A modular approach for remote operation of humanoid robots in search and rescue scenarios

Alessandro Settimi*†, alessandro.settimi@for.unipi.it, Italy

Corrado Pavan*, corradopav@gmail.com, Italy

Valerio Varricchio*†, valerio.varricchio@for.unipi.it, Italy

Mirko Ferrati*, mirko.ferrati@centropiaggio.unipi.it, Italy

Enrico Mingo Hoffman†, enrico.mingo@iit.it, Italy

Alessio Rocchi†, alessio.rocchi@iit.it, Italy

Kamilo Melo*, kamilo.melo@km-robota.com, Italy

Nikos G. Tsagarakis†, nikos.tsagarakis@iit.it, Italy

Antonio Bicchi*†, antonio.bicchi@centropiaggio.unipi.it, Italy

†Department of Advanced Robotics, Istituto Italiano di Tecnologia, via Morego, 30, 16163 Genova
*Centro di Ricerca "E.Piaggio", Dipartimento di Ingegneria dell'Informazione, Università di Pisa, Italy

Abstract

In the present work we have designed and implemented a modular, robust and user-friendly Pilot Interface meant to control humanoid robots in rescue scenarios during dangerous missions. We follow the common approach where the robot is semi-autonomous and it is remotely controlled by a human operator. In our implementation, YARP is used both as a communication channel for low-level hardware components and as an interconnecting framework between control modules. The interface features the capability to receive the status of these modules continuously and request actions when required. In addition, ROS is used to retrieve data from different types of sensors and to display relevant information of the robot status such as joint positions, velocities and torques, force/torque measurements and inertial data. Furthermore the operator is immersed into a 3D reconstruction of the environment and is enabled to manipulate 3D virtual objects. The Pilot Interface allows the operator to control the robot at three different levels. The high-level control deals with human-like actions which involve the whole robot's actuation and perception. For instance, we successfully teleoperated IIT's *COmpliant huMANoid* (COMAN) platform to execute complex navigation tasks through the composition of elementary walking commands (e.g. `[walk_forward, 1m]`). The mid-level control generates tasks in cartesian space, based on the position and orientation of objects of interest (i.e. valve, door handle) w.r.t. a reference frame on the robot. The low level control operates in joint space and is meant as a last resort tool to perform fine adjustments (e.g. release a trapped limb). Finally, our Pilot Interface is adaptable to different tasks, strategies and pilot's needs, thanks to a modular architecture of the system which enables to add/remove single front-end components (e.g. GUI widgets) as well as back-end control modules on the fly.

Keywords: Robot Tele-operation, Humanoid Robotics, Software Integration

1 Introduction

A number of recent calamities, such as the Deepwater Horizon oil spill and the Fukushima Dai-ichi nuclear disaster, have highlighted the enormous potential of robots capable to perform hazardous activities in future disaster response operations [1], thus producing a growing interest in Urban Search And Rescue (USAR) robotic research worldwide. In this context, initiatives such as the DARPA Robotics Challenge (DRC) introduced the idea of using humanoids to manage disaster situations. Although several robot typologies (e.g. wheeled [1] or snake robots [2]) have been considered to cope with USAR missions, humanoid robots can take advantage of a natively superior suitability to deal with environments and tools designed for humans.

On the one hand, despite the increasing low-level capabilities of humanoid robots, teleoperation is still essential to exploit the human competence in terms of decision making, strategic thinking, perception capabilities and overall awareness of a task [3]. On the other hand, telecommunication problems like intermittent availability, low-bandwidth and latency of the connection can occur in disaster scenarios and make the need for a certain degree of autonomy undeniable. The mentioned remarks enforce the ever increasing trend towards a *semi-autonomous* or *supervisory* control [3]. We hereby present ideas to build a semi-autonomous framework for the control of a humanoid robot in disaster scenarios and describe features of our teleoperation Pilot Interface. Among the main desirable functionalities recognized, we require that the human operator is able to issue symbolic commands to the robot, select the level of autonomy with which the robot performs each task and receive visual and status information feedback. In addition, the interface is designed to be modular and reconfigurable based on the peculiar needs for the task or the environment conditions and is thought to be general in order to be used by different kinds of robots performing in disaster scenarios.

This paper is organized as follows. We introduce the main concepts related to robot autonomy in section 2. In section 3 we address our Pilot Interface software architecture, while a few implementation details are provided in section 4. The main GUI components are depicted in section 5. Finally in section 6 some applications are shown.

2 The need for semi-autonomy

Due to the relatively low autonomous capabilities of state-of-the-art humanoid robotics, teleoperation still turns out among the most popular control solutions. In fact, literature includes examples of fully-teloperated robots able to perform complex tasks like driving lift trucks [4] with *ad-hoc* cockpits. At the forefront of this field, NASA's Robonaut humanoid can be reliably piloted to perform Extravehicular Activities [5] through haptic interfaces, predictive displays and telepresence devices, while in recent works motion capture technology substitutes master-slave systems for whole-body motions [6] and power tools manipulation [7].
However, despite the practical effectiveness of such methods, they embrace the paradigm of direct control, which neglects telecommunication problems and is not feasible to deal with disaster scenarios addressed in this work.
In order to tackle communication problems, literature proposes examples of semi-autonomous control architectures. The semi-autonomous control is a scheme which effectively integrates teleoperation with the increasing low-level robot autonomy. It allows a robot to focus on low-level tasks (e.g. terrain traversing), while leaving the human operator in charge of high-level control and supervisory tasks such as specifying the direction of motion or a point to be reached [8]. Basically, in semi-autonomous control "the remote is given an instruction that it can safely do on its own [9]".
In particular, adjustable autonomy allows the operator to choose the level of autonomy of the robot on the fly and according to the task, thus letting the operator continuously and transparently adjust it in order to meet the imposed performance expectations. Additionally, this approach can significantly reduce the amount of bandwidth needed [10].
In humanoid robotics, examples of semi-autonomous controllers have been presented recently to reduce the high-DOF humanoid control problem to the command space of a three axis joystick exploiting the improved low-level capabilities of robots. This approach is followed in [11], where an intelligent joystick for biped gait control is proposed, while a single joystick teleoperation system for whole body control is described in [12]. Both works refer to the HRP-2 robot.
A significant step towards the reduction of control complexity is found in [13]. In this work, symbolic command interfaces through mouse or keyboard are exploited - among other applications - to let the operator indicate a target object to be grasped on a graphic interface, whereupon the approach-grasp operation is autonomously carried out by the robot.
In a recent survey [3], Goodrich offered an overall view of Teleoperation for Humanoid Robotics, stressing particularly on the challenges that make teleoperation still essential for humanoids:

- object recognition,
- interpretaion and understanding of scenes and semantic spatial reasoning,
- prediction and planning.

[1] http://spectrum.ieee.org/automaton/robotics/industrial-robots/fukushima-robot-operator-diaries

In a recent survey on Urban Search And Rescue (USAR) robotics [8], various examples of semi-autonomous controllers are presented as trade-off solutions to combine the recent improvements on low-level robot autonomy and the necessity to include a human-in-the-loop in the control systems, showing better performances in terms of area coverage, number of victims identified, number of collision and decreasing the workload of the operator.
As mentioned in [14], in a well performing teleoperation system, the Human-Robot Interaction (HRI) must be as efficient and capable as possible, and "maximize the information transfer while minimizing cognitive and sensorimotor workload of the operator". It should be noted that "the importance of the operator interface does not diminish as the level of autonomy increases". In a supervisory control context, as the robot becomes more autonomous, the interface must "provide a mechanism for the operator and the robot to exchange information at different levels of details or abstraction", moving toward a supervising interface.
In [15], Yanco defined the design guidelines for an effective HRI interface, based on a multi-year study during USAR Competitions:

- Use of a single monitor for the interface with large video windows,
- Include a third-person view of the robot,
- Use a map of the reconstructed environment using sensor fusion,
- Use graphical representation of information rather than textual or numerical,
- Reduce cognitive fatigue, context switching and negative effect of neglect and at the same time improve situational awareness,
- Design of the interface based on the intended user rather than the developer.

2.1 Motion Description Languages

The problem of connecting the implementation of autonomy in tele-operated robotics and behavior-based, hybrid control methods emerged in [16]. One solution to this problem was provided by Roger Brockett, who first introduced the MDL approach in [17]. MDL is a formal language used to abstract different types of control into a simple set of basic control laws (*atoms*) that can be chained together in order to obtain *behaviors*. MDLe (extended MDL) [18], enriches the approach introducing the concept of a reactive behavior using triggers.
In humanoid robotics the attempt of simplifying the control problem and composing parallel whole-body primitives has been tackled with the operational-space or task-function approach, introduced in [19]. In this framework, the whole-body motion control problem of a humanoid robot is simplified considering the cartesian space instead of the state space of the robot. This solution eases the design of control laws, making it more intuitive, with the additional possibility to use the sensor space, thus closing the control loop in a more robust and accurate way.
A task in the task-function approach can be seen as an atom in the MDL, consisting in the lower level motion primitive of the robot behavior; tasks can be combined sequentially or simultaneously, thus defining behaviors in the MDL framework. Examples of this approach are the whole-body control introduced by Sentis [20] and the stack of tasks presented by Mansard in [21].

2.2 Semi-autonomy via MDLs

By means of a library of *parameterized* motion primitives (atoms) and behaviors based on the MDL framework, we implemented an adjustable semi-autonomous control with three different levels of autonomy, according to the classification proposed in [3]:

- *Traded Control*,
- *Shared Control*,
- *Direct Control*.

Every control level acts in a particular level of the MDL hierarchy. This gives to these levels a new meaning oriented to MDL: the operator can issue in the highest level of the control MDL plans, while in the middle level, atomic actions or behaviors. Finally, in the lowest level, the operator controls the robot at the joint level. Figure 1 depicts how the different levels of operation work. In *Direct* control, the operator regulates the joint displacement, in *Shared* control he can issue atoms or behaviors (depicted as segments), in *Traded* control he selects a point in the environment requesting a plan (on a wheel valve or a point in the ground).

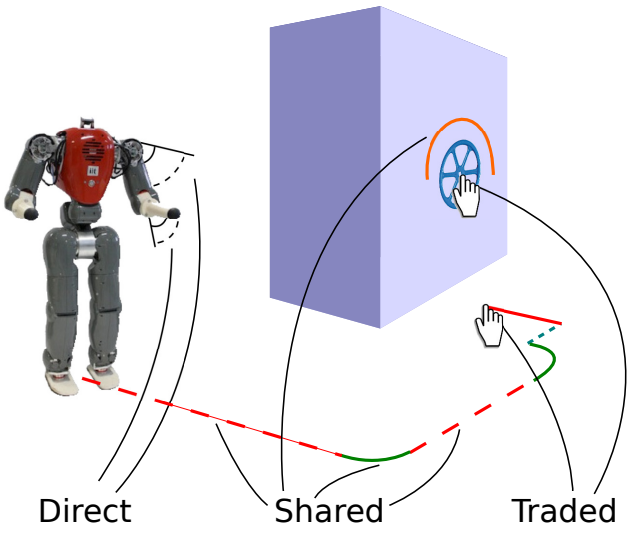

Figure 1: Graphical representation of control levels

3 Software Architecture

We developed a pilot control interface consistent with the three different levels of possible controls. The high-

level (*Traded*) control deals with the computation and execution of plans composed by primitives. *Shared* control is constituted by a set of 3D Interactive Markers that represents body parts of the robot or objects of interest to be positioned in the cartesian space. The operator can thus issue associated primitives or standalone primitives. Finally, we used *RobotMotorGui* (by YARP) to access each joint in *Direct* control.

As the control level gets lower, the robot loses autonomy but acquires more safety. It is reasonable that the operator could seamlessly switch between the levels of autonomy, depending on the task and on the environment condition.

The Pilot Interface is designed to provide both visual feedback to the user for validation purposes (or confirmation if needed) in a high level of autonomy, e.g. the planner shows the planned path before execution, and as a display control apparatus in the middle level, e.g. the operator can adjust the position of a 3d model of an object superimposing it onto a point cloud.

Our Pilot Interface is used to control IIT's *COmpliant huMANoid* (COMAN) [22], which is a torque controlled robot with 31-DOFs equipped with two Pisa/IIT - SoftHands [2] . The robot will have a Carnegie Robotics MultiSense S7 sensor [3] mounted as a head, but we are currently using a RGB-D camera (Asus Xtion Pro Live) mounted above the torso.

Modularity in complex systems is needed for robustness and reconfiguration. In our overall architecture many control modules have been developed using *YARP* as a middleware. These modules perform manipulation, locomotion, planning, perception and whole-body loco-manipulation tasks.

Each module is a standalone process that runs on the robot and interacts through messages with the Pilot Interface, from which it can receive a start/stop message and custom commands.

Due to the large number of different tasks that a *USAR* robot might perform, a modular and reconfigurable Pilot Interface is needed. Since each robot control module is an independent process, we want its respective operator widget to be an independent UI as well. With our approach we can run each individual control module widget as a standalone GUI, so that it can be tested or used without starting the whole Pilot Interface.

Our architecture (figure 2) easily allows the addition and removal of single graphic components into the main GUI window, and each component can be enabled or disabled from the Pilot Interface main control bar.

[2] http://softhands.eu
[3] http://carnegierobotics.com/multisense-s7/

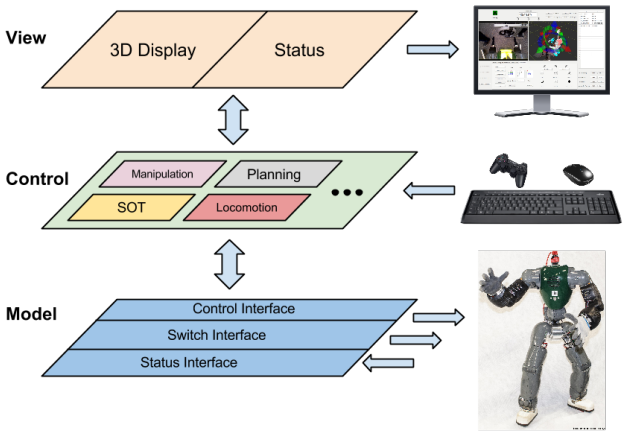

Figure 2: Architecture

4 Implementation

Similarly to most of our software, the Pilot Interface uses *YARP* as a communication facility. Nevertheless, while most of the control modules rely on *YARP* mainly to command the robot boards (or the simulated ones) through the low level library *Robolli*, the Pilot Interface uses *YARP* to send commands to the modules and receive feedback of their statuses.

As an implementation priority, we enforce into the modules some behaviours and structures specifically designed to ensure consistency, robustness and continuous monitoring of the software execution. Therefore, the communication between our Pilot Interface and the modules is wrapped into a number of standard interfaces shared by all the software components. Namely, we use a *switch*, a *status* and a *command* interface.

The *switch* interface makes it possible to *start, pause, resume* and *stop* modules. A practical advantage offered by a switch interface is the capability to recover the robot control in unforeseen situations. For instance, if the robot gets stuck in some position while performing a manipulation task, the operator can *stop* the manipulation module (such that actuators do not receive instructions), perform a reset procedure through a dedicated module and then *restart* the manipulation module.

The *status* interface is used to receive the module statuses at a constant rate. This offers the ability to visually check through the pilot interface that each module is running and possibly stream additional details of the operation being performed (e.g. percent progress of a computation, time left to complete a task).

The *command* interface is used to send custom commands to the modules.

In addition, the popular *Robotic Operative System* (ROS) framework is heavily relied on both as a bridge

with perception hardware (e.g. the 3D Camera sensor) and to visualize the state of the robot and a 3D reconstruction of the environment in the Pilot Interface, by means of the RViz displays and interaction utilities.

In figure 3 some uses of ROS and YARP in the communication layer are shown.

Figure 3: The usage of YARP and ROS in the communication layer

5 GUI Components

Our Pilot Interface is being used to tele-operate the robot COMAN to execute a number of practical tasks that can be encountered in disaster scenarios, some of which are described in the sequel.

One of the most basic tasks of a humanoid is *walking* on a flat ground. The currently implemented flat walking skills of our robot encompass four parametric motion primitives: *forward, backward and side walk*, specified with a distance parameter and *turn in place*, which comes with an angle amplitude parameter (figure 4).

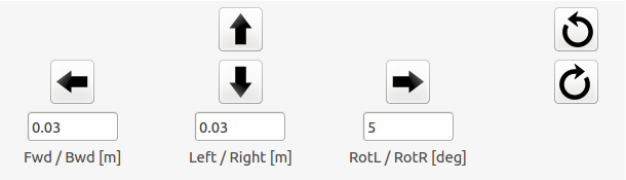

Figure 4: Walking widget

Coherently with the need for multiple control levels, walking can be performed through the pilot interface with two levels of autonomy. In the *Shared* control mode, the pilot can manually choose among walking primitives and navigate the robot through the environment by sending elementary commands one by one. In the *Traded* control mode, shown in the figure 5, the pilot selects a destination pose (green arrow) on a sensed 2D map of the environment, thus delegating the generation and online correction of an obstacle-free control sequence of walking primitives to a *flat walk planner*.

We developed a dedicated widget for a valve turning behaviour (section 6).

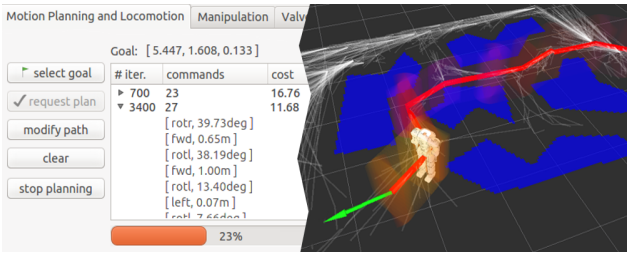

Figure 5: Planning widget and display

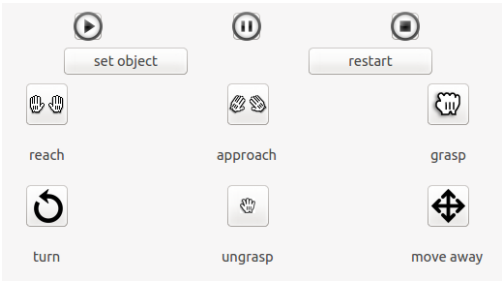

Figure 6: Valve Turning control widget

Every single pre-computed action can be triggered by clicking the related button (see figure 6).

The main rViz feature we use is 3D object visualization and interaction. We use a 3D display to represent the robot status as shown in figure 7.

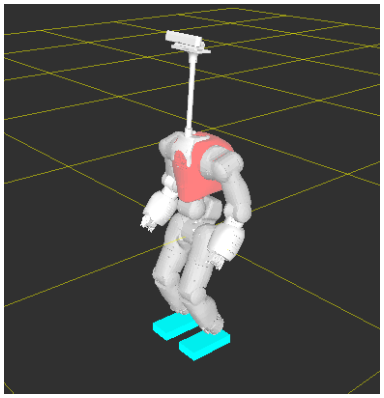

Figure 7: Robot Status

rViz 3D Interactive Markers are also widely used. The widget depicted in figure 9 collects various tools to manipulate 3D objects (e.g. a valve, in figure 8) or elements of the robot (e.g. hands, feet, COM, head, ...).

A significant issue we encountered on our robot *COMAN* is that when an overcurrent happens, the board related to the actuator that caused it will stop the control on the motor.

To reset the boards singularly and remotely we developed the widget shown in figure 10 that gives us the various information of the boards periodically or by request. The 'clear errors' button resets the state of the boards so that the standard work-flow can continue.

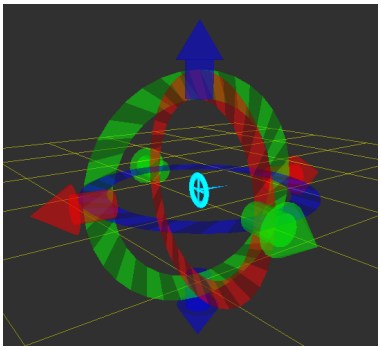

Figure 8: Valve 3D object

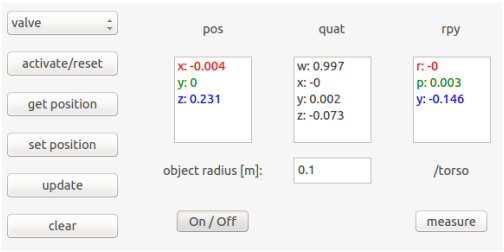

Figure 9: Object manipulation widget

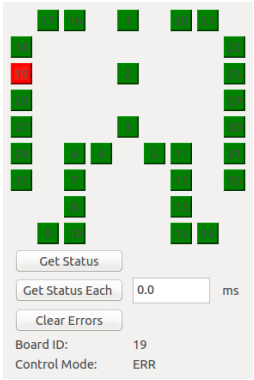

Figure 10: Control Boards Status widget

As mentioned in section 4, we can use the *switch* interface for single control modules through the Pilot Interface. In figure 11, we show the widget used to send the *start* and the *stop* commands to the various control modules.

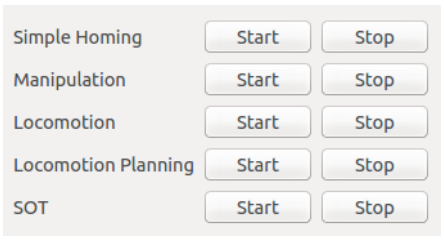

Figure 11: Module Manager widget

We decided to separate the *start/stop* commands from the *pause/resume* ones because the first ones are more 'dangerous', since they can actually stop the thread related to the control module.

6 Applications

The first practical task addressed has been the capability to *turn a valve*. In this task the robot is required to reach, grasp and turn a few valves of various sizes and orientations.

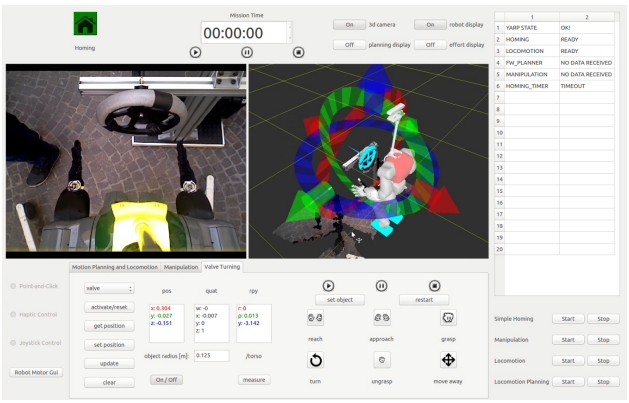

Figure 12: The Valve Turning

The current approach for this task is to first walk the robot towards a location close to the valve. Once a suitable proximity location is reached, the operator performs an estimation of the valve radius and pose by visually superimposing a 3D model of the valve on the point cloud. This operation is performed by moving and scaling a rViz 3D *interactive marker*. Eventually, a first guess of this estimation will be provided by a fully autonomous vision algorithm and confirmed or fine-tuned by the pilot. Once validated, the valve data is sent the manipulation module, which then computes suitable end-effector and joint trajectories. Finally, the operator can execute the computed task related commands, namely: *reach*, *approach*, *grasp*, *turn*, *ungrasp*, *move away*.

In the snapshots in figure 13 we display all the different steps of a *valve-turning* task.

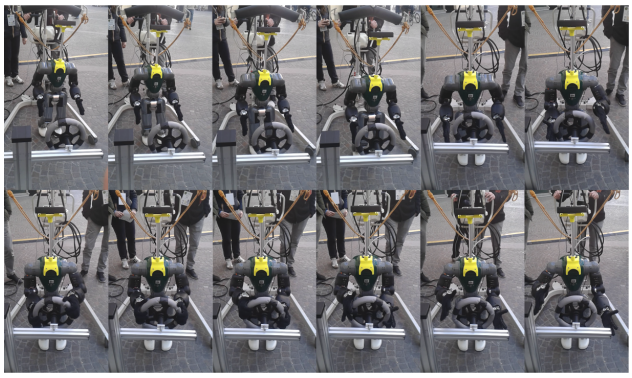

Figure 13: The Valve Task

Our team developed a *Stack of Tasks* (SOT) application to perform cartesian trajectories minimizing some loss function (to not work near joints limits for example).

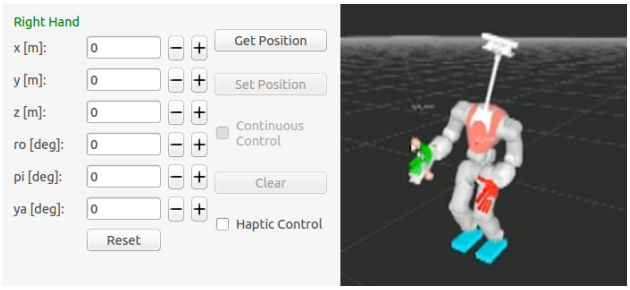

Figure 14: The Stask Of Tasks

As shown in figure 14 we use 3D rViz Interactive Markers to move a hand as said in section 5. The pilot can choose wether the command should be sent continuously or upon request. Furthermore, we can use the SOT to control the COM and a foot in a static equilibrium configuration.

7 Conclusions

In this paper, the design of a Pilot Interface for humanoid robots in USAR scenarios has been presented. We described the most relevant capabilities of our current implementation and showed its effectiveness during a demonstration based on the DRC valve-turning task. This task was fulfilled by sole means of a semi-autonomous *Shared* control level.

As a part of our future work, we plan to improve the capabilities of our robot by developing increasingly complex behaviors to address tasks that can be encountered in disaster scenarios. Our aim is to simplify the interface while reducing the pilot's cognitive fatigue and gradually move towards a more autonomous framework.

In addition, while the Pilot Interface can be seamlessly used to control the real and a simulated robot thanks to the *Gazebo-Yarp Plugins* [23] developed by our team, it appears desirable to embed simulation capabilities into the pilot interface, thus allowing a safe validation of planned actions prior to the execution on the real platform.

Acknowledgments

Research leading to these results has received funding from the European Union Seventh Framework Programme [FP7-ICT-2013-10] under grant agreements n.611832 WALKMAN.

References

[1] S. Kohlbrecher, J. Meyer, T. Graber, K. Petersen, O. von Stryk, and U. Klingauf, "Robocuprescue 2014 - robot league team hector darmstadt (germany)," tech. rep., Technische Universität Darmstadt, 2014.

[2] K. Melo, J. Leon, A. di Zeo, V. Rueda, D. Roa, M. Parraga, D. Gonzalez, and L. Paez, "The modular snake robot open project: Turning animal functions into engineering tools," in *Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on*, pp. 1–6, Oct 2013.

[3] M. A. Goodrich, J. W. Crandall, and E. Barakova, "Teleoperation and beyond for assistive humanoid robots," *Reviews of Human Factors and Ergonomics*, vol. 9, no. 1, pp. 175–226, 2013.

[4] H. Hasunuma, M. Kobayashi, H. Moriyama, T. Itoko, Y. Yanagihara, T. Ueno, K. Ohya, and K. Yokoi, "A tele-operated humanoid robot drives a lift truck," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 3, pp. 2246–2252, IEEE, 2002.

[5] K. Hambuchen, W. Bluethmann, M. Goza, R. Ambrose, K. Rabe, and M. Allan, "Supervising remote humanoids across intermediate time delay," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pp. 246–251, IEEE, 2006.

[6] C. Stanton, A. Bogdanovych, and E. Ratanasena, "Teleoperation of a humanoid robot using full-body motion capture, example movements, and machine learning," in *Proc. Australasian Conference on Robotics and Automation*, 2012.

[7] R. O'Flaherty, P. Vieira, M. Grey, P. Oh, A. Bobick, M. Egerstedt, and M. Stilman, "Humanoid robot teleoperation for tasks with power tools," in *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, pp. 1–6, April 2013.

[8] Y. Liu and G. Nejat, "Robotic urban search and rescue: A survey from the control perspective," *Journal of Intelligent & Robotic Systems*, vol. 72, no. 2, pp. 147–165, 2013.

[9] R. Murphy, *An introduction to AI robotics*. The MIT press, 2000.

[10] J. M. Bradshaw, P. J. Feltovich, H. Jung, S. Kulkarni, W. Taysom, and A. Uszok, "Dimensions of adjustable autonomy and mixed-initiative interaction," in *Agents and Computational Autonomy*, pp. 17–39, Springer, 2004.

[11] J. Chestnutt, P. Michel, K. Nishiwaki, J. Kuffner, and S. Kagami, "An intelligent joystick for biped control," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 860 – 865, May 2006.

[12] M. Stilman, K. Nishiwaki, and S. Kagami, "Humanoid teleoperation for whole body manipulation," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 3175–3180, IEEE, 2008.

[13] N. Sian, K. Yoki, Y. Kawai, and K. Muruyama, "Operating humanoid robots in human environments," in *Proceedings of the Robotics, Science & Systems Workshop on Manipulation for Human*

Environments, Philedelphia, Pennsylvania, Citeseer, 2006.

[14] T. Fong and C. Thorpe, "Vehicle teleoperation interfaces," *Autonomous robots*, vol. 11, no. 1, pp. 9–18, 2001.

[15] H. A. Yanco and J. L. Drury, "Rescuing interfaces: A multi-year study of human-robot interaction at the aaai robot rescue competition," *Autonomous Robots*, vol. 22, no. 4, pp. 333–352, 2007.

[16] M. A. Goodrich and A. C. Schultz, "Human-robot interaction: a survey," *Foundations and Trends in Human-Computer Interaction*, vol. 1, no. 3, pp. 203–275, 2007.

[17] R. Brockett, "Formal languages for motion description and map making," *Robotics*, vol. 41, pp. 181–191, 1990.

[18] V. Manikonda, P. S. Krishnaprasad, and J. Hendler, *Languages, behaviors, hybrid architectures, and motion control*. Springer, 1999.

[19] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *Robotics and Automation, IEEE Journal of*, vol. 3, no. 1, pp. 43–53, 1987.

[20] L. Sentis, *Synthesis and control of whole-body behaviors in humanoid systems*. PhD thesis, Citeseer, 2007.

[21] L. Saab, O. E. Ramos, F. Keith, N. Mansard, P. Soueres, and J.-Y. Fourquet, "Dynamic whole-body motion generation under rigid contacts and other unilateral constraints," 2013.

[22] N. G. Tsagarakis, G. M. Cerda, Z. Li, and D. G. Caldwell, "Compliant humanoid coman: Optimal joint stiffness tuning for modal frequency control.," in *ICRA*, pp. 665–670, 2013.

[23] E. M. Hoffmann, S. Traversaro, A. Rocchi, M. Ferrati, A. Settimi, L. Natale, A. Bicchi, F. Nori, and N. G. Tsagarakis, "A yarp based plugin for gazebo simulator," *MESAS*, 2014.