WALK-MAN: A High-Performance Humanoid Platform for Realistic Environments

N. G. Tsagarakis, D. G. Caldwell, F. Negrello, W. Choi, L. Baccelliere, V.G. Loc, J. Noorden, L. Muratore, A. Margan, A. Cardellino, L. Natale, E. Mingo Hoffman, H. Dallali, N. Kashiri, J. Malzahn, J. Lee, P. Kryczka, and D. Kanoulas
Istituto Italiano di Tecnologia, Genoa, Italy
e-mail: nikos.tsagarakis@iit.it
M. Garabini, M. Catalano, M. Ferrati, V. Varricchio, L. Pallottino, and C. Pavan
Centro Piaggio, Universita di Pisa, Pisa, Italy
A. Bicchi, A. Settimi, A. Rocchi, and A. Ajoudani
Istituto Italiano di Tecnologia, Italy and Centro Piaggio, Universita di Pisa, Italy

Received 14 October 2015; accepted 11 July 2016

In this work, we present WALK-MAN, a humanoid platform that has been developed to operate in realistic unstructured environment, and demonstrate new skills including powerful manipulation, robust balanced locomotion, high-strength capabilities, and physical sturdiness. To enable these capabilities, WALK-MAN design and actuation are based on the most recent advancements of series elastic actuator drives with unique performance features that differentiate the robot from previous state-of-the-art compliant actuated robots. Physical interaction performance is benefited by both active and passive adaptation, thanks to WALK-MAN actuation that combines customized high-performance modules with tuned torque/velocity curves and transmission elasticity for high-speed adaptation response and motion reactions to disturbances. WALK-MAN design also includes innovative design optimization features that consider the selection of kinematic structure and the placement of the actuators with the body structure to maximize the robot performance. Physical robustness is ensured with the integration of elastic transmission, proprioceptive sensing, and control. The WALK-MAN hardware was designed and built in 11 months, and the prototype of the robot was ready four months before DARPA Robotics Challenge (DRC) Finals. The motion generation of WALK-MAN is based on the unified motion-generation framework of whole-body locomotion and manipulation (termed loco-manipulation). WALK-MAN is able to execute simple loco-manipulation behaviors synthesized by combining different primitives defining the behavior of the center of gravity, the motion of the hands, legs, and head, the body attitude and posture, and the constrained body parts such as joint limits and contacts. The motion-generation framework including the specific motion modules and software architecture is discussed in detail. A rich perception system allows the robot to perceive and generate 3D representations of the environment as well as detect contacts and sense physical interaction force and moments. The operator station that pilots use to control the robot provides a rich pilot interface with different control modes and a number of teleoperated or semiautonomous command features. The capability of the robot and the performance of the individual motion control and perception modules were validated during the DRC in which the robot was able to demonstrate exceptional physical resilience and execute some of the tasks during the competition.

1. INTRODUCTION

1.1. The Disaster Response Challenge

Recent natural disasters such as the 2011 earthquake and tsunami in Japan and subsequent problems at the Fukushima nuclear power plant have dramatically highlighted the need for effective and efficient robotic systems that can be deployed rapidly after the disaster, to assist in tasks too hazardous for humans to perform. The conditions that a disaster response robot will encounter during a mission in a harsh environment can vary depending on the nature of the physical catastrophe or man-made crisis. To operate and be effective though within realistic unstructured environments designed for humans or in environments that have become hostile or dangerous, a robot should possess a rich repertoire of capabilities and be able to demonstrate excellent performance.

Direct correspondence to: N.G. Tsagarakis, e-mail: nikos.tsagarakis@iit.it

Concerning locomotion the robot should to be able to handle and navigate over debris, terrains, and pathways of different characteristics and difficulty, ranging from flat terrains to structured uneven terrains and inclined surfaces and finally unstructured rough ground with partially or totally unstable foothold regions and randomly varying height maps. The ability to use human-designed equipment such as ladders to gain access to elevated areas is also a fundamental requirement. It should also have the ability to transverse passages that have limited ground support, and walk through narrow gaps that require versatile locomotion with significant body posture maneuvering and sharp turning capabilities. In a harsh environment, the robot should be capable of these unmodified human tools for solving complex bimanual manipulation tasks, such as connecting a hose or opening a valve, in order to relieve the situation or for performing repairs. It should also be able to autonomously perform elementary manipulation tasks, such as grasping or placing objects while also having the manipulation strength and power capacity to exert significant forces to the environment, for example, lifting/carrying or pushing collapsed debris to open the path way, operate heavy power tools to break concrete blockages, apply strong forces to open blocked doors, or generate the forces need to turn on/off valves and other related heavy duty power or fluidic line switches.

1.2. Motivation for Robot Embodiment

The above requirements pose significant challenges for existing disaster response mobile manipulation platforms based on wheeled or caterpillar mechanisms. These robots may provide optimal solutions for well-structured and relatively flat terrains environments; however, outside of these types of workspaces and terrains, their mobility decreases significantly and usually they can only overcome obstacles smaller than the size of their wheels.

Legged robots have an advantage in these ground/terrain conditions as they have the kinematic capabilities to adapt to terrain variations and successfully maintain the robot body postural state in a well-balanced equilibrium. The number of legs used in such a robot has a significant effect on the overall mobility performance in terms of balance stability and locomotion versatility. Quadruped robots demonstrate better balance stability and are more tolerant and robust against external perturbation and contacts. The body profile though of a quadruped robot can form a limited factor that can prevent quadrupeds from being able to walk through narrow spaces with limited support pavements or perform sharp turning and maneuvering as well to climb ladders. Bipedal systems of humanoid form are more difficult to control in terms of balancing due to limited support area and a relative high center of mass (CoM). Bipedal systems though have better mobility versatility and they can cope with situations such as passing through narrow gaps and spaces, walk on very limited support pavements, and climb ladders and stairs. Furthermore, although their balancing control represents a major challenge when coping with uneven terrains, they have advantage over the quadrupeds as their body kinematic flexibility potentially allows them to execute bipedal locomotion under severe postural modulation or even to switch to other forms of more stable locomotion in challenging situations such as crawling and quadruped.

Apart from manipulating the dynamic/mobile environment contacts with the static environment through the manipulation physical interface (arms and hands), arms and hands can potentially also enhance the locomotion capabilities of the robot and its ability to balance while crossing uneven grounds or during climbing stairs and ladders. Considering all these locomotion and manipulation capabilities and the need for the robot to be compatible with human environments and tools and be able to perform multiple tasks and also be compatible with the human operators potentially working close, it is evident that robots designed with specialized functionality are not suitable and have limited capabilities. The most suitable form of robot that is compatible for such needs and comes to our mind is the robot that has a humanoid form and embodiment. The WALK-MAN platform was therefore designed to have a humanoid form.

1.3. Literature

During the past two decades, there has been considerable progress in the mechatronic development of humanoids and bipeds, with robots based on designs ranging from those with entirely passive dynamics to fully powered systems having been explored. The first modern humanoid, WABOT-1, formed the template for most subsequent designs. Hence, ASIMO, which is one of the best-performing powered humanoids, was developed from E0 (1986), E1-E2-E3 (1987–1991), E4-E5-E6 (1991–1993), P1-P2-P3 (1993–1997), through to the original ASIMO (2000), and the new ASIMO (2005) (Hirai, Hirose, Haikawa, & Takenaka, 1998). The P3 prototype unveiled in 1998 (Hirose & Ogawa, 2007) was one of the breakthrough designs, initiating and spurring research on a number of other key platforms. The Humanoid Robot Platform (HRP) started with an adapted Honda P3 and subsequently HRP-2L/2P/2/3/4 were released (Akachi et al., 2005; Kaneko, Harada, Kanehiro, Miyamori, & Akachi, 2008). Similarly, KAIST built KHR-1/2/3 (Hubo) (Park et al., 2007), Waseda continued its long and successful tradition to build many different models through to Wabian-2R (Ogura, Aikawa, Shimomura, Morishima, & Lim, 2006), and University of Tokyo look at improving the power performance of humanoids (Ito, Nakaoka, Urata, Nakanishi, Okada, & Inaba, 2012). The iCub humanoid represents a coordinated European effort in the humanoid arena aiming at producing a "child-like" humanoid platform for understanding and development

of cognitive systems (Parmiggiani et al., 2012; Tsagarakis et al., 2007), but other successful humanoid/bipedal implementations within Europe include the humanoid LOLA, which is an enhancement over the Johnnie robot (Lohmeier, Buschmann, Ulbrich, & Pfeiffer, 2006), and the recently developed torque-controlled humanoid TORO (Englsberger et al., 2014).

There are two main actuation approaches in the development of humanoids with impedance modulation versatility and improved full body motion agility skills. Robots such as PETMAN and ATLAS take advantage of the increased mechanical robustness, high power (up to 10 kW/kg), and torque control bandwidth offered by hydraulic actuation to improve the dynamic performance and external perturbation (impact/interaction) rejection.

These hydraulic powered systems as well as those actuated by stiff motorized units rely on sensors and software control to regulate their intrinsically very high mechanical impedance and replicate compliant behaviors. Safety is a very real concern making them unsuitable when operating in human-centered environments, while their energy efficiency is still remains a major hurdle. Further, the high mechanical impedance and the lack of any physical elasticity do not allow all humanoids, which are powered by stiff motorized or hydraulic actuation, to make use of their natural dynamics.

The second common actuation technique currently used to improve the motion performance of humanoids and reduce their intrinsic mechanical impedance uses physically compliant actuation systems. Here, elasticity is introduced between the load and the actuator to effectively decouple the high inertia of the actuator from the link side. The series elastic actuator (SEA; Pratt & Williamson, 1995), which has a fixed compliance element between a high impedance actuator and the load, was one of the earliest of these designs. State-of-the-art robots powered by SEAs include notably the M2V2 bipedal robot (Pratt et al., 2012), the NASA-JSC Valkyrie humanoid robot (Paine et al., 2015), and the IIT compliant humanoid COMAN (Tsagarakis, Zhibin, Saglia, & Darwin, 2011; Tsagarakis, Cerda, Li, & Caldwell, 2013). Two of the main benefits of the SEA actuation are the physical protection provided to the reduction drives due to the impact torque filtering functionality and the improved tolerance and accommodation of unexpected interactions constraints or inaccuracies. Furthermore, introducing fixed compliance improves the fidelity of torque control at low bandwidths and robustness (Pratt & Williamson, 1995), and may have energetic benefits (Laffranchi, Tsagarakis, Cannella, & Caldwell, 2009). However, it also imposes constraints on the control bandwidth, and these systems are less able to quickly generate forces and motions. The level of incorporated compliance is therefore a significant parameter of the design of SEA actuated systems.

The capability of dynamic whole-body motion is mainly stemmed from the abundant kinematic redundancy of humanoid robots, allowing us to utilize the torso and arms to assist balancing and locomotion. Since the pioneering work on generalized inverse kinematics (IK; Nakamura & Hanafusa, 1987), a concept of task-priority based on IK (Nakamura et al., 1987; Siciliano & Slotine, 1991) has been proposed to enhance the capability of redundant robot manipulators to perform a multiple number of tasks. One major work enabling whole-body manipulation of highly redundant robots in the Cartesian task space was the operational space framework (Khatib, 1987), where a force-level redundancy resolution provides the useful property known as dynamic consistency. The operational space formulation is first introduced for fixed-base robotic manipulators, and it is extended to control whole-body behaviors for humanoid robots performing multiple tasks, and further developed to deal with multiple contacts including the floating base (Sentis, Park, & Khatib, 2010; Sentis, Petersen, & Philippsen, 2013). Recent development of hierarchical quadratic programming (QP) further exploits a dynamic motion to execute multiple tasks including equality and inequality constraints (Escande, Mansard, & Wieber, 2014; Herzog, Righetti, Grimminger, Pastor, & Schaal, 2014; Saab, Ramos, Keith, Mansard, Soueres, & Fourquet, 2013). Some of these frameworks have been implemented in popular libraries such as the Stack of Tasks (Mansard, Stasse, Evrard, & Kheddar, 2009) and iTasc (De Schutter et al., 2007), and recently in the modern ControlIt! (Fok & Sentis, 2016) software.

Despite the advancements made in the above technologies and their application in some excellent humanoids platforms, significant barriers still remain, preventing robot hardware (physical structure and actuation) and humanoid control from reaching closer to the performance of human in locomotion and whole-body motion capability and performance.

While WALK-MAN is based on an actuation principle utilizing SEA drives, it contains unique performance features that differentiate the robot from previous state-of-the-art compliant actuated robots. Driven by the hypothesis that the level of physical interaction performance is the result of both active and passive adaptation, WALK-MAN actuation combines customized high-performance modules with tuned torque/velocity curves with transmission elasticity to provide high-speed adaptation response and motion reactions to disturbances. To the authors' knowledge, the power (torque/velocity) capabilities of WALK-MAN actuation exceed the performance of the actuation drives of the previous developed motorized/compliant humanoid robots.

Furthermore, WALK-MAN design incorporates design choices based on optimization studies to select the kinematic structure for the legs (hip) and the arm (shoulder), and make use of actuation relocation along the body structure to maximize the robot dynamic performance.

Concerning the motion generation and control, a novel library has been developed with the objective to be also

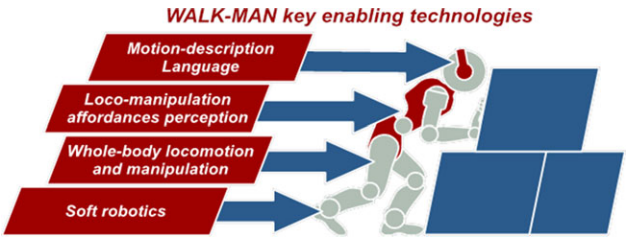


Figure 1. Enabling technologies of WALK-MAN towards the development of a humanoid that will be capable of going outside the lab environment and to operate in destructured spaces

flexible and extensible. The library has high modularity through the separation of task descriptions, control schemes, and solvers implementation. Furthermore, it provides a large set of already implemented tasks that can be combined to design complex whole-body motions.

1.4. WALK-MAN Objectives and Contribution

WALK-MAN humanoid robot was developed within the European Commission project WALK-MAN (www.walk-man.eu), which aims to develop a humanoid robot that will demonstrate the following three challenging skills: (1) powerful manipulation—for example, turning a heavy valve of lifting collapsed masonry, (2) robust balanced locomotion—walking, crawling over a debris pile, and (3) physical sturdiness—for example, operating conventional hand tools such as pneumatic drills or cutters.

The work on the development of the WALK-MAN robot made use of two main concepts (Figure 1) to achieve these goals.

- The use of powerful, yet soft actuator technologies combined with proprioceptive sensing and active impedance control, to provide more natural adaptability, interaction, and physical robustness.
- An integrated framework to whole-body locomotion and manipulation (termed loco-manipulation) and the development of loco-manipulation primitive behaviors that link and control the robots perception and action at the whole-body level.

WALK-MAN (Figure 2) should eventually possess sufficient abilities to allow it to operate semiautonomously or under teleoperation and show human levels of locomotion, balance, and manipulation during challenging operations.

This paper provides an overview of the WALK-MAN humanoid platform with emphasis on the mechatronic developments and integration.

On the hardware side, one of the main contributions of WALK-MAN design is its actuation system and the design of the robot in general that consider innovative design optimization features including the selection of kinematic structure for the legs and the arms and the placement of the actuators with the body structure to maximize the robot performance. The physical robustness of the robot is ensured with the integration of elastic transmission, proprioceptive sensing and control, and impact absorbing covers.

Another contribution of this work is the introduction of WALK-MAN software framework details and the integration aspects adopted during the development phase that was time limited (approximately one year). The WALK-MAN software architecture is discussed in this paper describing the complete software stack: custom firmware, control modules tackling different tasks, a remote pilot graphical interface, and the whole architecture to manage and connect the different applications. Our approach considered a layered component-based architecture where each task of the DARPA Robotics Challenge (DRC) is handled by a single control module and modules interact with the hardware and each other through well-defined APIs. In that way and once a rough and primitive API was defined, modules could be developed in parallel, in the meantime shared functionalities could be improved under the hood of the high-level control software without requiring code changes.

The developed whole-body loco-manipulation framework is another contribution of WALK-MAN development. The framework was designed to be flexible and easily extensible. Furthermore, it was developed to be extremely modular through the separation of task descriptions, control schemes, and solvers implementation. We provide an overview of this framework that enables WALK-MAN to execute loco-manipulation behaviors synthesized by combining different primitives defining the behavior of the center of gravity, the motions of the hands, legs and head, the body attitude and posture, and the constrained body parts such as joint limits and contacts.

Finally, the features of the user interface developed for the participation to the DRC are presented. This pilot interface allows the operators to drive WALK-MAN with different control modes and a number of teleoperation or semiautonomous command features.

The following sections introduce details on how the above technologies were implemented and integrated to develop WALK-MAN and effectively demonstrate its capabilities during the DRC Competition Finals.

2. WALK-MAN MECHATRONICS

2.1. Mechanics Overview

WALK-MAN humanoid (Figure 2) approximates the dimensions of an adult human; its height from the foot sole to the head top is 1.915 m. The width between the two shoulders is 0.815 m, while its depth at the torso level is 0.6 m. The total weight of the WALK-MAN robot is 132 kg of which 14 kg is the weight of the power pack and 7 kg is the weight of the protection roll bar structure around the torso and head.

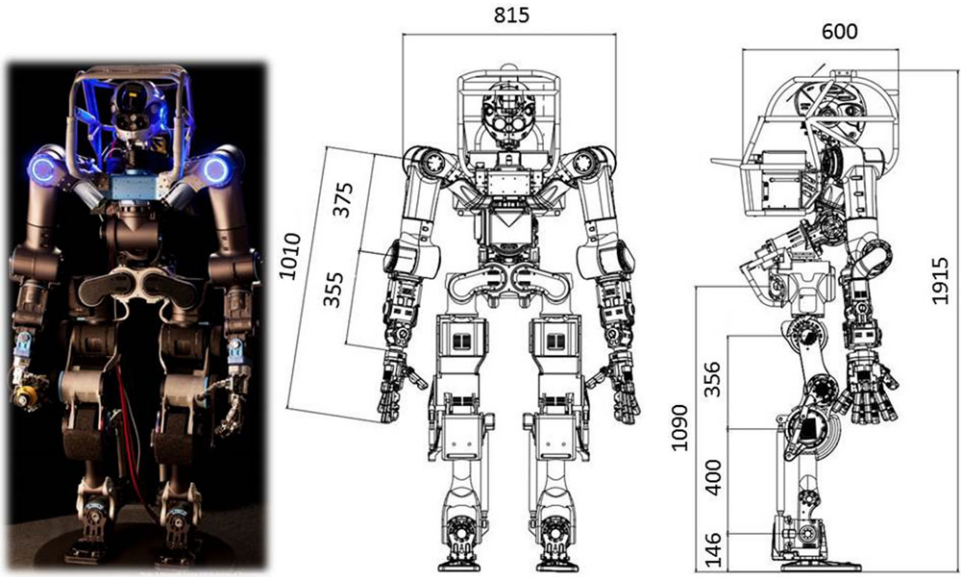


Figure 2. WALK-MAN body size specifications; all dimensions are in mm

The design of the WALK-MAN robot has been driven by the following objectives: (1) high power-to-weight ratio and reduced inertia at the legs to maximize dynamic performance, (2) large joint range of motion to achieve human like movement, and (3) enhanced physical sturdiness. A number of innovative design optimization features were considered to address the above objectives and maximize its physical performance. These design features include the selection of kinematic structure, the arrangement of the actuators, and their integration with the structure to maximize range of motion, reduce the limbs mass and inertia, and shape the leg mass distribution for better dynamic performance. Physical robustness is ensured with the integration of elastic transmission and impact energy absorbing covers. Kinematics, mobility, overall size, and structural strength together with actuation performance (strength, power, speed, range of motion) have been defined considering the requirements of the intervention scenario defined in collaboration with Italian Civil Defence Corps and from the requirements imposed by the current rules and task definition adopted in the DRC. One of the key technologies of the WALK-MAN robot is the new Series Elastic high-end Actuation (SEA) unit that has been explicitly designed for the purpose of the project. This actuation can demonstrate high power density and excellent physical robustness during impacts. This performance is combined with other important engineering aspects, such as modularity, scalability, and reliability, together with uniformity of interfaces, costs, and maintenance, in order to create a platform capable to match with the requirements and challenges that a humanoid robot design imposes.

2.1.1. Upper Body Design

The WALK-MAN upper body (excluding hands and neck) has 17 degrees of freedom (DOF), each arm has 7 DOF, and the trunk has a 3 DOF waist. WALK-MAN arm kinematics closely resembles an anthropomorphic arrangement with 3 DOF at the shoulder, 1 DOF at the elbow, 1 DOF for the forearm rotation, and 2 DOF at the wrist (Figures 3(a) and (d)). This is a typical arm configuration that will provide the humanoid the ability to manipulate the environment with adequate dexterity as well as using the one additional degree of redundancy in the arms to cope with constraints that may be introduced in the task space by the surrounding environment. Concerning the length of WALK-MAN arm segments, it is evident that this does not follow the anthropomorphic ratio with respect to the size of the rest of the body parts. The choice to extend the length of the robot arms was made for the purpose of enlarging the manipulation workspace and particularly for reducing the distance of the hands from the ground level. This makes easier to approach and grasp objects located at low height without the need to perform severe torso posture bending. In addition, the longer arms were adopted considering that they will be probably more effective in reaching the ground or the surrounding environment during critical balancing recoveries to prevent failing.

To derive the values of the upward angle and forward angle of the WALK-MAN shoulder frame, we perform an optimization in which important manipulation indices were considered and evaluated in a prioritized order (Bagheri, Ajoudani, Lee, Caldwell, & Tsagarakis, 2015). The range of

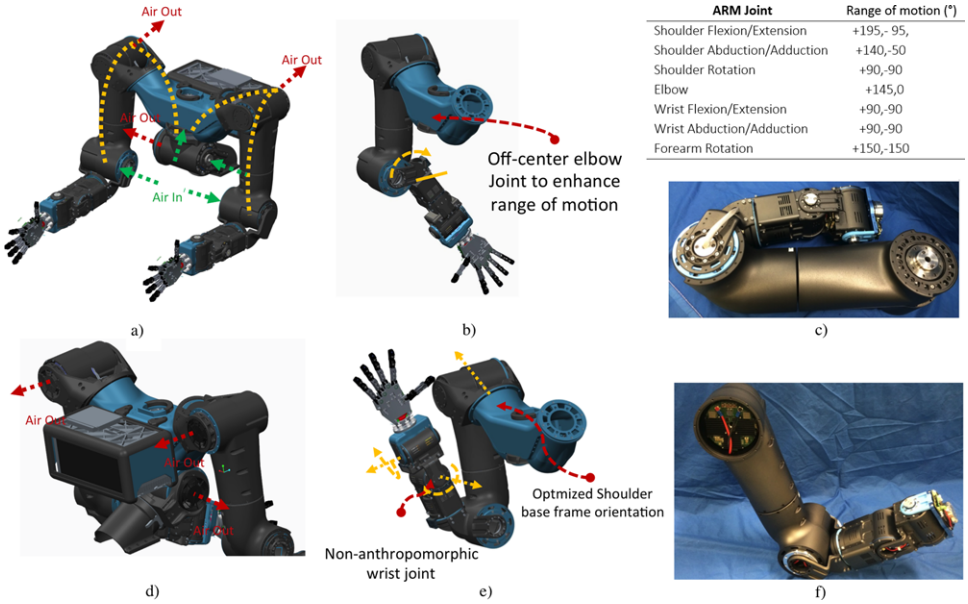


Figure 3. The overall WALK-MAN torso kinematic structures and features. Panels (a) and (d) show a 3D CAD view of the torso and highlights the cooling system working principle together with the power-pack integration and arm kinematics. Panels (b) and (e) show in detail the kinematics of the arms and the optimized displacement of the actuated joints, finally, panels (c) and (f) show some pictures of the arm prototype highlighting the anthropomorphisms of the structure both in terms of kinematics and ranges of motion

motion of a standard human was used as a starting point. Wherever possible, a greater joint range of motion was considered to enhance the motion and manipulation capability of the arm (Figures 3(b) and (e)). In particular, the range of wrist and elbow joint were significantly extended. This was done by considering an off-center elbow joint arrangement (see Figure 3 (b)) for the latter that results in a wide elbow flexion joint and a nonintersecting axis wrist joint that provide a large range of motion for both wrist pitch and yaw motions. The actuation of the arm (see Figures 3(c) and (f)) was based on the integration of seven SEA units along the kinematic structure of the arm. The actuators of the arm are based on the modular design principle of the actuator unit introduced in Section 2.2. For the interconnection of the actuator units, the arm design followed an exoskeleton structure approach in which the body of the actuator is floating inside this exoskeleton structure and the actuator is fixed to the structure and the follow link using only two flanges located at the same side of the actuator (Negrello et al., 2015). Finally, at the same time the exoskeleton cell structure design forms a closed tunnel in which forced air is used to cool down the actuators that are floating inside the cell structure (see Figures 3(a) and (d)).

The end-effector is designed with an anthropomorphic shape to adapt to objects, tools, and fixtures designed to match the ergonomics of the human hand. To increase the robustness, reliability, and efficiency of the system, the design approach has been based on a substantial and guided reduction of the complexity, concerning both aspects of mechanics and control. The Pisa/IIT SoftHand (Catalano, Grioli, Farnioli, Piazza, & Bicchi, 2014) is the ground platform from which the new end effector has been developed, taking as reference three main guideline principles: a *Synergy-based framework*, a *Soft joints design*, and the use of *Soft materials*. The first principle allows a simplification in the actuation layout, control approach, and grasp capabilities. The WALK-MAN hand has 19 DOF, distributed in an anthropomorphic structure and a single actuation unit. Power from the actuation unit is transmitted to all joints with a distributed differential mechanism. The distribution system is obtained through the employment of a tendon-based structure that connects all the joints of the hand (Catalano et al., 2014). To make the system able to withstand powerful impulsive events (as impacts) and nonforecast mechanical solicitations (as finger disarticulations and clenching) a *Soft joints design* is pursued. The fingers joint are rolling joints kept together by an elastic ligament that also implements the elastic return force (Catalano et al., 2014). To further improve the robustness and the adaptability of the hand, the fingers of the WALK-MAN hands are covered by an outer shell made of printed soft polymer, which also provides suitable friction coefficients for those parts of the hand that need to come in contact with objects to be grasped and manipulated. The choice of realizing these components

as outer shells makes them easy to substitute in case of damage or tearing due to use.

Figure 4 shows a picture and a 3D view of the WALK-MAN hand implementation and highlights its overall dimensions. The hand is actuated by a KollMorgen 30 Watts motor (RBE 00510) with a Harmonic Drive HFUC-8-100 with a reduction ratio of 100:1. The actuation system act on a Dynema fiber ligament with a diameter of 0.8 mm and a maximum strength of 1100 N. Fingers, palm, and wrist interface are built with high-strength aluminum alloy, electronic boards are placed in the wrist interconnection and protected by an aluminum alloy frame. Fingers and palm are covered by a special soft rubber shell with a hardness of 40 SHORE. The total weight is approximatively 1.3 kg. The hand is capable to exert a maximum static grasp force (in power grasp) between 80 and 150 N and a maximum static grasp torque (in power grasp) between 2 and 5 N m (all these values change in function of the closure of the hand). Moreover, it is capable of exerting a maximum static vertical lifting force of 160 N. Figure 5 shows the hand performing different tasks: grasping a drill tool, driving a car, manipulating a valve, and grasping a wood block.

2.1.2. Lower Body Design

The WALK-MAN lower body (Negrello et al., 2016) has 12 DOF, 6 DOF for each leg. WALK-MAN leg kinematics closely resembles an anthropomorphic arrangement with 3 DOF at the hip level, 1 DOF at the knee, and 2 DOF at the ankle (see Figure 6).

To improve the dynamic performance of the leg, it is beneficial to minimize the leg inertia. This allows us to increase the peak acceleration and velocity of the joints; this is particularly important for the pitch joints to allow fast swing motions as well as reduce the disturbance generated by this motion to the rest of the body. To achieve this, the hip complex has the roll-Yaw-Pitch hip configuration, depicted in the top right of Figure 6, which places the pitch motion at the last DOF of the hip.

To further reduce the leg inertia seen at the pitch joints, the mass of the leg should be distributed closely to the hip and in general as close as possible to the upper leg. To achieve this, the knee and ankle pitch actuators have been relocated upward with the knee pitch motor placed at the thigh just after the hip pitch, and the ankle pitch motor is located inside the knee joint. This has an influence on the adopted leg design and on the transmission system as is shown in the bottom right of Figure 6. The transmission of the motion from the relocated knee and ankle pitch actuators to the corresponding joints has been realized using the 4-bar mechanisms shown in Figure 7. Although this actuation relocation approach and the use of two 4-bar mechanisms add some complexity, it is significantly beneficial for the reduction of the leg inertia, the effect of which can be seen in the graphs on the bottom-right side of Figure 6. It can be seen in the left graph that placing the knee pitch actuator immediately after the hip pitch joint and in a distance approximately of $d_k = 100$ mm results in almost half thigh inertia compared to the thigh inertia when the knee pitch actuator is placed inside the knee and in a distance of $d_k = 360$ mm from the hip pitch joint center. Similarly in the second graph on the right, it can be observed that placing the ankle pitch actuator inside the knee joint ($d_a = 0$ mm) results in about 25% reduction in the calf inertia compared to the case when the ankle pitch actuator is placed at the ankle ($d_a = 400$ mm)

For selecting the leg joint range of motion in some cases it was necessary to enlarge the desired ranges due to the constraint of the design or to allow a larger mobility for a particular task. In general, pitch joints are more relevant for those tasks related to forward stepping, squatting, up-the-hill walking, etc. Note that in WALK-MAN the feet have no toe articulation; thus the ankle pitch joint range has been enlarged to compensate it for performing deep squatting motion (Figure 8) or other motions requiring large ankle-pitch. The roll joints are directly related to lateral stepping, lateral stability, and leg crossing. Pitch joints (hip, knee, ankle) are powered by high-power actuators, while hip yaw and ankle roll have medium-power actuators.

The foot of WALK-MAN, Figure 8 (right), has a flat plate profile composed of four layers that implement a shock absorbing structure. Two metal plates encompass a rubber layer that acts as an impact absorber. The relative motion of the two metal plates, obtained via a proper conformation of their edges, allows the compression of the rubber along the vertical direction only. A final rubber layer is mounted at the bottom metal plate to increase the grip between the foot and the ground.

For the purpose of monitoring the resultant forces at the end effector, the foot incorporates a custom six-axis force torque sensor.

2.2. Actuation

The WALK-MAN actuator consists of a frameless brushless DC motor, a harmonic drive (HD) gearbox, with reduction ratios between 80:1 and 120:1 (G) depending on the joint, and a flexible element (a torsion bar) that connects the output of the gearbox to the output flange of the actuator (Figure 10). The assembly of the motor, HD, and the torsion bar, as well as the actuator housing, was fully customized to reduce size and weight, and follow a hollow shaft design approach. The actuator unit is equipped with a complete set of sensors for measuring the joint position, torque, and temperature. As shown in Figure 10, two absolute high-resolution position sensors (IC-Haus/Balluff 19-bit) are employed to read the output of the actuator unit: the first is mounted at the output of the HD and before the elastic element, while the second at the link side after the elastic element. Such kind of arrangement allows us to

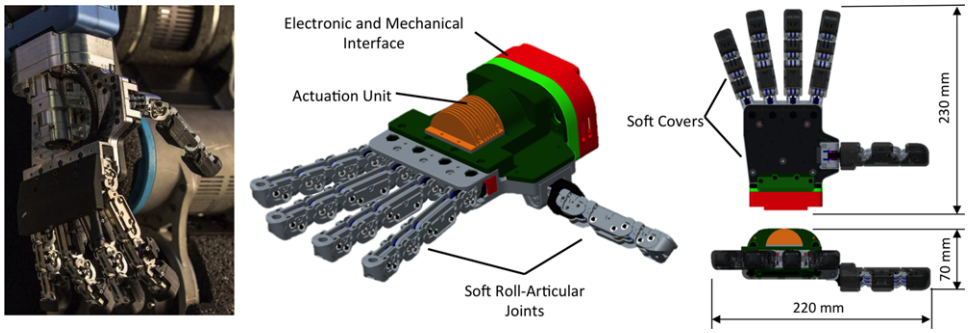


Figure 4. The WALK-MAN hand assembled on the full robot (left panel), its 3D CAD view (central panel), and a top and side view (right panel), highlighting its main features and overall dimensions

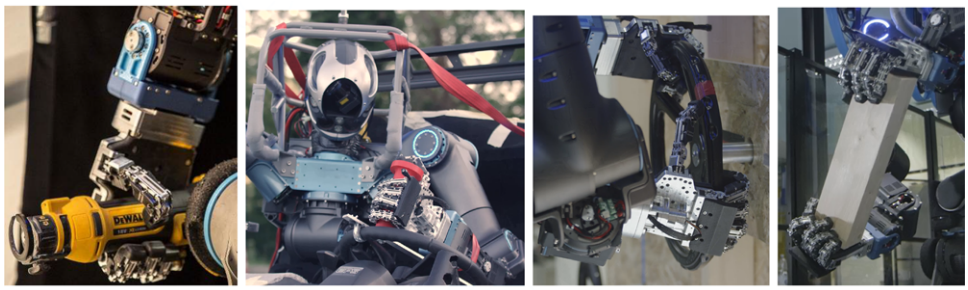


Figure 5. The adaptiveness, versatility, and robustness of the hand employed in the execution of grasping and manipulation tasks

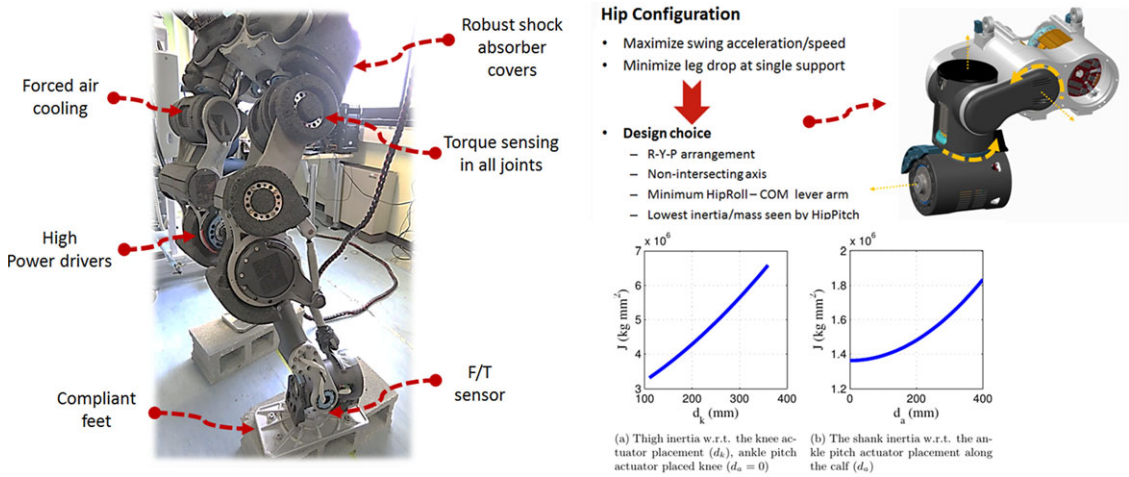


Figure 6. WALK-MAN leg features, hip configuration, and effect of the location of the knee and ankle pitch actuators on the leg inertia

realize a torque sensor embedded in the mechanical design of the actuator by simply monitoring the relative deflection of the torsion bar spring.

Figure 9 demonstrates an initial zero torque control result obtained for an A type WALK-MAN motor (see Table I) based this torque sensor measurement. The control objective in the depicted experiment is to make the joint transparent to any motion externally applied to the joint output with minimal reaction torque. To record the shown data, a rigid bar has been attached to the motor. A human subject applies a motion to this bar and therefore backdrives the motor. The torque controller is initially switched on and keeps the resistance torque felt by the human subject within a 2 N m band. The control effort in terms of a

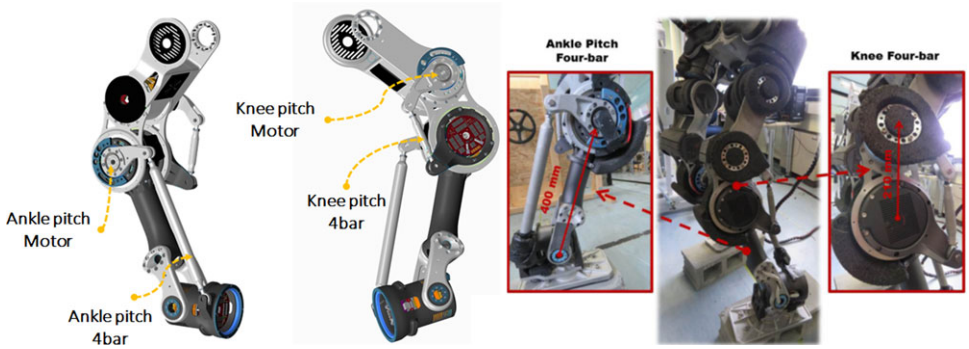


Figure 7. The relocation of knee and ankle pitch actuators and the 4-bar transmissions

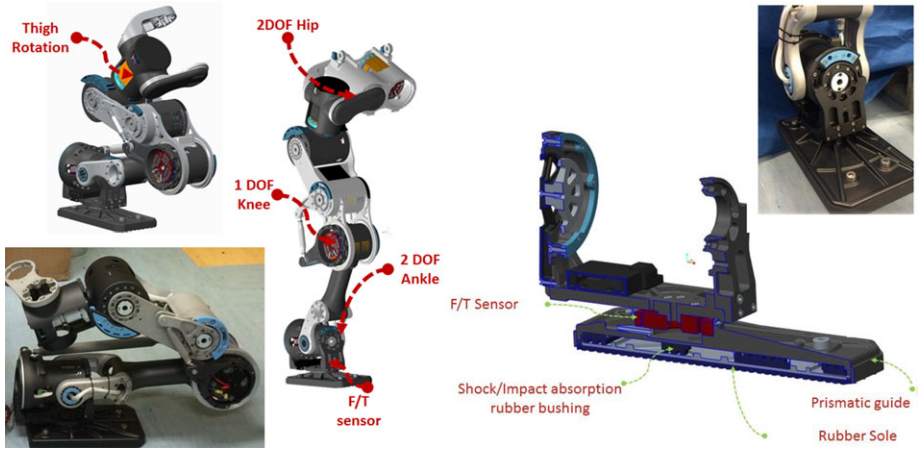


Figure 8. WALK-MAN leg kinematics and foot design

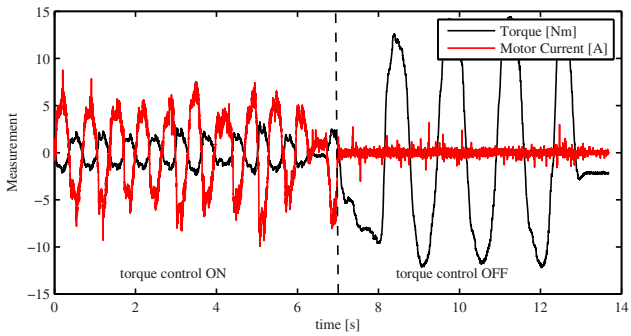


Figure 9. Initial zero torque control results

motor current is shown along with the torque profile. After 7 s, the torque controller is deactivated. While trying to maintain the same bar motion without active zero torque control, the human subject now has to overcome the full internal motor damping and feels reaction torques that are six times larger than before. Additional experiments demonstrating the zero torque control performance based on low apparent friction pendulum motions and zero torque control augmented by a gravity compensation with increased output loads are available at the following link:

Table I. WALK-MAN actuation specifications.

	A	B	C
Continues Power (W) @ 120C rise	900	500	222
Peak Torque (N m), (G=80:1, eff=90%)	270	140	56
Peak Torque (N m), (G=100:1, eff=90%)	330	170	–
Peak Torque (N m), (G=120:1, eff=90%)	400	210	–
No load speed (rad/rad)	14	16.7	11.3
Weight (kg)	2.0	1.5	0.7
Stiffness range (N m/rad)	10 000	1200–6000	500
Overall dimensions D×L (mm)	110×150	100×140	60×100

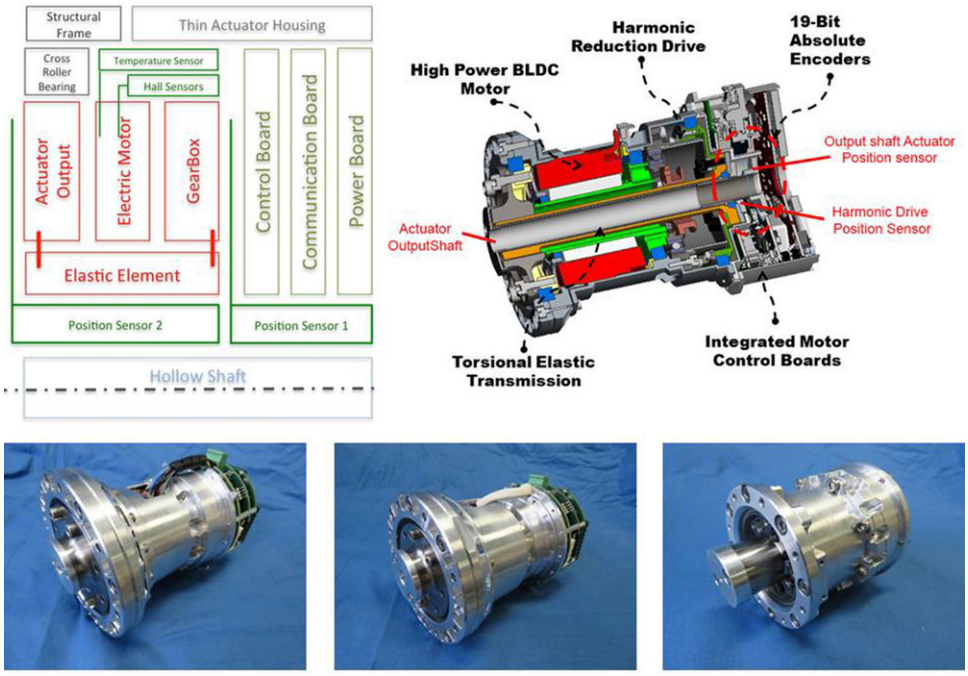


Figure 10. WALK-MAN actuation unit layout, CAD section, and prototype joints from left to right: high, medium and low power size

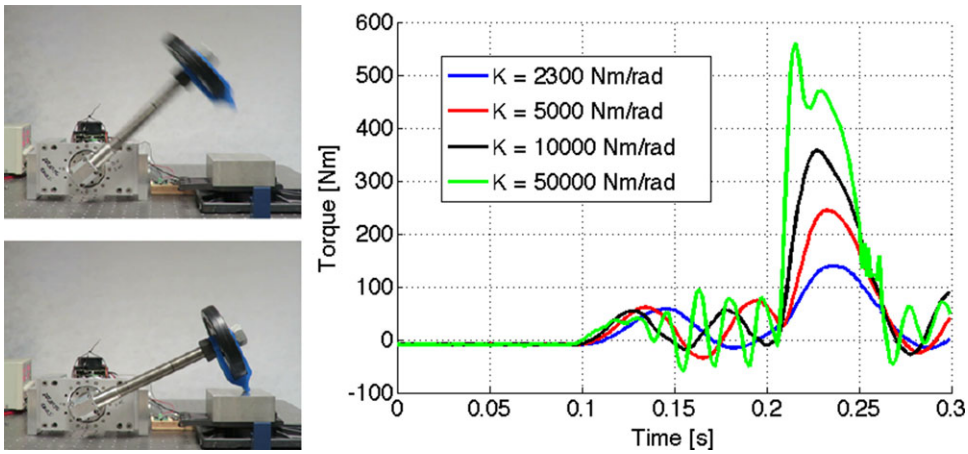


Figure 11. Left: the experimental setup used for the impact tests to demonstrate the effect of intrinsic compliance on the reduction of impact torques reaching the reduction drive. The test bed is composed of an actuation unit, a hammer system, an F/T sensor, and rigid and soft covers with different stiffness. Right: impact torque profiles as a function of the compliance of the torsion bar

http://walk-man.eu/results/videos/item/walk-man-drc-video-collection.html.

For the implementation of the elastic element, the torsion bar has been chosen for its linearity and low hysteresis properties when the deflection is inside the elastic region. Moreover, for an actuator that aims to be modular and scalable, it is an essential feature to have the ability to easily vary the stiffness of the bar without major design and fabrication changes. This eventually allows us to select and tune the stiffness of the joints without radically redesign. The peak torque and stiffness levels of WALK-MAN drives have been selected to match joints requirements derived from extensive simulation studies of the robot model executing manipulation and locomotion tasks.

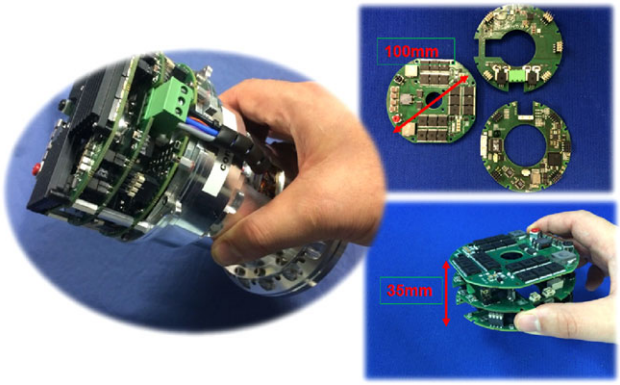


Figure 12. Motor driver integration with the actuator mechanics

Table I reports the torque and speed limits of the actuators modules. The incorporation of physical elasticity in the transmission system of the WALK-MAN drive enhances the physical robustness of the actuation module and the entire robot body in general. To demonstrate this beneficial effect, a series of impact trials were performed. For the experiments, a single DOF test bed has been realized, connecting the actuation unit to a link with a length of 320 mm and weight of 2 kg at the link end. The contact during the impact happens on the tip of a custom-designed hammer, which has a contact area of 300 mm^2. To perform the impact trials, the joint was commanded to follow a 3 rad/s joint velocity reference. The influence of the transmission stiffness on the torque experienced by the gearbox during the impact is shown in the graph of Figure 11. These results demonstrate the significant effect of intrinsic elasticity in the reduction of the peak impact torque that reaches the reduction drive of the actuator, therefore providing physical protection and improved robustness during accidental collisions and impacts. More details about the actuator and its testing can be found in Negrello et al. (2015) and Roozing, Li, Medrano-Cerda, Caldwell, and Tsagarakis (2016).

One of the main features of the WALK-MAN actuator is the seamless integration with the electronics related to the actuator control and sensing. The electronics of the actuator are integrated in a form of a stack of three printed circuit board (PCB) layers at the back of the actuator consisting of the digital signal processing based control board, the data acquisition and communication layer, and the power drive board. Intercommunication within the PCB stack, cabling and connector placements for the motor power lines, the hall sensors, encoders, torque, and temperature sensing were tightly optimized in terms of wire length and routing considering also easy access for maintenance operations. WALK-MAN motor drivers are presented in Figure 12.

2.3. Perception System

The WALK-MAN perception system incorporates several sensing features that permit the robot to perceive the environment and the associated physical interactions with it, and sense the robot posture and the effort generated by its drives. The thermal state of the robot is also monitored with a distributed network of temperature sensors located close to the heat sources such as the motors, the power electronics, and the power source of the robot. An overview of WALK-MAN perception components is introduced in Figure 13. The green highlighted perception features are those implemented during the first period of the project. These features were functional during the DRC and will be described in the following paragraphs.

- Absolute joint position sensing: absolute position sensing provides system initialization at power on and was provided by incorporating two absolute magnetic encoders in the WALK-MAN actuator unit. The first encoder is placed immediately after the harmonic gear measuring the motor side angle after gear, while the second is located after the series elastic bar monitoring the link angle.
- Joint torque sensing: as presented in Section 2.2, joint torque sensing in WALK-MAN drives is implemented using an elastic torsion bar in which the deflection when loaded is measured using the two high-resolution absolute encoders used for the position sensing. To derive the torque measurement from the torsion bar deflection, the stiffness of the bar is required to be known accurately. This is obtained from a calibration that involves the loading of the actuator output after the assembly with a series of known loads. Based on deflections measured and the applied loads, the stiffness of the torsion bar of each actuator unit is obtained prior to the assembly in the robot and is hard coded in the firmware of each drive.
- Force/Torque sensing at the end-effectors: customized 6 DOF force torque sensors are tightly integrated at the wrists and the ankles of the WALK-MAN robot. The foot 6 DOF load cell has a size of 82 mm in diameter and 16 mm in width. It is based on a three-spoke structure where six pairs of semiconductors strain gauges are mounted to measure the strain generated on the load cell as a response to the load applied. The sensor has integrated data acquisition and signal conditioning electronics, and communicates with the rest of the system using the same EtherCAT bus accommodating the interfacing and the low-level communication. The second sensor used in the wrists of the robot is also a custom design with dimensions of 50 mm in diameter and 6 mm in width.
- WALK-MAN head: the head module is the housing of the vision system of the WALK-MAN robot. The WALK-MAN head was designed to incorporate the Multisense M7 sensor that provides a stereo vision system with an integrated FPGA unit, an Inertial Measurement Unit

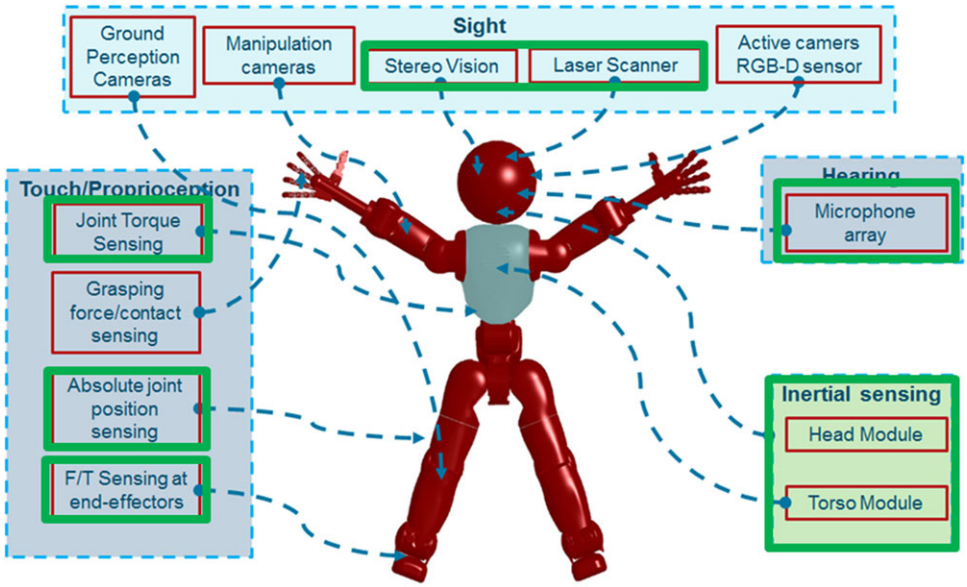


Figure 13. WALK-MAN perception system components

(IMU), and a laser sensor. The head design is shown in Figure 14. The M7 sensor occupies the front side of the head, while at the back side, the vision processing unit based on an i7 Quad core processor COM express PC has been installed. A microphone array system has been installed around the ears allowing the robot to monitor sounds and potentially transmit them back to the operator station if needed. Finally, the head is mounted on the base of a neck module that provides head mobility around the pitch and yaw axis allowing the control of the view direction without the necessity to use the torso motions or the rest of the robot body to orient the vision sensor along a specific view direction.

- IMU: addition to the IMU integrated inside the Multisense M7 sensor, a second IMU has been installed in the pelvis area to monitor the pelvis state in terms of

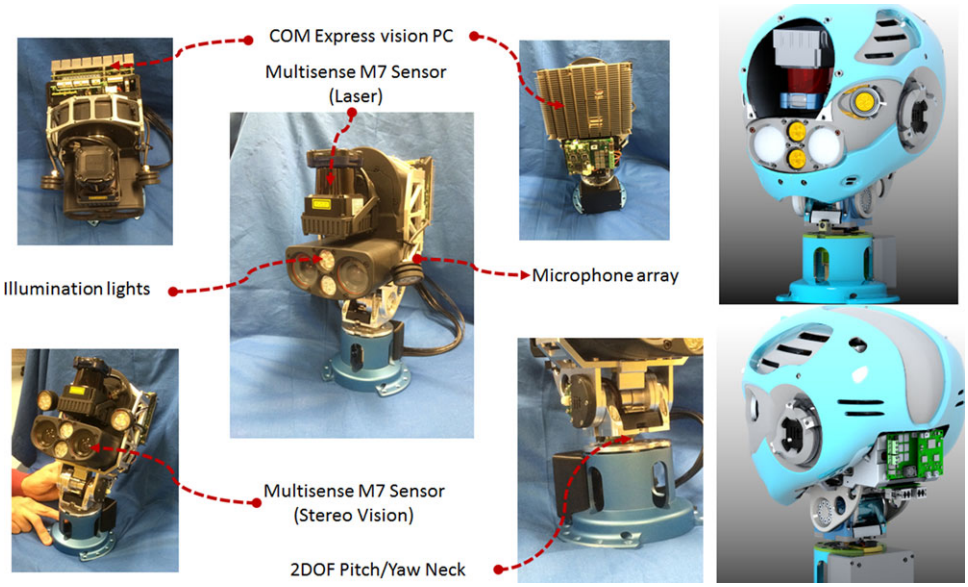


Figure 14. WALK-MAN head: perception components, processing unit, and neck module

acceleration and orientation. Both IMUs accommodate the development of locomotion and balancing controllers by providing useful information for the robot CoM estimation and the derivation of the terrain inclination.

3. WALK-MAN SOFTWARE

3.1. Architecture

In this section, we report an overview of the WALK-MAN software architecture. Similar to other DRC teams, we built a complete software stack: a custom firmware, control modules tackling different tasks, a remote pilot graphical interface, and the whole architecture to manage and connect the different applications. Due to the limited time constraint (around 10 months) and the variety of programming skills among our robotics researchers, our design choices are oriented to:

- avoid code duplicates and improve code reuse;
- provide common shared C++ classes and utilities to the team;
- ease and speed up the production of significant code by hiding code complexity in simple APIs;
- faster test and debug, even without the physical robot, through simulation.

As a consequence of these principles, our core developers focused on low-level interfaces, middleware management, and network and performance optimization.

We devised a layered component-based architecture, where each task of the DRC is handled by a single control module and modules interact with the hardware and each other through well-defined APIs. Once a rough and primitive API was defined, modules could be developed in parallel; in the meantime, shared functionalities could be improved under the hood of the high-level control software without requiring code changes.

The necessity of testing different modules at the same time on the same robot, and the initial lack of the robot itself, leads to the creation of an hardware abstraction layer (HAL) between the robot and the control modules. This HAL was initially implemented for a simulated robot in Gazebo, and it lately was replicated in the real robot.

The architecture is organized into four software layers (see Figure 15(a)).

- The top layer is the operator control unit, which we call *pilot interface*.
- A network bridge connects the pilot and the robot, where different *control and perception modules* compose another layer.
- An HAL remotizes the robot hardware and provides to the control modules a set of shared libraries (*GYM*) used to interact with the remote driver, called *Ethercat Master*.
- The lowest layer is represented by the firmware running in embedded boards, each controlling one actuator.

In Figure 15(b), we show a detailed view of the threads (and the related frequency in Hz) running inside the control modules and the HAL.

A COM Express computation unit based on a Pentium i7 quad core processor was used to execute the motion control of WALK-MAN. The communication to the low-level motor drivers can reach the frequency of 2 kHz. However, the available onboard computation resources did limit the frequency of execution of the different control modules. The control frequency of the different modules was therefore tuned from 100 Hz (manipulation modules) to 500 Hz (walking modules), while the communication rate of the trajectory references to the joints was also set to 500 Hz. These

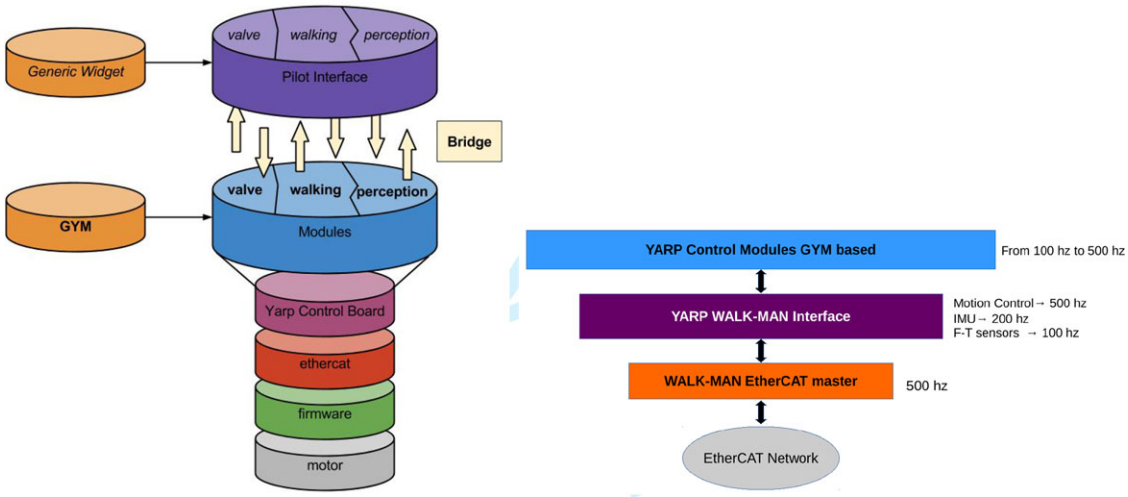


Figure 15. Software architecture

were though still adequate to generate the robot motion and regulate its states both for the manipulation and locomotion tasks. Finally, the IMUs had a slower rate because of the sensor measurement and communication bandwidth constraints.

3.1.1. Firmware-Ethercat

At the lowest level, each joint of WALK-MAN is controlled by a proportional-derivative-integral (PID) position loop in a distributed embedded electronic system with one board per joint. Our main aim was to have a hard real time loop in the firmware: the execution time of each firmware function was measured and tuned so that a 1 kHz loop could be implemented.

3.1.2. Ethercat Master—YARP

In the robot, the hardware manager runs on the *control pc* and is called *Ethercat Master*. It is responsible for managing Ethercat slaves (i.e., the electronic boards), keeping them synchronized and sending/receiving position references in a real-time fashion. The *Master* can be seen as a hardware robot driver, which handles low-level communication and exposes a simpler and asynchronous API to the higher levels. Between the Master and the controlling modules, we choose to introduce a middleware capable of remotizing the robot driver. Given our high speed and low latency requirements, a simple and fast communication framework was required, such as YARP (Metta, Fitzpatrick, & Natale, 2006). The Master creates an input and output YARP port for each control module and for each type of information required by them.

3.1.3. Generic YARP Module

A control module software can be summarized as a *sense-compute-move* loop, where *sense* receives all the inputs from the robot, and these inputs are used by *compute* to implement the control law of the module. Finally, *move* sends to the robot the newly computed desired references of the joints. We designed a generic module as a C++ abstract class that provides a common and standard way to execute these initialization steps, along with a *sense* and *move* default implementation that hides YARP remotization interface. The Generic YARP Module (GYM) functions handle all the YARP required communication between a module, the Master and the Pilot Interface, effectively hiding YARP communication mechanisms and classes. This GYM was iteratively improved based on the team feedback about needed functions and on an effort to search and remove duplicate code across different modules. One of the features implemented in the GYM code is a set of communication interfaces between the module and the pilot: Command, Status, and Switch. These interfaces in their default implementation send through the network an array of characters; the Command and Status interfaces support the addition of a custom data serializer that can be implemented by the user in order to send any type of data. GYM is organized into two threads: a watchdog and a main control loop. Developers can write their own code inside the control loop function; they also have access to a set of helper functions providing a standard kinematic description of the robot based on a Universal Robotic Description Format (URDF). The watchdog thread is not customizable and listens for standard commands from the pilot, through the Switch interface.

The *Switch interface* is used to send the following commands to each module: start, pause, resume, stop, quit. Since some of these commands are critical, they cannot be overridden with different implementations and modules are only allowed to re-implement pause and resume functions. This approach guarantees that any bug or misbehavior of the code running inside a GYM does not propagate to the whole system, since a module can always be forced to stop by the pilot with a stop command. The *Command interface* is used to send commands to the robot related to the precise task being executed, such as "go_straight 10" to make the robot walking for 10 m or "set_valve 0.5 0 0.1 0 0 0 1 Waist" to set the valve data for the turning valve task with respect to the *Waist* robot reference frame. The *Status interface* is used to send back to the pilot any necessary information

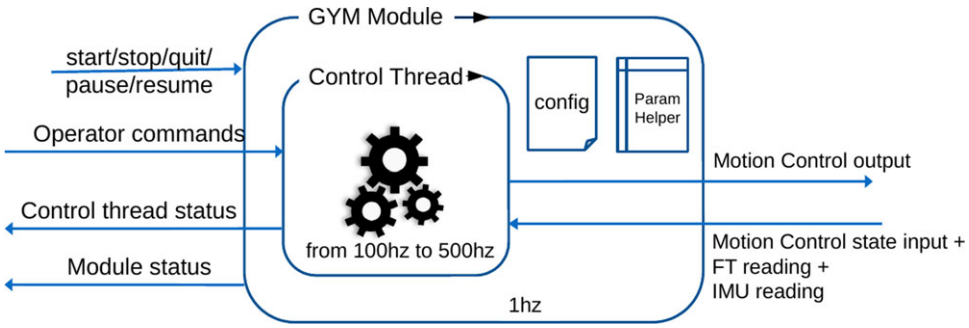


Figure 16. GYM control thread and communication interfaces

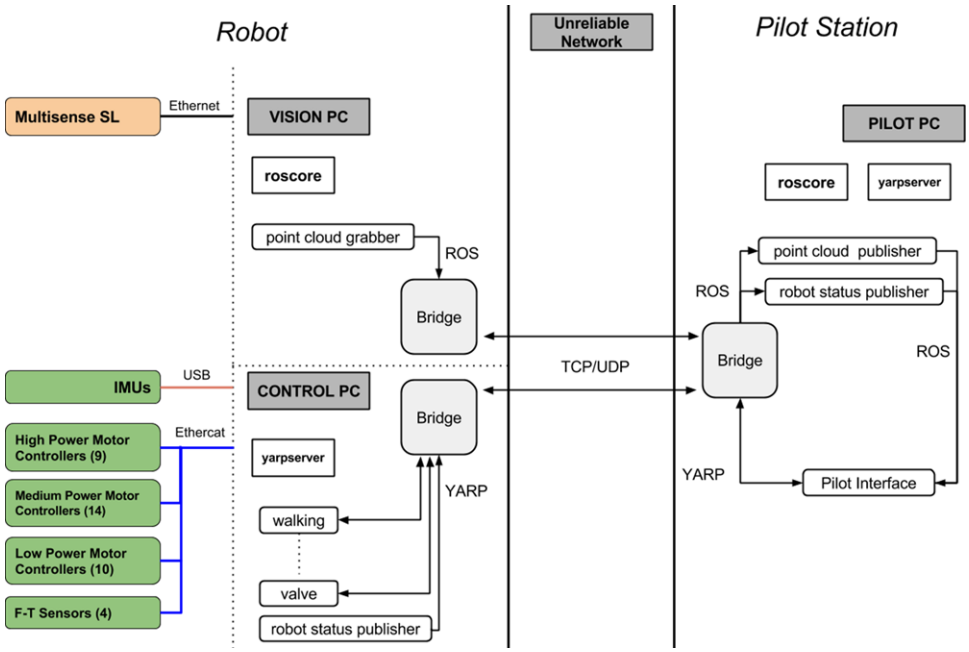


Figure 17. System PC connections and interactions

to have complete knowledge on the internal state of the control modules, such as "turning valve," "walking," "ready," and so on.

In Figure 16, we report an overview of the GYM with the watchdog thread (GYM Module), the control loop thread, and all the communication interfaces between GYM, the operator, and the robot.

3.2. System Communication

Our robot is used with two common types of network configurations between the *pilot pc* and the robot. The first setup is similar to a lab environment, where the network is fully operational and the bandwidth is at least 100 Mb/s. The second one is inspired by real-world scenarios, where a wireless network is discontinuously working and the average bandwidth is less than 1 Mb/s. It is desirable to have most of the software architecture independent from the network capabilities; in particular, the code running in control modules and in the pilot interface should not require any changes depending on the network.

When working in the first configuration, we use a single YarpServer and RosCore, and modules can communicate directly with each other; there are no networking issues from pilot to robot.

In the real-world scenario, a direct communication may result in frequent disconnections, and the centralized YARP/ROS servers may not be able to recover from such disconnections. Thus, we move to a strong separation between *pilot pc* and the robot, with two pairs of RosCore/YarpServer running, respectively, on the *pilot pc* and the *control pc*, splitting modules into a robot and a pilot ecosystem. A low-level network application is required in order to connect modules running in one ecosystem with modules on the other, under the constraint of no modification to the modules source code. Our network manager behaves as a two-way bridge between *pilot pc* and the robot; it is completely transparent to the processes it connects, meaning that there is no way for the process to understand if they are communicating through a bridge or directly. Our bridge is developed as a pair of processes, running on two different computers, called BridgeSink (in the sender pc) and BridgeSource (in the receiver pc). The Boost Asio library was used to abstract UNIX sockets and obtain an asynchronous behavior in the communications.

As an example, if Module Alice on PC1 is sending info to Module Bob on PC2 using YARP, and the bridge is disabled, Alice will try to connect to Bob and will find a YARP port on the remote PC2, while Bob will listen from Alice's remote YARP port from PC1. If the bridge is enabled, Alice will see a local fake Bob YARP port that is actually the BridgeSink running on PC1, while Bob will listen from a local fake Alice YARP port that is actually the BridgeSource on PC2. BridgeSink and BridgeSource will internally transfer information from PC1 to PC2 using the bridge heuristics for network management, where the most important option is the bridge channel protocol (UPD or TCP) and the middleware (YARP or ROS). In Figure 17, we report the location (motor PC, vision PC, pilot PC) where the various programs are executed, focusing on the TCP/UDP bridge role.

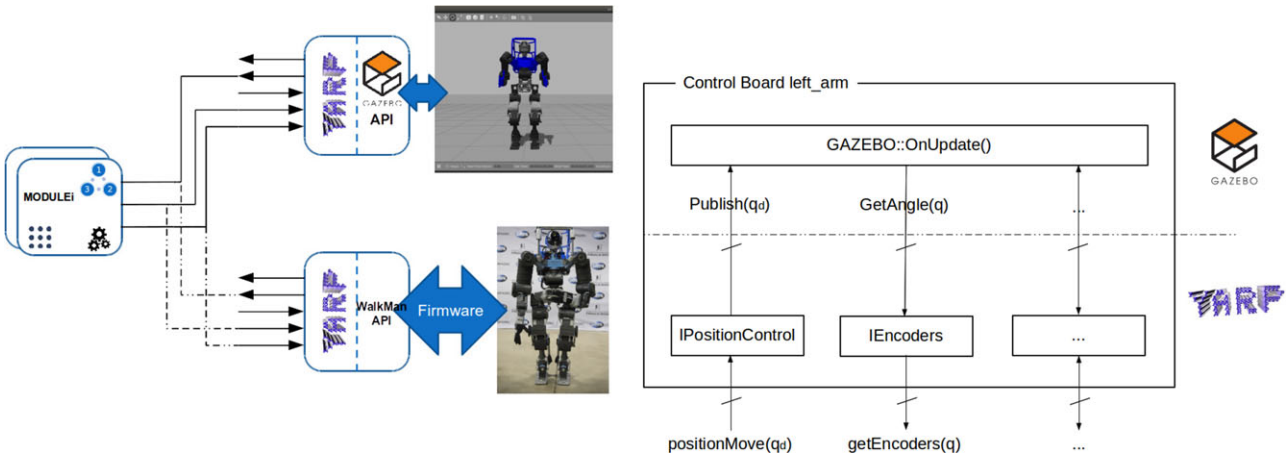


Figure 18. Simulator infrastructure

3.3. Simulation Environment

The development of efficient simulation tools to model the humanoid robot dynamics lies within the core of the WALK-MAN project. The WALK-MAN simulator is based on Gazebo and aims at accelerating the development and simulation of motion control modules in tasks involving interaction with complex environments and planning. For this purpose, we developed a set of plugins that enable the interoperability of YARP modules between a real robot and a simulated one in Gazebo. Since these plugins conform with the YARP layer used on the real robot, applications, written for WALK-MAN, can be tested and developed also on the simulated robot without changes (Figure 18(a)).

These plugins consist of two main components: a YARP interface with the same API as the real robot interface, and a Gazebo plugin that handles simulated joints, encoders, IMUs, force/torque sensors, synchronization, and so on. Different modules and tasks for WALK-MAN have been developed using Gazebo and the presented plugins as a test bed while preparing for the DRC Finals. In our software framework, the *simulator* is a module that represents the real robot at the interface level. Such simulator module accepts control input (desired joint torques, desired joint position, etc.) and outputs sensory feedback (cameras, joint positions, etc.) from the simulated world.

By accurately simulating robots and environments, the code designed to operate on a real robot can be executed and validated on the simulated equivalent system. This avoids common problems associated with hardware such as hardware failures, and unexpected and dangerous behaviors, particularly during the initial stages of development and tuning of new modules and controllers. In this way, the simulator becomes a fundamental part of the robot software development cycle as the first step to validate algorithms, thus minimizing the risks of hardware breaks.

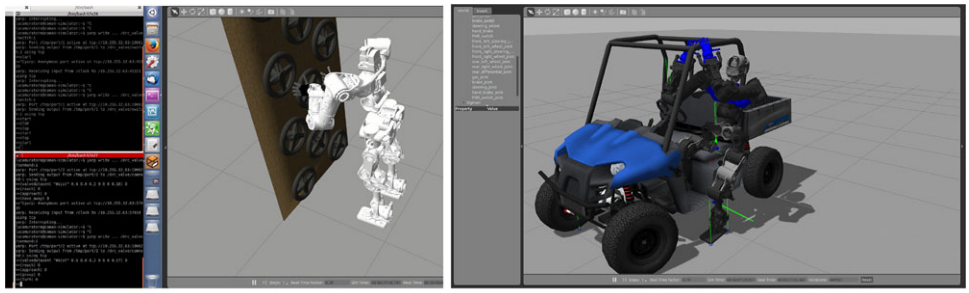
Our decision to add a YARP interface to Gazebo is motivated by the following considerations. The possibility of switching between fast, not accurate simulations and slow, accurate ones, and thus the capability of choosing among different dynamic engines, was needed. Furthermore, a simulator that is both easy to use and flexible to be extended was required. It is useful to understand Gazebo plugins and YARP device drivers before describing the structure of the developed plugins (from now on **gazebo_yarp_plugins**). Gazebo plugins are C++ classes that extend the functionalities of Gazebo, while YARP device drivers are C++ classes used in YARP for abstracting the functionality of robot devices. Usually, each class of gazebo_yarp_plugins embeds a YARP device driver in a Gazebo plugin. YARP provides special devices that act as network proxies and make interfaces available through a network connection. This allows accessing devices remotely across the network without code change. A **device driver** is a class that implements one or more interfaces. There are three separate concerns related to devices in YARP:

- Implementing specific drivers for particular devices
- Defining interfaces for device families
- Implementing network wrappers for interfaces.

For example, the Control Board device driver implements a set of interfaces that are used to control the robot (IPositionControl, ITorqueControl, etc.) and another set of interfaces to read data from the motors (IEncoders, etc.) as shown in Figure 18(b).

A gazebo_yarp_plugin is made of:

- Gazebo plugins that instantiate YARP device drivers, and
- YARP device drivers that wrap Gazebo functionalities inside the YARP device interfaces.



(a) Reaching and grasping the valve (b) Egressing the Polaris

Figure 19. Simulated tasks in Gazebo

Some examples of implemented plugins are the **Control Board**, **6-axis Force Torque sensor**, IMU, and the **Clock** plugin used for synchronization. The first three plugins are directly related to the simulated objects and sensors, while the last one is a system plugin that synchronizes all the other YARP modules with the simulation time. Another fundamental aspect in simulations is the synchronization between control modules and the simulated robot. A YARP control module is a process in which one or more threads are started. When such modules are used in the real robot, the thread rate is timed by the machine (system) clock, also called the wall clock. When the simulation is running, we want the rate of such modules to be synchronized with the simulated time; otherwise the control loop could run faster or slower with respect to the simulated robot dynamics. The clock plugin is implemented as a System plugin and publishes on a YARP port the time information from the simulator. For every simulation step, the simulation time is incremented and the timestamp is sent via socket. YARP functions that provide access to the computer internal clock and support thread scheduling can be synchronized with an external clock . YARP classes supporting periodic threads are therefore automatically synchronized with the clock provided by the simulator. Examples of simulated tasks for the DRC Finals are shown in Figure 19.

3.4. WALK-MAN Pilot Interface

To tackle the execution of the DRC tasks, we developed a remote operator interface to both receive information of the environment in which the robot is operating and send commands to the robot.

The *PI* (pilot interface) has been implemented using Qt Libraries[1] and ROS libRViz[2] for 3D rendering. A description of a preliminary version of the interface can be found in Settimi et al. (2014).

The *PI* has been developed in a modular way, such that different widgets can be included or not into the graphical user interface depending on the application or the user need. As depicted in Figure 20, many interfaces can be generated by changing simple *XML* files, and these can be used by different pilots (as occurred during the DRC) in order to make them focusing on different critical aspects (execution of the task, perception of the environment, status of the robot, and so on).

The structure of the standard interface is organized in three layers from the top to the bottom (see Figure 21). In the first layer, the pilot can enable/disable *advanced* and *all-button-enabled* modes. Moreover, a mission time is displayed in the interface together with several buttons dedicated to toggle the different displays visualized in the middle layer. Indeed, the second layer is dedicated to visualize both the robot point of view (on the left) and the 3D visualization of the robot immersed in the environment. The environment is reconstructed based on the point clouds received form the robot (from laser scan and stereo vision). In the third layer, there are different tabs that depend on the particular needs (basic control, manipulation, locomotion, perception, status, and so on). Each control module has a dedicated widget that inherits from a *Generic Widget*. With this approach, the control modules already have available the switch and status interfaces (see Section 3.2).

As an example, in Figures 22, 23, and 24, we report the door opening widget, the locomotion widget, and boards status widget, respectively. As an advanced feature, based on the forgiveness design principle, a special timed button has been implemented: after the click, a countdown of 3 seconds is displayed on the button before sending the command; the command can be stopped by reclicking on it (this is used for dangerous commands, such as the "Go There!" button in the locomotion widget, to undo erroneous clicks).

This feature was designed to prevent some wrong commands to be sent from the pilot to the robot during the training for the DRC. In fact, after a wrong critical command, the only way to prevent robot damages was the emergency power button. During the DRC, the timed button was used once and prevented a robot fall. In particular,

[1]http://www.qt.io
[2]http://wiki.ros.org/rviz

Figure 20. Distributed operating station

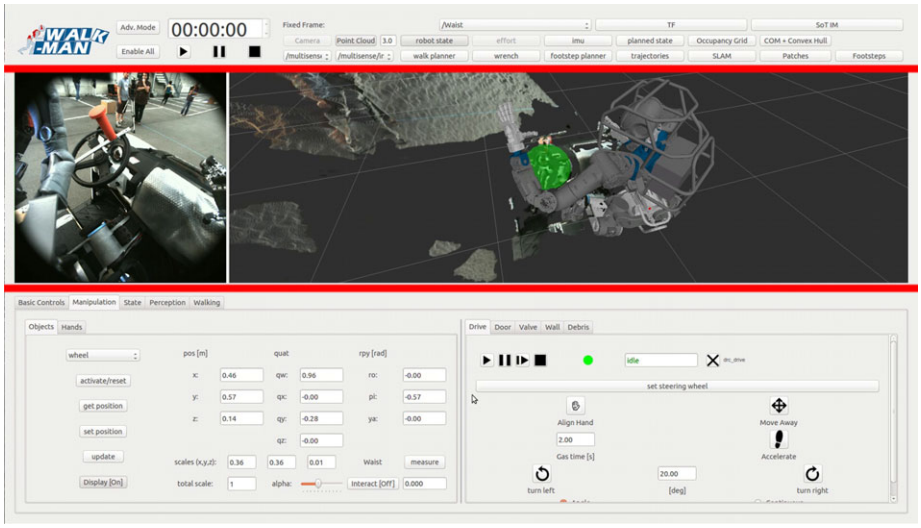


Figure 21. Layered structure of the standard pilot interface, displayed during the driving task execution

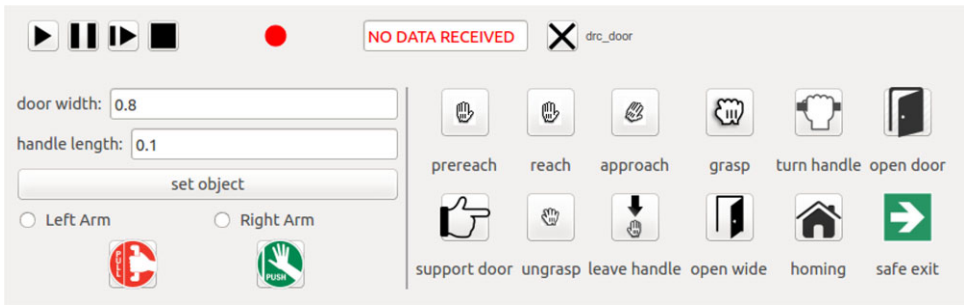


Figure 22. Door opening task widget. The operator can specify the door data and with which arm the robot should open the door; then the single actions are triggered by associated buttons

the operator was trying to cross the door after it was opened, but had forgot to evaluate for the terrain inclination. Just after the *walk forward* button was clicked, the support operator noticed the missing procedure and called for a stop of the timer, giving the main operator the possibility to execute additional routines to evaluate the inclination of the terrain and allow the robot to safely continue and make the step.

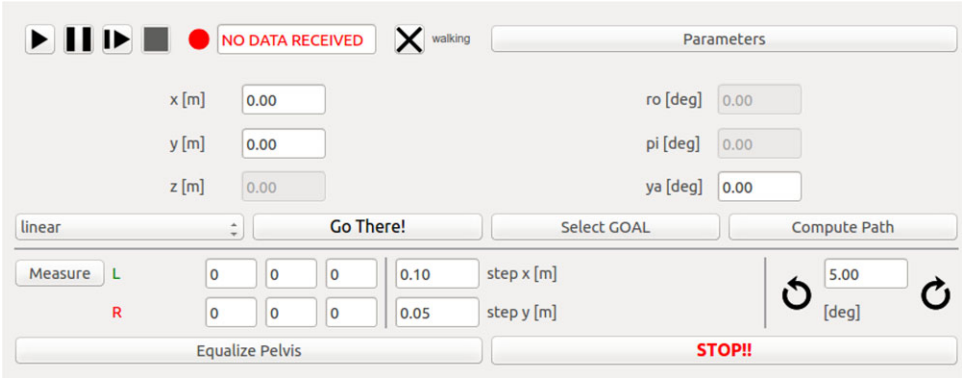


Figure 23. Locomotion widget. The pilot can ask the robot to perform basic locomotion primitives (walk forward, backward, left, right, rotate on the spot) and can change the type of trajectory that the internal footstep planner of the walking module will use to reach the goal position

4. WALK-MAN MOTION CONTROL

4.1. Whole-Body Control

One of the main components of the WALK-MAN software stack is the library used to solve whole-body inverse kinematics (WBIK) problems, called *OpenSoT* (Rocchi, Hoffman, Caldwell, & Tsagarakis, 2015). *OpenSoT* is a whole-body control framework inspired by the *Stack of Tasks* (Escande et al., 2014; Mansard et al., 2009) with the main idea of decoupling the tasks/constraints description and the solvers implementation. It provides base classes and standard interfaces to specify tasks, constraints, and solvers. This yields the following features that make the implementation of OpenSoT unique and attractive:

- Demonstrates high modularity through the separation of task descriptions, control schemes, and solvers maximizing customization, flexibility, and expandability.
- Provides user-friendly interfaces for defining tasks, constraints, and solvers to promote integration and cooperation in the emerging field of whole-body hierarchical control schemes.
- Demonstrates computation efficiency to allow for real-time performance implementations.
- Allows ease of use and application with arbitrary robots through the URDF and Semantic Robotic Description Format.
- The architecture of OpenSoT encourages collaboration and helps integration and code maintenance.[3]

4.1.1. Inverse Kinematics

When performing tasks in a real scenario, the IK is a fundamental part of the control architecture as it maps the desired references in the operational space to desired references in joint space. One challenge to solve the IK problem is to render a singularity robustness and a capability to handle the constraints/bounds into an algorithm of the solver. To resolve this an IK solver based on QP optimization with the possibility of specifying *hard* (Kanoun, Lamiraux, & Wieber, 2011) and *soft* (Chiacchio, Chiaverini, Sciavicco, & Siciliano, 1991) priorities between tasks as well as linear constraints and bounds (Escande et al., 2014).

Each task in the stack of the hierarchical IK problem can be formulated as the following QP problem:

$$\begin{aligned} \underset{\dot{\mathbf{q}}}{\operatorname{argmin}} \; & \|\mathbf{J}_i\dot{\mathbf{q}}_i - \mathbf{v}_{d,i}\|_W + \lambda\|\dot{\mathbf{q}}_i\| \\ s.t. \; & \mathbf{c}_{l,i} \le \mathbf{A}_i\dot{\mathbf{q}}_i \le \mathbf{b}_{u,i} \\ & \mathbf{b}_l \le \mathbf{A}\dot{\mathbf{q}}_i \le \mathbf{b}_u \\ & \mathbf{u}_l \le \dot{\mathbf{q}}_i \le \mathbf{u}_u \\ & \mathbf{J}_{i-1}\dot{\mathbf{q}}_{i-1} = \mathbf{J}_{i-1}\dot{\mathbf{q}}_i \\ & \vdots \\ & \mathbf{J}_0\dot{\mathbf{q}}_0 = \mathbf{J}_0\dot{\mathbf{q}}_i, \end{aligned} \tag{1}$$

where $\mathbf{J}_i$ and $\mathbf{v}_{d,i}$ are, respectively, the Jacobian and the desired velocity reference for the ith task; λ is the regularization coefficient; $\mathbf{A}_i$, $\mathbf{c}_{l,i}$, and $\mathbf{c}_{u,i}$ are constraints for the ith task; $\mathbf{A}$, $\mathbf{b}_l$, and $\mathbf{b}_u$ are global constrains; and $\mathbf{u}_l$ and $\mathbf{u}_u$ are global bounds (i.e., active for all tasks). The final set of constraints represents the optimality conditions inherited from higher priority tasks: the previous solutions $\dot{q}_i$, $i < n$, are taken into account with constraints of the type $A_i\dot{q} = A_i\dot{q}_i \; \forall i < n$, so that the optimality of all higher priority tasks is not changed by the current solution. The weighted minimization of the task errors can be achieved by adding a joint space task (postural or minimum velocity) at the lowest priority level such as minimum velocity of which resulting optimization is equivalent to a weighted pseudoinverse (Siciliano, Sciavicco, Villani, & Oriolo, 2008). As shown in Nakamura (1990), the regularization term can be applied in the cost function

[3]The OpenSoT library is open-source and downloadable at: https://github.com/robotology-playground/OpenSoT

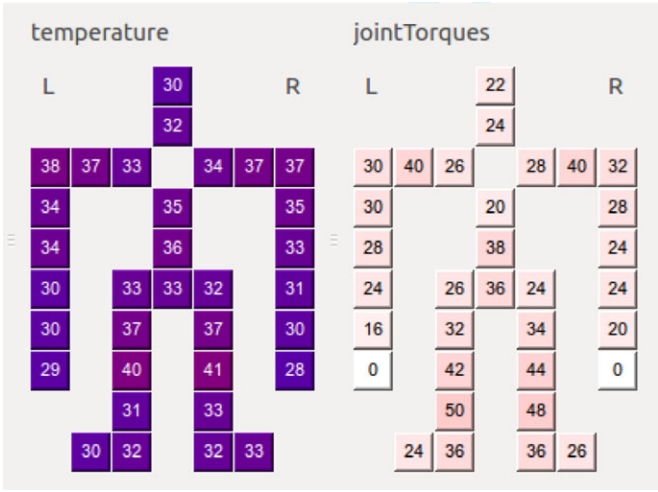


Figure 24. Boards status widget. Left: the board temperatures are shown. Right: the torques associated with the different joints are reported

to guarantee the robustness near kinematics singularities. Bounds and constraints are mandatory to be robust to joint position/velocity/acceleration/torque limits. A mixture of hard and soft priorities is in general needed to describe a stack of tasks. The solution obtained can then be integrated and sent as a position reference.

In OpenSoT, we implemented a set of tasks and constraints that can be composed to obtain stacks tailored to different control and task scenarios. Other than fundamental operations like aggregation (to create augmented tasks), creating subtasks and masking the task Jacobians to use only on a subset of joints, a pool of constraints and task had to be implemented: Cartesian, CoM, postural, minimum effort, manipulability, minimum joint velocity and acceleration, and interaction (admittance control; Rocchi et al., 2015). The implemented constraints are position and velocity constraints in Cartesian space, convex hull constraints, joint position and velocity limits, and self-collision avoidance (Fang, Rocchi, Mingo Hoffman, Tsagarakis, & Caldwell, 2015).

4.1.2. A Robust IK Solver

The currently implemented solver is based on the popular QP library *qpOASES* (Ferreau, Kirches, Potschka, Bock, & Diehl, 2014) that implements an active-set approach to

# Priority	Tasks
0	Cartesian right sole
1	aggregation of 4 tasks: Cartesian waist Cartesian torso Cartesian right arm Cartesian left arm
2	aggregation of 2 tasks: joint posture minimize joint accel.

	Initialization	Solve
Sparse	0.00143	0.000490
Dense	0.00297	0.000475

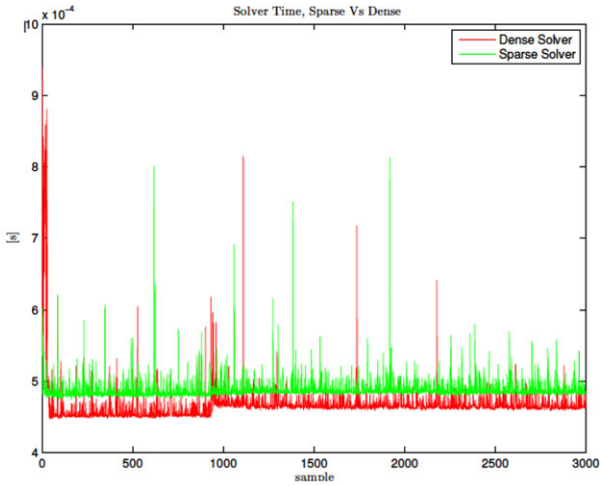


Figure 25. An example of stack description and the time needed to solve a stack of tasks, Sparse versus Dense implementation, on an Intel Core i7. The problem also has two global constraints: keeping the Center of Mass inside the support polygon and keeping the velocity of the Center of Mass bounded and two bounds (joint position and velocity limits)

handle inequality constraints. The library also provides a *warm-start* and a *hot-start* approach to solve QP Problems. Basically, in the *warm-start*, an initial guess from the previous solution and previous active set is used to solve the QP Problem. In the *hot-start*, a previous decomposition of the matrix for the Karush–Kuhn–Tucker conditions is reused to decrease solving time.

For each task, the cost function is computed as

$$f(\mathbf{q}_i) = \dot{\mathbf{q}}_i^T \mathbf{J}_i^T \mathbf{W}_i \mathbf{J}_i \dot{\mathbf{q}} + 2(\mathbf{J}_i \dot{\mathbf{q}})^T \mathbf{W} \mathbf{v}_{\mathrm{d,i}}. \quad (2)$$

If local constraints are presented in the task, they are added to the matrix of the constraints together with the global ones. The optimality constraints are added together with the other constraints automatically by the *front-end*. Equality and inequality constraints are treated together.

Since most of the time the task Jacobian will result in a sparse matrix, it is convenient to use the sparsity of the matrices in order to speed up computation: in particular, *qpOASES* allows us to define QP Problems with sparse Hessian matrices. Performances of the sparse implementation against the dense implementation are illustrated in Figure 25 for a medium size IK Problem (29 variables, up to 63 constraints), where it is shown how the computation speed is enhanced in the *initialization* phase, where the sparse solver is approximately twice as fast as the dense solver.

A side-effect of faster initialization times is reflected on a lower solving time variance, which is $9.4777 \times \mathrm{e}^{-10}$ for the dense solver and $2.3904 \times \mathrm{e}^{-09}$ for the sparse. Of course, to obtain good results from the solver a good tuning of the regularization term λ had to be performed. With $\lambda = 2.221 \times 10^{-3}$ a good compromise between joints trajectory smoothness and task error is achieved.

In the DRC Finals, *OpenSoT* was used to implement all the manipulation tasks (*driving*, *door opening*, *wall cutting*, and *valve turning*) while keeping balance as well as considering joint position and velocity limits. The IK solver was running on the onboard computer.

4.1.3. Example of High-Level Task: Squat

In this section, an overview of an *OpenSoT* implementation of a whole-body squat motion is presented. The involved components are the Cartesian, CoM, and Postural tasks; Joint Limits and Joint Velocity Limits bounds; and Self Collision Avoidance, Support Polygon, and Torque Limits constraints, as described briefly in Table II. The task consists of moving the left arm forward and near the ground generating a squat motion of the whole-body and high joint torques. In particular, we will show not only that the joint torques are bounded in the limits, but also that the task makes the robot fall if performed without the robot dynamics constraint. Using our *Math of Tasks* formulation, the stack

Table II. Definition of Cartesian, CoM, Postural, Joint Limits, Joint Velocity Limits, Self Collision Avoidance, Support Polygon, and Torque Limits constraints. These are just a small set of the constraints and tasks that the *OpenSoT* library provides.

Task	**Formulation**
Cartesian Position/CoM	$\mathcal{T}({}^b\mathbf{J}_{d,p}, \quad \dot{\mathbf{p}}_d + \mathbf{K}_p(\mathbf{p}_d - \mathbf{p}))$
Cartesian Orientation	$\mathcal{T}({}^b\mathbf{J}_{d,o}, \quad \boldsymbol{\omega}_d + \mathbf{K}_o(-(\eta_d \boldsymbol{\epsilon} - \eta \boldsymbol{\epsilon}_d + [\boldsymbol{\epsilon}_d \times]\boldsymbol{\epsilon})))$
Postural	$\mathcal{T}(\mathbf{I}, \quad \dot{\mathbf{q}}_d + \lambda(\mathbf{q}_d - \mathbf{q}))$
Bound	**Formulation**
Joints Limits	$\mathcal{B}(\sigma(\mathbf{q}_{\min} - \mathbf{q}), \quad \sigma(\mathbf{q}_{\max} - \mathbf{q})$
Joint Velocity Limits	$\mathcal{B}(-\sigma \dot{\mathbf{q}}_{\max} \Delta t, \quad \sigma \dot{\mathbf{q}}_{\max} \Delta t)$
Constraint	**Formulation**
Self Collision Avoidance	$\mathcal{C}(\boldsymbol{N}, \boldsymbol{D})$
Support Polygon	$\mathcal{C}(\mathbf{A}_{\mathrm{CH}}, \mathbf{b}_{\mathrm{CH}})$
Torque Limits	$\mathcal{C}(\mathbf{M}(\mathbf{q}), \mathbf{u}_{\mathrm{dyn}}(\boldsymbol{\tau}_{\min}), \mathbf{u}_{\mathrm{dyn}}(\boldsymbol{\tau}_{\max}))$

Notes [a]In particular, $T_i = \mathcal{T}(\mathbf{A}_i, \mathbf{b}_i)$ defines a task where $\mathbf{A}_i^T \mathbf{A}_i$ is the task Hessian and $\mathbf{A}_i^T \mathbf{b}_i$ the task gradient. For the Cartesian task, $\mathbf{p}_d = [x_d \quad y_d \quad z_d]$ is the desired position and $\boldsymbol{\alpha}_d = [\eta_d \quad \epsilon_{1,d} \quad \epsilon_{2,d} \quad \epsilon_{3,d}]$ is the desired orientation expressed as a quaternion (Nakanishi, Cory, Mistry, Peters, & Schaal, 2008), $\mathbf{K}_p$ and $\mathbf{K}_o$ are positive definite matrices, and $\boldsymbol{\xi}_d = [\dot{\boldsymbol{p}}_d \quad \boldsymbol{\omega}_d]$ is the desired Cartesian velocity for the end-effector. For the support polygon constraint, every row of $[\mathbf{A}_{\mathrm{CH}} \quad \mathbf{b}_{\mathrm{CH}}]$ is the vector $[a_i \quad b_i \quad -c_i]$ of coefficients from the implicit equation of the line $a_i x + b_i y + c_i = 0$, normalized so that $a_i^2 + b_i^2 = 1$. The torque limits constraint is implemented so that $\mathbf{u}_{\mathrm{dyn}}(\boldsymbol{\tau}) = \sigma(\Delta T(\mathbf{D}(\mathbf{q}, \dot{\mathbf{q}}) + \boldsymbol{\tau}) + \mathbf{M}(\mathbf{q})\dot{\mathbf{q}}_{\mathrm{prev}})$ (Mingo Hoffman, Rocchi, Tsagarakis, & Caldwell, 2016). In the the self-collision avoidance task, every row of the constraint is of the form $[\boldsymbol{n}_i^T \, {}^{cp_{1,i}}\boldsymbol{J}_{cp_{2,i}}(q) \quad \varepsilon(d_i - d_{s,i})/t]$ which corresponds to the ith pair of links for which the self-collision (Fang et al., 2015) avoidance is applied.

can be written as

$$\begin{pmatrix} T_{\substack{\text{Right}\\\text{Foot}}} \setminus \\ T_{\text{CoM XY}} \ll C_{\substack{\text{Support}\\\text{Polygon}}} \setminus \\ \left(T_{\substack{\text{Right}\\\text{Wrist}}} + T_{\substack{\text{Left}\\\text{Wrist}}}\right) \setminus \\ T_{\substack{\text{Joint}\\\text{Posture}}} \end{pmatrix} \ll \left(B_{\substack{\text{Joint}\\\text{Limits}}} + B_{\substack{\text{Joint Velocity}\\\text{Limits}}} + C_{\substack{\text{Torque}\\\text{Limits}}} + C_{\substack{\text{Self Collision}\\\text{Avoidance}}}\right), \quad (3)$$

where $S = T_1 / T_2$ creates a stack with T_1 has higher priority than T_2, $T_3 = T_1 + T_2$ is an augmented task (augmented Jacobian formulation), and $T_1 \ll C_0$ applies the constraint C_0 (or the bound B_0) to the task T_1 (can also be applied directly to a stack S, meaning the constraint applies to all tasks in the stack).

Torque limits constraint has $\sigma = 0.2$ and the sensed (simulated) wrenches at the force/torque sensors are filtered:

$$\mathbf{w}_t \mathrel{+}= (\mathbf{w}_t - \mathbf{w}_{t-1})\, 0.6. \tag{4}$$

Furthermore, for the three joints in the torso maximum torques of 72[N m], 132[N m], and 72[N m] are set, respectively, for the roll, pitch, and yaw joints (around 40% less than the maximum available peak torques in the real robot).

The Cartesian task consists of a linear trajectory for the left hand, from the initial pose, 0.7[m] forward, 0.08[m] on the left, 0.5[m] down, and a desired rotation around the local z-axis of $\frac{\pi}{3}$[rad], which is repeated from start to end and then back again. The trajectory has to be executed in 6 seconds. Desired joint trajectories are sent to the robot open-loop integrating the results obtained from the IK:

$$\mathbf{q}_{\mathrm{d}} = \mathbf{q} + \dot{\mathbf{q}}\Delta T. \tag{5}$$

In Figure 26, the final motion performed by the robot when the torque limits constraint is not active (upper sequence) and when it is active (lower sequence) can be observed. Without considering torque limits, the robot falls in the second part of the squat motion. Figure 27 shows (on the left) that the torques at the torso remain in the limits when using the torque limits constraint, while saturate when not using it.

Cartesian errors are shown in Figure 27 (right). Despite the fact that the Cartesian errors are small when not using the torque limits constraint, the robot falls with high joint torques trying to keep the Cartesian error small. With the constraint activated, the Cartesian errors are larger but the robot does not fall and the torques on the joints are inside the bounds.

4.2. Locomotion

Several tasks in DRC required the robot to be able to walk and balance while progressing through the challenge. An overview of our locomotion module is shown in Figure 28. The module operation starts from footsteps planning done either automatically or manually by pilot. The footsteps are later transformed into task space references, such as feet and ZMP trajectories. These are used inside the pattern generator that computes the CoM reference (pelvis) trajectory to realize stable locomotion. This then generates the gait pattern, which is then executed on the robot. The gait pattern execution loop runs at 3 ms cycle and involves pulling of the configuration reference, robot state estimation, reference correction by gait stabilizer, IK, and reference execution.

4.2.1. Individual Components

In this section, we will shortly describe tools and implementation of each component for the locomotion control module.

4.2.2. Step Planning and Reference Trajectory Generation

We used two approaches to generate footsteps: (a) automated generation of footsteps given goal point and (b) manual foot placement. The first solution used in all flat surface walking scenarios when the robot does need to avoid obstacles. In this mode, through the pilot interface (Section 3.4), we define the final desired position and orientation of the robot. Next, we generate spline trajectory connecting the present and final position of the robot. Finally, we generate a series of footsteps that follow the spline and guarantee collision-free foot placement. In the latter solution, through the pilot interface, we can manually specify the position and orientation of the individual footholds. This is used to plan the locomotion on uneven terrain such as cinder blocks.

Before passing the footsteps to next stage, we can use the information from perception module with 3D reconstruction of the environment and automatically realign the generated steps to the walking surface. This way, we can compensate for small unevenness or inclination of the ground. In the next stage based on the footsteps, the task-space trajectory of the end- effectors and ZMP are automatically generated. The ZMP reference is placed in the middle of the sole during the single support phase and linear transitions from the previous to next support foot during the double support phase. The transition takes place not only in the horizontal but also in the vertical direction, for example, when climbing up steps. The foot trajectory can have one of two shapes: either smooth rising and lowering, interpolated by the fifth-order polynomial, or rectangular trajectory used for climbing steps or stepping over obstacles, also interpolated with the polynomial. The parameters of the individual steps trajectory can be modified through the pilot interface.

4.2.3. Pattern Generation

In this stage, the trajectory of the pelvis is generated based on the trajectory of the end-effectors and ZMP. The controller is based on the preview controller developed by Kajita, Kanehiro, Kaneko, Fujiwara, and Yokoi (2003), which in the first iteration generates the initial CoM trajectory, and then simulates the motion using a multibody model of the robot to calculate the expected ZMP trajectory. Finally, the discrepancy between the initial ZMP reference and ZMP from multibody model simulation is used to modify the CoM reference to improve the ZMP tracking. Thanks to the second stage, even though the Preview Controller employs an inverted pendulum model that assumes that

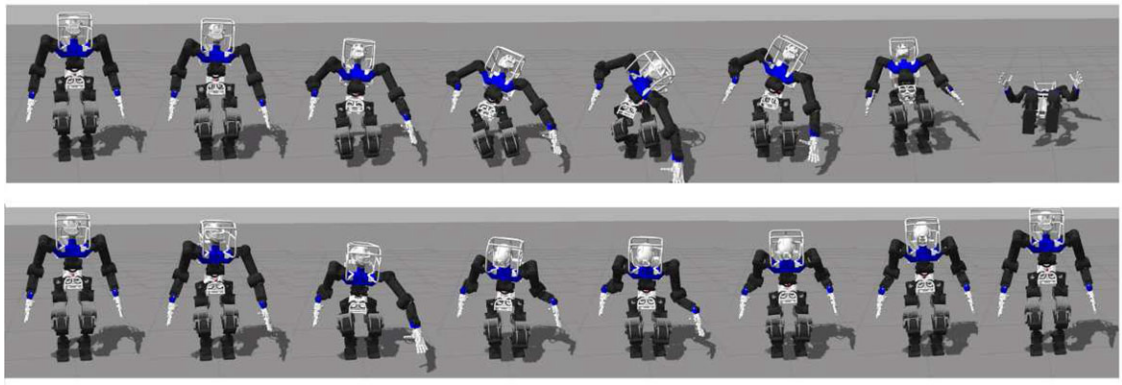


Figure 26. In the upper sequence, WALK-MAN falls due to a dynamically unfeasible motion, while in the lower sequence, the motion is dynamically feasible thanks to the dynamics constraint

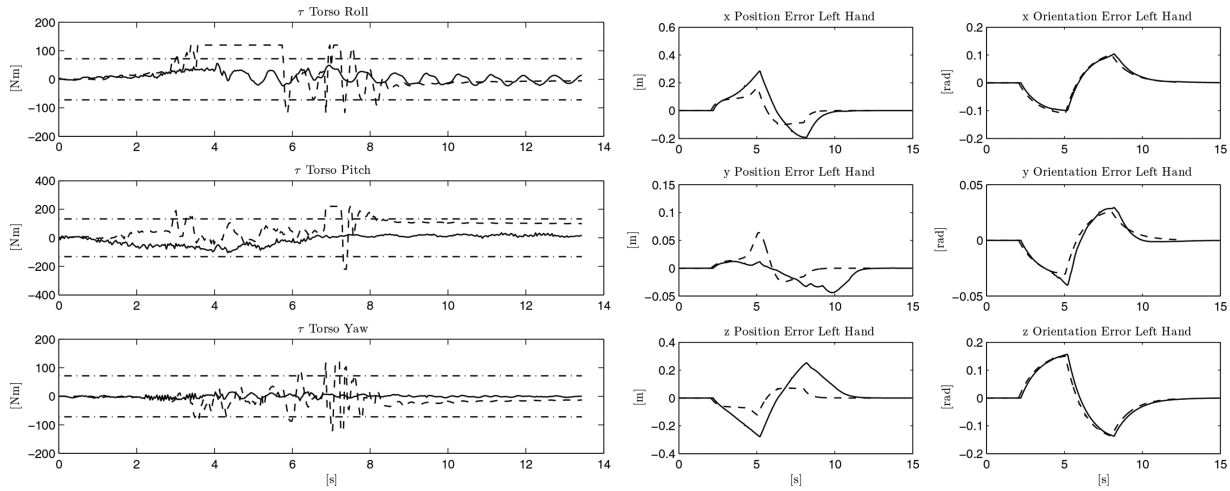


Figure 27. Left: measured torques on the joints of the torso while performing the task without (dashed lines) and with (continuous lines) the robot dynamics constraint. The constant lines show the limits on the torques. Right: Cartesian error on the left hand while performing the task without (dashed lines) and with (continuous lines) the robot dynamics constraint

CoM trajectory is within a plane, we are able to compensate for vertical motion of the CoM. Also the ZMP position in multibody model simulation for every sample is calculated in the horizontal plane that is derived from vertical ZMP reference. This is especially important when climbing steps or modulating COM height when stepping over an obstacle. Finally, the CoM reference is translated into the pelvis reference at every sampling time of the multibody simulation.

4.2.4. Stabilization

When the gait pattern is executed in the feed-forward manner, the errors in the modeling and environment reconstruction can induce unstable locomotion, especially on the WALK-MAN platform equipped with SEAs. To stabilize the locomotion, we use the torso position compliance controller by Nagasaka, Inaba, and Inoue (1999). The controller based on the estimated ZMP position modifies the pelvis reference to simultaneously track the ZMP reference and prevent divergence of CoM from original reference.

4.3. Manipulation

Each manipulation module implemented for the DRC tasks is a GYM with some additional functionality. The underlying structure of every module is thus composed of the basic methods of a GYM and the following additional components: a Finite State Machine (FSM), a trajectory generation library and a whole-body IK trajectory generator. The working principle of a manipulation module is the following. When a message (it can be either a string representing a parametrized action or a string representing an object and its pose inside the environment) arrives through the

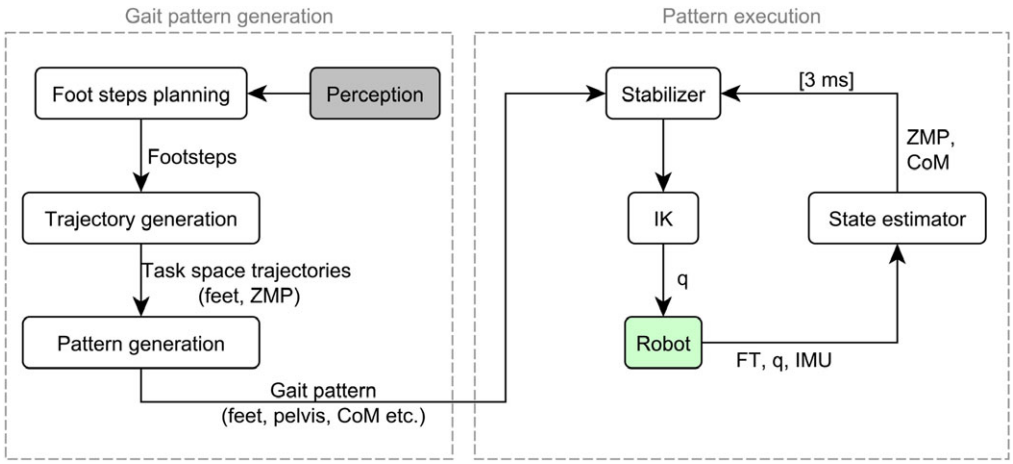


Figure 28. Locomotion control diagram

command interface, it constitutes the triggering condition for the FSM to be changed from one state to another. In every state, a trajectory generator object is called to create a trajectory for a given end-effector. At run-time a portion of the trajectory is then passed to the whole-body IK solver, described in Section 4.1, which computes the corresponding portion of joint displacement to be sent to the robot actuators. The WBIK solver takes into account all the joints presented in the kinematic chains of the robot and control the CoM position to be always inside the convex hull defined by the two feet as the highest priority task.

4.3.1. FSM

To cope with complex tasks, the FSM is needed to switch among different actions of the robot. Once the new status is received, the operator can send a message through the command interface. This changes the module state accordingly to the structure of the FSM. As an example, once the pilot receive the status "reach," one can send the "approach" command to the module. The FSM of the Valve module is depicted in Figure 29.

Every state corresponds to either a specific action or a waiting state.

Every module starts in an idle state. On the reception of a message containing a string of the type "*object*_data_sent," where *object* is a string specific to the module (e.g., "valve"), the transition to the ready state is made and the data structure contained in the message is stored within the module.

For the manipulation phase, we thus defined a common library of actions as follows: the *reach* action to let a robot end-effector reach the proximity of the object, the *approach* action to let the end effector touch the object, the *grasp* and *ungrasp* actions to close and open the hand/s, respectively, and the *move_away* action to move the end effector away from the object, once the robot releases it through the *ungrasp* action.

All those actions were executed via linear trajectories in the Cartesian space (see the following subsection).

4.3.2. Trajectory Generator Library

The trajectory generator library consists of two types of trajectory: linear and circular. The linear trajectories (from now on LT) are created via fifth-order polynomials, interpolating from the initial and final positions

$$x(t) = x_0 + a \cdot \frac{x_F - x_0}{2t_f^3} t^3 + b \cdot \frac{x_F - x_0}{2t_f^4} t^4 + c \cdot \frac{x_F - x_0}{2t_f^5} t^5, \tag{6}$$

where x_0 and x_F are the initial and final values, respectively, t is the current time, and t_f is the final time. a, b, c represent the coefficients of the polynomial.

The circular trajectories (CT) are parametrized on the rotation angle, where the polynomial interpolates from the initial to the final angular displacement of the trajectory. The Cartesian trajectory is thus computed as

$$\begin{aligned} \alpha(t) &= a \cdot \tfrac{\alpha_F}{2t_f^3} t^3 + b \cdot \tfrac{\alpha_F}{2t_f^4} t^4 + c \cdot \tfrac{\alpha_F}{2t_f^5} t^5 \\ x(t) &= R(\alpha(t)) \cdot x_0, \end{aligned} \tag{7}$$

where x_0 is the initial state, t is the current time, and t_f is the final time. a, b, c represent the coefficients of the approximation polynomial, and R is a rotation matrix with respect to a certain axis.

4.3.3. Car driving

To tackle the car driving problem some *ad hoc* solution is devised for both the steering wheel and the gas pedal

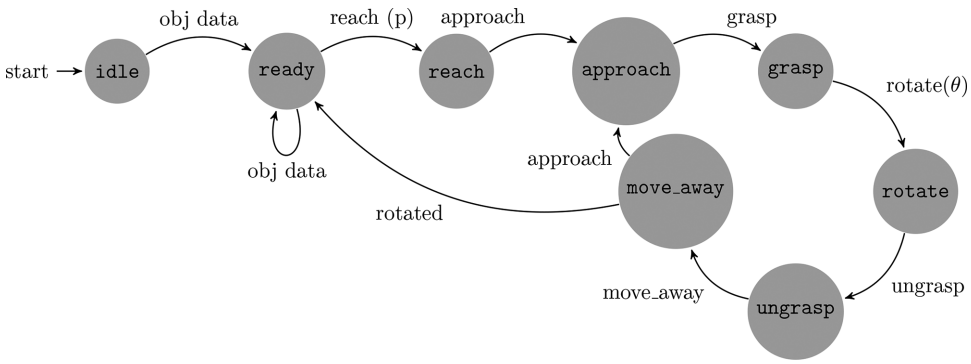


Figure 29. Finite State Machine of the Valve turning task

(the left leg steps on the gas pedal and the left arm steers the wheel). Three modifications were made on the vehicle: (1) an additional handle was mounted on the steering wheel, (2) a mechanical limitation was put under the accelerator pedal to limit the acceleration of the car, and (3) a special seat was designed to cope with the height of the robot. As the other modules, the car driving module consists of common actions plus a custom action to rotate the steering wheel.

4.3.4. Door opening

The door opening task consisted of the common set of actions described in Section 4.3.1 plus a set of custom actions reported in Table III. Moreover, it uses custom messages such as the *left/right* message to give the possibility of specifying the hand to be used and the *push/pull* message to determine the opening procedure of the door.

4.3.5. Valve turning

To cope with the valve turning task, we used a strategy similar to the door opening task. The pilot can specify if the robot has to use one hand (and which one: *left/right*) or both hands, through the corresponding messages and also the direction of turning (*CW/CCW*).

4.4. Perception

Exteroceptive and proprioceptive perceptions were important aspects both for the manipulation and the locomotion tasks of the DRC, either in a (semi) autonomous or in a teleoperated system. The main challenge for the perception system is to give as fast as possible enough information about the environment for handhold and foothold affordances depending on the manipulation or the locomotion task, respectively. For this reason, filtered data need to be acquired correctly in the lower level of the system and then either be used for teleoperation or (semi) automatically model the environment for a higher level robot–environment interaction. In particular, the data are going to be acquired and processed in three different levels. The acquisition and filtering take place either on the robot or on the field PC, while foothold/handhold modeling is executed on the pilot side, respecting the network bandwidth restrictions.

4.4.1. Point Cloud Acquisition and Filtering

The amount of data to be acquired on the robot PC and to be sent along the network to the field/pilot PC depends on the bandwidth specifications. For the stereo camera, 1MegaPixel unorganized RGB-D data are acquired in 4 fps. The same framerate has been used for the IMU data. A higher framerate has been chosen for the laser data. 1081 line point cloud data are acquired from the 2D laser sensor, while it is rotated with a speed of 0.5 rad/s (Figure 30). The amount of data were enough for having accurate and enough information for each task completion. All the data were transformed in the same fixed frame of the robot. A set of different RGB-D data grabbers were implemented in case the network bandwidth was decreased further, by reconstructing the point clouds on the side of the pilot and transmitting only the depth map and the camera information through the network, while RGB images could be further compressed with losing only little of the accuracy.

A set of real-time point cloud filters were first applied at the robot PC level to 3D points that were further away from the robot (e.g., 4 m). Given that local action planning with respect to the point cloud was enough for task completion, while the RGB images could be used for a pilot-driven global action planning. To reduce the number of outliers and the noise on the stereo cloud point data, we apply a set of bilateral filter (Paris & Durand, 2009) and a radius outlier removal, while for the laser cloud, we apply a shadow points removal filter for the ghost points on discontinuity edges. The IMU data coming from the accelerometer, gyroscope, and magnetometer were combined with a Madgwick filter (Madgwick, 2010), and the gravity vector was extracted. To guarantee that the size of unorganized point cloud data meets the bandwidth requirements, an automated uniformly random filter was used to extract

Table III. The module primitives specifications.

Task	Primitives	Description
Common	reach	LT to the specified object pose at a safety distance
	approach	LT to the specified pose in order to touch the object
	grasp	Hand closure in order to grasp the object with the specified hand
	ungrasp	Hand opening in order to detach the hand from the object
	move away	LT to a safety distance from the object
Car	accelerate (α)	CT around the robot ankle
	turn (θ)	CT around the center of the steering wheel
Door	turn handle	CT around the axis of rotation of the door handle
	push/pull door	CT around the axis of rotation of the door hinge
	support door	LT towards the specified point on the door
	open wide	CT around the axis of rotation of the door hinge
Valve	rotate(ϕ)	CT around the axis of rotation of the wheel valve

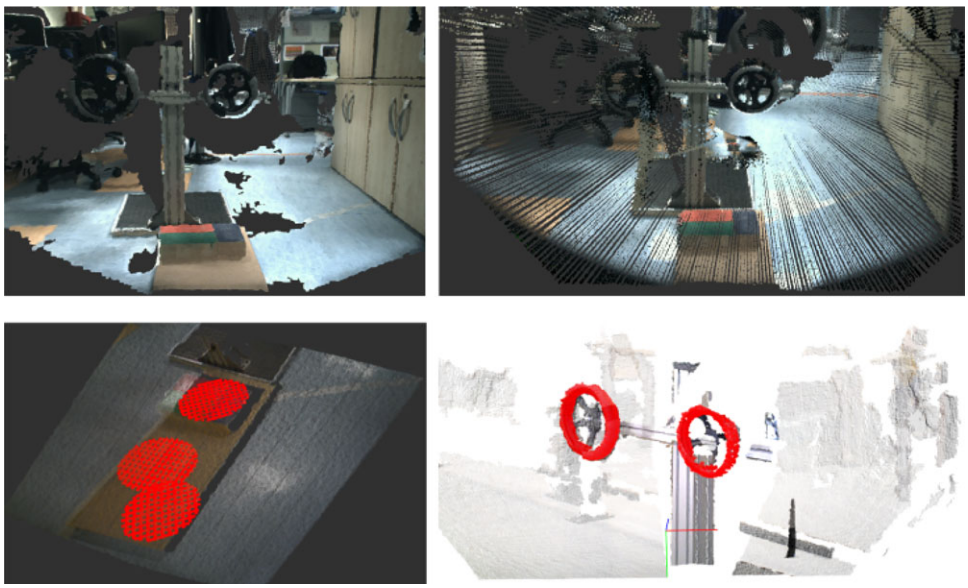


Figure 30. A stereo cloud (upper left) and the corresponding laser cloud (upper right). A set of automatically fitted circular patches used as foothold affordances (lower left), and valve detection results using RANSAC 3D circle fitting and evaluation in the acquired stereo cloud (lower right)

the extra points. At the field PC level, the line point cloud data from the laser are accumulated in a circular buffer to cover the whole scene before the full cloud is transmitted to the pilot. At the pilot PC level, a history of data over time is kept in a circular buffer such that they can be used in case some of the latest data are very noisy or incomplete, as well as for any potential data fusion process.

4.4.2. Handhold and Foothold Modeling

All the remaining data processing takes place at the pilot PC level. The laser data were used for very precise handhold or foothold detection when the robot was not moving, while the more uncertain stereo cloud was used for initial estimations, where usually the pilot could tweak the hand/foot poses accordingly. For the manipulation tasks, the pose of the grasp frame on the object to be aligned with the hand frame is defined from the task and the object itself. Thus, it is enough to detect the pose and the corresponding properties of each object in the environment. For the locomotion tasks, the footholds are detected as frames to be aligned with the foot.

In particular, various techniques were used for automatic object detection being as generic as possible for the manipulation tasks. An initial point cloud Euclidean distance segmentation (Trevor, Gedikli, Rusu, & Christensen,

2013) followed by a RANSAC model fitting (Fischler & Bolles, 1981) and model evaluation was enough to localize objects (Figure 30). In some cases, like in the valve on a wall, the drill on a table, or the door handle, the segmentation was replaced by a simple plane removal filtering, while for particular objects like the drill, a 3D point cloud template matching algorithm (Rusu et al., 2009) replaces the fitting process. For instance, the steering wheel and the valve were detected automatically with a torus or 3D circular fitting process, and their model evaluation was with respect to the size, color of the object, orientation with respect to the gravity vector coming from the IMU, and the height from the ground. Similarly, the door handle and the debris were modeled as line segments, while the drill was localized using template matching. In all cases, the main characteristics of each object were extracted, for example, the pose and radius for the valve, the size of the door handle, and its distance from the door's rotation axis. Given the difficulty of automatically localize objects in a very clutter and dynamically changing environment, a semiautonomous segmentation system was developed where the user clicks on a 3D point of the object of interest helping with the model fitting, by basically detecting the object himself. Finally, the pilot could tweak manually the pose and the properties of the fitted model to adjust it to the real-world data.

The representation of the foothold affordances is also a crucial aspect of the perception system, which was used only for rough terrain locomotion on the bricks and the steps. For this a set of circular planar patches of the size of the foothold were used (Kanoulas, 2014; Kanoulas & Vona, 2013, 2014a; Vona & Kanoulas, 2011). Even though these patches could be used in a fully automated local footstep planner method, to ensure reliability and safety, the process was semiautonomous. The pilot was clicking a sequence of points in the environment around where the sequence of footsteps need to be fitted. Then a sequence of automated nearest neighborhood searching of the size of the foot for each clicked point was taking place, followed by a flat circular patch fitting process, while the final output of the algorithm is a set of 6 DOF foothold frames along with the right or left foot label, and the foot sequence number. A set of four sequential footsteps per time could be handled considering the robot's drift. The pilot could tweak the pose of the footsteps manually to improve the fitting in cases where the point cloud noise was leading to errors.

The dynamically changing environment, its size, and the limitation of the current visual simultaneous localization and mapping systems in which automatically recovering from a registration error and failure was difficult were the main reason why a visual state estimation system was not integrated. The most promising and reliable system that was tried was the 3D visual/IMU Moving Volume KinectFusion one (Roth & Vona, 2012). During the experiments, it was realized that a single point cloud was enough to plan a sequence of four footsteps as well as complete all the manipulation tasks. Moreover, system calibration and ground truth data for foothold and handhold affordances were created using AprilTags (Olson, 2011).

5. WALK-MAN VALIDATION

The testing and validation of the WALK-MAN humanoid was performed prior to DRC with several executions of the DRC tasks inside the lab as well as during our participation in the DRC where the robot effectively performed the driving and the door opening task in the two runs in day one and day two of the competition. A failure in the power source of the robot prevented the continuation of the robot toward the turning of valve and subsequent tasks. This section introduces the material from some indoor-lab trials as well as from the trials during the DRC. The video media material summarizing the successful execution of these experiments can be found in the WALK-MAN EU Project website: http://walk-man.eu/results/videos/item/walk-man-drc-video-collection.html.

5.1. Manipulation

5.1.1. Car driving

The robot succeeded to perform the driving task both in the IIT facilities and in the DRC (see Figure 31). Two obstacles were positioned on the two sides of the track. Grasping of the wheel was performed with the operator superimposing the mesh of the steering wheel on the point cloud and subsequently commanding the robot to grasp the handle of the wheel. Using steps of throttle and commanding the rotation angle of the steering wheel for turning, the pilot guided the robot to drive the car to the end of the track.

5.1.2. Door opening

The door opening task was performed (Figure 32) both in the labs and in the DRC. Once the robot was positioned in front of the door, the operator sent the handle position and the width of the door as estimated from the perception data to the robot. Using the predefined actions, the robot grasped the handle and opened the door successfully.

The door opening task was attempted twice during the two runs of the DRC finals. In both runs while we successfully executed the phase of the door opening, we then experienced a robot collapse incident in front of the door. In both runs, this was caused by a sudden power cut caused by the release of the main power relay switch on the power management board in the power backpack. A malfunctioning on the regulator component of the relay driving circuit was the reason for the power relay release. Unfortunately, we did not manage to track this power shut down issue after our first run, and the same event also happened during the second run that eventually prevented us to continue with the following tasks.

Figure 31. Driving car execution in the DRC

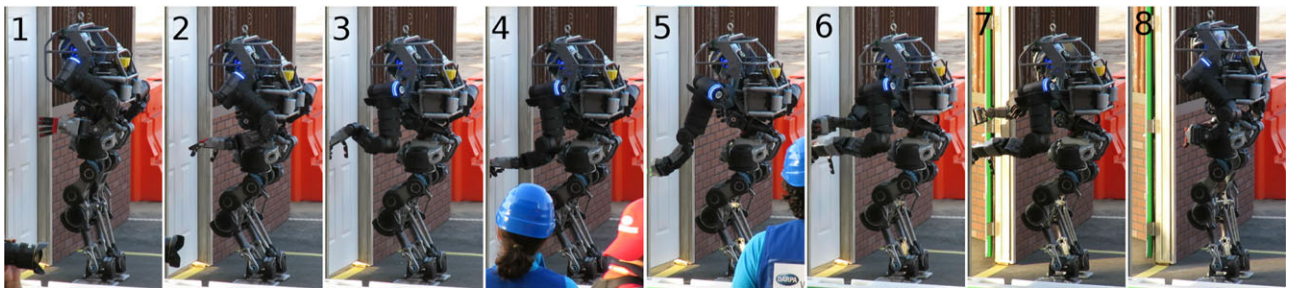


Figure 32. Door opening task execution in the DRC

It is important to mention here that in both fall events, the robot collided with the ground starting from a standing posture. In one of the two falls, the robot's initial contact to the ground was concentrated on a single and small area on the side of the left elbow joint resulting in high-impact torques on the shoulder abduction joint. In both incidents, the robot survived from the falls with no damage in any of its mechatronic components and was operational immediately after. Although no data were collected during the fall events and we cannot positively argue that SEAs assisted the robot to escape without damage, we believe that SEAs had a beneficial contribution in the protection of harmonic reduction drives during the fall events. This is in agreement with impact torque reduction demonstrated in Figure 11.

5.1.3. Valve turning

To test the valve turning module, a test-bed for mounting valves of different size was built at IIT premises, as shown in Figure 33. The robot performed the task by firstly executing a walking task to arrive at a reachable distance from the valve. With the operator assistance, the localization of the valve from the perception data was derived and sent to the robot. A series of subtasks that involved reaching, grasping, turning, and releasing the valve were then executed to complete turn the valve. The sequence of these actions was repeated by the robot until the operator realized that the valve had completed a full rotation. To mention here that during the valve turning task perception inaccuracies as well as pilot actions in the valve localization resulted on errors in the estimation of the valve location/orientation. As a consequence, during the approach and grasp phase as well as during the turning action, the commanded trajectories were deviated from the ideal ones introducing constraints to the manipulation motions during the interactions. In these cases, the integrated compliance has demonstrated its benefits showing ability to cope with few centimeters of errors between the end effector and the environment constraints minimizing the resulted disturbances to the robot body by accommodating these inaccuracies during interaction.

5.2. Locomotion

During the DRC as well as prior to with lab trials, we confirmed that the robot is able to stably walk and turn on the level ground. Figure 34 (left) shows the ZMP plot from forward walking experiment with stride length 0.1 m, step width 0.28 m, step time 1.5 s, and double support time 0.3 s. Although the measured ZMP does not follow exactly the reference, it is within the support polygon that during stance phase is 0.1 m in the lateral and 0.06 m in the medial direction from the ZMP reference.

Figure 34 (right) shows a comparison between the torque measured by the ankle pitch motor sensor and the torque around the parallel axis as reported by the foot force torque sensor. We can see very strong correlation between the data with small differences caused by the displacement between the sensors.

Figure 35 shows snapshots from an experiment of dynamic stepping over an obstacle. In this case, the step size was 0.35 m, step time was 0.7 s, and the obstacle height was 5 cm.

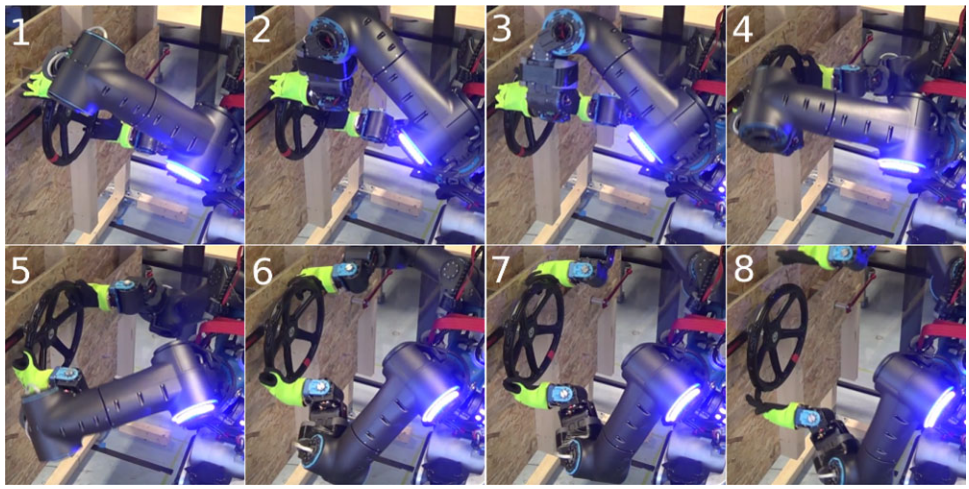


Figure 33. Valve turning task execution inside the IIT facility

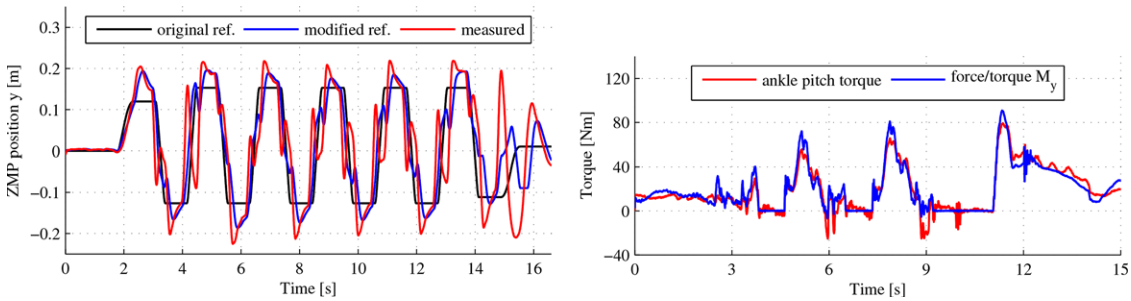


Figure 34. Left: lateral ZMP data from forward walking experiments. The black, blue, and red lines represent the original ZMP reference of the planned motion, reference modified by the ZMP feedback controller, and measured ZMP position, respectively. Right: comparison of torque measured by the ankle pitch torque sensor and foot force/torque sensor

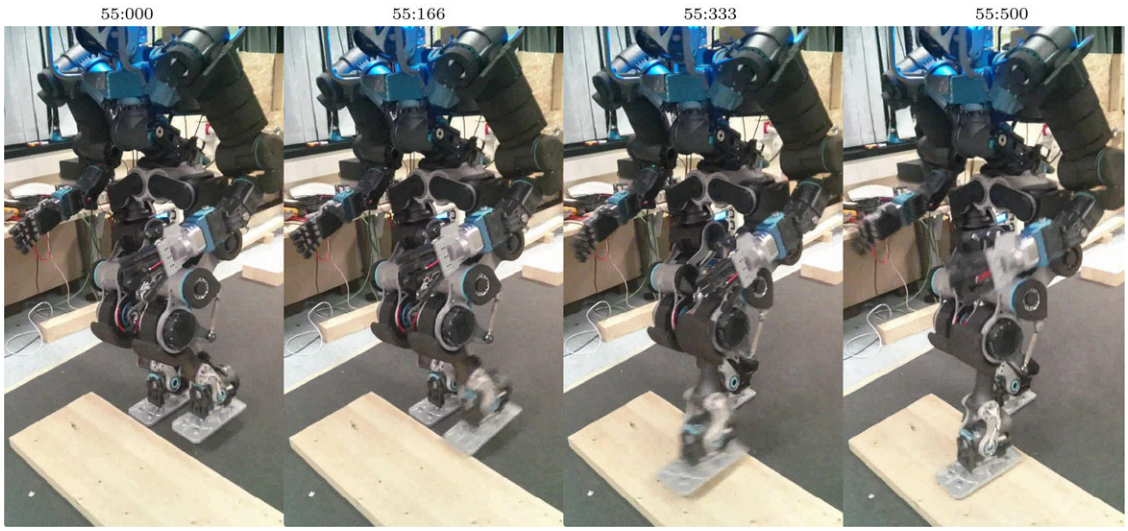


Figure 35. Snapshots from dynamic stepping on the obstacle. The obstacle is 5 cm high and the step time is 0.7 s. The digits above pictures represent the time of the experiment in ss:mmm format

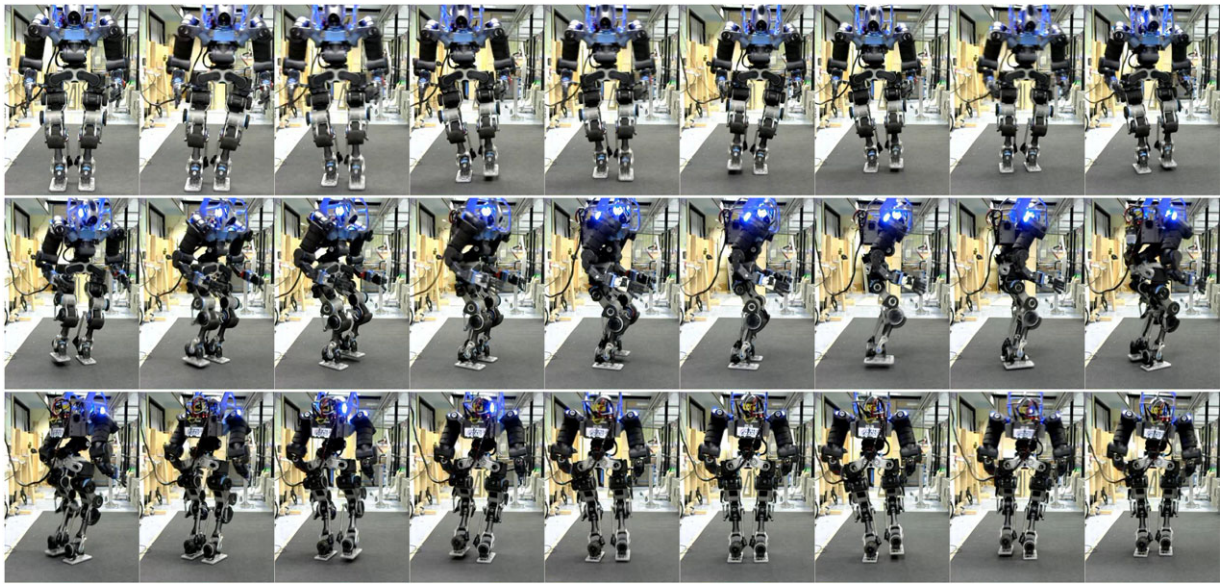


Figure 36. Snapshots from locomotion experiments. The robot starts from walking backward, then turns left on the spot 180°, and finally walks forward

Finally, Figure 36 shows snapshots from locomotion experiments during which the robot was walking backward, turning on the spot, and walking forward.

5.3. Perception

A set of visual experiments is presented in Figure 37, which validates the point cloud quality after the filtering and the model fitting for various objects and foothold affordances.

In particular, the stereo point clouds of a single camera frame were used for testing, while the pilot was part of the modeling loop by providing a single point on the object or foothold of interest, as described above, to help with the segmentation. The code was developed using the Point Cloud (Rusu & Cousins, 2011) and the Surface Patch (Kanoulas & Vona, 2014b) libraries.

For the case of the door handle, the dominant door plane was removed from the cloud, while a line segment was fitted using the RANSAC approach. Similarly for the debris, the dominant ground plane was removed first. In the case of the valve, a 3D circle fits to the point cloud, while for the drill the template matching was used, since the model of the drill was a priori known. For the footstep patches, the pilot provided the center points of each footstep in the correct order, by also specifying each time whether the foot is the left (in green) or the right (in red) one. Most of the times the object or foothold detection is successful, but the pilot is able to tweak the properties of each affordance when the error is observed visually to be big.

6. DISCUSSION

The participation to a complex and large challenge such as the DRC required a lot of human effort and time: every aspect of the robot, from hardware to electronics, software, and control approaches, was tested and improved during the months before the challenge. In this section, we would like to highlight some lessons learned focusing on four main topics: Mechatronics and Low Level Control, Task Execution Strategy, Software Modularity and Utilities, and Pilot Training and Management.

6.1. Mechatronics and Low Level Control

WALK-MAN was designed and realized within the very short period of time of less than 1 year. Although this is considered as a great achievement for the team, it certainly had some significant consequences to the robot readiness that strongly affected its performance in the DRC. During the design phase, we had limited time to perform iterations of the robot design, and sometimes we had to adopt and follow design choices without the possibility of fully evaluating them. This was relevant for example to the design and validation of the actuation units. One critical parameter was the choice of the intrinsic elasticity level that in the first version of the actuators was selected considering two main objectives. For favoring the joint torque control, the first objective was to maximize torque sensing resolution by providing the largest deflection possible, subject to the second objective that was a constraint for the lowest stiffness level computed based on the maximum deflection range and stress

Figure 37. Visual perception results for the door handle, the circular valve, the drill, the debris, and the rough terrain. The RGB image, the point cloud data, and the fitted model appear in each row correspondingly

level on the elastic component at the peak torque of each joint. This initially resulted in stiffness level in the range of 2500 N m/rad for the high-power actuation modules. However, although our target was to have WALK-MAN running in torque/impedance control mode, the torque controller design and testing was not completed in time. As a result, we were forced to use position-based joint control schemes that could not cope well with the low intrinsic stiffness of WALK-MAN joints limiting the performance particularly of our locomotion ZMP and COM controllers. As a correction action and since there was not adequate time to tune the torque and impedance control on the whole robot, we increased the stiffness of the series elasticity up to 6000 N m/rad. This improved the performance of the joint position control and eventually enabled the robot to demonstrate basic locomotion functionality compromising however the intrinsic adaptation to small terrain irregularities. WALK-MAN will soon demonstrate its torque/impedance control operation that will enable us to improve further the current performance of the SEA joints using the more suitable torque regulation instead of the position control used during the DRC. The limited available time for testing the robot before the DRC was also relevant to the robot collapse incident in front of the door. Our onboard power source was installed 1 week before the departure of the robot for the DRC, and it was not extensively debugged for prolonged period of operation. In both runs of the door task, the falling event was a result of a sudden power cut caused by the release of the main power relay switch on the power management board in the backpack. The above mechatronic/low-level control examples of limited performance or malfunctioning in the hardware show that even if the hardware potential is high in terms of expected performance, to achieve reliable and consistent operation with such complex machine, and to compete effectively during the challenge necessitated much more extensive prior testing and debugging and eventually some mechatronic revisions on the first prototype to tune its performance.

6.2. Task Execution Strategy

In the short time before the DRC, it was very hard to organize a technical discussion about the software and control approaches, and we often had to pick specific strategies even if not all the members of the team considered them to be the best approaches. As an example, most of the team approved a two-arm control strategy for valve turning, which is a very good and generic solution in order to handle very large and hard friction valves. Nevertheless, given the hard deadline, in our opinion a better choice is to use tactical solutions that are easy and fast to implement and debug, focused on solving the specific task requirements instead of solving the general problem; thus, the valve module was developed with a single-arm strategy, guaranteed to work only on valves compliant with DRC rules specification. Usually, in large companies and in organized open-source projects, coding quality standards, style, and procedures are mandatory and adopted by the whole team. Such approach requires dedicated advanced training and hence time. In our case, the team was formed on purpose for the DRC by including researchers of different groups with different expertise and standards. In similar situations, we strongly suggest to let

every programmer choose his programming style and control approaches designing a flexible architecture to support the different users. Our architecture reflects this need by not enforcing any specific control algorithm in the modules implementation, so that developers were free to read just the sensors and to control the joints they required to achieve their specific tasks.

6.3. Software Modularity and Utilities

As already said, the development of the WALK-MAN software architecture has been organized to enhance code reusability and modularity. Designing a modular architecture does not always come for free: each layer requires its own API to interface with each other, and each API has to be maintained and updated. Nevertheless, our team could have never been able to develop and change the modules without such APIs. The benefits of APIs have also been exemplified by what happened after the DRC rush, where some parts of the architecture have gone under a redesign process, while some others have been abandoned without impacting the whole system. The most striking example of the effort done in avoiding the boilerplate code together with the use of GYM, is the DRC driving module. Indeed, it was developed in a very small amount of time by a master student (i.e., nonexpert code developer), which managed to control the gas pedal and to steer the wheel in less than two weeks. The module was then refined and tested for a week by two developers of the team and eventually used in the challenge. During the development of the control modules and during the DRC, multiple logging utilities, both on the robot and on the pilot PCs, were storing information useful for debugging. Such information was: sent commands, status of the robot, point clouds, failures, and warnings from the control modules. These logging utilities were very custom designed and primitive in their capabilities due to lack of time, but they provided enough information to speed up the debugging process. Future work will include a generic logging class, integrated in each module with the same style of GYM and GW. In our opinion, the architecture, with its APIs and GYM, was mature enough to allow the development of all the DRC tasks. For example, both the valve and drill task have been performed reliably in the lab many times. Moreover, we were also ready to perform the surprise task, since we had a generic manipulation module for those situations. Unfortunately, we have not been able to perform said tasks in the DRC due to the problems with the door task. Generally speaking, the architecture structure and implementation did not affect any task during the DRC and did not impose any constraint on the control strategies implemented in each task module. Few main issues (e.g., multithreading issues, network bridge incompatibility with custom YARP ports) were detected during the months before the competition, and they were solved in a small amount of time without affecting the software users.

6.4. Pilot Training and Management

Finally, there was a trade-off between the effort required from the pilots during the challenge and the development effort required to offload them from some tasks. As an example, we decided to skip the development an artificial vision system for object recognition, and trained the perception pilot in order to be very fast and accurate in that task. We also noticed that training the pilot in order to know better a module behavior pays off as much as an improvement in the module code or control law in the short time. Note that this solution cannot be used in other situation, where pilots may be untrained people or where the task complexity, if not solved in the software, may require impossible pilot efforts. Our architecture requires tens of modules to be running at the same time across multiple computers, and the modules starting order may become complex to maintain. After the first tests with the whole architecture running, we noticed that lot of pilot effort had to be put in starting the modules in the right order. We decided to reduce such requirements as much as possible, and finally ended up with only the ROS and YARP name servers to be started before all the other modules. We believe that the effort to provide asynchronous starting order is compensated as the architecture increases in complexity.

7. CONCLUSIONS

We introduced WALK-MAN, a humanoid robot that is being developed inside the European Commission project WALK-MAN with the target to demonstrate advanced capabilities including powerful manipulation, robust balanced locomotion, high-strength capabilities, and physical sturdiness, and be able to operate in realistic challenging workspaces. An overview of the WALK-MAN hardware, which was designed and built in less than a year, was presented in this paper. The robot software architecture was discussed, and the main software components and their interconnection were introduced. The loco-manipulation motion-generation and control framework that was developed to enable the robot to execute manipulation and locomotion tasks as well as the pilot interface functionality and features were described in detail. The first validation of the WALK-MAN robot was performed with the participation of our team in the DRC competition where the robot was able to function and execute some of the challenging tasks under the control of a pilot operator. With the participation in the DRC, the first milestone of the project was achieved, and it now continues to reach beyond DRC. In the second part of the project, civil defense bodies are being consulted to tune the robot abilities and future developments and assist to define specifications for a true real-world challenge with realistic and realizable scenarios for WALK-MAN.

ACKNOWLEDGMENTS

The development of the WALK-MAN platform is supported by the WALK-MAN FP7-ICT-2013-10 European Commission project. This work would not have been possible without the major support from the Italian Institute of Technology and the University of Pisa, and the great talent skills and incredible commitment and passion of all members of WALK-MAN team.

REFERENCES

Akachi, K., Kaneko, K., Ota, S., Miyamori, G., Mirata, M., Kajita, S., & Kanehiro, F. (2005). Development of humanoid robot hrp-3p. In IEEE-RAS International Conference on Humanoid Robots (pp. 50–55).

Bagheri, M., Ajoudani, A., Lee, J., Caldwell, D. G., & Tsagarakis, N. G. (2015). Kinematic analysis and design considerations for optimal base frame arrangement of humanoid shoulders. In IEEE International Conference on Robotics and Automation (ICRA) (pp. 2710–2715). Seattle, WA.

Catalano, M. G., Grioli, G., Farnioli, E. A. S., Piazza, C., & Bicchi, A. (2014). Adaptive synergies for the design and control of the pisa/iit softhand. International Journal of Robotics Research, 33, 768–782.

Chiacchio, P., Chiaverini, S., Sciavicco, L., & Siciliano, B. (1991). Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy. The International Journal of Robotics Research, 10(4), 410–425.

De Schutter, J., De Laet, T., Rutgeerts, J., Decré, W., Smits, R., Aertbeliën, E., ... Bruyninckx, H. (2007). Constraint-based task specification and estimation for Sensor-Based robot systems in the presence of geometric uncertainty. The International Journal of Robotics Research, 26(5), 433–455.

Englsberger, J., Werner, A., Ott, C., Henze, B., Roa, M. A., Garofalo, G., ... Albu-Schaffer, A. (2014). Overview of the torque-controlled humanoid robot toro. In IEEE-RAS International Conference on Humanoid Robots (pp. 916–923).

Escande, A., Mansard, N., & Wieber, P.-B. (2014). Hierarchical quadratic programming: Fast online humanoid-robot motion generation. International Journal of Robotics Research, 33(7), 1006–1028.

Fang, C., Rocchi, A., Mingo Hoffman, E., Tsagarakis, N. G., & Caldwell, D. G. (2015). Efficient self-collision avoidance based on focus of interest for humanoid robots. In IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids) (pp. 1060–1066).

Ferreau, H. J., Kirches, C., Potschka, A., Bock, H. G., & Diehl, M. (2014). qpOASES: a parametric active-set algorithm for quadratic programming. Mathematical Programming Computation, 6(4), 327–363.

Fischler, M. A.& Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6), 381–395.

Fok, C.-L. & Sentis, L. (2016). Integration and usage of a ROS-Based whole body control software framework. In Robot Operating System (ROS), Studies in Computational Intelligence (pp. 535–563). Springer. Switzerland

Herzog, A., Righetti, L., Grimminger, F., Pastor, P., & Schaal, S. (2014). Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics. In Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (pp. 981–988).

Hirai, K., Hirose, Y., Haikawa, Y., & Takenaka, T. (1998). The development of honda humanoid robot. In IEEE International Conference on Robotics and Automation ICRA (pp. 1321–1326).

Hirose, M. & Ogawa, K. (2007). Honda humanoid robots development. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences (pp. 11–19).

Ito, Y., Nakaoka, T., Urata, J., Nakanishi, Y., Okada, K., & Inaba, M. (2012). Design and development of a tendon-driven and axial-driven hybrid humanoid leg with high-power motor driving system. In Proc. 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids) (pp. 475–480).

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., & Yokoi, K. H. K. (2003). Biped walking pattern generation by using preview control of zero-moment point. In IEEE International Conference on Robotics and Automation (ICRA) (pp. 1620–1626).

Kaneko, K., Harada, K., Kanehiro, F., Miyamori, G., & Akachi, K. (2008). Humanoid robot hrp-3. In IEEE International Conference on Intelligent Robot Systems (IROS) (pp. 2471–2478).

Kanoulas, D. (2014). Curved surface patches for rough terrain perception. PhD thesis, CCIS, Northeastern University.

Kanoulas, D. & Vona, M. (2013). Sparse surface modeling with curved patches. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (pp. 4209–4215).

Kanoulas, D. & Vona, M. (2014a). Bio-Inspired rough terrain contact patch perception. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (pp. 1719–1724).

Kanoulas, D. & Vona, M. (2014b). The surface patch library (SPL). In IEEE ICRA Workshop: MATLAB/Simulink for Robotics Education and Research. Available at: http://ccis.neu.edu/research/gpc/spl.

Kanoun, O., Lamiraux, F., & Wieber, P.-B. (2011). Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task. IEEE Transactions on Robotics, 27(4), 785–792.

Khatib, O. (1987). A unified approach for motion and force control of robot manipulators: The operational space formulation. IEEE Transactions on Robotics and Automation, 3(1), 43–53.

Laffranchi, M., Tsagarakis, N. G., Cannella, F., & Caldwell, D. G. (2009). Antagonistic and series elastic actuators: a comparative analysis on the energy consumption. In IEEE/RSJ

International Conference on Intelligent Robots and Systems (pp. 5678–5684). IEEE.

Lohmeier, S., Buschmann, T., Ulbrich, H., & Pfeiffer, F. (2006). Modular joint design for performance enhanced humanoid robot lola. In IEEE International Conference on Robotics and Automation (ICRA) (pp. 88–93).

Madgwick, S. O. (2010). An efficient orientation filter for inertial and inertial/magnetic sensor arrays. Technical Report, University of Bristol, UK.

Mansard, N., Stasse, O., Evrard, P., & Kheddar, A. (2009). A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks. In International Conference on Advanced Robotics (pp. 1–6).

Metta, G., Fitzpatrick, P., & Natale, L. (2006). Yarp: yet another robot platform. International Journal on Advanced Robotics Systems, 3(1), 43–48.

Mingo Hoffman, E., Rocchi, A., Tsagarakis, N. G., & Caldwell, D. G. (2016). Robot dynamics constraint for inverse kinematics. In International Symposium on Advances in Robot Kinematics, ARK Grasse, France, June 27–30, 2016.

Nagasaka, K., Inaba, M., & Inoue, H. (1999). Stabilization of dynamic walk on a humanoid using torso position compliance control (in Japanese). In Proceedings of 17th Annual Conference of the Robotics Society of Japan (pp. 1193–1194).

Nakamura, Y. (1990). Advanced robotics: redundancy and optimization, 1st edition. Boston, MA: Addison-Wesley Longman Publishing.

Nakamura, Y.& Hanafusa, H. (1987). Optimal redundancy control of robot manipulators. International Journal of Robotics Research, 6(1), 32–42.

Nakamura, Y., Hanafusa, H., & Yoshikawa, T. (1987). Task-priority based redundancy control of robot manipulators. International Journal of Robotics Research, 6(2), 3–15.

Nakanishi, J., Cory, R., Mistry, M., Peters, J., & Schaal, S. (2008). Operational space control: A theoretical and empirical comparison, 27(6), 737–757.

Negrello, F., Garabini, M., Catalano, M., Malzahn, J., Caldwell, D., Bicchi, A., & Tsagarakis, N. (2015). A modular compliant actuator for emerging high performance and fall-resilient humanoids. In IEEE-RAS International Conference on Humanoid Robots (pp. 414–420).

Negrello, F., Garabini, M., Catalano, M. G., Kryczka, P., Choi, W., Caldwell, D. G., ... Tsagarakis, N. G. (2016). Walk-man humanoid lower body design optimization for enhanced physical performance. In IEEE International Conference on Robotics and Automation (ICRA) (pp. 1817–1824). IEEE.

Ogura, Y., Aikawa, H., Shimomura, A., Morishima, A., & Lim, H. T. A. (2006). Development of a new humanoid robot wabian-2. In IEEE International Conference on Robotics and Automation (ICRA) (pp. 76–81).

Olson, E. (2011). AprilTag: a robust and flexible visual fiducial system. In IEEE International Conference on Robotics and Automation (ICRA) (pp. 3400–3407).

Paine, N., Mehling, J. S., Holley, J., Radford, N. A., Johnson, G., Fok, C.-L., & Sentis, L. (2015). Actuator control for the nasa-jsc valkyrie humanoid robot: A decoupled dynamics approach for torque control of series elastic robots. Journal of Field Robotics, 32(3), 378–396.

Paris, S.& Durand, F. (2009). A fast approximation of the bilateral filter using a signal processing approach. International Journal of Computer Vision, 81(1), 24–52.

Park, I., Kim, J., & Oh, J. (2007). Mechanical design of the humanoid robot platform hubo. Journal of Advanced Robotics, 21(11), 1305–1322.

Parmiggiani, A., Maggiali, M., Natale, L., Nori, F., Schmitz, A., Tsagarakis, N., ... Metta, G. (2012). The design of the icub humanoid robot. International journal of humanoid robotics, 9(04), 1250027.

Pratt, G. & Williamson, M. (1995). Series elastic actuators. In IEEE International Conference on Intelligent Robot Systems (IROS) (pp. 399–406).

Pratt, J., Koolen, T., De Boer, T., Rebula, J., Cotton, S., Carff, J., ... Neuhaus, P. (2012). Capturability-based analysis and control of legged locomotion, part 2: Application to m2v2, a lower-body humanoid. International Journal of Robotics Research, 31, 1117–1133.

Rocchi, A., Hoffman, E. M., Caldwell, D. G., & Tsagarakis, N. G. (2015). Opensot: a whole-body control library for the compliant humanoid robot coman. In IEEE International Conference on Robotics and Automation (ICRA) (pp. 1093–1099).

Roozing, W., Li, Z., Medrano-Cerda, G. A., Caldwell, D. G., & Tsagarakis, N. G. (2016). Development and control of a compliant asymmetric antagonistic actuator for energy efficient mobility. IEEE/ASME Transactions on Mechatronics, 21(2), 1080–1091.

Roth, H. & Vona, M. (2012). Moving Volume KinectFusion. In British Machine Vision Conference (BMVC) (pp. 1–11).

Rusu, R. B., Blodow, N., & Beetz, M. (2009). Fast point feature histograms (FPFH) for 3D registration. In International Conference on Robotics and Automation (ICRA) (pp. 3212–3217).

Rusu, R. B. & Cousins, S. (2011). 3D is here: point cloud library (PCL). In IEEE International Conference on Robotics and Automation (ICRA) (pp. 1–4), Shanghai, China.

Saab, L., Ramos, O., Keith, F., Mansard, N., Soueres, P., & Fourquet, J. (2013). Dynamic whole-body motion generation under rigid contacts and other unilateral constraints. IEEE Transactions on Robotics, 29(2), 346–362.

Sentis, L., Park, J., & Khatib, O. (2010). Compliant control of multicontact and center-of-mass behaviors in humanoid robots. IEEE Transactions on Robotics, 26(3), 483–501.

Sentis, L., Petersen, J., & Philippsen, R. (2013). Implementation and stability analysis of prioritized whole-body compliant controllers on a wheeled humanoid robot in uneven terrains. Autonomous Robots, 35(4), 301–319.

Settimi, A., Pavan, C., Varricchio, V., Ferrati, M., Hoffman, E. M., Rocchi, A., ... Bicchi, A. (2014). A modular approach for remote operation of humanoid robots in search and rescue scenarios. Modelling and simulation for autonomous

systems, Vol. 8906, (pp. 192–205). Springer, Cham, Switzerland.

Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2008). Robotics: modelling, planning and control, 1st edition. London: Springer.

Siciliano, B. & Slotine, J.-J. E. (1991). A general framework for managing multiple tasks in highly redundant robotic systems. In "Robots in Unstructured Environments," 91 ICAR., Fifth International Conference on Advanced Robotics (pp. 1211–1216), IEEE.

Trevor, A. J., Gedikli, S., Rusu, R. B., & Christensen, H. I. (2013). Efficient organized point cloud segmentation with connected components. Semantic Perception Mapping and Exploration (SPME).

Tsagarakis, N., Metta, G., Sandini, G., Vernon, D., Beira, R., Becchi, F., ... Caldwell, D. (2007). icub -the design and realization of an open humanoid platform for cognitive and neuroscience research. Journal of Advanced Robotics, 21(10), 2151–1175.

Tsagarakis, N., Zhibin, L., Saglia, J., & Darwin, G. C. (2011). The design of the lower body of the compliant humanoid robot ccub. In IEEE International Conference on Robotics and Automation (ICRA) (pp. 2035–2040).

Tsagarakis, N. G., Cerda, G. M., Li, Z., & Caldwell, D. G. (2013). Compliant humanoid coman: Optimal joint stiffness tuning for modal frequency control. In IEEE International Conference on Robotics and Automation (ICRA) (pp. 665–670).

Vona, M. & Kanoulas, D. (2011). Curved surface contact patches with quantified uncertainty. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 1439–1446).