Humanoid teleoperation system for space environments

Paolo Pierro, *Student Member, IEEE*, Daniel Hernández, Miguel González-Fierro, Lorenzo Blasi, Andrea Milani and Carlos Balaguer, *Member, IEEE*

***Abstract*— Our goal is to study collaborative working environments in which human and robotic agents can work together in the achievement of different tasks. The introduction of robots in the field of space applications becomes really useful when performing tasks that are too dangerous, too difficult or even impossible for humans. In this paper we present a teleoperation system for interacting with a humanoid robot in a space environment. A "lunar scenario" was built in which the HOAP-3 humanoid robot is able to detect and manipulate objects, with the help of a human operator. A human machine interface (HMI) and a high level command protocol have been developed for the teleoperation of the robot. The HMI allows an operator to control the robot movements and visualize the environment from robot cameras.**

I. Introduction

With the growth of robotics there has been a great progress in space exploration. For a long time, mobile robots have been used to explore and research other planets; the twin rovers Spirit [1] and Opportunity [2] have been employed satisfactorily to perform a geological analysis on Mars surface, as it has been for many other vehicles like Mars 2 and 3 of the Soviet Union or the Surveyor rover of NASA [3].

Future space human missions will aim at colonizing and inhabiting other planets. Therefore, it will be necessary to find out solutions for the construction of permanent habitats or research labs and perform tasks, such as the initial communication systems setup, that expose humans to dangerous environments. In order to support such activities, we will require designing robots that are able both to collaborate and interact with humans sharing the same working environment and even to substitute them in dangerous situations. Humanoid robots are suitable for these duties because they are able to interact with the environment using the same tools designed for humans.

One example of a humanoid robot specifically created to perform tasks in the space is Robonaut [4], developed by NASA and DARPA, whose anthropomorphic appearance facilitates teleoperation. This robot has been created to assist astronauts during space walks and it is able to handle extra-vehicular activity (EVA) tools and geologic.

Manuscript received March 23, 2009. This work was supported in part by the European Commission under FP6-2005-IST5.

P. P., D. H., M. G. and C. B. Authors are with Universidad Carlos III de Madrid, Avda Universidad 30, 28911, Leganés, Madrid, Spain, (e-mail: {ppierro, mgpalaci, dhernandez, balaguer}@ing.uc3m.es).

L. B. and A. M. Authors are with Hewlett-Packard Italy Innovation Center, via Grandi 4, 20063 Cernusco sul Naviglio (MI), Italy (e-mail: {lorenzo.blasi, andrea.milani}@hp.com).

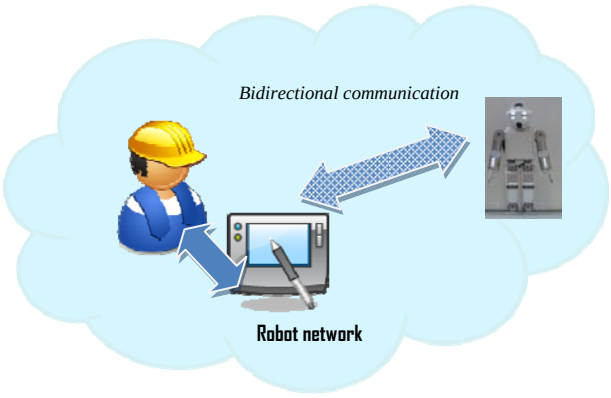

Fig. 1. Teleoperation system.

HRP-2 humanoid robot is another example of a robot that can cooperate with humans [5]. This robot is able to manipulate objects under the orders of a human [6] and also to assemble a panel cooperatively with a human [7]. ASIMO [8] can perform tasks like serving drinks, pushing a cart or taking tools from a table. Other robots with similar characteristics are Wabian-2 [9], Khr-2 [10] or Reem-B [11].

Humans and robots can collaborate in several ways, as presented in [12]. In [13] a personal digital assistant (PDA) interface has been designed to control a small doll-shape robot; in [14] and [15] a handheld interface is used to perform remote driving.

This paper deals with the collaboration between humanoid robots and humans in order to achieve tasks in space environments. In this research we use the robot HOAP-3 teleoperated by a human agent.

The small humanoid robot "HOAP-3" is about 60cm high and its weight is about 8 Kg, so that it becomes quite easy to control and move while maintaining the whole stability.

The robot is able to explore the surroundings and detect an object that is placed in the scenario. The robot can go towards the object and take it. A human machine interface (HMI) and a high level command protocol have been designed to help the operator in moving the robot. As a control device we use a lightweight and portable tablet pc. The system is presented in Fig. 1.

The content of this paper is divided in six sections. In Section II we describe the scenario built for the experiment. The control architecture used in the robot is exposed in Section III. The HMI developed to move the robot is presented in Section IV and the command protocol is deeply described in Section V. The experimental results are

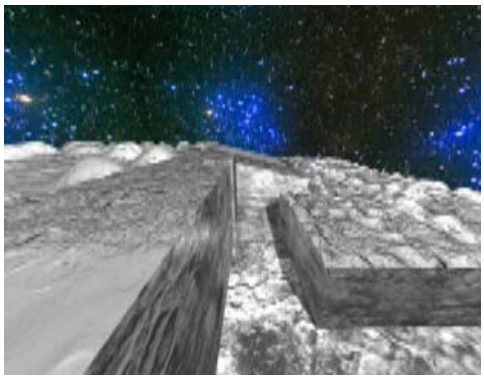

Fig. 2. Recreation of the lunar scenario

depicted in Section VI. The conclusions are in Section VII.

II. Description of the Scenario

A scenario has been designed to simulate a lunar ambient. It consists on a long corridor surrounded by cliffs where the robot can walk and interact with the environment. In Fig. 2 it can be seen a 3D recreation of the scenario. A similar model has been built at Carlos III University to recreate the moon surface.

The surface of the cliffs has been built with planes of polystyrene where it has been made holes to simulate craters. The floor of the scenario has been made of hard cardboard as the robot has to walk on it. To paint it, we have used a uniform grey to avoid interferences in the vision of the robot.

This scenario allows the human operator to interact with the robot. Through teleoperation and using the HMI, human gives the humanoid the order of walk until it sees an object that simulates a satellite dish. The robot detects this object and then the operator orders the robot to grip the object and move it.

III. Control of the HOAP-3 Robot

The Hoap-3 robot is provided with an internal PC operating in RT-Linux, which can communicate with other PCs via wireless network.

The control strategy is presented in Fig. 3. In such a scheme, several blocks have to be considered. Once that a command has been received, the robot distinguishes if it is a command for the walking generation or for the arms movement.

The walking patterns of the robot have been designed basing on the theory of the 3D Linear Inverted Pendulum Mode presented in [16].

Different trajectories were simulated in Matlab and Simulink® using kinematical and dynamical information of the robot. Different trajectories have been tried with the real platform: going forward, backward, turn left and right with a specified angle. Some simulation results are in Fig. 4.

The posture stability control has not been implemented yet, but several studies are being done in order to

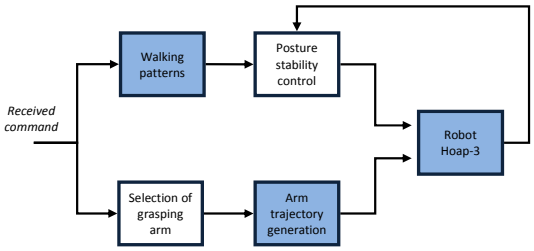

Fig. 3. Control strategy.

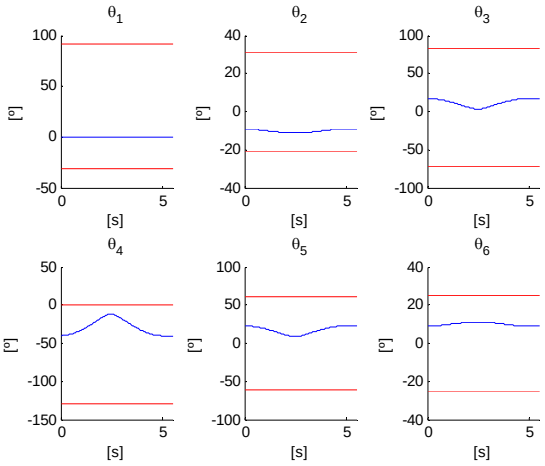

Fig. 4. Joints positions of Hoap-3 right leg in walking forward trajectory. The blue lines represent the joint trajectories. These positions are ensured to be within the red lines which represent joints mechanical limits.

accomplish it[12].

If the received command requires a movement of the arms, as in the case of a grasping task, the first problem to be considered is the selection of the suitable arm. This is the goal of the “Selection of the grasping arm” block. The arm to be moved can be selected as the one which can reach the object, but in several cases both arms can achieve this aim. In this case, our present research is focused in choosing the arm that can reach the object with higher manipulability.

Finally, the trajectory of the arm is evaluated online through the algorithm of kinematic inversion presented in [18], once that the command provides the distance and the orientation from the object. The orientation reference for the object is calculated with the support of the unit quaternion presented in [19].

IV. The Human-Machine Interface

An HMI was developed for the teleoperation of the robot HOAP-3 in the lunar scenario. Using the HMI a human agent can work collaboratively with the robot in the achievement of the proposed tasks.

The HMI allows an operator to see the environment from the robot cameras, as well as to control various movements of the robot and give orders for doing some tasks, e.g. “grab object”.

The HMI provides several functionalities to the human agent working with the robot:

- video feedback from robot cameras and visual cues of

object recognition;

- control movement of the robot head (pan and tilt);
- control walking and turning movements of the robot.
- command the robot to perform higher order task, such as go to specific location, grab object or drop object.
- communication feedback with a log of the commands between operator and robot.

The developed HMI is shown in Fig. 5.

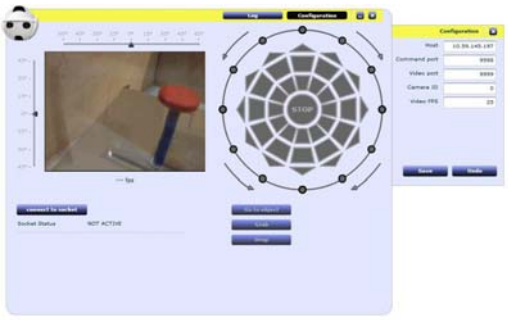

Fig. 5. The HMI for teleoperation of robot HOAP: On the left there is the video panel and the head movements controls. On the right side of the HMI there is the communication configuration panel. On the bottom side there is the 'connect' button and the communication log. On the center wheel of the HMI there are the walking and turning movements' controls and the higher order command buttons, 'Go to object', 'Grab' and 'Drop'.

We have improved the HMI from previous versions. In Fig. 6 we compare the HMI developed for this task with the previous interface. A more intuitive interaction between the user and the interface was one goal in the redesign of the HMI. A new panel layout improves the usability of the system. The new interface also improves feedback to the operator with a log of the received and performed commands. The look and feel has also been improved.

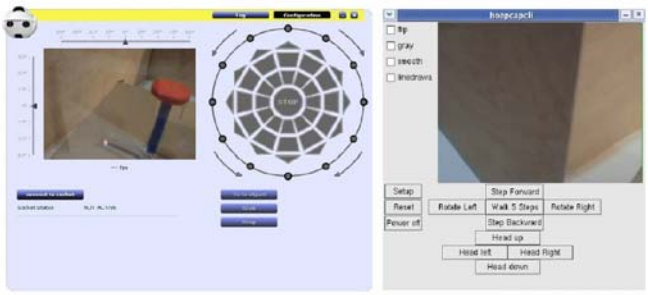

Fig. 6. Comparison between the HMI developed for the teleoperation system and previous version of HMI. The functions and buttons of the HMI have been redistributed to improve usability. Greater functionality has been added.

The developed HMI greatly improves the usability from the previous interface and provides the operator with added functionality. The goal of the HMI is to have a platform for communication with the robotic system in working environments that allows the operator to give the robot direct action commands like "grab an object", "go to a place" or "move number of steps". Several of these orders were implemented but further effort needs to be done for increasing the number of actions the robot can perform.

The HMI should also give to the operator more feedback from the robot environment and the state of the robot actions.

In the case of spatial applications, several problems have to be issued. Planet rotations may cause troubles in communications: in fact, due to the rotation, the object can get out of the line of sight.

The other main problem is the time delay. For a moon-earth communication the delay could be around 3 seconds and for mars-earth communication the delay could be up to 10 minutes, as the time delay increases with the distance from the earth [17]. In order to simulate the conditions and problems that arise in a real space communications application, a controllable time delay module has been added to the HMI. This module will permit to set a variable delay and to test the performance of the proposed teleoperated system in space environments.

V. The Command Protocol

In order to control the robot we designed a Robot Command Protocol (RCP). Design goals for the protocol are: simplicity, generality, flexibility and expressiveness. The protocol should be simple in that no unneeded features should be added; the protocol should be general and flexible enough to be used for several use cases without modifications. A powerful characteristic that leads to both flexibility and expressiveness can be identified as orthogonality, which can be achieved by clearly separating disconnected functionalities while at the same time allowing their combination without unneeded constraints. RCP is a text-based protocol which has its roots in Unix protocols like SMTP or FTP. Each RCP command is a text string terminated by a newline character. Using text commands has several advantages. First of all the resulting protocol is simple to understand and implement; this means that support for robot control can also be easily added to programs different from our HMI. Moreover, the protocol is lightweight; since the robot has limited computational resources that can be dedicated to command parsing, this was an important design goal. Finally, the human-readable text commands make debugging easy. Communication traces can be understood immediately and you can even do simple tests by typing commands directly in a telnet session. Our protocol is concerned with application-level communication only; we assume that a reliable channel (in our case a TCP/IP connection) is used for transmission. The protocol is also general in that it hasn't been designed for a specific target robot, but for a generic target robot described by a high-level robot model.

RCP was originally defined in [20] and can be decomposed into several sub-protocols. Each sub-protocol contains a set of commands used for a single purpose. The list of RCP sub-protocols is shown in Table I.

TABLE I
RCP SUB-PROTOCOLS

Name	Number of commands
Connection	2
Control negotiation	2
Basic movement	3
Direct command execution	1
Configuration	2
Sensor reading	1
Positioning	2
Notification	tbd
Goal-setting	1
Object grabbing	2
Strategy selection	2

One can start and end a communication with the robot using the *connection sub-protocol*. Once connected, you can use any of the other sub-protocols, for example the *sensor reading sub-protocol*, which allows operator access to the output of robot sensors.

In order to execute some commands the robot needs to know its current position in a cartesian coordinate system. However without a GPS receiver, even if the robot knows its initial position and updates it according to its movements, the resulting position is just an estimate which gets more and more inaccurate as the robot moves. The *positioning sub-protocol* defines how the robot and the operator can dialogue about the robot's position and collaborate to improve its accuracy: the operator and the robot can both ask for the current robot's position or give a value for it.

The subprotocols summarized so far could be used by multiple users connected to the robot at the same time, since they are mostly composed of query commands, but other sub-protocols which really control robot movements need exclusive access to it. Thus it is necessary for the operator to acquire a sort of "exclusive lock" using the *control negotiation sub-protocol* before issuing action commands. This ensures only one user at a time can control the robot. Once a user has acquired exclusive control over the robot, she can use the *basic movement sub-protocol* to translate or rotate it.

The generic robot model includes a command queue, where commands are inserted before being executed. So, even if commands are sent by the user before the current movement has been completed, they are put into the queue and executed sequentially. However, while the robot is moving the user may notice she has made a mistake (e.g. the robot is going too far away) and want to stop the robot immediately, without waiting for all commands in the queue to complete. This is a situation in which the *direct command execution sub-protocol* is useful, because it allows sending a command which is executed immediately, bypassing the queue. For example our user will issue the command

```
DIRECT STOP
```

The robot executes the STOP command immediately. It terminates the current movement by reaching the nearest stable position, clears out the queue and sends a reply to the user.

The DIRECT command described above is a good example of orthogonality because it can be combined with any other command, used as a parameter, to make it bypass the queue. Also the STOP command is orthogonal because it can be used to terminate any command, not only movements.

The reply sent by the robot after executing the STOP command is:

```
OK COMMAND <cmd_id> INTERRUPTEDBY <cmd_id>
```

Note from the above line that IDs are used to refer to commands. Every command is assigned an ID by its receiver (i.e. the robot or the HMI). Then the receiver sends the counterpart a reply indicating whether the command has been accepted or not. Successful replies always start with "OK", while unsuccessful ones start with "KO".

The basic movement sub-protocol defines a general MOVE command with the following structure:

```
MOVE <movement_type> <direction> <count> <unit>
```

Currently three flavors of the command are supported:

```
MOVE WALKING [FORWARD|BACKWARD] <count> STEPS
MOVE TURNING [LEFT|RIGHT] <count> DEGREES
MOVE HEAD [UP|DOWN|LEFT|RIGHT] <count> DEGREES
```

The MOVE command is a good example of the flexibility of our protocol, in that its structure allows adding new movement types easily. For example we could add a new BOWING movement.

As movements are not an instant action, the robot sends multiple replies in response to a MOVE command:

```
OK COMMAND <command_id> QUEUED
OK COMMAND <command_id> STARTED
OK COMMAND <command_id> COMPLETED
```

The first reply is sent as soon as the command is accepted; the second one when the command is considered for execution and the third one after the movement has been completed.

Robot movements depend on parameters such as speed and step length. These (and other) parameters can be read or set via the *configuration sub-protocol*.

A more advanced way of controlling robot's movements is the one defined in the *goal-setting sub-protocol*. When the user knows the exact position where the robot must go, she can define it as a target position and send it to the robot. In the current implementation the route to the target position is autonomously determined by the robot navigation module.

The protocol allows defining the robot's target position in two ways; one is to define the position with a pair of coordinates, while the other is to indicate an object as a target.

The relevant command is GOTO, which is shown below in its two variants:

```
GOTO OBJECT(<object_id>)
GOTO <x> <y>
```

Note that if the target position is occupied by an object the robot cannot stop exactly there; in this case the robot stops within a certain range (defined by a configuration parameter) from the target.

The same target-object specification of the goal-setting sub-protocol is used also in the *object grabbing sub-protocol*. In order to tell the robot to grab or drop an object the user issues the commands:

```
GRAB OBJECT(<object_id>)
DROP OBJECT(<object_id>)
```

Typically these kind of high-level operations are not unambiguously defined by the target object only, but involve some decision about which strategy should be used for executing the operation. For this purpose a specific *strategy selection sub-protocol* has been defined with which both the robot and the user collaborate in deciding which strategy to use for the operation at hand.

The strategy selection dialogue is initiated from the robot side with a request listing the possible strategies:

```
SELECT STRATEGY FOR <cmd_id> [<strategy_1>,
<strategy_2> ... <strategy_n>]
```

Then the user chooses a strategy and communicates her decision with the command:

```
USE STRATEGY FOR <command_id> <strategy>
```

After the strategy has been selected the robot can actually grab or drop the object.

Finally, note that the transmission of the video stream from the robot camera is not defined as part of the command protocol, as it happens on a separate channel using an ad-hoc streaming protocol.

Within the second year of the Robot@CWE project we implemented a subset of the full protocol described above, namely the basic movement, goal-setting and object grabbing sub-protocols. Also, in the current implementation there is no command queue: if a new command arrives before the previous one has completed, it is rejected.

A reference of all the commands currently defined in the RCP protocol is shown in Table II.

TABLE II
RCP COMMANDS

Sub-protocol	Command
Connection	CONNECT <profile>
	DISCONNECT
Control negotiation	CONTROL BEGIN
	CONTROL END
Basic movement	MOVE <movement_type><direction><count><unit>
	STOP
	COMBINE <command>
Direct command execution	DIRECT <command>
Configuration	QUERY PARAM <parameter_name>
	SET <parameter_name> <parameter_value>
Sensor reading	QUERY SENSOR [<label>,... , <label>]
Positioning	QUERY POSITION
	POSITION <x > <y > <confidence>
Notification	tbd
Goal-setting	GOTO OBJECT(<object_id>)
	GOTO <x> <y>
Object grabbing	GRAB OBJECT(<object_id>)
	DROP OBJECT(<object_id>)
Strategy selection	SELECT STRATEGY FOR <cmd_id> [<strategy_1>, ..., <strategy_n>]
	USE STRATEGY FOR <cmd_id > <strategy>

VI. EXPERIMENTAL RESULTS

At the University Carlos III de Madrid a lunar scenario was built to simulate the operation of a robotic agent working in collaboration with a human in a space environment. The task to be performed consists of teleoperating the HOAP-3 robot, first walking through an enclosed hall and finding an object, in this case an ‘antenna’, then grasping the object and placing it in a different location. For the teleoperation task, the communication between the robot and the HMI is performed over a standard wi-fi 802.11 network.

In order to evaluate the teleoperated system proposed in this paper, several tests were conducted with the HOAP-3 robot. The robot walks in an enclosed corridor while being teleoperated by a human agent. Trough the HMI the operator sends walking and turning movement commands. Video feedback from the robot cameras indicates to the operator that the robot has located the ‘antenna’.

Fig. 7. a) the recreation of lunar scenario. b) HOAP-3 teleoperated through a corridor looking for the ‘antenna’.

Then, the robot approaches the object to a close enough distance so that it can grab it when requested by the human operator.

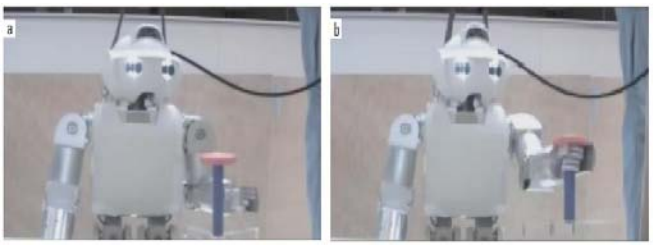

Fig. 8. Robot HOAP grasping the ‘antenna’ a) the robot computes grasping trajectories and receives a command by the operator. b) robot successfully picks up the ‘antenna’.

Then the robot computes the best trajectory for the grasping movement and performs accordingly to the operator decisions.

Fig. 8 and Fig. 9 show the experimental setup for the

demonstration conducted with the proposed teleoperated system. A human agent works collaboratively with a humanoid robot by supervising, controlling and helping in the decision taken by the robot.

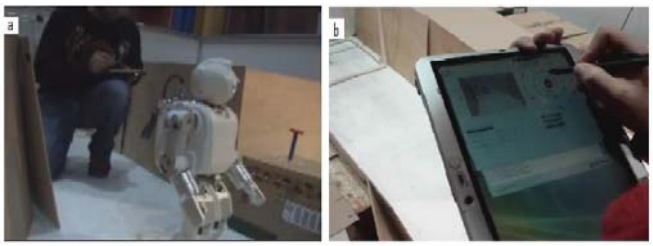

Fig. 9. Demonstration of the proposed teleoperated system on a 'lunar scenario' a) Robot HOAP-3 and a human operator work collaboratively on finding and moving the 'antenna'. b) The operator teleoperates the robot with the HMI using a pocket PC.

VII. CONCLUSIONS

A teleoperation system for control of a humanoid robot has been presented in this paper. A collaborative working environment was demonstrated; using a lunar scenario a humanoid robot and a human operator work together in achieving a task.

Walking patterns for a humanoid robot have been presented with different trajectories for forward, backward, turn left and turn right movements, all tested on a HOAP-3 robot.

We have presented a HMI to help a human agent work collaboratively with the robot. The HMI allows the operator to give the robot direct actions commands like "grab and object", "go to a place", etc. The HMI also gives the operator feedback from the robot environment and the state of the robot actions.

An RCP for the communication with the robot is presented in this paper. The main goals of the protocol are simplicity, generality, flexibility and expressiveness. The RCP is a text-based protocol, is simple to understand and debug. It is lightweight and general, meaning that it has not been designed for a specific target robot, but for a generic target robot described by a high-level robot model.

The system was tested on two different tasks. First the robot walks in an enclosed corridor while being teleoperated by a human agent using the developed HMI. For the second task the robot recognizes an object which it grasps when given the command by the operator.

Future works in space collaborative working environments would include working in new tasks with the robot like the construction of a space shelter. Further work on the RCP and the HMI is also necessary to add more features and functionalities to future applications.

REFERENCES

[1] P.C. Leger, A. Trebi-Ollennu, J.R. Wright, S.A. Maxwell, R.G. Bonitz, J.J. Biesiadecki, F.R. Hartman, B.K. Cooper, E.T. Baumgartner, M.W. Maimone, M.W., "Mars Exploration Rover surface operations: driving spirit at Gusev Crater," in *IEEE International Conference on Systems, Man and Cybernetics*, Volume 2, pp. 1815-1822, Oct. 2005.

[2] J.J. Biesiadecki, E.T. Baumgartner, R.G. Bonitz, B. Cooper, F.R. Hartman, P.C. Leger, M.W. Maimone, S.A. Maxwell, A. Trebi-Ollennu, E.W. Tunstel, and J.R. Wright, "Mars exploration rover surface operations: driving opportunity at Meridiani Planum," in *IEEE Robotics & Automation Magazine*, Volume 13, Issue 2, pp. 63–71. June 2006.

[3] R.G. Bonitz, T.T. Nguyen and W.S. Kim, "The Mars Surveyor '01 Rover and Robotic Arm," in *IEEE Aerospace Conference Proceedings* 2000, Volume 7, pp. 235-246, March 2000.

[4] R.O. Ambrose, H. Aldridge, R.S. Askew, R.S.; R.R. Burridge, W. Bluethmann, M. Diftler, C. Lovchik, D. Magruder, F. Rehnmark, "Robonaut: NASA's space humanoid," in *IEEE Intelligent Systems and Their Applications*, Volume 15, Issue 4, pp. 57–63, 2000..

[5] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, T. Isozumi, "Humanoid robot HRP-2," in *IEEE International Conference on Robotics and Automation 2004*, Volume 2, pp. 1083–1090. Apr 26-May 1, 2004.

[6] Ee Sian Neo, T. Sakaguchi, K. Yokoi, "A humanoid robot that listens, speaks, sees and manipulates in human environments," in *IEEE International Conference on Multisensor, Fusion and Integration for Intelligent Systems*, pp. 419–425, 20-22, Aug. 2008.

[7] K.Harada, S.Kajita, F.Kanehiro, K.Fujiwara, K.Kaneko, K.Yokoi,and H.Hirukwa, "Real-Time Planning of Humanoid Robot's Gait for Force Controlled Manipulation," in *IEEE International Conference on Robotics and Automation*, 2004.

[8] Y. Sakagami, R. Watanabe, C. Aoyama, C.; S. Matsunaga, N. Higaki, K. Fujimura, "The intelligent ASIMO: system overview and integration," in *IEEE/RSJ International Conference on Intelligent Robots and System*, Volume 3, pp. 2478-2483, 2002.

[9] Y. Ogura, H. Aikawa, K. Shimomura, A. Morishima, Hun-ok Lim and A. Takanishi, "Development of a new humanoid robot WABIAN-2," in *International Conference on Robotics and Automation 2006*, pp 76–81. 2006.

[10] S.-W. P. Ill-Woo Park, Jung-Yup Kim and I.-H. Oh, "Development of humanoid robot platform KHR-2 (Kaist Humanoid Robot-2)," in *International Journal of Humanoid Robotics*, vol. 2, no. 4. pp. 519-536, 2005.

[11] R. Tellez, F. Ferro, S. Garcia, E. Gomez, E. Jorge, D. Mora, D. Pinyol, J. Oliver, O. Torres, J. Velazquez and D. Faconti, "Reem-B: An autonomous lightweight human-size humanoid robot," in *IEEE-RAS International Conference on Humanoids 2008*, pp. 462–468, 2008.

[12] P. Pierro, C. A. Monje and C. Balaguer, "Modelling and Control of the Humanoid Robot RH-1 for Collaborative Tasks," *in IEEE RAS/RSJ Conference on Humanoids Robots*, Daejeon, Korea, 2008, pp. 125–131.

[13] S. Calinon and A. Billard. "PDA interface for humanoid robots," in *IEEE International Conference on Humanoid Robots (Humanoids)*, October 2003.

[14] T. Fong, C. Thorpe, and C. Baur. "Advanced interfaces for vehicle teleoperation: collaborative control, sensor fusion displays, and remote driving tools". in *Autonomous Robots*, 11(1), pp. 77-85, 2001.

[15] T. Fong, C. Thorpe, and B. Glass. Pdadriver: "A handheld system for remote driving," in *IEEE International Conference on Advanced Robotics*, 2003.

[16] S. Kajita, F. Kanehiro, K. Kaneko, et.al, "The 3D Linear Inverted Pendulum Mode: A simple modeling for a biped walking pattern generation, in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.239-246, 2001.

[17] C. Balaguer, R. Aracil, M. Ferre, M. Buss and C. Melchiorri, *Advances in Telerobotics*, Springer Tracts in Advanced Robotics (STAR), 2007.

[18] Siciliano, L. Sciavicco, L. Villani, G. Oriolo, *Robotics: Modelling, Planning and Control*, London, Great Britain, Springer-Verlag, 2009.

[19] S. Chiaverini and B. Siciliano, "The unit quaternion: A useful tool for inverse kinematics of robot manipulators", *Systems Analysis Modelling Simulation*, vol. 35, pp. 45–60, 1999.

[20] L. Blasi, O. Stasse, *Robot@CWE Deliverable D3.3@M22*: Design support software for ROBOT@CWE, Annex II.