Evaluation of recent game interfaces for the command of a humanoid robot

Duarte Aragão

Abstract—**This dissertation explores novel and typical interaction devices as an interface for a humanoid robot control. The novel devices used were the Wii gaming console remote control and the Kinect sensor remote control for the Xbox 360 gaming console, the typical devices used were a computer graphical interface and a remote control directional pad.**

The complexity of humanoid robots such as the iCub, and the need for people to interact with them, either simply for the control of its pose, or for imitation learning, lead to the interest in new forms of interactions with the robot. The new forms of control are expected to be as precise as the previous ones, but to allow a more natural and intuitive interaction. Recent interaction devices released by the gaming industry are a good way to achieve this goals.

To understand how these interfaces compare an evaluation was done, consisting of tests with objects for the iCub to reach while being controlled using one of the proposed interaction system. Differences between users, comments, their involvement, and ease of use were compared and associated with users profile and interface characteristics.

The tests results suggest that different users adapt differently to the same interface, having this in mind, it was successfully shown that the new interaction devices can be adapted to obtain better performance while controlling an humanoid robot.

Keywords: interaction, humanoid, comparison, Kinect and Wii

I. Introduction

Robots have been increasing in complexity as they have increased on ability. Particularly in humanoids this complexity is evident. Many times this complexity is solved providing autonomy to the robot. Although the autonomy level achieved today is not enough for some of the tasks that robots are trying to solve. In another front, games have also become very complex, and in a way some games have similar needs to the needs in Human-Robot Interaction (HRI).

Recently gaming companies have started to release new and appealing interfaces to simplify and make more natural the Human-Computer Interaction (HCI). These interfaces are intended to be used by non core gamers so they are typically simple and robust. Also because these devices are sold massively, the price tends to be very low for a high tech device. The easy access to these devices enabled the creation of Internet gathering points where hobbyists, professionals and researchers, discuss and expose their work, which many times features novel and unexpected interaction methods. HRI has also been taking advantage from this movement, using it as inspiration for the resolution of interaction problems between complex robots and humans.

The iCub is an humanoid robot that replicates a child in size and ability of movements[1]. This humanoid is intended to be used in projects that range from psychology to services. Currently the interaction with the iCub is made through a Graphical User Interface (GUI), or through coding, although this solution is not simple to use.

The novel interface devices released and the movement created around them, serve as inspiration to this work as a new solution to an classical problem of HRI. The goal of this work is not only to develop useful solutions using these devices but also to understand how they compare to typical interfaces as a human commands using the iCub robot.

The two main contributions of this work are a set of applications for the iCub control using these interface devices, and the results of a evaluation done with human subjects and several different interfaces.

II. Related work

A robot autonomy level can be measured by the percentage of time that a robot does not need intervention to realize a task [2]. Nowadays robots are still not completely prepared to be autonomous and understand our language, but they have become useful in many tasks as helpers [3]. HRI is responsible for the way of "communicating" with a robot and control how it may help us in a task.

The humanoid control problem as been approached by several works in the past and used multiple different solutions. One of those solutions has been telepresence [4] that allows the remote control of a robot. The novel devices that are used in this dissertation allow for that type of remote control. Using these devices it is possible to do motion capture of the users movements and map it to the robot to enable learning by demonstration techniques, as shown in other works [5].

In the last few years there were several gaming interfaces introduced by gaming companies that have contributed for the development of the HCI, and from which HRI has taken advantage. These interfaces have been compared with each other by other works [6], and considered as suitable for motion capture. With these controllers it is possible to build cheap and robust motion capture systems for HRI.

The iCub is a 140cm tall humanoid robot child, that is able to crawl, sit, and do sophisticated manipulation with its hands. The iCub has 53 Degree of Freedom (DOF), 39 DOF that are distributed through the upper part of the robot body, and 14 DOF are in the lower part. The iCub project is all open source, which makes all the development of the iCub available to anyone interested. This software can be all downloaded form the iCub website[1] [7]. Each joint of the iCub as at least one motor that can be controlled through a server framework called Yet Another Robot Platform (Yarp). All the low level control is made through a set of Digital Signal Processor (DSP) based cards, connected via Controller area network (CAN) bus, specifically designed for the iCub robot[1]. The sensor and motor-state data is passed on to the PC104 card located in the head of the robot, where all that data is reformatted and synchronized with data streams to a Gbit Ethernet connection. Typically the heavier computation is made on independent machines external to the iCub, connected through the iCub Ethernet connection.

An Yarp network provides access and control to the iCub motor joints. The main abstractions of Yarp are the `ports`. `Ports` implement the observer pattern that allows to send a message through one port, to a multiple number of ports that are distributed across different computers, this way the sender and receiver can work independently. Another abstraction implemented by Yarp is the `device`, that allows to specify which device the programmer/user wants to interact with and the framework provides an object that hides the low-level programming details. This pair of `port`/`device` allows for the development

[1] http://www.robotcub.org/

of a remote device driver that can be used across a network, making the heavier processing independent from the robot hardware.

Fig. 1. Wiimote with the Motion Plus extension attached.

The Wii Remote (Wiimote) (figure 1) is the novel remote control used with the Wii gaming console. Its many attractions are that it provides unique interaction possibilities, it is inexpensive [8], [9] and durable. What allows for unique interaction possibilities is the accelerometer that can get the current orientation of the remote, and the amount of force applied to the remote in respect to the gravity [10]. This categorizes this device as a spatial convenient device [11], making it respect the three main spatial characteristics: provides spatial data, has sensors and emitters, also it is robust, cheap, and easily configured. Also the Wiimote has an extension port, through which it is possible to extend the Wiimote capabilities. The extension used for the work developed was the Wii Motion Plus (WM+) extension, that enables a better motion capture by adding a gyroscope to the Wiimote features. Inside the WM+ there is a dual-axis gyro by InvenSense, the IDG-600(pitch and roll), and a single-axis gyro by EPSON TOYOCOM labeled X3500W (yaw). These two gyroscopes allow to measure the rate of rotation along all 3-axes of X (pitch), Y (roll), and Z (yaw). The works that use this device range from motion control systems [12], to robotic user interface [13], or even art projects [14]. The flexibility and ease of use has been presented as a efficient interface for robotic control on several works [8], [13], [10]. These works typically compare traditional interface methods with the Wiimote interface, where the accelerometer and Infrared (IR) camera are used to control a robot or a 3D environment.

Fig. 2. Microsoft Kinect sensor.

The Kinect for the Xbox 360 console from Microsoft[2], is a webcam style device add-on that allows a controller free gaming interaction. The technology used by the Kinect to produce its result is quite old, but the algorithms that it uses for skeleton detection and user tracking are the ones being presented as innovations in the gaming world. The systems that are used with this sensor allow to control games using the user body without any type of controller in the user hands. As Microsoft puts it "you are the controller"[3]. Due to its low price and since there are a few free Application Programming Interfaces (APIs) being released, there has been an exponential rise in the amount of projects done using the Kinect. From NASA[4] to researchers and hobbyists, have contributed to a very steep and interesting evolution in the natural

[2]http://www.xbox.com/en-US/Kinect/

[3]http://www.microsoft.com/presspass/features/2010/oct10-/10-21kinectads.mspx

[4]http://www.sfgate.com/cgi-bin/article.cgi?f=/c/a/2011/01/10/BUO01H4ISI.DTL

interfaces community using this device. Those projects show many times original, and unexpected uses of the Kinect for different projects.

Fig. 3. Kinect sensor depth image example.

The Kinect can capture a depth-image where each pixel indicates also its depth. To achieve such an image several different techniques can be used, the most typical are: stereo imaging, Time-of-Flight (ToF) cameras, and structured light. The Kinect uses a structured light pattern that is projected by the IR light source. The image of that projection is captured by an IR camera in the Kinect. The deformations in that light pattern are interpreted by the Kinect software to detect what is the distance of each pixel captured. Recently Microsoft acquired a company that produces ToF cameras and it is rumored that the next version of Kinect might use ToF instead of the structured light technique[5]. This preference is due to the quality of the depth image calculated, that is higher in a ToF camera, and also more robust to different environments.

III. Proposed interface system

This work proposes to compare four interfaces for the command of the iCub robot: a GUI, a Directional Pad (D-Pad), a Wiimote device, and a Kinect device.

The interactions can be made with two different control methods either at the single motor level, or Cartesian level. The single motor control is done by setting the angle value of each motor individually. The Cartesian control uses inverse kinematics to define how a set of motors should be positioned so that an end-effector reaches its goal.

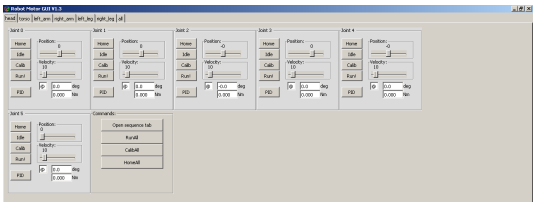


Fig. 4. Computer motor interface - named robotMotorGui.

The GUI interface, shown in figure 4, was already available in the iCub repository, so no development was needed. This interface has implemented a single motor and a Cartesian control, although in this work only the Cartesian control is used. This option was taken because the motor control with the GUI interface was too complex. To define the goal 3D point of the end-effector the GUI interface has three sliders, one slider per axis. Altering the position of the sliders will alter the desired position of the end-effector. Through this interface it is possible, also by the mean of sliders, to define the amount of time to reach the desired position from the current point, and the orientation of the end-effector. There can be only one slider altered at a time, each time that a slider is altered the user must wait until the end-effector reaches the desired position.

The D-Pad interface uses the Wiimote buttons, although its functionality could be easily passed to a computer or any other device with six programmable buttons. The front, back, left, and right buttons, move the robot end-effector in the same height plane in the respective directions relatively to the iCub robot. To alter the height of the end-effector the - and + buttons of the remote can be used, moving it down

[5]http://www.nytimes.com/2010/10/30/technology/30chip.html

and up respectively. The robot only moves while the user is pressing in one of the buttons. The movements allowed by this interface are illustrated in figure 5.

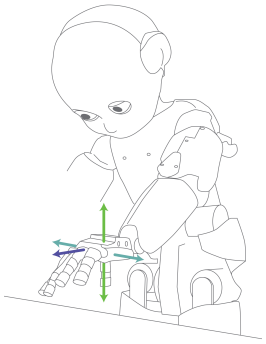


Fig. 5. D-Pad interface movements.

The Wiimote and Kinect interfaces were both developed specifically for this work. All of these interfaces only work while the trigger button (button B) on the bottom of the Wiimote is pressed. This button has no other use than setting the control on and off in both interfaces.

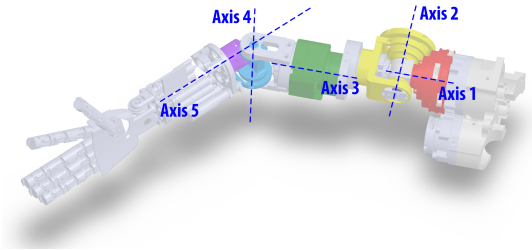
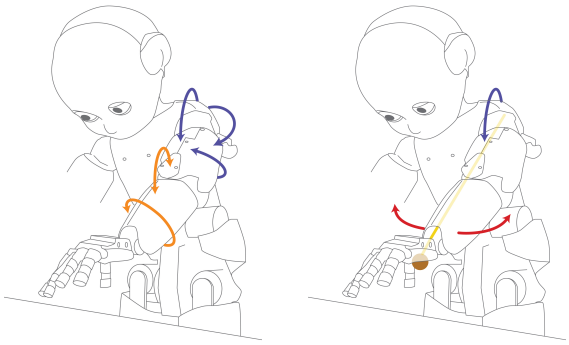


Fig. 6. Wiimote controlled motors in the iCub arm.

The Wiimote motor control maps the rotation made around each axis to a different motor. Because there are more motors to be controlled in the iCub arm than rotation axis, the arm is divided into two parts. The arm and the forearm, the selection of which part to control can be made by pressing buttons one and two. The angle of a motor only changes while the user is rotating the remote around the intended axis, when the user stops moving the Wiimote the robot stops immediately. The motors controlled and the axis of each motor are shown in figure 6. In figure 7a the arrows show the type of rotation per each part, blue arrows arm part, orange arrows forearm part.

The Wiimote kinematic control is mapped to the rotations of the arm around the iCub shoulder with a constant radius. The radius can be altered by pressing the - and + buttons, that decrease or increase the radius size. What is meant by this explanation is that the end-effector always follows a virtual point that is controlled by the Wiimote. That virtual point rotates around an origin point, the shoulder, from which maintains a constant distance. A rotation of the Wiimote around its X axis, results in a rotation of the virtual point with the same amount of degrees around the origin point in the X axis. In figure 7b the yellow line represents the radius, the brown ball represents the end-effector.



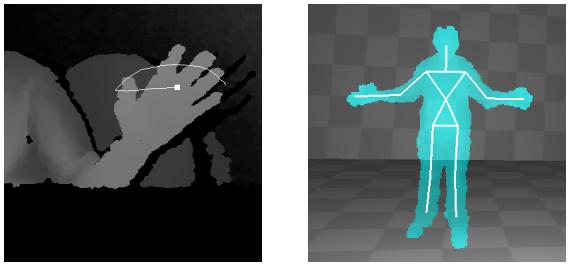
(a) Motor control. (b) Cartesian control.

Fig. 7. Wiimote control types.

The Kinect motor control maps the skeleton detected to the iCub motors. Figure 8b shows the skeleton detection made by a sample program. To map the skeleton detected by the Kinect to the iCub each of the rotation matrices that define the skeleton is converted to Euler angles, if its confidence level is higher than a defined threshold. The Euler angles are

mapped to the motors directly depending on the motor axis. This way a pose that is detected by the Kinect is mimicked by the iCub in the most similar way possible.

The Kinect kinematic control maps the user hand position to the iCub end-effector. Figure 8a shows the hand detection made by a sample program. The metaphor used for this interaction is grabbing the robot hand, while standing face to face with it. So the iCub will follow the user hand detected by the Kinect, in a mirrored way. Pushing the hand front will mean for the iCub to pull the hand towards itself. Moving the hand to the left, will mean moving the hand to the right relatively to the iCub origin. The iCub only moves relatively to the hand initial position, where the initial position is the position of the hand on the first moment the Wiimote button is pressed.

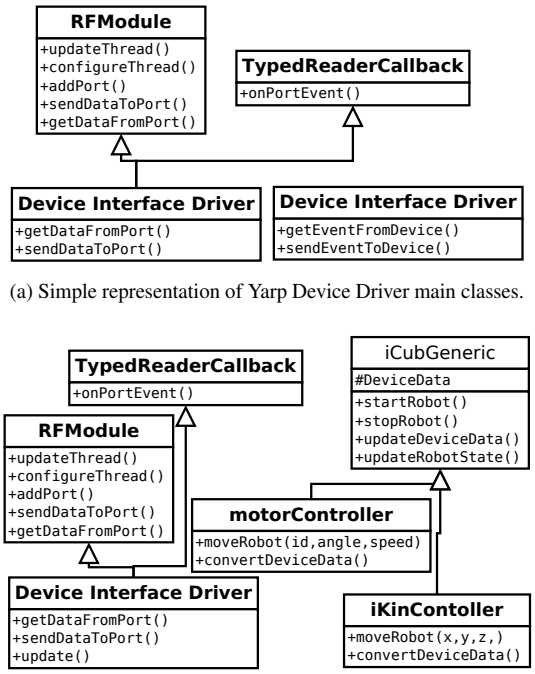


(a) Kinect hand detection. (b) Kinect skeleton detection.

Fig. 8. Kinect detection made by sample programs.

IV. Applications Developed

There were five applications developed that can be separated into two groups. The first group is the Wiimote device group, the second is the Kinect device group. The Wiimote group has three distinct applications, an interface application, a solver application, and a control application. The Kinect group has an interface application and a control application. The structure of the two common applications is the same, only being distinguished by its inner logic.



(a) Simple representation of Yarp Device Driver main classes.

(b) Simple representation of the Interaction Module main classes.

Fig. 9. Interaction applications structure.

The interface application implements the Yarp device abstraction. This application is responsible for streaming the data from the interface device into a Yarp port in a simple and usable format, so that other programs can take advantage from that data by connecting to the port or using it as an Yarp device. The control application uses the Yarp device abstraction implemented by the interface application. This application is responsible for converting the data retrieved from the interface application into values that are used to control the iCub. Figure 9a shows the main classes of the interface and control applications, in the case of the Wiimote the control application is also responsible for altering the settings of the Wiimote.

As can be easily seen by the figure both applications have a device interface driver class that inherits from the RFModule and TypedReaderCallback classes. The functionality of this common class is similar in both cases. Both applications have a data receiving port

that executes a function per each set of data received, and a data sending port that is maintained by a thread that writes data to that port. This system enables communication between this two modules, or other helper modules that might be implemented. An example of a helper module is the Wiimote solver application which is used to decide based on the WM+ data which is the best position for the end-effector to try and reach.

The applications were developed using C++ and used the Yarp and iCub libraries. For the development of the interface applications, two extra libraries were used to get data from the devices the wiiuse library[6] and the OpenNI/NITE library[7]. The wiiuse library[8] was used to get data and set settings of the Wiimote. The wiiuse version used was altered by the fwiineur library[9] developers so that data from the WM+ could be retrieved. The OpenNI/NITE library uses the IR projector and IR camera from the Kinect to calculate a depth map. Using this depth map and the NITE algorithms it is possible retrieve a set of rotation matrices or points that describe the pose and position of a users body or body parts.

V. Evaluation

The evaluation goal was to understand what were the main flaws in each system developed by looking at the users difficulties. From its results it was also possible compare the several interaction systems.

For the evaluation test subjects with very different backgrounds, and little technological knowledge were chosen with the exception of one user that had a strong technological understanding. The group of users was composed of men and women, with age from twenty to sixty years old. All of the subjects use a computer every day, although only two users had used Wiimote like devices, and none of the users had ever used the Kinect. This was also the first time any of the users interacted with the iCub robot. There was a deliberate choice of users with no technical background because those users do not have any predetermined idea of what to expect from such an interface, or from a humanoid robot. For the evaluation, users were invited to come to the lab during a weekend, so that they could be comfortable.

The evaluation was split into two parts, the tests and the questionnaire. There were six tests made. All of the tests only used the right arm of the iCub, from shoulder to hand. Two of them were based on typical interfaces, a GUI, and a D-Pad (the Wiimote cursor). The other four were based on two novel interfaces, the Wiimote, and the Kinect. The novel devices interfaces were used in two tests each, one using individual motor control, the Wiimote motor test, and the Kinect skeleton test, and the other on the Cartesian control, the Wiimote kinematic test, and the Kinect hand kinematic test. For the tests there was no feedback other than the robot reaction, this made the tests harder. Using a graphical feedback would result on having to consider the design of that feedback and not only the interaction allowed by the interface. These tests followed the interaction described in section III, and used the applications described in section IV.

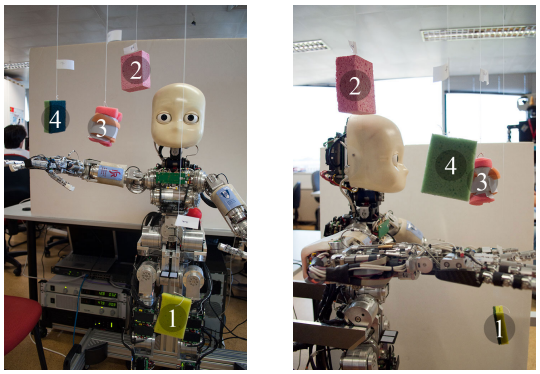
Before and during the tests a presentation was shown describing what was the iCub project, what was the goal of that evaluation, what should the test subjects do, and how the interfaces worked.

The tests were divided into five tasks. The three first tasks of the tests were made to make the user comfortable, and to check if the user was able to do the most basic tasks. The first task was to “play

[6]http://sourceforge.net/projects/wiiuse/
[7]http://www.openni.org/
[8]http://sourceforge.net/projects/wiiuse/
[9]http://fwiineur.blogspot.com/search/label/release

around" with the interface, the second task was to move the arm up and down, the third task was to move the arm left and right. All users successfully conclude these tasks without any difficulty.

The fourth and fifth tasks used a set of four objects placed at the right arm reach distance and suspended from the ceiling. The objects were numbered from one to four, as can be seen in figure 10. For this tasks the time spent, the amount of errors and the user comments were annotated. Before any of the tasks started the robot arm was positioned into an initial "safe" position.

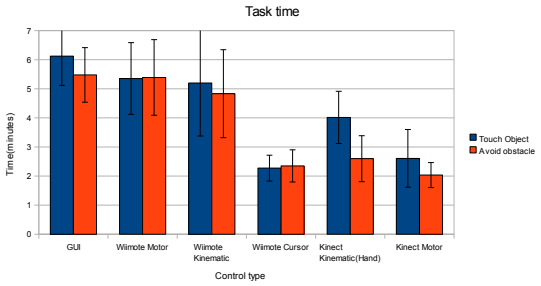


(a) Front view. (b) Left view.

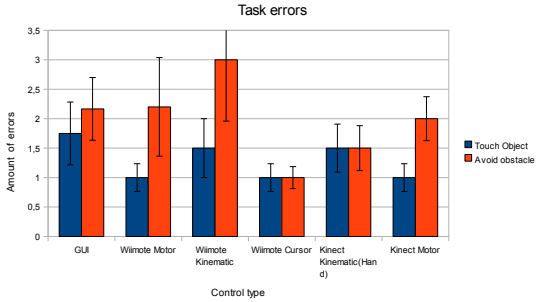
Fig. 10. Interfaces test scenery.

The goal of task four was to touch all the objects, from one to four, without colliding with any of the other objects if possible. The goal of task five was to touch objects four one and two, by this order. But in task five besides not colliding with the unintended objects, there was an obstacle (an A4 sheet of paper), placed in the object three position. Collision with the obstacle should be avoided. Figures 11 shows the amount of time needed to conclude the tasks, and the amount of errors per task.

The graphs support the idea that the most successful interaction was the Wiimote cursor. The D-Pad interface is the one that got the best time and smaller error values. From user comments it was



(a) Task time comparison.



(b) Task error comparison.

Fig. 11. Tasks results.

possible to understand that this interface was the most well known by users, helping to visualize how action with the interface would result in a robot reaction. By contrast the Kinect interface was the least known interface. Although when comparing the Kinect interface performance with the D-Pad interface performance, the results were very similar in number of errors and amount of time. The GUI interface was a classic interface, but the control of the robot was considered too complicated by the users, even with only three sliders. The Wiimote interactions were the most difficult to make the user understand how the interaction worked. That difficulty in understanding was reflected into the results. It is assumed that the understanding of how the interaction works is related to the difficulty of control. One of the subjects achieved very good results in the Wiimote motor interface only after being extremely clear about how the interaction worked. This interface was specially bad with the older users that did not seem to be

comfortable with the explanation.

VI. Conclusion

The motivational question for this work was "how can novel gaming interfaces compare as a humanoid interface?". The answer was explored through an evaluation made where several interfaces were used with the same goal so that their performance could be compared.

The strongest conclusion statement of this work is that familiar interfaces perform better for the untrained people. This could be taken from the fact that the most successful interface during the evaluation was the D-Pad interface. Even with limited control as with this interface users were able to perform very well the tasks proposed. The great strength of this interface was that users knew what to expect, because they are used to have this kind of interface and could easily relate what they expected to what happened.

The Kinect although a novel interface was the second most successful interface. This was the interface were users saw more possibilities and the easiest to learn. But being the interface that was the most unknown augmented the uncertainty among users that it would not really obey in the expected way, making users spend much time testing it out to understand what the iCub reaction would be. Being a common interface eliminates this skepticism making the users have a predefined idea of what might happen, as it was the case with the D-Pad interface. Having a predefined idea helps to adapt to the real interaction.

The Wiimote interaction supports this idea because users that were not able to grasp the concept of the interface, performed poorly, while users that were able to understand it, slightly performed better. A user that during a test understood very well how the interaction worked was able to have a very impressive performance, although in the beginning of the test the user performance was very low. The Wiimote was the less common control method, and also the most difficult to relate to. Those were considered the main reasons for the difficulty in understanding the interface.

References

[1] G. Metta, L. Natale, F. Nori, G. Sandini, D. Vernon, L. Fadiga, C. von Hofsten, K. Rosander, M. Lopes, J. Santos-Victor, A. Bernardino, and L. Montesano, "The icub humanoid robot: An open-systems platform for research in cognitive development," *Neural Networks*, pp. 1125–1134, 2010.

[2] H. A. Yanco, "Classifying human-robot interaction: An updated taxonomy," in *Proc IEEE SMC*, 2004, pp. 2841–2846.

[3] W. Bluethmann, R. Ambrose, M. Diftler, S. Askew, E. Huber, M. Goza, F. Rehnmark, C. Lovchik, and D. Magruder, "Robonaut: A robot designed to work with humans in space," *Autonomous Robots*, vol. 14, pp. 179–197, 2003, 10.1023/A:1022231703061. [Online]. Available: http://dx.doi.org/10.1023/A:1022231703061

[4] J. Kofman, X. Wu, T. Luu, and S. Verma, "Teleoperation of a robot manipulator using a vision-based human-robot interface," *Industrial Electronics, IEEE Transactions on*, vol. 52, no. 5, pp. 1206 – 1219, October 2005.

[5] S. Calinon, F. D'halluin, E. Sauser, D. Caldwell, and A. Billard, "Learning and reproduction of gestures by imitation," *Robotics Automation Magazine, IEEE*, vol. 17, no. 2, pp. 44 –54, June 2010.

[6] S. Purkayastha, N. Eckenstein, M. Byrne, and M. O'Malley, "Analysis and comparison of low cost gaming controllers for motion analysis," in *Advanced Intelligent Mechatronics (AIM), 2010 IEEE/ASME International Conference on*, july 2010, pp. 353 –360.

[7] G. Metta, G. Sandini, D. Vernon, L. Natale, and F. Nori, "The icub humanoid robot: an open platform for research in embodied cognition," in *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, ser. PerMIS '08. New York, NY, USA: ACM, 2008, pp. 50–56. [Online]. Available: http://doi.acm.org/10.1145/1774674.1774683

[8] S. Olufs and M. Vincze, "Robot on the leash − an intuitive inexpensive interface for robots using the nintendo wii remote," in *Advances in Robotics Research*, T. Kröger and F. M. Wahl, Eds. Springer Berlin Heidelberg, 2009, pp. 311–321, 10.1007/978-3-642-01213-6 28. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-01213-6_28

[9] A. Azmi, N. Alsabhan, and M. AlDosari, "The Wiimote with SAPI: Creating an Accessible Low-Cost, Human Computer Interface for the Physically Disabled," *IJCSNS*, vol. 9, pp. 12–63, 2009.

[10] C. Guo and E. Sharlin, "Exploring the use of tangible user interfaces for human-robot interaction: a comparative study," in *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, ser. CHI '08. New York, NY, USA: ACM, 2008, pp. 121–130. [Online]. Available: http://doi.acm.org/10.1145/1357054.1357076

[11] C. A. Wingrave, B. Williamson, P. D. Varcholik, J. Rose, A. Miller, E. Charbonneau, J. Bott, and J. J. L. Jr., "The wiimote and beyond: Spatially convenient devices for 3d user interfaces," *IEEE Computer Graphics and Applications*, vol. 30, pp. 71–85, 2010.

[12] Y. Chow, "Low-cost multiple degrees-of-freedom optical tracking for 3d interaction in head-mounted display virtual reality," *International Journal of Recent Trends in Engineering*, vol. 1, no. 1, pp. 52–56, 2009.

[13] A. Basçetinçelik, "WiiRobot: Controlling Robots with Wii Gestures," *Providence, RI: Department of Computer Science Brown University*, 2009.

[14] H.-J. Lee, H. Kim, G. Gupta, and A. Mazalek, "Wiiarts: creating collaborative art experience with wiiremote interaction," in *Proceedings of the 2nd international conference on Tangible and embedded interaction*, ser. TEI '08. New York, NY, USA: ACM, 2008, pp. 33–36. [Online]. Available: http://doi.acm.org/10.1145/1347390.1347400