller)	Hardware problem, preventing the Power ON operations.
4	Parameter out of Range	Parameter set beyond the hardcoded limits.
5	Torque Array not Loaded	Torque array not found in the flash memory, or calibration not performed
6	Torque Read Error	Torque sensor broken
7	Spare	
8	Link Encoder Error	More than 20 bad samples in a row from the Link Encoder Reading
9	Deflection Encoder Error	More than 20 bad samples in a row from the Deflection Encoder Reading
10	Temperature Warning	Temperature from the board or the actuator beyond 60 C°
11	Temperature Fault	Temperature from the board or the actuator beyond 70 C°
12	Motor Encoder Error	More than 200 bad samples in a row from the Motor Encoder Reading
13	DC Bus Voltage read Error	Bad DC Bus Voltage reading
14	Sandbox Active	Entering the Sandbox (if active)
15	Spare	

Figure 7: Fault word: meaning and triggering conditions.

This fault management simplifies the debug operations of the robot when an abnormal state occurs reducing the maintenance time of the platform. As maintenance time was a significant problem during the first evaluation camp, we expect a large benefit from reducing it.

3.1.3 Software Architecture

As already described in *Deliverable D7.3 First Integrated CENTAURO System* [11] the software architecture adopted in the Centauro robot is based on the XBotCore [13] framework with ROS integration.

XBotCore contains a plugin-based Real-Time(RT) controller at 1 kHz and a non-RT Communication Handler working at 200 Hz to manage communications with the rest of the (ROS-based) system. XBotCore was designed with a ”single-master” policy, i.e. either the RT controller or the ROS (non-RT) controller is able to send references to the robot. This means that we are not able to command at the same time a subset of motors of the robot through the RT controller and the others using ROS-based communication.

In the first evaluation camp we faced some issues for the above mentioned limitation: in

particular, while the robot was controlled by the Alex exoskeleton (Inverse-Kinematic RT controller) it was not possible to move the robot using the wheels (ROS-based wheeled locomotion controller).

Since this feature was needed to adjust the robot position with respect to the task it had to execute, we developed a shared-control plugin which gives us the opportunity to command at the same time the upper-body of the robot (for Cartesian control of the arms' end-effector) using an RT controller and the lower body (for wheeled, quadruped, or hybrid locomotion) from external ROS reference.

From the high-level diagnostic software point of view, after the first evaluation meeting it was obvious that a GUI (Graphical User Interface) to debug the hardware issues on the Centauro robot was needed. To this purpose we developed the WebGUI, shown in Figure 8.

Figure 8: Centauro WebGUI.

The WebGUI is a JavaScript-based interface which communicates with the XBotCore Communication Handler using web sockets. It allows the operators to visualize the low-level state of the robot and to understand if a fault condition is happening in one or more joints, as indicated by red highlighting of the motor in the 3D visualization of the robot model. At the same time we are able to check, plot and visualize all the data received from the robot and sent to it, as shown in Figure 9.

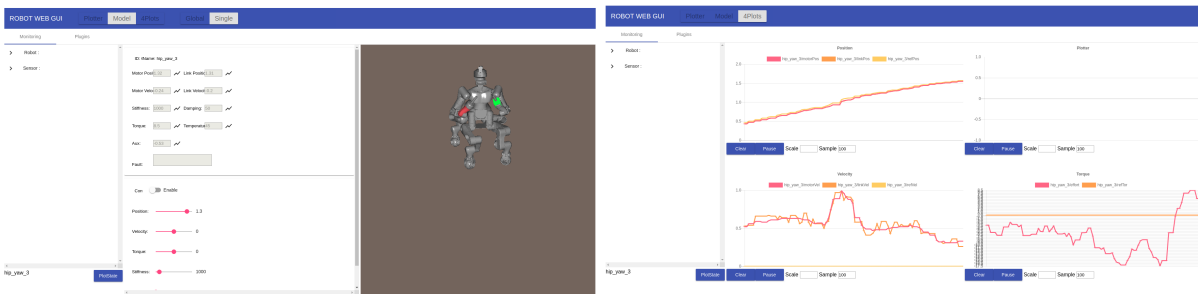

Figure 9: Centauro WebGUI: Joint state visualizations. Left: Immediate view. Right: Time-series view.

4 Centauro Operator Interfaces

4.1 Full-Body Teleoperation Control Station: Development and Integration with the CENTAURO System

The interface for the main operator has to track the movements of the operator to control the remote robot and to provide force feedback for its upper body, including the fingers, in order to display the forces that the robot exerts on the environment. Thus the Full-body Telepresence Control Station is composed of different exoskeleton modules covering arms, wrists and hands, and pedals for navigation control. It especially enables the first operator to teleoperate complex manipulation tasks while providing force feedback. In the last period of the CENTAURO project, all the modules composing the full-body telepresence control station have been developed and tested. Since the telepresence control station is symmetric, all the modules have been implemented and tested at first on the right operator side, then robotic modules have been optimized and built for the left side.

Bilateral Arm Exoskeleton Module In the final system, both arms of the bilateral arm exoskeleton have been built and assembled. Actuators and electronics have been integrated and tested. Also, a low level control algorithm has been implemented for providing transparency and force feedback to the operator when teleoperating the robot. A summary of the final bilateral arm exoskeleton capabilities and performance is described hereafter:

The final implementation of the bilateral arm exoskeleton features a large range of motion which does not restrict the movements of the operator, a full active support of shoulder (three DoF) and elbow (one DoF), and a low weight to reduce the inertia of the device and to improve dynamic response.

The exoskeleton provides full support for shoulder and elbow rotation, implementing an innovative lightweight and backdrivable transmission for the complex 3 DoF shoulder joint. The whole upper limb exoskeleton can reach about 90% of the natural workspace of the human arm without singularities, covering an extended range of motion for each DoF.

The device is a mechanically compliant robotic manipulator, actuated by means of electric actuators, remotely located with respect to the robotic joints. The remote location of the actuators allows to keep the masses and the encumbrances of the moving parts small, in order to achieve the highest possible dynamic performance and the widest admissible joint ranges of motion without interference with the human body. For ALEx, the total weight of the moving parts is only 3 kg, of which about 2 kg are due to the first two proximal links.

The arm exoskeleton kinematics is isomorphic to that of the human arm (i.e. the axes of

Figure 10: The bilateral arm exoskeleton module, featuring four actuated DoFs (shoulder and elbow) in a wide operative workspace.

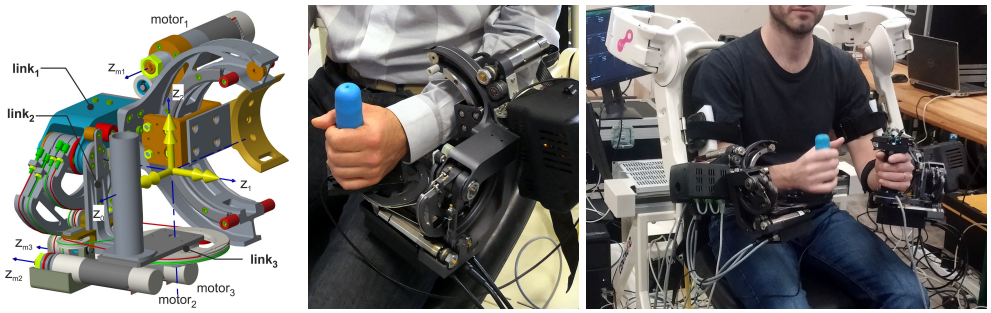

Figure 11: The wrist exoskeleton module (WRES): design, prototype and final bilateral implementation designed to be mounted at the arm exoskeleton.

rotation of the DoFs of the device are substantially aligned with those of the user physiological articulations) and has the important property of presenting no singularities in the natural workspace of the upper limb. The wide ranges of motion at the shoulder are $120°$, $95°$, $165°$, for the first, second and third DoFs respectively.

A chair, having a manual hydraulic height regulation of the seat, allows the user to operate the device from a comfortable seating position.

Wrist Exoskeleton Modules Two WRES exoskeleton modules, for the right and left wrist, have been developed and integrated in the final version of the CENTAURO Control Station providing tracking and force feedback for the operator's wrist rotations and torques. WRES is a purely rotational 3-DoF forearm-wrist exoskeleton based on serial kinematics and actuated by a tendon-based transmission. The device is able to apply torques for forearm pronation/supination (PS), wrist flexion/extension (FE) and radial/ulnar deviation (RU). Main requirements concerned the lightness of the device, its easiness to be worn, and the need to have an open structure in order to allow the user to manipulate real objects or avoid collision between the upper bilateral arms exoskeleton ALEx during bimanual operational tasks.

In the final version, both WRES modules feature renewed control electronics, fully interfaced with the rest of the Control Station modules and with the main control unit through the reliable and real-time ETHERCAT communication protocol. To achieve good transparency, the low-level control of the wrist exoskeleton takes into account the gear head efficiency and two main feed-forward compensation terms: gravity compensation and the gear motors' viscous friction compensation.

Hand Exoskeleton Modules The Hand Exoskeleton has been designed to track the operator's fingers in closing and opening, and to provide force feedback of grasping forces exerted by the teleoperated robotic hands. Due to the numerous DoFs of the human hand, the device has been designed adopting the under-actuation concept, focusing on grasping operations with independent fingers. Importantly, wearability issues were taken into account and addressed by a peculiar design solution with parallel, underactuated kinematics. It provides independent finger operation with underactuation for the two DoFs of the first and second phalanges. It exchanges only normal forces between user's finger and exoskeleton links and includes finger kinematics in the kinematic chain of the exoskeletons, thus obtaining the required self-adaptability to operator's finger sizes. The hand exoskeleton is composed of five finger components and is equipped with five actuators in total. A more complex kinematic solution has been designed and implemented for the thumb, underactuating its five DoFs along a preferred direction of grasping. This

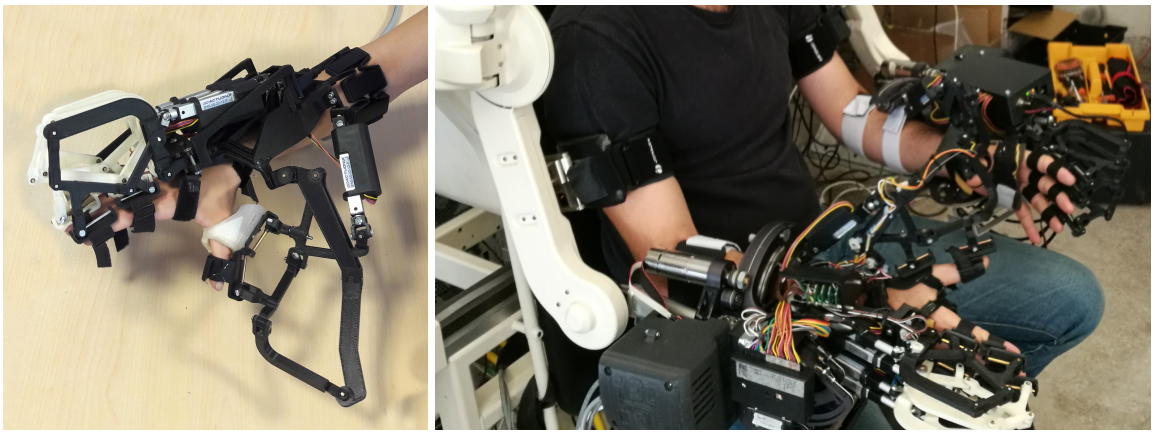

Figure 12: The final hand exoskeleton prototype, implemented in two devices for bilateral operation of the two hands.

allows to perceive force feedback along one DoF (resembling grasping direction), at the same time without constraining the five DoF of the operator's thumb in teleoperating the robotic hand.

The initial prototype of the hand exoskeleton has been optimized in kinematic dimensions and attachment of the final links to the user hands, providing a more comfortable and rapid donning of the device. Intermediate joints of the parallel mechanism have been redesigned for improving robustness and for integration of a potentiometer, in order to better track position of the fingers and of the thumb. An additional mechanism and sensor has been integrated for the index finger, in order to measure finger spread and thus to teleoperate that specific degree of freedom of the robotic hand. Concerning the actuation of the device, linear screw actuators with DC motors have been implemented as a compact and lightweight solution. Since backdriveability is fundamental in teleoperation, miniaturized strain gauge force sensors have been adopted.

A force control algorithm has been implemented in the device and tested with transparency and virtual stiffness rendering. Force sensors placed between the ground link and the actuators measure the intentional forces applied by the user's fingers on each of the finger-exos. Measured forces are used to set the direction of the closing/opening the hand as much as the speed of motion.

Final Full-Body Telepresence Control Station The different modules of the Full-Body Telepresence Control Station have been assembled in the prototype shown in Fig. 13. The prototype was composed of the bilateral upper arm module, covering full shoulder and elbow rotation, two WRES modules, covering wrist abduction, flexion and pronosupination, and two HandExos modules, providing underactuated, individual force feedback for each finger of the hands. The Full-Body Telepresence Control Station also integrates pedals for controlling navigation of the Centauro robot in remote environment. The module has been integrated in the final version of the Centauro Control Station and provides three DoFs (it makes use of a couple of pedals with independent rotation on the horizontal axis, plus an additional rotation mechanism coupling the two pedals on the vertical axis).

4.2 Development and Assessment of the Operator Interface Control

The main objective of this task is the development of a bilateral teleoperation framework that is able to ensure a good transparency level while maintaining the system stability also under

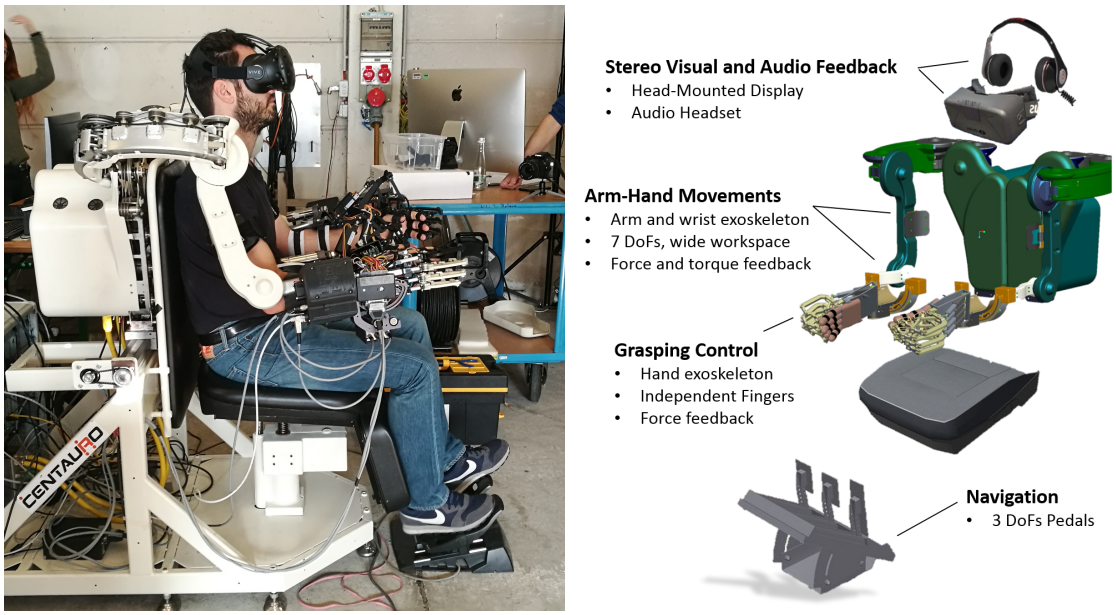

Figure 13: The concept scheme and the final assembled Full-body Telepresence Control Station.

communication delays. Two different approaches have been investigated and implemented for teleoperation of the arm (up to the wrist point) and for teleoperation of fingers. This difference in the telepresence approach was due to the different nature in terms of masses, rigidity and type of actuators of robotic modules related to hands and arm respectively for both the Centauro robot and the full-body telepresence suit.

Hand Teleoperation Control Teleoperation at the level of hand grasping and finger movements was developed to control the Schunk anthropomorphic robotic hand, mounted at the right arm of the Centauro Robot. The aim of the teleoperation was to control finger movements and force modulation of the operator's hand directly on the remote robotic hand, and to provide force feedback of the grasping interaction with physical objects. One of the challenges in such teleoperation scenarios is that the master and slave systems (the hand exoskeleton and the robotic hand) share similar but not identical kinematics and possess different actuation and control methods. Due to the above, specific pose estimation and force-feedback algorithms on the underactuated hand exoskeleton where developed depending on the involved DoFs for each finger. In particular, the very high number of DoFs of the human hand is in conflict with the limited number of actuators for both, the robotic hand and the exoskeleton. Moreover, due to the variability of the human hand size, underactuation and adaptive kinematics are design solutions implemented in the exoskeleton which further reduce the number of implemented actuators together with other wearability constraints.

The overall teleoperation approach has been set on the basis of the local admittance control of the hand exoskeleton and of the local impedance control embedded in the Schunk hand. Angular position references estimated by the hand exoskeleton were sent as references to the Proportional-Derivative position control system embedded on the Schunk Hand. The embedded PD control of the Schunk hand turned position references in actuator torques proportional (at stall) to the position error. Torques applied by the Schunk hand were then sent back to the hand exoskeleton and used as force reference in the local admittance control (see Figure 14). Hence, the operator can control finger positions while moving in free space by means of the local transparency control of the hand exoskeleton and by means of the position control implemented

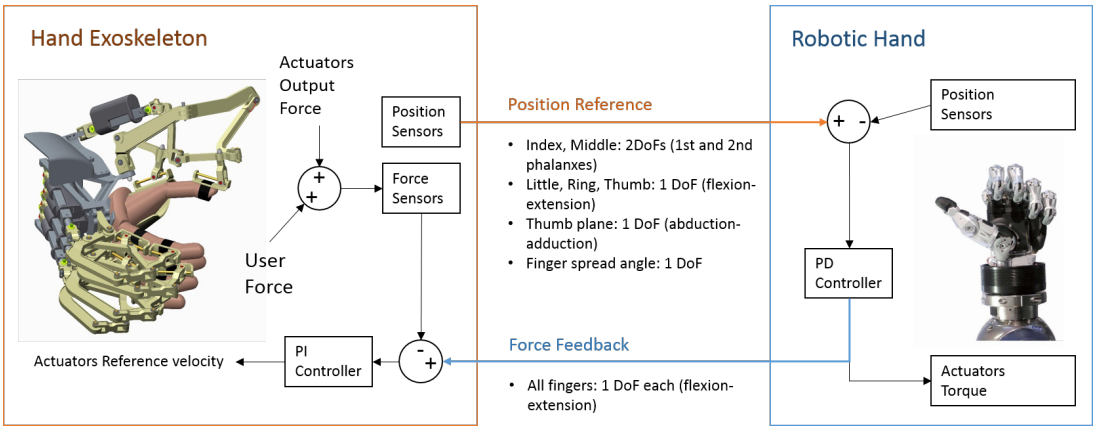

Figure 14: Teleoperation schema between the Hand Exoskeleton and the Schunk robotic hand.

on the robotic hand. Once in contact with an object, the applied force can be increased by further closing the fingers. This acts like a virtual spring due to the proportional component of the position control loop at the robotic hand. At the same time, the amount of force applied by the robotic hand was fed back to the operator as a non-zero force reference added in the transparency control.

The kinematic matching of the different DoFs slightly differed between fingers: Thumb, Little and Ring fingers were operated on the basis of a single DoF each by both sides, resembling the overall closing/opening of the finger. Index and Middle fingers were position controlled by the operator using two DoFs, matching rotation of the first and second phalanxes of each finger, sensed on the hand exoskeleton, and allowing a more precise grasping. Force feedback was however provided on 1 DoF, using the single actuator implemented in the adaptive and underactuated mechanism of the hand exoskeleton. Two additional DoFs, position sensed on the hand exoskeleton but not actuated for force feedback, were used for controlling spread between fingers and for rotating the plane of closing/opening of the thumb (abduction-adduction).

4.3 Full-body Telepresence with the CENTAURO Robot

In a preliminary testing phase, different parts of the complex Full-Body Telepresence Station were tested separately: Teleoperation of the hands, teleoperation of the arms, navigation and visualization.

Regarding teleoperation of the hands, teleoperation of the Schunk Hand with the HandExos has been performed, as well as integration and teleoperation with the HERI hand.

Regarding teleoperation of the Schunk hand, initial experiments investigated and tuned different control modalities at the low-level control of the robotic hand. After initial tests a calibration of the operator's hand with respect to the pose of the Schunk Hand and pose limits has been further improved: a pose estimation algorithm onboard of the hand exoskeleton has been developed which is based on a calibration procedure that can be performed once the operator wears the hand exoskeleton. This helps to overcome differences in parameters introduced by different fixation and tightening of fixation belts between the operator's hand and the hand exoskeleton links.

For teleoperating the HERI hand, a similar pose-estimation algorithm has been developed, although parameters were less critical since, due to the kinematics of the HERI hand, there existed a direct 1-DoF mapping between fingers of the robotic hand and of the exoskeleton. A ROS node was developed for communicating position/force references from the hand-exoskeleton to

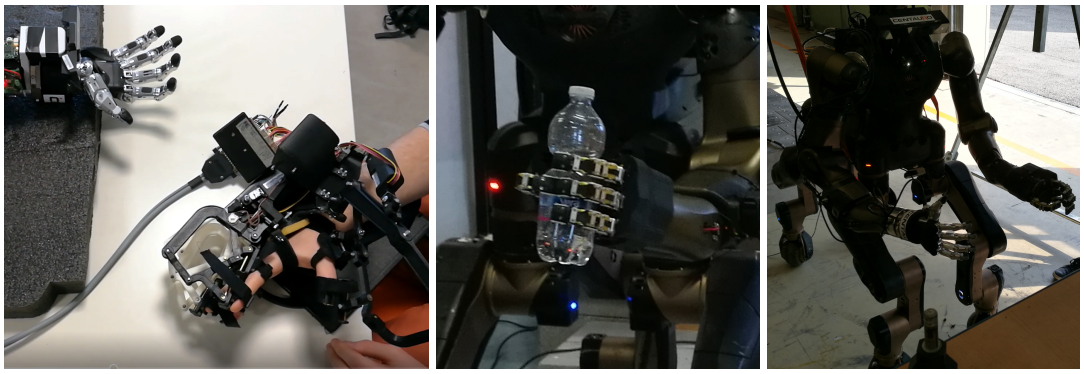

Figure 15: Test and tuning of hand teleoperation (pose estimation, pose limits and force feedback parameters) between the HandExos modules, the Schunk Hand and the HERI Hand.

the HERI hand, and to retrieve force sensor measurements from the robotic hand to the hand exoskeleton.

Also, a Passivity Approach teleoperation algorithm has been developed and added to the control of the hand exoskeleton in presence of force feedback, in order to improve stabilization of the teleoperation on the contact threshold with rigid objects.

Regarding teleoperation of the arms, preliminary tests have been conducted teleoperating the Centauro robot through the upper arm (wrist and shoulder) exoskeleton modules of the Full-Body Telepresence Station. Individual tests were performed to investigate the data communication protocols between the robot and the control station and to track the position of the robot wrist point. Position tracking also involved tuning of the workspace limits and offsets, a critical aspect since arms and wrists of the Centauro robot have different proportions and also kinematics with respect to the operator inside the control station.

Then, teleoperation tests in presence of force feedback were performed with enabled passivity control. In the experiments, the 6 DoF force/torque sensor mounted at the robot's wrist was used to provide force and torque information. The weight of the robotic hand was compensated. The setup allowed for effective force feedback to the operator: test included transparency in free-space movements, contact threshold with solid objects and sliding over surfaces (see Fig. 16). A second approach to determine forces and torques at the robot's wrists was tested as well: Indirect force/torque information was obtained from measurements of arm actuator torques. With this option, only linear force-feedback resulted reliable, whereas feedback of torques was affected by the indirect measurement of torques at the robot side. Teleoperation using both hands (Schunk Hand and HERI Hand) was performed grasping objects (plastic bottles in order to tune teleoperation, electric tools as drill) with different poses (full hand grasping, pinch).

Figure 16: Test and tuning of arm teleoperation (workspace, offsets, force feedback parameters) between the HandExos, the Schunk Hand and the HERI Hand.

5 Simulation & Visualization

As a major link of the overall system the Modeling and Simulation workpackage interfaces and integrates with three other major components: the Robot Platform (Section 5.1), the Operator Interfaces (Section 5.2), and the Navigation & Manipulation Pipeline (Sections 6 and 7). The workpackage is tasked with different concerns, which will be explained in the following. More details about individual tasks and progress can be found in D4.1, D4.2, D4.3 and D4.4. The general integration concept was already presented in D7.3.

5.1 Robot Platform

As basis both for visualization and simulation, a high-fidelity robot model is required. In addition to that, an interface to the real robot system is needed. Using these components, in combination with the VEROSIM simulation environment, a Digital Twin (DT) of the real robot can be simulated or visualized (see Fig. 17).

Interfaces (In- and Output) The interfaces here encompass the automated URDF model import and the ROS interface. The URDF model pipeline accompanies the development process to early integrate the different modules and components of the final system.

Dependencies The main dependency here lies in the modeling pipeline. As this involves WP2 in terms of the robot setup but also the sensor setup from WP5/6 a general workflow was established to react to changes in the robotic system by IIT, then integrate necessary adaptations from UBO (mainly regarding the sensor setup) and finally import the model by RWTH (see Fig. 17). As the middleware ROS has been established as the central communication interface the dependency here only lied in a concluding definition of the different message types used.

Status of Implementation and Integration The modeling pipeline as well as the ROS interface has completely be integrated and is in use.

Testing and Evaluation The modeling pipeline as well as the ROS interface has successfully been used in the evaluation camps.

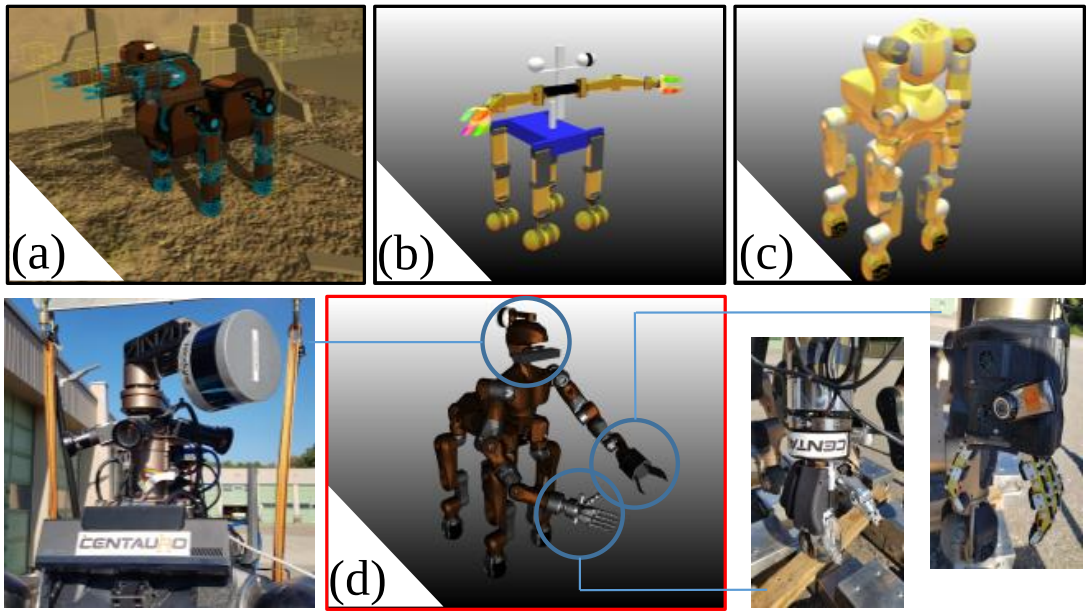

Figure 17: Evolution of the Digital Twin: (a) a RWTH Design Study, (b) Momaro, (c) simple Centauro, (d) final Centauro, based on real sensor head and variable hand setup.

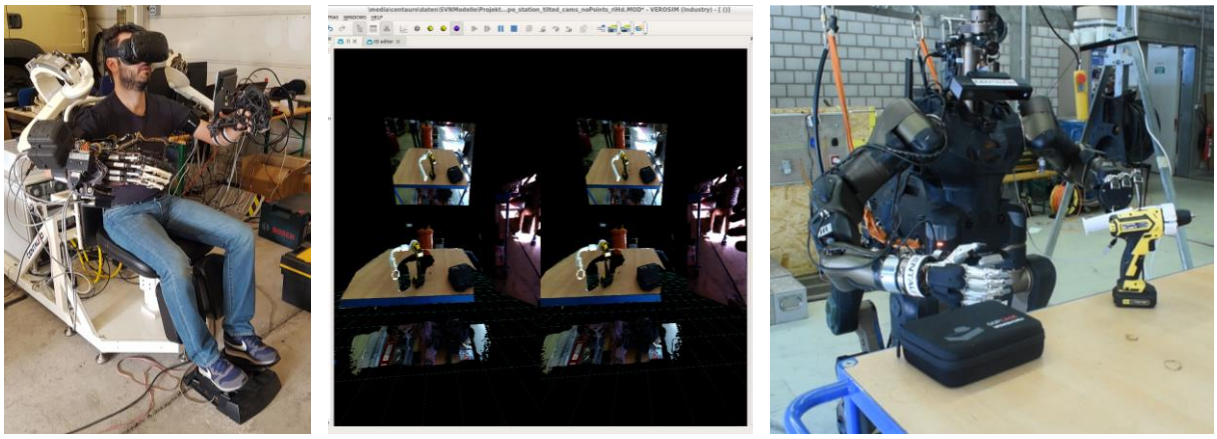

Figure 18: Main operator first-person interface for manipulation task: *Left:* Exoskeletal control. *Center:* Visualization in a head-mounted stereoscopic display. *Right:* Real robot executing the task teleoperated including force feedback.

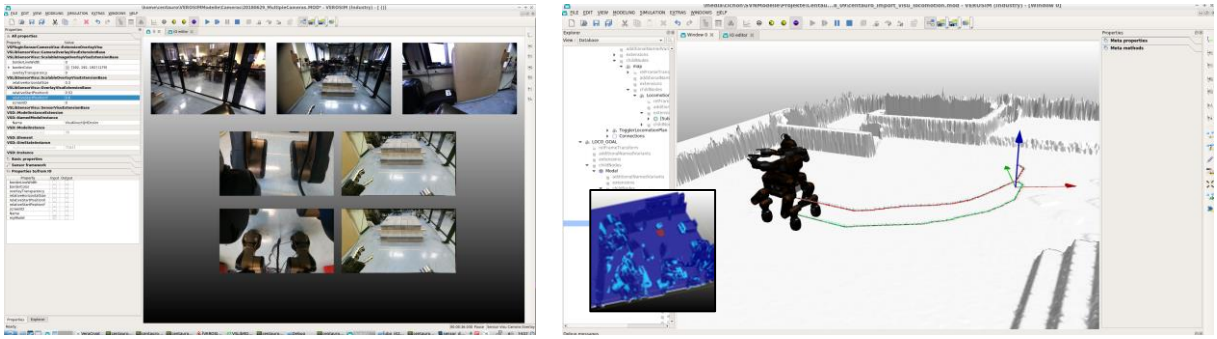

Figure 19: Support operator third-person interfaces (*Left:* 2D UI elements, *Right:* 3D VisuModel) in a locomotion task involving (i) creation of height map, (ii) creation of cost map, (iii) definition of locomotion goal (iv) planning of different paths (v) selection of path (vi) supervised execution of motion.

Debugging Issues Debugging modalities have been implemented to online inspect the ROS messages in VEROSIM. Additional issues only arose when data types were adapted and changed during the development (without global knowledge) and had to be integrated in the simulator as well. The lesson learned here is mainly that the selection of ROS as the message passing medium is very suitable.

5.2 Operator Interface

The operator interface offered by the modeling and simulation workpackage consists mainly of the first person operator and the support operator using different views of the simulator. The ***main operator*** uses a first-person view with a head-mounted stereoscopic display, in particular the HTC Vive. This first-person view is controlled by the main operator (i.e., by moving his head) and bound to pre-defined (yet movable) viewpoints, e.g., in the head of the robot which can be set and changed by the main operator, as well as the support operator. This first-person view is shown in Figure 18 and more information is given in [5].

The ***support operators*** mostly use a third-person view onto the whole scene by means of regular computer monitors (but can also use an immersive room-scale VR view) where each part of this system is connected via ROS (see also [5]). This third-person view is implemented

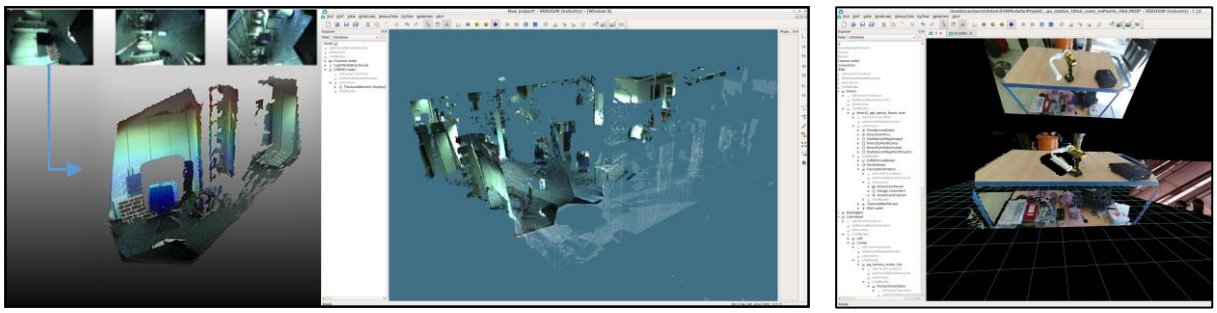

Figure 20: Textured Mesh. *Left:* Velodyne points and 3 RGB camera images - single camera with sub-sampled (colored) point cloud, then all three RGB cameras meshed with the overall Velodyne point cloud - missing image data is meshed with uniform half-transparent gray texture. *Right:* High-resolution Kinect v2 mesh for manipulation tasks.

as a free-floating camera for each support operator that can be navigated using keyboard and mouse control, game pads, or space mouse devices. The developments are based and still rely on the 3rd person operator interfaces for Momaro and have been extended with valuable user interfaces also using the simulation/DT in-the-loop. In addition, one can project data into the 3D scene, select objects in the environment to read and change their properties, particularly to change the visibility of 3D data in the scene. By adapting and applying the existing user interface framework of the simulation environment, developers of the CENTAURO project are also enabled to implement regular workflows and dialogues with typical 2D UI elements, e.g., to configure 2D overlay windows for visualizing additional data, e.g., from sensors or to develop manual control panels for aspects of the robot mission. In Fig. 19, the support operator interface is shown for a locomotion task.

To summarize the UI, we have two operators with different control devices and middlewares (exoskeleton (UDP), direct control with standard PC input devices (ROS)), visualization can be done either in 3D immersive mode or in 2D mode rendered in VEROSIM aided by external ROS tools, and the support operators either support the 1st person operator or directly control the RT or DT via ROS.

Visualization and Rendering The Visualization and Rendering includes visualization of the current robot state, visualization of raw sensor data, visualization of pre-processed sensor data (output of WP5, WP6), and monoscopic and stereoscopic rendering of all this data.

Textured Polygon Meshes Based on the optical sensors of the robot (Kinect v2 RGB and depth cameras, Velodyne laser scanner, three wide angle RGB cameras) a textured polygon mesh renderer has been implemented in VEROSIM (cf. Figure 20). This allows to mesh the Velodyne point cloud and to lay the textures of the three RGB cameras on top. Additionally, meshing the Kinect sensors into a high-resolution short-range overlay (see also [5] Section 5.2). The combination of both gives a good overview over the scene (for locomotion tasks) and the additional high resolution Kinect mesh for manipulation tasks. Unfortunately, the limitations of the Kinect (minimum range and bad performance for shiny surfaces) lead to problems in very short-range applications.

Interfaces (In- and Output) The input to the simulation is established via ROS. This standardized interface leads to an efficient way of raw and preprocessed data flow into the simulation system. The second interface here is the visualization pipeline of the simulator. Using the OpenVR library we established 100% adaptation of all rendering either monoscopically or stereoscopically. Thus, everything that has been established in the simulator was ready for use in the end of the project in both ways and by both operator types.

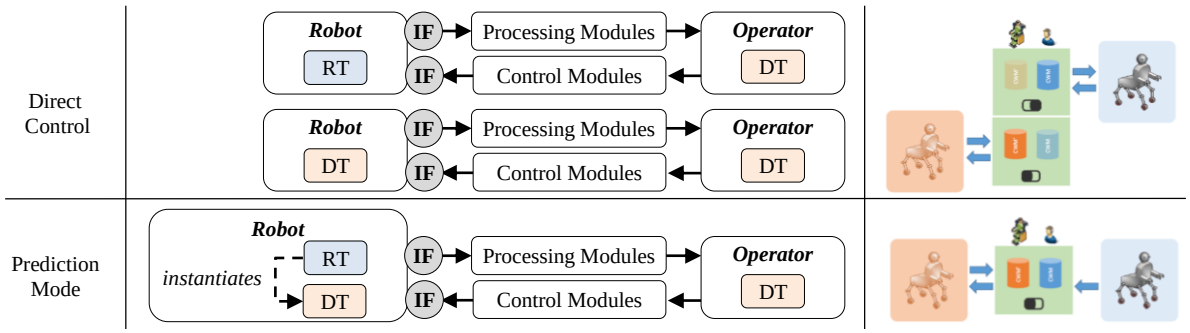

Figure 21: Centauro Modes - Hard- and Software Setup in Terms of Data Flow: (i) Using the DT for visualization on the operator side, (ii) directly controlling either RT or DT by means of the same in- and output data streams and (iii) the switch of instantiating a DT based on the RT data for prediction mode.

Dependencies The operator interface in simulation mainly depends on the ROS input from the sensor data. Using standard message types for raw data this only depended on the specialized data types for locomotion tasks or textured mesh visualizations for example. The additional interface of controlling the Digital Twin directly is as well based on standard ROS message types using the same interface as was already used for the Momaro robot.

Status of Implementation and Integration The operator interface from simulation was successfully implemented and integrated.

Testing and Evaluation In the evaluation camp at KHG, the visualization pipeline was tested and evaluated. The switch into simulation was tested at KHG as well but could not be evaluated yet as we encountered mismatches of mass distributions between Real and Digital Twin that made operation impossible. These issues are resolved now.

5.3 Predictive Robot Model

The predictive robot model comprises the overall integration of all relevant data into a holistic virtual world, the synchronization and clone of the CWM, and the possibility to directly control the DT with the same (ROS) interface as the real robot.

Centauro Modes In a mission the CENTAURO setup consists of the (i) Centauro robot (RT), (ii) modules for incoming and outgoing data and (iii) the Digital Twin (DT). On the operator side, the DT is used for visualization purposes and has a reduced complexity (cf. "Visu" model). Additionally, on the robot side, there needs to be a DT instantiated from the RT percepts when needed. With this setup the different mode of the CENTAURO system can be achieved according to Figure 21.

Monoscopic/Stereoscopic rendering for support operator Depending on the mentioned modes, the operator can use the Visu model of the Digital Twin and choose between the various visualization possibilities the DT in-the-loop offers. Thus, all aforementioned information and visualization metaphors can be used and rendered either monoscopically or stereoscopically for the main operator as well as for the support operator

Switch into Prediction Mode The established ROS interface can also be used in the mission to evaluate planned actions in simulation first, before executing them with the real system. The dynamic environment generated by the robots percepts combined with the DT model of the Centauro robot can then be directly controlled by the support operator (cf. Figure 22). This switch is established in the ROS control software by using an additional ROS name space to

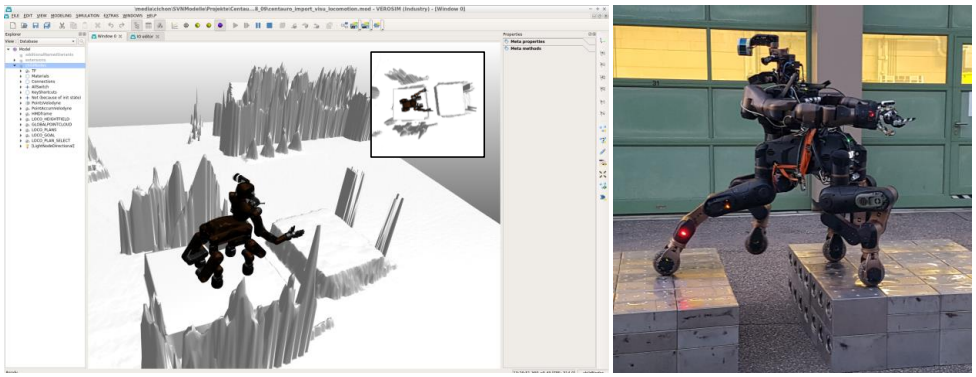

Figure 22: Switch into simulation for “stepping over a gap”: Online generation of a “dynamic” height map DT with the DT model of the Centauro robot.

redirect all ROS commands to the virtual robot while the real robot is in a resting state. This switch is described in detail in Deliverable D4.4 [6] Section 7.2.

6 Autonomous Navigation

6.1 Terrain Classification Pipeline

The terrain classification system consists of two complementary processing pipelines that assess the traversability of the local terrain from geometry-based and appearance-based perspective. We separate the terrain into three classes: safe terrain, risky terrain, and obstacle. The safe terrain is an area on which the robot can drive smoothly, the risky terrain is an area on which the robot will experience bumps and vibrations, and obstacle is an area which is non-drivable. The flow diagram of the system is given in Figure 23.

We derive terrain characteristics from geometric and visual inputs. The geometry-based analysis takes the terrain elevation point cloud generated from the raw laser scanner data as input and provides straight-forward features of the terrain structure which can be applied directly to assess the terrain traversability. Similarly, the vision-based analysis takes the camera images as input and classifies the terrain type into a set of terrain classes. The terrain traversability score is then inferred from the discrete terrain class.

The two image-based pipelines are trained using different dataset. The appearance-based feature extraction pipeline is trained using the CityScapes dataset and the stair detection pipeline is trained using the data we collected at KTH campus, UBO and KHG experiment site. We hand-labeled a small amount of the data to train the final random forest classifier for fusing the results.

Firstly, we describe the geometric terrain feature extraction in Section 6.2, then we explain the visual terrain segmentation architecture in Section 6.3.

6.2 Geometry-based Traversability Analysis

As described in the introduction, features from both spatial domain and frequency domain can be extracted from the terrain elevation map. Features from the spatial domain such as slope and roughness summarize the relationship of a point with respect to its surrounding points. Features from the frequency domain such as FFT and PSD can predict the robot's response to entire terrain segments. In our approach, only spatial features are used to represent the terrain traversability since the frequency features require a large window size to generate a reasonable result. Therefore, it is hard to generate a high resolution cost map with frequency-based features. The process of geometry-based traversability analysis includes:

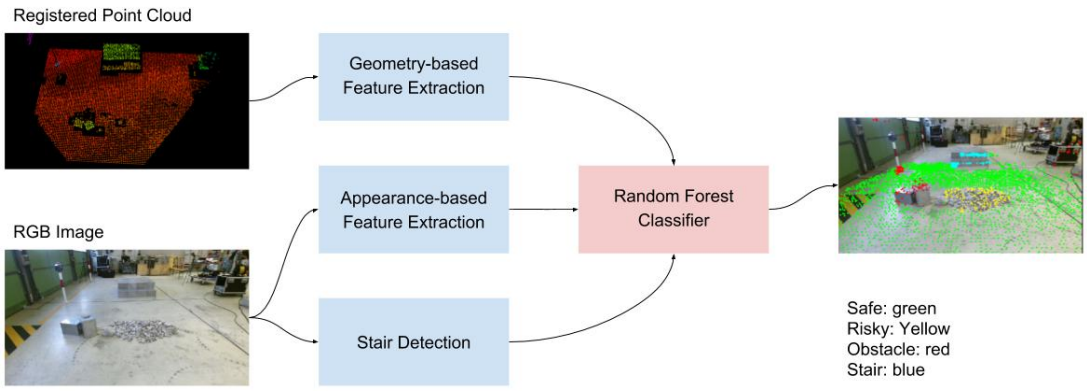

Figure 23: The flow diagram of the terrain classification system.

1. Pointcloud pre-processing,
2. terrain feature extraction, and
3. 2D cost map projection.

The input of the process is the point cloud generated by laser scanner and the output is a 2D cost map that contains a traversability score for each map cell. The laser scanner used in our approach is the Velodyne VLP-16. It scans the environment using 16 sweeping beams which cover the field of view between ± 15 degrees vertically and 360 degrees horizontally. In order to obtain denser point clouds, we register the scans using a Normal Distributions Transform (NDT) presented by Saarinen et al. [21]. An example of a full Velodyne scan registration is shown in Figure 24. The focus of this deliverable is terrain classification which can be evaluated using this local scan alignment. In the integrated system, we plan to use our terrain classification methods in conjunction with the mapping framework being developed in Task 5.1. The corresponding results [23, 8] are enclosed at the end of this deliverable.

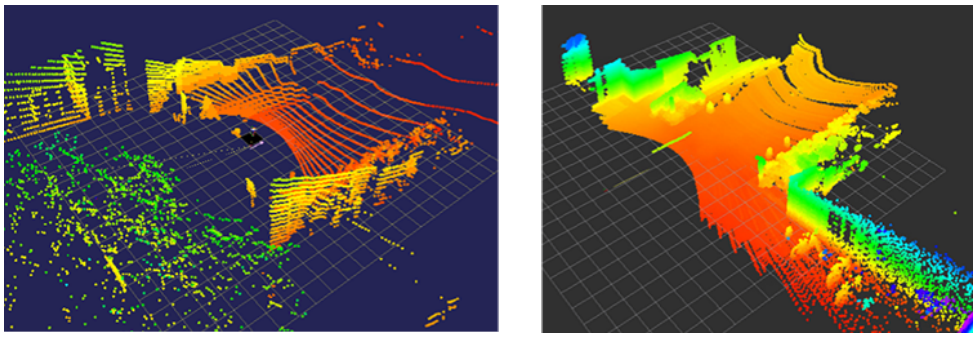

Figure 24: Left: one scan from the Velodyne VLP-16. Right: registered pointcloud from 30 scans.

6.2.1 Pointcloud Pre-processing

From Figure 24, we can see almost one third of the points belong to big obstacles like the wall, which can never be accessed by the robot. In order to speed up the computation we remove the points that belong to obstacles before we calculate the terrain features for each point. Apart from the obstacle points, we define the ground points to be the points where we can place the robot without colliding with any obstacle. For every point in the point cloud, we investigate whether it belongs to the ground or an obstacle. We define a cube surrounding a point to represent the minimum space required by the robot body to perform actions. The robot can lower its body to avoid obstacles above the head and use its legs to step over obstacles with low height. In Figure 25, $h_{\text{head\_min}}$ denotes the minimum height of the robot body and $h_{\text{leg\_max}}$ the maximum height the robot can step on, respectively. The red points between $h_{\text{head\_min}}$ and $h_{\text{head\_min}}$ are the points which cannot be accessed by the robot. A point is classified as a ground point only if there is no points exist between $h_{\text{head\_min}}$ and $h_{\text{leg\_max}}$. An example of the pre-processing result is shown in Figure 26.

6.2.2 Terrain Feature Extraction

Methods for road quality evaluation have been studied extensively. Various standards at national and international level such as ISO 2631 [1], the international roughness index and the riding index [22] are available to represent and evaluate the road quality and riding comfort for the

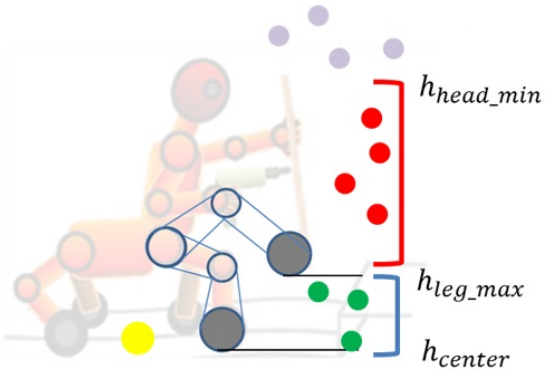

Figure 25: Obstacle points. The robot is placed on the yellow point; the grey points above the robot can be avoided by shrinking down the body; the green points on the bottom can be stepped over using the legs and the red points cannot be accessed by the robot.

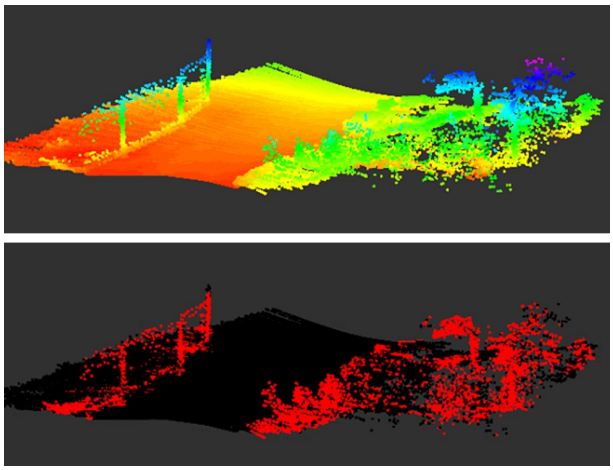

Figure 26: Point cloud pre-processing. The top image is the registered point cloud where the color represent the height. The bottom image is the result of pre-processing, the black points are the ground points and the red points are the obstacles.

passengers sitting in the vehicle. Human perception of the riding quality is highly dependent on the vibrations of the vehicle. ISO 2631 describes an evaluation method on how vibrations affect health, comfort, perception, and motion sickness. The basic evaluation measured in the standard is the weighted acceleration root-mean-square value in a certain time domain.

We select the standard ISO 2631 as one of our terrain feature and slightly modify the standard to use the vertical acceleration root-mean-square value in a spatial domain instead of time domain. The vertical accelerations are the product of the underlying terrain profile. To simplify the problem, we assume there is no friction and deformation between terrain and wheel. Then the vertical acceleration can be calculated purely based on the gravity slope angle of the terrain under the robot, as shown in Figure 27.

For every ground point we define a cube of the same size as the robot's wheel footprint. Four features are computed using all the points inside the cube, which are namely its mean slope, maximum height difference, height variance, and root-mean-square vertical acceleration. These four features describe the terrain characteristics and are independent from the robot model. Both slope and height difference are related to the pose stability when the robot is standing on the point of interest; the height variance can be interpreted as terrain roughness and the root-mean-square of the vertical acceleration captures the robot's expected vibration when driving over a

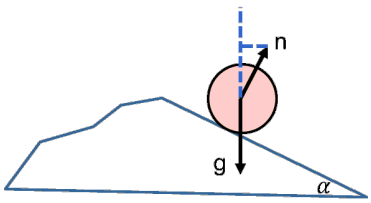

Figure 27: The vertical acceleration is only related to the gravity terrain slope angle a.

specific point.

6.2.3 2D Traversability Cost Map Projection

We define a 2D structured map that contains one traversability value for every pixel to be the interface between terrain features and the navigation component. The 2D map is a convenient structure for data processing and visualization, and it is compatible with most of the off-the-shelf navigation algorithms. We first assign a traversability value to each ground point and then project the points to the horizontal map surface. The traversability value can be designed in different ways using the combination of our four geometric features and the actual robot model. We compute the cost C as

$$C(x,y) = k_1 \cdot \triangle H(x,y) + k_2 \cdot HV(x,y) + k_3 * S(x,y) + k_4 * A(x,y) \quad (1)$$

which is the weighted sum of the maximum height difference $\triangle H(x,y)$, the height variance $HV(x,y)$, the slope $S(x,y)$ and the vertical acceleration $A(x,y)$ at (x,y). We assign a larger weight to the root-mean-square acceleration feature $A(x,y)$ since our platform is very sensitive to body vibrations.

Based on the map resolution, several ground points can be projected onto the same pixel and the pixel traversability value is the mean of the point traversability value that projected on the pixel. A special case has to be considered when the robot is standing in front of a table or a bridge where it can both go over and under the table/bridge. If the points of the ground surface and the points of the table/bridge surface are visible at the same time, they are also projected onto the same pixel. In this case, we cannot fuse them together because they belong to different surfaces. We check the maximum open space between points to identify multiple access paths. If the space is big enough for the robot to go though, we separate them into two groups. We define another traversability map to represent the second accessible surface and the traversability value is the mean traversability computed by the points belonging to each group. Figure 28 shows the process of detecting multiple surfaces.

Two examples of the cost generation is shown in Figure 29. After we project the point cloud to the cost map we apply a morphological interpolation algorithm to fill the holes that are not covered by the sensor data. The color of the interpolated cost map is normalized for better display.

The traversability cost extracted from the laser point cloud measures the quality of the terrain, but when driving on soft terrain, the elevation profile measured by the laser scanner may not be the same profile the robot actually encounters. The laser scan can easily be fooled by the properties of the terrain surface, especially for highly deformable terrain types such as long grass. They can thus be marked as obstacles although they are perfectly drivable. The deformation of wheel and terrain, weight of the robot, terrain sinkage rate and slippage rate will all affect the interaction between the wheels and the terrain surface. Bing et al. [3], Smith and Peng [26], and Smith [27] attempt to simulate the interaction between the vehicle and soft terrain but all

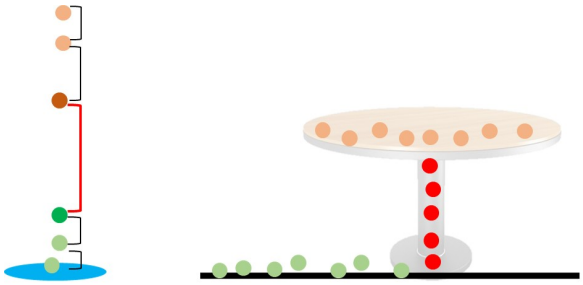

Figure 28: Multiple accessibility detection. The left image shows the case when multiple points are projected to the same pixel. The blue circle represents the pixel; the small circles represent the points. The blocks are the size of open spaces between points and the red block is the maximum open space. The points above and below the maximum open space are separated into two group. The right image shows an example of a table, the green and yellow points are two surface groups and the red points are obstacles.

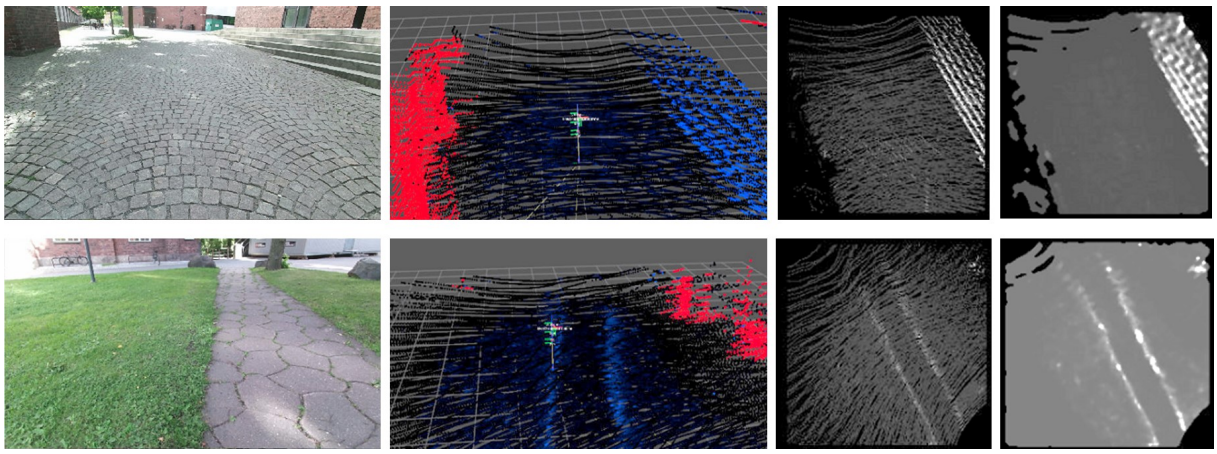

Figure 29: Cost map generation. The first column is the RGB image. The second column is the registered point cloud with cost value. The red points are the obstacles detected during the pre-processing step. The cost value is colored in blue, where darker blue denotes lower cost and lighter blue denotes higher cost. The third column is the cost map projected by the point cloud and the fourth column is the cost map after interpolation.

models mentioned in these papers require additional information that a laser cannot provide. To compensate for this problem, we apply another terrain classification system based on image features to recognize the surface material for each terrain type.

6.3 Vision-based Traversability Analysis

In addition to the information on terrain geometry obtained from the laser scanner, we employ a semantic segmentation architecture to process image data from the RGB camera. The goal of this subsystem is to augment the geometry-based terrain information with semantic labels to gain a better understanding of the scene in general, and specifically to assess its navigability. A semantic understanding of the local terrain is vital in cases where the laser scanner is not able to fully assess its traversability. Two possible scenarios can be identified and defined by the following two examples.

In the first example, the robot is facing a large patch of flat ground and based on laser scanner data alone, the system would suggest to drive over it. If the alleged flat ground were

water, a system that does not assess the appearance of the terrain and issues the drive command will inadvertently put the robot into immediate danger.

In the second example, the robot is surrounded by patches of large height variability and deems the terrain non-navigable solely based on the traversability cost from the laser scanner. However, the environment is made up of (deformable) knee-high grass and bushes, which the robot could handle by walking or even driving. Again, a system that does not take terrain appearance into account might not find a path with sufficiently low risk and thus get stuck in this type of environment.

To address these types of issues, we define a model architecture that is inspired by state-of-the-art convolutional networks for semantic segmentation. In order to perform fast inference on our model, we relax our requirements from per-pixel to superpixel accuracy, which we believe to be sufficient for navigability assessment and path-planning purposes. The model is defined in Section 6.3.1, along with general system requirements and possible limitations.

Since the creation of a large task-specific segmentation dataset is arguably unfeasible, we make use of transfer learning techniques. Section 6.3.2 provides information about the dataset for our baseline model, as well as requirements for task-specific training data for fine-tuning.

6.3.1 Segmentation Architecture

We define a segmentation architecture that can provide both accurate labeling while preserving the spatial structure of the input. Although variable input size is not a strict requirement, we opt for a fully convolutional architecture without the loss of representational power. In order to reduce computational complexity, as well as speeding up training and inference, we employ multiple stages of down sampling between stacks of convolutional layers to reduce the spatial size of the input. The encoder part of our network resembles the popular VGG-16 [25] architecture, whereas the decoder mirrors the encoder with deconvolutions and unpooling layers as shown in Figure 30.

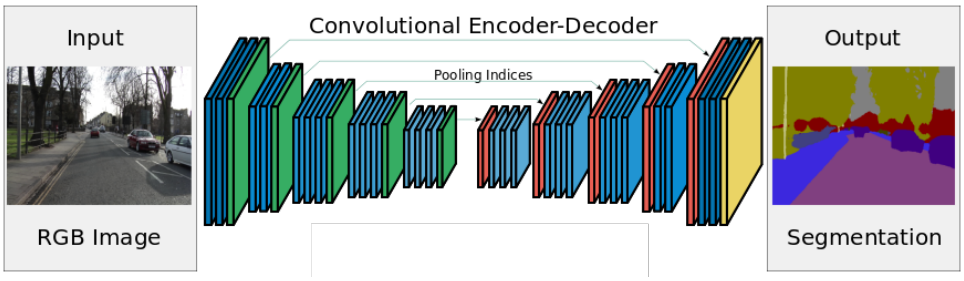

Figure 30: Our networks resembles a convolutional encoder-decoder segmentation architecture. The blue layers denote convolution and deconvolution operations of the encoder and decoder, respectively. The green layers denote spatial max-pooling, whereas the red layers denote spatial unpooling operations according to the saved pooling indices. The final yellow layer denotes a pixel-wise softmax operation. Figure modified from [2].

When down sampling, we save the indices of maximum activations in the encoder part of the network in order to perform up sampling that relates the spatial location of the predictions back to the original input.

6.3.2 Segmentation Datasets

Training deep learning architecture such as convolutional neural networks require a large amount of labeled training data. Popular benchmark datasets for semantic segmentation are, for in-

stance, PASCAL VOC [9] and MS COCO [4]. The main shortcoming of these datasets is their focus on objects in the foreground of the scene, whereas the interest in terrain classification is to assess the navigability of the entire field of view, including the background.

We aim to train our model on a dataset that resembles the terrain classes of the final task to a large extent, the choice fell on the CityScapes dataset [7]. CityScapes is a semantic segmentation dataset geared towards autonomous driving in urban scenes. Interestingly, the group labels of the CityScapes dataset already provide a good approximation for our classes of interest. The interesting classes of the dataset are namely: hard ground (road and sidewalk), soft ground (grass, soil, and sand), vegetation (trees, bushes, shrubbery, etc.), vehicle (car, truck, motorcycle, etc.), construction (buildings, road signs, etc.), and humans. At this point we deemed it to require too many resources to create our own dataset for training before we have tested the full system and identified the bottlenecks in performance.

To enrich the output of the terrain classification system, we added an additional pipeline to recognize the fourth class stair which is explained in Sec. 6.3.3.

6.3.3 Image-based Stairs Detection

For detecting stairs using image data, we have employed a Dilated Residual Network (DRN) [29]. The reason for using a DRN compared to a regular CNN architecture is that DRN has more detailed activation maps compared to a regular CNN architecture. This is shown in Fig. 31. As we go deeper in the regular ResNet, the activation maps get coarser. However, in DRN architecture, thanks to the dilation operation, the activation maps do not shrink. As a result the performance of the DRN is better than the regular ResNet without a big memory penalty.

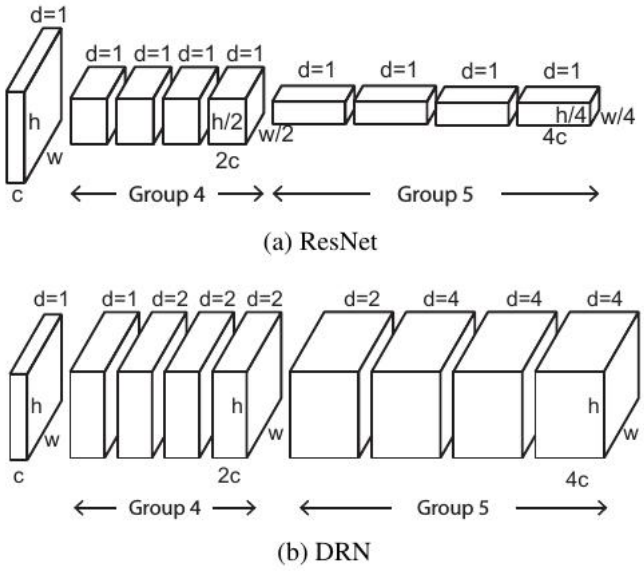

Figure 31: The architectural difference between a regular ResNet and DRN [29].

For stairs detection, we have used the 54-layer Encoder/Decoder architecture of DRN. Data augmentation is applied on the fly to increase the training data variability. Moreover we have added class-specific weights in the loss function to improve the segmentation result. We have setup the stairs detection problem as pixel-wise binary classification. A pixel can be labeled either as a stairs or as background. The output of this network is a pixel-wise probability map of an input image. Some of the obtained results are given in Fig. 32. Here, the pink color represents the pixel with stairs label while the purple color represents the background.

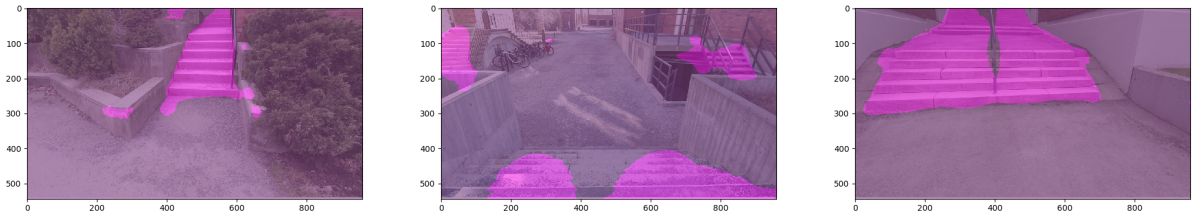

Figure 32: Example stairs detection results.

6.4 Integration Status

The system is fully integrated to the WP5 navigation system and is evaluated together with the full-body navigation planning during the Final Evaluation camp at KHG.

Figure 33 shows input maps and a generated cost map of the test scene. It also shows two path alternatives between which the operator was able to choose. The operator interface such as visualization of height and cost maps, definition of a desired goal pose and visualization and selection of path alternatives was integrated with the VEROSIM visualization.

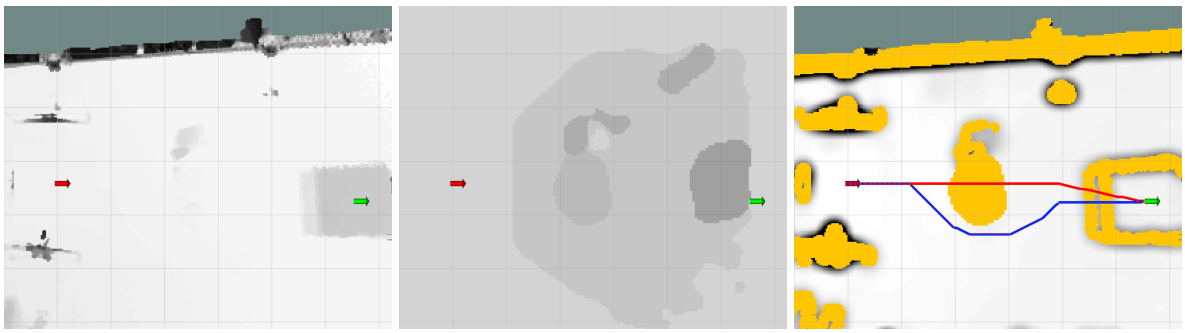

Figure 33: Foot cost map generation for the scenario at the Final Evaluation camp at KHG. The robot starts (red arrow) in front of a field of gravel and has to climb a flight of stairs afterwards to reach the goal pose (green arrow). l.: height map, c.: terrain class map, r.: foot cost map with two path alternatives without (red) and with (blue) consideration for terrain class information.

6.5 Integration Status of the Autonomous Navigation Planner

In D7.3, we already described the hybrid driving-stepping locomotion planner approach with all the required interfaces between sub-modules. Laser scanner range measurements and RGB images are input and used to generate an environment representation. While the former one is fed into a multiresolutional surfel grid module to obtain registered point clouds and localization, RGB images are used for a parallel terrain classification process which incorporates point clouds as well. Subsequently, point clouds are processed to 2D height maps before both, height maps and terrain class maps are fed into the planner. The planner processes all input maps to cost maps. It plans paths from the current robot state, which it receives from the SLAM component, to the desired goal position, which is defined by the operator. Finally, the planned path is given to a controller.

In the current reporting period, we significantly improved the locomotion planner performance, e.g., by employing multiple levels of representation, but those changes had no influence on the interfaces to other modules. In addition, we extended the locomotion planner to generate

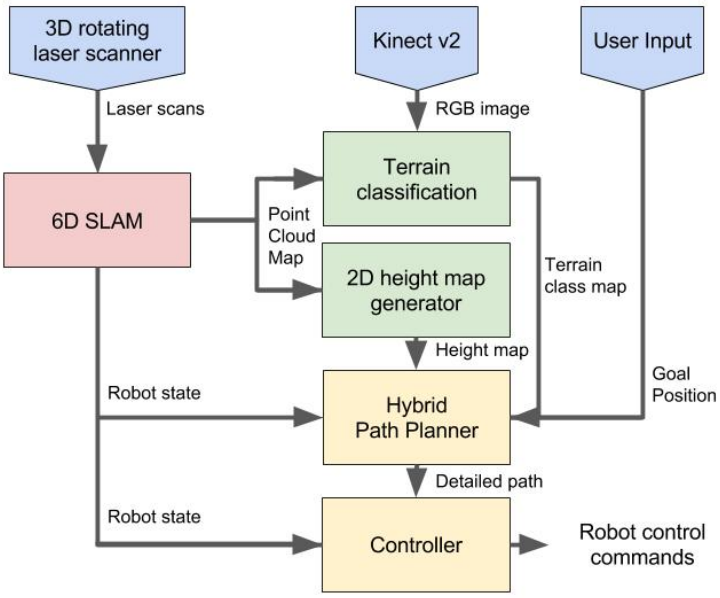

Figure 34: Overview of the locomotion planner pipeline.

two different path alternatives the operator can select from. Finally, we establish the interface to VEROSIM which is described in detail here.

In the current reporting period, we extended the locomotion planner to generate two different path alternatives. In addition we establish the interface to VEROSIM which is described in detail here. Figure 35 visualizes the operation of the planner via VEROSIM.

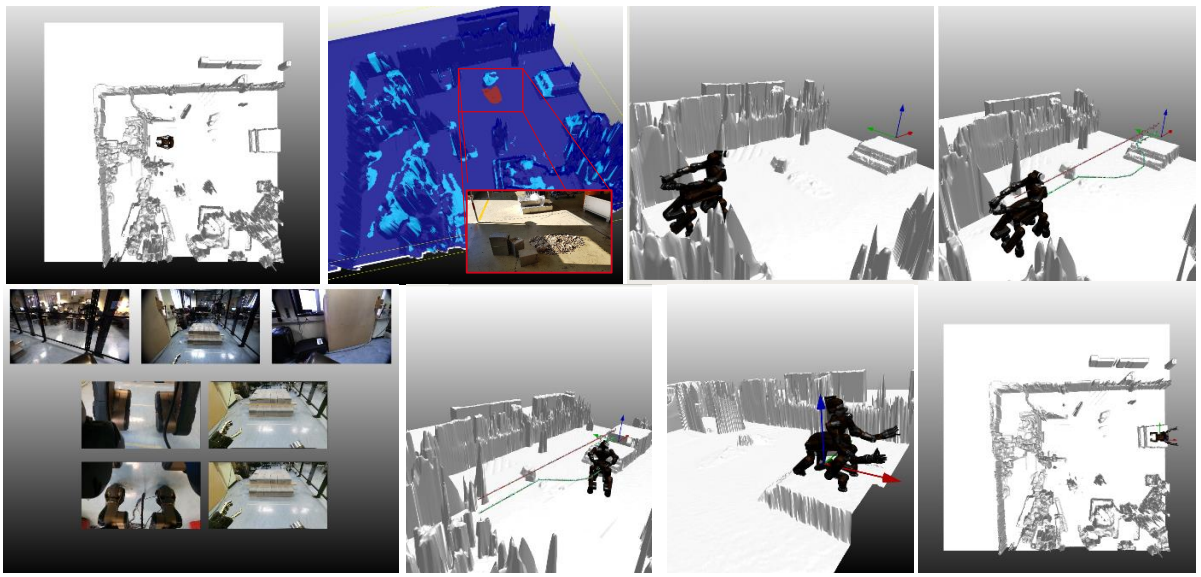

Figure 35: Locomotion pipeline: (top) Height map → Cost map (including impassable gravel on the ground) → Goal pose definition → different paths and select → (bottom) supervised execution of the task.

VEROSIM can be used to visualize data to the operator and to receive input from them which can be sent to other modules. Data visualization is addressed by providing height maps and cost maps to VEROSIM. Besides integrating these 2D maps into the 3D world visualization, VEROSIM is also capable of generating 3D surfaces out of these height maps. The interface definition for those maps is defined in Table 2. The interface has been established and tested

Table 2: Map interface definition between the locomotion planner and VEROSIM.

Data	From	To	Type	Definition
2D height maps	2D height map generator	VEROSIM	ROS message *SingleDataMap*	**Header** header *Size in Cells* **uint32** cells_x **uint32** cells_y *Cell coordinates of (0,0)* **float32** origin_x **float32** origin_y *Resolution (m / cell)* **float32** resolution *Cell values (row step: cells_x)* **float32[ ]** data
Cost maps	Hybrid path planner	VEROSIM	ROS message *SingleDataMap*	s.o.

Table 3: Goal pose interface definition between VEROSIM and the locomotion planner.

Data	From	To	Type	Definition
Desired goal pose	VEROSIM	Hybrid path planner	ROS message *LocomtionGoal*	**Header** header *Base pose* *(orientation in* $[0, 2\pi)$*)* **float32** x **float32** y **float32** orientation

during the Final Evaluation Camp at KHG. Some issues occurred concerning the correct definition of the transformation between the global frame (which is different for the ROS based robot software and VEROSIM) and the *map* frame for which the map is defined. Those issues have been solved.

To operate the autonomous navigation, it is required to define a desired robot goal pose, which is a 3D (x,y,θ) vector of the desired robot base pose. VEROSIM provides a graphical interface to define this pose in the visualized 3D world. The interface, defined in Table 3, has been defined to send this pose to the planner.

After receiving the desired goal pose, the locomotion planner generates two path alternatives which are again sent to VEROSIM in two messages to be displayed. The corresponding interface is defined in Table 4.

Finally, the operator can choose one of the two path alternatives and input this choice via VEROSIM buttons. The information is send to the planner by the interface defined in Table 5. The planner then sends the corresponding path to the controller.

Table 4: Path interface definition between the locomotion planner and VEROSIM.

Data	From	To	Type	Definition
Path	Hybrid path planner	VEROSIM	ROS message *Locomotion-Plan*	**Header** header **LocomotionPose[ ]** poses
			ROS message *Locomotion-Pose*	**Header** header *Base pose* **float32** base_x **float32** base_y **float32** base_orientation *Relative foot poses* **float32** foot_fl_x **float32** foot_fl_y **float32** foot_fl_z **float32** foot_bl_x **float32** foot_bl_y **float32** foot_bl_z **float32** foot_br_x **float32** foot_br_y **float32** foot_br_z **float32** foot_fr_x **float32** foot_fr_y **float32** foot_fr_z *Maneuver information* **bool** base_shift_required **bool** roll_required **bool** step_required **bool** pitch_required **bool** careful_driving_required **float32** step_length **float32** step_max_rel_height **float32** step_rel_end_height **float32** y_com **float32** pitch_angle

Table 5: Path selection interface definition between VEROSIM and the locomotion planner.

Data	From	To	Type	Definition
Path selection	VEROSIM	Hybrid path planner	ROS message *std_msgs::Int32*	**int32** data *value must be either* 1 *or* 2

7 Autonomous Manipulation

Previous to the first evaluation, several methods for object/workspace perception were developed. To facilitate tight integration, final choices had to be made. We will describe the final pipeline (see Fig. 36) in the following.

7.1 Scene Segmentation

We settled on a semantic segmentation method for initial object perception, since it provides the preciseness required for later pipeline stages. In contrast to the last periodic report, we use an improved semantic segmentation method based on the RefineNet architecture [12], which offers higher resolution while retaining high-level features through intermediate upsampling stages.

Since training data is scarce in robotic applications, we generate synthetic training scenes from data captured using a turntable setup. Single training frames are generated by composing extracted object segments on top of a randomly chosen cluttered background image (see Figure 37). For details on this process, we refer to [24].

For known objects, estimating their 6D pose is meaning- and helpful. To this end, we designed an initial pose estimation network that directly predicts translation and rotation com-

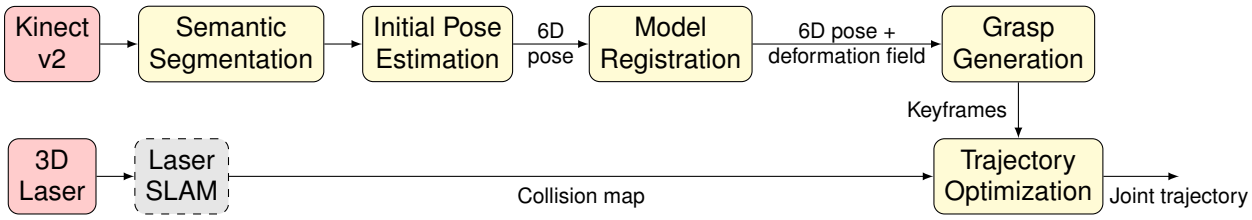

Figure 36: Pipeline for autonomous manipulation. Sensors are colored red, pipeline components yellow, and external modules from other workpackages are colored gray.

Figure 37: 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.

ponents from intermediate feature representations of the semantic segmentation module. The pose estimation is focused on a particular object through cutting, re-scaling, and masking using the semantic segmentation result. See [17] for more details. As an alternative, we can also obtain a rough pose estimate for most oblong objects using PCA on the extracted 3D point clouds. This method has the advantage of requiring no training time at all.

7.2 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 to 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 deformation fields relating the canonical instance to all other instances of the category using Coherent Point Drift (CPD) [14]. Next, we find a linear subspace of these deformation fields, which defines the category's deformation model (Figure 38).

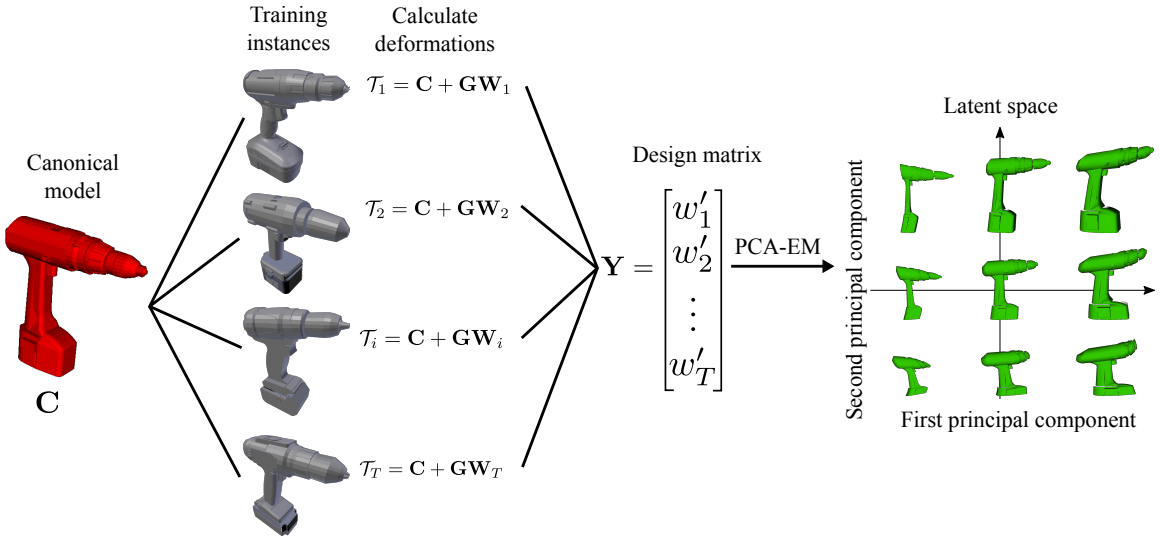

Figure 38: Training phase. Deformations between the canonical model and each instance are calculated by using CPD. The deformations are then assembled into the design matrix $\mathbf{Y}$. Finally, the principal components (latent space) are found by using PCA-EM.

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 [20], this transformation can be expressed as:

$$\mathcal{T}_i = \mathbf{C} + \mathbf{G}\mathbf{W}_i, \tag{2}$$

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.

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.

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. 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:

$$E(\mathbf{x}, \boldsymbol{\theta}) = \sum_{m=1}^{M} \sum_{n=1}^{N} P \left\| \mathbf{O}_n - \Theta(\mathcal{T}_m(\mathbf{C}_m, \mathcal{W}(\mathbf{x})_m), \boldsymbol{\theta}) \right\|^2, \tag{3}$$

where M is the number of points of the canonical model, N is the number of points of the observed instance $\mathbf{O}$, $\mathbf{x}$ is the latent vector, Θ is a function that applies a rigid transformation with parameters θ, and P represents the probability or importance weights between points expressed as:

$$P = \frac{e^{\frac{1}{2\sigma^2}||\mathbf{O}_n - \Theta(\mathcal{T}(\mathbf{C}_m, \mathcal{W}(\mathbf{x})_m), \boldsymbol{\theta})||^2}}{\sum_{k=1}^{M} e^{\frac{1}{2\sigma^2}||\mathbf{O}_n - \Theta(\mathcal{T}_k(\mathbf{C}_k, \mathcal{W}(\mathbf{x})_k), \boldsymbol{\theta})||^2}}. \tag{4}$$

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. Figure 39 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 [20] and [19].

The grasp generation pipeline outputs end-effector poses on the object. These are converted into joint-space trajectories using inverse kinematics and a keyframe-based interpolation system.

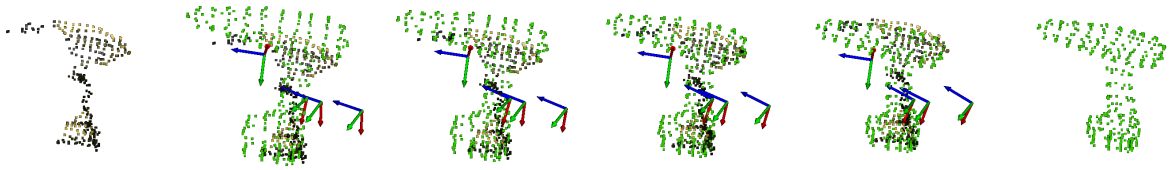

Figure 39: Transferring grasping knowledge to the presented novel instance. The input point cloud is at the leftmost while the inferred shape is at the rightmost.

7.3 Motion Planning & Trajectory Optimization

After joint trajectories have been generated, we can optimize the found trajectory regarding several objectives, such as collision avoidance, joint torques, or speed. We use our stochastic trajectory optimization approach [16] which is based on Stochastic Trajectory Optimization for Motion Planning (STOMP) [10]. The approach was already described in [11].

To summarize, our method optimizes the trajectory of the arm which is represented in joint space. The trajectory is represented as a finite sequence of N waypoints - keyframes. The optimization is performed with respect to a multicriteria cost function which consists of the following components: obstacle costs, joint limit costs, orientation constraints costs, torque costs and duration costs. These costs are calculated for each transition between two consequent keyframes of the trajectory. The final trajectory costs are defined as a sum of transition.

7.4 Operator Interfaces for Autonomous Manipulation

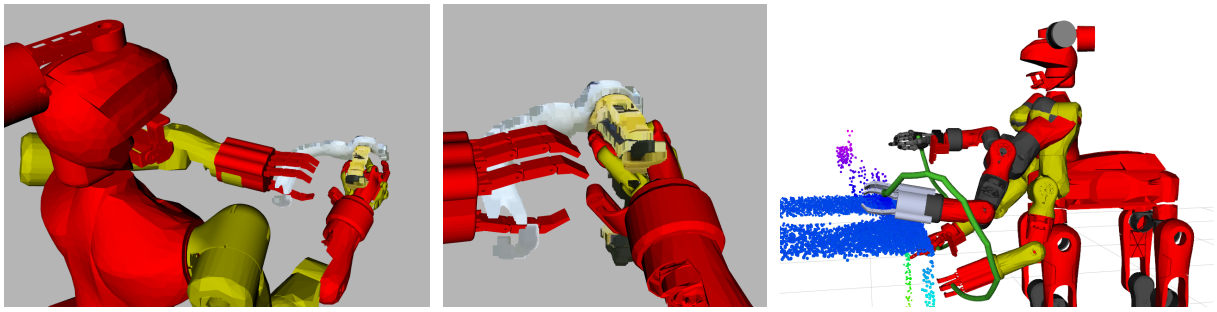

Figure 40: Candidate grasps and trajectories displayed on the support operator interface. *Left:* Found bimanual grasp candidate on a drill. *Center:* Detail view. *Right:* Resulting arm trajectory for reaching the grasp poses.

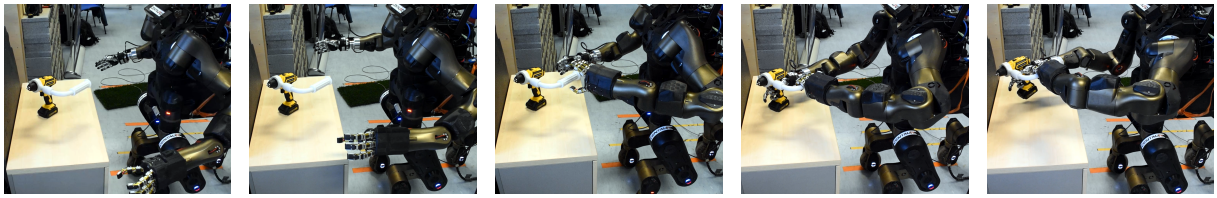

Figure 41: Centauro performing a dual-arm functional grasp of two-handed drill. *From left to right:* Initial pose, reaching the pre-grasp pose, grasping the drill, and lifting it.

Finally, we developed operator interfaces for the secondary operator. Each stage of the pipeline produces results that can be verified by the operator before confirming execution (see Fig. 40): First, the object is segmented and the pose is estimated. After the non-rigid registration step, the transferred grasp poses can be verified. Finally, the generated endeffector trajectory is displayed to the operators. If the operators approve of the results, the motion can be executed. This behavior prevents catastrophic failures while still off-loading a lot of work from the operators.

8 Evaluation

We evaluated the final integrated 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 occuring 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 D8.4 Final CENTAURO System Evaluation* [28].

9 Lessons Learned

In this section, we summarize the experience gained and the resulting insights from the second evaluation camp with particular focus on the integration efforts.

Concerning the actual robot hardware, most of the issues occurred during the first evaluation camp were resolved with the revisions integrated in the final system. The final robot demonstrated stable performance allowing long operations during the evaluation days that sometimes reach or exceed several hours (6-7 hours) of continues operation. The upgraded cooling system and efficient thermal management revisions in the firmware of the robot actuation proved to be effective, eliminating the thermal issues occurred during the first evaluation camp. This permitted the robot to operate for long times even under high joint stress postures, e.g. during the uneven terrain crossing and the stairs climbing task. Particularly the latter task failed last year due to actuation electronics thermal issues but was successfully completed this year in semi-autonomous mode without any thermal issue effect or actuation system saturation.

One minor issue on one of the leg joint encoders during the third evaluation day was immediately identified thanks to the new debugging and monitoring tool developed during the last year which allows the robot operators to monitor the conditions of all robot states in real time, enabling to identify abnormal states that are intuitively visualized through the debugging/monitoring tool GUI. The development of this tool was motivated by the observations we collected in the previous year evaluation camp, where it was evident that a module/component that could detect system abnormal system conditions and notify at high level the operator about potential system warnings and faults and their type could assist and make more efficient the identification of the system issues. The lack of such a tool on the robot software caused large delays during the debugging process in the 2017 evaluation camp but its use in the second evaluation camp proved to be very effective.

A communication issue on the right arm force/torque sensor device during the trials was successfully resolved by sensory substitution and estimation of the right arm end-effector forces using the arm joint torque sensors. This demonstrated the benefit of the proprioceptive sensing redundancy incorporated on the robot with the integration of both joint torque and end-effector force/torque sensing.

Another limitation observed in the 2017 evaluation concerning the robot hardware was on the grasping functionality of the first prototype hand used on the left arm of the robot, making the grasping of some of the tools used in the tasks challenging. To improve on this, in the evaluation camp of this year the left hand of the robot was replaced with the HERI hand end-effector that demonstrated better grasping performance. A minor issue related to the grasping of the small circular valve was due to the particular palm/finger arrangement of the new hand that prevented the hand to conform to this particular valve shape. Concerning the performance of the right hand (commercial Schunk hand) it was evident from the evaluation trials that although this hand is capable in performing grasping of different tools, its physical robustness is limited and great care is needed to prevent damaging it. That can be critical in real operations. This did not prevent, though, its successful employment during the evaluation camp. Nevertheless, the performance of robotic end-effectors in general remains a significant challenge and barrier for the execution of many manipulation tasks. During the evaluation camp, tasks that required to firmly grasp and manipulate small objects like a door key were made easier by using an interface handle between the object and the HERI end-effector as the direct grasping and manipulation of, e.g., the door key was not possible.

Overall, the final robot hardware and low-level software components demonstrated very good and stable performance—permitting the execution of the evaluation tasks with minimal or

no delays for maintenance services. This is a significant progress and improvement with respect to the first release of the robot used in the first evaluation camp.

Two other important upgrades of the robotic platform in the second evaluation camp were the integration of the on-board power module (battery) and the wireless router that permitted the tether-free operation of the robot (this was not possible in the first release of the robot). During the evaluation camp, this allowed us to verify the power autonomy of the system as well as the quality of the WiFi link. The use of the battery in some tasks demonstrated that the robot autonomy can be estimated with good accuracy to be approximately two hours. This was the time of the robot operation during the execution of the uneven terrain trials starting from a fully charged battery. It is considered adequate for short missions but more prolonged autonomy will be required in most cases of real operations.

Concerning the quality of the WiFi link, it was possible to command the robot to execute some of the task trials but it was also evident the effects on the communication latency depending on the location of the ground wireless router with respect to the robot. This introduced some stability issues during the execution of tele-manipulation tasks when the force feedback was introduced, with the latter also affected from latency in the input/feedback loop especially in fast motions due to motion filtering implemented to smooth potential discontinuous reference commands to be sent to the robot joints.

To conclude, the most serious issue occurred during the evaluation camp was an instability on the steering yaw joints of the ankles that happened during the last trial of the evaluation camp while the robot was far from the WiFi ground router operation in in tether-free mode. Prior to this incident, a large latency/drop of the wireless communication link together with a synchronization issue between the on-board computers was observed resulting to step reference commands to be sent to the robot. The reason for this incident is still under investigation by the UBO and IIT teams, although a routing problem in the internal robot network is suspected.

Concerning the use of the tele-presence operation station, an incident on the transmission system of the arm exoskeleton caused some delay on the use of the telepresence station in the manipulation trials. This was eventually fixed and some manipulation tasks were executed successfully though the instability effects mentioned above in some cases under bi-lateral (with force feedback) teleoperation will require further improvement to allow the full utilization and to explore the benefits of this teleoperation mode.

The situation awareness of the operator was improved with the use of the 3D stereo camera providing depth cues to the operator of the tele-presence station. In challenging tasks like the connection of power plugs, where occlusion may occur, additional cameras were needed and installed on the forearms of the robot to provide further views of the grasped objects and their interfaces, e.g. for aligning the male and female electrical plug interfaces. This was also the case when the manipulation tasks were executed using other input devices like 6D mice or when commands were generated using the operator station GUIs.

Similarly for perceiving the workspace underneath the robot pelvis for the purpose of semi-autonomous or fully autonomous locomotion execution, additional perception sensors were needed and installed on the robot and should be considered as a primary upgrade feature in a follow-up version of the robotic platform.

The introduction of the shared control functionality permitting independent control of the lower and upper body of the robot from different operator interfaces, e.g. the upper body through the use of the tele-presence station and the lower body using UBO locomotion interfaces gave flexibility on the execution of manipulations tasks requiring mobility that was not available through the pedal interface of the tele-presence station.

Concerning the VEROSIM simulation tools, issues also arose regarding the rendering per-

formance in the head mounted display but could be resolved on site by swapping calculations into another thread and thus giving the necessary performance to the HMD. Additionally, the first person operator nevertheless was in some situation more satisfied by using a normal camera stream instead of the virtual immersive HMD rendering. In reaction, additional 2D visualizations of the camera streams were added to the 3D HMD display. The lesson learned here is that each situation and operator has different requirements and as a successful execution of a mission is the main goal, one should always use the best suited interface for the given setup.

Testing and evaluating the autonomous modules and functionality remained a very challenging task due to the dependency on critical performance of many different modules and interfaces, e.g., calibration accuracy of kinematic and perception sensors, Lidar measurements, point cloud generation, inertial measurement unit (IMU), inverse kinematics, joint control tracking performance and robot kinematic model precision, etc. The lesson learned is that all these modules have to demonstrate the critical accuracy performance needed and work reliably to achieve a successful execution of autonomous tasks, in which sometimes even small inconsistencies and deviations can affect significantly the execution success as it was the case during the autonomous bi-manual manipulation task or the autonomous stair climbing where small estimation errors on the object pose or the height of the stair step could lead to task execution failures. The robustness of autonomous execution could be significantly benefited through additional unit test evaluation of the individual involved components to ensure that all demonstrate consistency in terms of the critical performance need for autonomous execution. This will also help to understand the level of errors that can be tolerated during the execution. This though will require additional integration and evaluation effort and resources to execute these activities on this high complexity system. Additionally, more closed-loop control is desirable in both cases, where the system could react to changed perception results and thus compensate previous estimation errors. Especially in manipulation, though, this is beyond the current research frontier in many problem settings.

10 Conclusions

This report described the final integrated Centauro system and presented the details of the integration effort and the status of the final system. The validation of the updated CENTAURO disaster-response system including the recently integrated components was performed in the second evaluation camp that took place at the premises of Kerntechnische Hilfsdienst GmbH (KHG), Karlsruhe, Germany, which is part of the German nuclear disaster response organization.

Overall, the final system performance was significantly improved with respect to the first version with most of the issues of the first version fixed and significant new functionality added. This was demonstrated in higher-complexity tasks. Certainly, the experience gained will form a useful foundation for future developments and directions beyond the end of this project that the consortium will look to explore.

References

[1] Mechanical vibration and shock – evaluation of human exposure to whole-body vibration, 1997. ISO 2631.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[3] Zhou Bing, Bi Tianle, and Li Weiping. Ride comfort research of-off road vehicle based on soft terrain. In *Computer-Aided Industrial Design & Conceptual Design (CAIDCD)*, 2010.

[4] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO captions: Data collection and evaluation server. *CoRR*, abs/1504.00325, 2015.

[5] T. Cichon. Deliverable D4.3 Constructing simulated world from robot percepts.

[6] Torben Cichon. Deliverable D4.4 switching between direct control and prediction mode.

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] 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.

[9] M. Everingham, Van Gool L., C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/, 2012.

[10] 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.

[11] 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, and S. Behnke. Deliverable D7.3 First Integrated CENTAURO System.

[12] 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.

[13] 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.

[14] 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.

[15] D. Pavlichenko, D. Rodriguez, M. Schwarz, C. Lenz, A. Periyasamy, and S. Behnke. Deliverable D6.4 Autonomous Dual-Arm Pick and Place Manipulation Skills.

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

[17] Arul Selvam Periyasamy, Max Schwarz, and Sven Behnke. Robust 6D object pose estimation in cluttered scenes using semantic segmentation and pose regression networks. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2018.

[18] Zeyu Ren, Navvab Kashiri, Chengxu Zhou, and Nikos Tsagarakis. Heri ii: A robust and flexible robotic hand based on modular finger design and under actuation principles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018*, 2018.

[19] D. Rodriguez and S. Behnke. Transferring category-based functional grasping skills by latent space non-rigid registration. In *IEEE Robotics and Automation Letters (RA-L)*, pages 2662–2669, 2018.

[20] 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)*, 2018.

[21] Jari Saarinen, Henrik Andreasson, Todor Stoyanov, and Achim J Lilienthal. Normal distributions transform monte-carlo localization (ndt-mcl). In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 382–389. IEEE, 2013.

[22] MW Sayers. *The little book of profiling*. Transportation Research Institute (UMTRI), 1998.

[23] 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, 2016.

[24] 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. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[26] William Smith and Huei Peng. Modeling of wheel-soil interaction over rough terrain using the discrete element method. *Journal of Terramechanics*, 2013.

[27] William Clarke Smith. Modeling of wheel-soil interaction for small ground vehicles operating on granular soil. Technical report, The University of Michigan, 2014.

[28] Uwe Süss, Klas Nordberg, Navvab Kashiri, Luca Muratore, Nikos Tsagarakis, Tobias Klamt, and Sven Behnke. Deliverable D8.4 First CENTAURO System Evaluation.

[29] Fisher Yu, Vladlen Koltun, and Thomas A Funkhouser. Dilated residual networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 3, 2017.