Eurographics Italian Chapter Conference (2011)
A. F. Abate, M. Nappi, and G. Tortora (Editors)

# Gestural Interaction for Robot Motion Control

Giuseppe Broccia, Marco Livesu, and Riccardo Scateni

University of Cagliari, Dept. of Mathematics and Computer Science - Italy

**Abstract**

*Recent advances in gesture recognition made the problem of controlling a humanoid robot in the most natural possible way an interesting challenge. Learning from Demonstration field takes strong advantage from this kind of interaction since users who have no robotics knowledge are allowed to teach new tasks to robots easier than ever before. In this work we present a cheap and easy way to implement humanoid robot along with a visual interaction interface allowing users to control it. The visual system is based on the Microsoft Kinect's RGB-D camera. Users can deal with the robot just by standing in front of the depth camera and mimicking a particular task they want to be performed by the robot. Our framework is cheap, easy to reproduce, and does not strictly depend on the particular underlying sensor or gesture recognition system.*

## 1. Introduction

The main goal of learning from demonstration is to have a robot which learns from *watching* a demonstration of the task to be performed [ACVB09]. Users who have no robotics knowledge can take strong advantage while teaching new tasks to humanoid robots. To achieve that, a robust and reliable gesture recognition systems is needed, in order to detect the non-verbal information which are sent to them by the human gestures.

Gestures recognition aims at recognizing meaningful expressions of motion performed by a human being. Mainly such recognition involves hands, arms, and face, but can be extended to the head and/or the whole body too. Gesture recognition has great importance the in Human-Robot Interface (HRI) field [GS07], where its simplicity is useful in designing an intelligent and efficient system which allow humans and robots to communicate easier. Gesture applications are manifold, ranging from virtual reality to medical rehabilitation, sign language and so on. For a comprehensive survey one can refer to [MA07].

In this paper we present a novel, cheap, humanoid robot implementation along with a control and interaction interface which allows users to control it just by standing in front of a depth camera and mimicking the task to be performed. To achieve this we employ a gesture recognition system which is composed by different modules. The vision system is provided by the Microsoft's Kinect depth camera, an RGB-D sensor which couples RGB images with depth information provided by an Infra-red (IR) sensor. From its first release, in 2010, Microsoft Kinect [Kin] gathered the attention of the scientific community due to its low price (which is more than 5 times lower than a usual depth camera) and its precision, which meets the gesture recognition requirements.

**Figure 1:** *The robot used in our project.*

The RGB-D sensor can be accessed and controlled by the open-source OpenNI [Ope] framework via the NITE [NIT] middleware. While the former provide interfaces for several sensors, the latter provides a control skeleton model of the subject calculated using depth information of the sensor



DOI: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2011/061-066

along with real time tracking of the skeleton feature points. We will discuss more in depth about NITE control skeleton in Section 3. Depth sensors can be employed in many other different fields, to have an example one can refer to [Wil10].

Our gesture acquisition system is similar to the one proposed by [RGPS11], but additionally we provide pure mimicking capabilities along with a gesture recognition system which do not depends on the particular underlying framework employed. Other notable examples of HRI are given in [SC11], [AWU*08] and [VJiCGP09].

The rest of the paper is organized as follows: in Section 2 we introduce the humanoid robot we built for our tests. Section 3 is about the gesture control system; Sections 4 and 5 are respectively about actual limitation and future improvements of our robot while Section 6 is about conclusions.

## 2. The Robot

In our experiments we used a *homebrewed* humanoid robot (Figure 1). The robot is equipped with eight servo commanders (from now on *servos*) which controls the following eight DOF (degrees of freedom) (see Figure 2):

- A and B control the rotation of the right forearm
- C controls the rotation of the right arm around $Z$ axis
- D controls the rotation of the right arm around $X$ axis
- E controls the rotation of the left arm around $X$ axis
- F controls the rotation of the left arm around $Z$ axis
- G and H control the rotation of the left forearm

The robot is mounted on wheels, instead of walking on its legs for two main reasons: first, and most important, balancing a robot on its legs would have gone far beyond the scope of our project, which is centered mostly on the reproduction of the gestures; moreover, it would have costed much more to have more moving parts and servo commands.

The servos we employ allow rotations of at least 180°, thus reducing the overall mobility respect to a human (see Figure 3). We then decide to limit the shoulder mobility, forbidding the arms to go behind the body. Even if this choice seems to drastically reduce the expression power of the robot respect to a human in our experiments we found that user's gesture expressing power is weakly reduced (we will discuss more in depth about this issue in Section 4).

Servo commanders are controlled whit an Arduino [Ard] logic board, which is an open-source single-board microcontroller which allows to interface electronic devices to the computer quickly and easily. The version of the board we employ is named *Arduino Uno* and integrates a serial interface (through USB cable) to enable the exchange of data with the computer.

Mobility is guaranteed by *DFRobot 4WD Arduino Mobile Platform* which, as the name suggests, offers the robot four motored wheels and the support for fixing the Arduino board

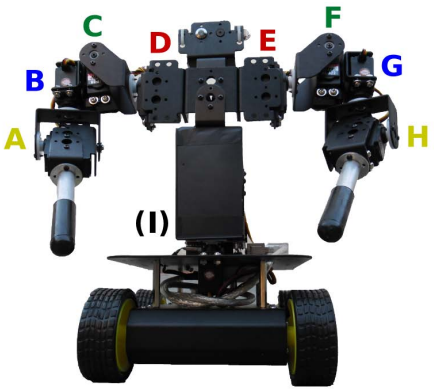


**Figure 2:** *The eight DOF of our robot.*

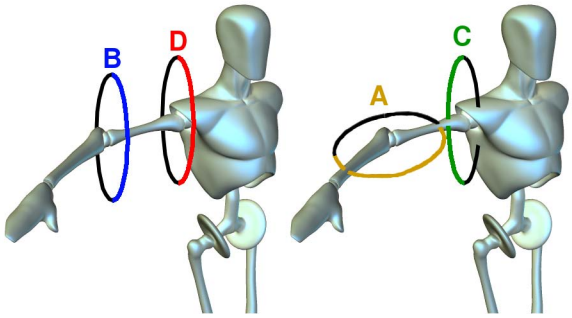


**Figure 3:** *Rotation limits for servos A, B, C and D. For each ellipse, the colored parts show the allowed rotation angles while the black parts indicate the forbidden angles.*

over it. Such kit, however, does not offer any control card for the engines, so one need to build it by himself or buy it. In order to use as few pin as possible, we created an electronic control card which handles couple of wheels, as the user can move left, right or both left and right wheels. By combining this three movements with the two possible rotation directions of the wheels the user is able to control the whole robot mobility (see Table 1 for movement commands details).

| | Left Wheels | Right Wheels | Enable |
|---|---|---|---|
| Forward | fw | fw | yes |
| Backward | bw | bw | yes |
| Turn right | fw | bw | yes |
| Turn left | bw | fw | yes |
| Stop | - | - | no |

**Table 1:** *Encoding control to handle motors motion. Each wheel can rotate either forward (fw) or backward (bw) respect the robot.*

The robot is also equipped with an VGA RGB webcam, placed in its head, which allow users to drive the robot successfully even when they lose direct visibility to it.

Mobility plays an important role in the robot handling, thus, since both webcam and Arduino require a direct USB connection with the workstation, in order to reduce the constrictions given by cables we employ a small 4-port USB hub which allow us to control the robot with a single USB cable. Besides solving the problem of cables, this choice provide two additional USB ports useful for future upgrades.

Figure 4 shows the schematic diagram used for the wiring of the project. The depth sensor is connected to the computer via USB. On the computer runs the recognition software that recognizes the gestures of the user and communicates with the robot. The scheme doesn't include links for the power of the robot, as this occurs through the USB bus to the onboard hub, Arduino and webcam, whereas with an external power supply (5 Volts, 5 Ampere) for servos, control board wheels and wheel motors; this choice will be discussed in Section 4.

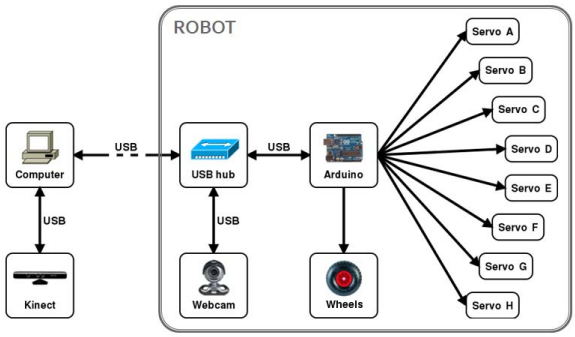


**Figure 4:** *Electrical diagram of the robot.*

## 3. Gesture Recognition Framework

The main goal of our project was to catch gestures from a human user which stands in front of a depth sensor, and control a robot able to reproduce them. In this section we will present all the technical issues of the gesture recognition system, which plays an important role in the whole framework.

We employed the Microsoft Kinect [Kin] as depth sensor, (using the OpenNI [Ope] APIs to interface it) and the NITE [NIT] framework for depth image analysis and control skeleton extraction. Even if NITE includes the support for gesture recognition, we choose to re-implement the recognition because to control how our robot reproduces the gestures, we needed low-level information which NITE is not able to manage.

### 3.1. User detection and control skeleton

The RGB-D sensor computes the distance at which the user's body is located while the underlying API are able to detect shapes and build a skeleton model of the human who's being detected. Moreover, the API tracks the position and orientation parameters of all the joints of the skeleton model in real time. This information enable us to track down arms, trunk and legs of the subject in real time (the NITE middleware provides an actual frame-rate of 25 FPS).
To start a run the user must stand in front of the sensor assuming the *calibration position*, that is, with arms parallel to the ground and forearms perpendicular (see Figure 5). After a short calibration phase the subject's silhouette becomes blue and the NITE control skeleton appears over the silhouette. As summarized in Figure 6 such skeleton is composed by the following 15 control points:

- head ($h$)
- neck ($n$)
- torso ($t$)
- left hand ($lh$) and right hand ($rh$)
- left elbow ($le$) and right elbow ($re$)
- left shoulder ($ls$) and right shoulder ($rs$)
- left hip ($lh$) and right hip ($rh$)
- left knee ($lk$) and right knee ($rk$)
- left foot ($lf$) and right foot ($rf$)

Until now in our experiments we defined only with few gestures/commands, hence we worked only with a subset of the skeleton control points.

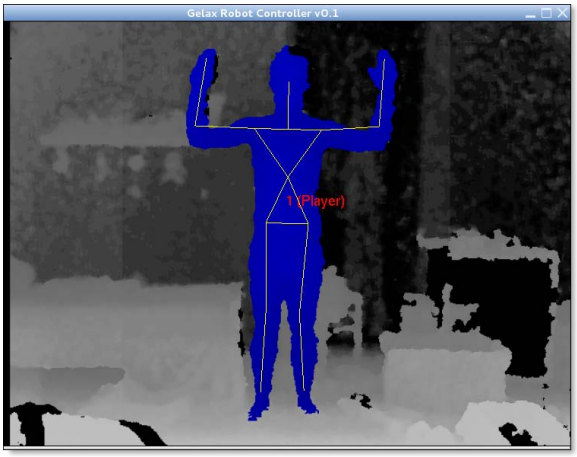


**Figure 5:** *To start a run with our framework the user must stand in front of the depth sensor assuming the calibration position (i.e., with arms parallel to the ground and forearms perpendicular)*

### 3.2. Upper-body gestures recognition

At any time, the robot's upper body pose depends directly from the angle assumed by the servo commanders which control arms and shoulders (from servo A to servo H in Figure 2). To estimate such angles in the control skeleton we take advantage of the Carnot's theorem, which allows the

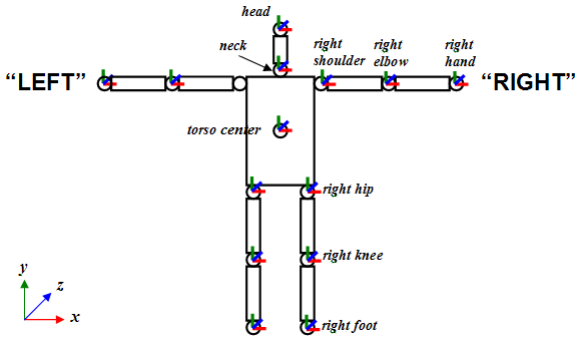


**Figure 6:** *The 15 feature points provided by NITE control skeleton: left and right shoulders, left and right elbows, left and right hand, left and right hips, left and right knees, left and right foots, torso, neck and head.*

calculation of the angle of a triangle just by knowing the size of its edges. The angle is given from the following formula

$$\alpha = \arcsin\left[\frac{b^2+c^2-a^2}{2bc}\right], \qquad (1)$$

where $a$, $b$ and $c$ are the triangle edge lengths and $\alpha$ is the angle of the corner opposite to the edge $a$. An example of angle estimation (for the servo commander C) is showed in Figure 7, where the edge lengths are calculated in terms of distances between the skeleton feature points showed in Figure 6, in particular

```
a = distance(rightUnderarm,rightElbow)
b = distance(rightElbow,rightShoulder)
c = distance(rightShoulder,rightUnderarm)
```

The angles obtained by equation (1) are already in the working range of the servo, in fact to $0^\circ$ corresponds a low vertical arm, to $90^\circ$ corresponds a horizontal arm and to $180^\circ$ corresponds an up vertical arm.

However, in our experiments, we found that the arm controlling obtained with the previous algorithm is not satisfactory, because is too much sensitive to small displacements of the skeleton control points. To achieve a more robust tracking we employed, only for the servos C and F (placed in the robot's shoulders), a different angle estimation algorithm: let $d$ be the arm length measured from the shoulder to the elbow and $q$ the difference between the elbow and shoulder heights (e.g., their distance form the ground); when the arm is raised the angle $\alpha$ is $180^\circ$ while when the arm is lowered $\alpha$ becomes $0^\circ$. To estimate inner values of $\alpha$ the following straightforward proportion has to be solved

$$q : d = \alpha : 2\pi\,,$$

then

$$\alpha = \frac{q \cdot 2\pi}{d}\,.$$

Estimating the angles of each servo commander frame by frame, and updating their position we yield the mimic properties of the robot.

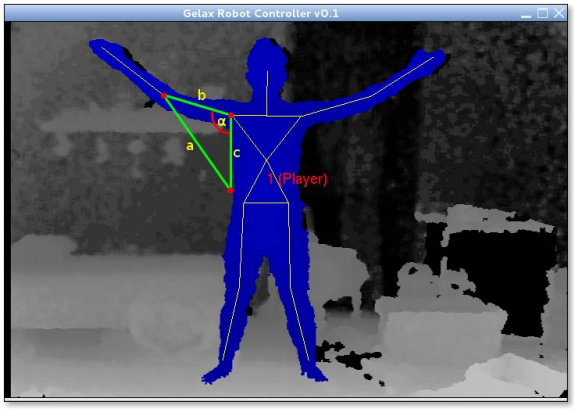


**Figure 7:** *Angle estimation for servo commander C. Thanks to Carnot's theorem we are able to estimate angle $\alpha$ just by knowing the lengths of the edges a, b and c, whose endpoints corresponds to the feature points of the control skeleton provided by NITE framework.*

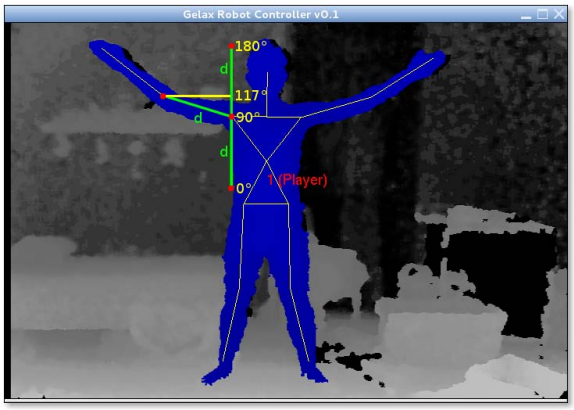


**Figure 8:** *To estimate the angle of the shoulders we compare the arm length (d) with the height difference between the elbow and its shoulder (e.g., their distance from the ground). When the arm is lowered such difference is zero, while when the arm is raised the difference is 2d. The first setting corresponds to an angle of $0^\circ$ while the latter corresponds to an angle of $180^\circ$. For inner angles a simple proportion has to be solved.*

### 3.3. Gestures for robot motion

Robot motion is controlled by the user with leg and hips gestures. To control robot's backward and forward movements the user must move respectively one foot backward or forward. To avoid unwanted robot displacements, the system recognise that the gesture has been performed if and only if the movement amount is greater of a threshold, which is calculated taking into account the user leg's length. The choice of a user adaptive threshold works better respect to a static one because it makes gestures invariant to users having different physical stature or different conformations. The condition to be satisfied is the following

$$\left| \mathsf{Depth}\,(lf) - \mathsf{Depth}\,(rf) \right| \geq d\,(rf, rk)\ , \qquad (2)$$

meaning that the depth displacement between the foots must be greater or equal the distance between one foot and its knee.
To distinguish between a forward and a backward gesture the system checks the sign of the difference showed in equation (2): a positive sign means that the user moved one foot forward, while a negative sign means that the user moved one foot backward.
The recognition of *left* and *right* gestures is performed in the same way, but taking into account hips and shoulders instead of knees and foots. When the depth difference between control skeleton's hips is greater than a threshold defined in function of the distance between one hip and one shoulder lying o the same side, a gesture has been performed

$$\left| \mathsf{Depth}\,(lh) - \mathsf{Depth}\,(rh) \right| \geq d\,(rh, rs)\ , \qquad (3)$$

Once again, the sign of the difference allows the system to distinguish between a *left* and *right* gesture.

When no one of the above conditions holds, the gesture recognized is *stop*. Table 2 summarizes all the motion gestures that can be performed by a user.

| Command | Gesture |
|---|---|
| Forward | Move and keep one foot forward |
| Backward | Move and keep one foot backward |
| Turn left | Rotate the torso to the left |
| Turn right | Rotate the torso to the right |
| Stop | None of the previous moves |

**Table 2:** *Motion capabilities of our robot with the gestures associated to them.*

## 4. Limitations

Most of the limitations discussed in this section are strictly related with the robot introduced in Section 2. By our choice, we tried to keep it as simple and cheap as possible, as anyone with few knowledge and little cost should reproduce our results.

### 4.1. Movement restrictions

Our robot has movement restrictions as is equipped with servo commanders that allow rotations of at least $180^\circ$. This should be good for elbows or knees, which in human bodies are restricted to less than $180^\circ$, but represents a problem for shoulders, when the overall mobility respect to a human turns out to be reduced. To overcome this limitation continuous rotation motors should be used instead, along with encoders for detecting the current position. However, this choice would drastically increase both weight and complexity of the robot.

### 4.2. Mobility restrictions

The robot is linked to the workstation by two cables: one USB cable for data exchange and one power cable for servo commanders. Although they are long enough, cables represent a limit in the overall mobility anyway. For what concerns data communications the cable should be removed in favor of a transmitter/receiver for wireless Arduino, one for the robot and one (with USB connection) for the workstation. To remove the power cord a battery pack is necessary. The use of a battery would significantly increase the total weight: the solution of this technical hitch occurs replacing the motors in current use with more powerful and expensive models. Webcam should be replaced with a wireless camera.

### 4.3. Wheels vs Legs

There are several reasons why we choose to use a mobile platform with wheels instead of legs. Equipping the robot with legs (similar to the arms already present) would have entailed additional work much more demanding than the project. The legs would create balance problems very complex to solve.

### 4.4. Hardware dependencies

Even if we did everything possible to make the project as generic as possible, we must say that this is so much related to the hardware in use. The calculation of servos angles that control the robot depends on their range of motion. The transmission encoding used it's up to the type of transmission channel, as well as the Arduino software robot side.

## 5. Future Works

In this project we used Microsoft Kinect [Kin] as depth sensor. An interesting feature of the Kinect is the possibility to employ microphone arrays to impart voice commands. The

use of an array of microphones makes easy the calibration depending on the environment where the user acts (e.g. by detecting the echo due to the walls or furnishings), in this way it's being removed ambient noise and music playing, permitting the correct recognition of voice commands. At the moment, the framework OpenNI/NITE doesn't support this feature, but in the future, when the support will be completed, the project will be upgraded to receiving voice commands to access special features. These special features includes the "Start" and "Stop" obviously used to start and stop the robot. In addition, we will add voice commands to lock a specific servo in a fixed angle; then, we could add the ability to perform precision movements by specifying a single actuator and control it individually.

The robot we realized is only a prototype to verify the potentiality offered from this new type of control. We obtained very positive results. In the future, the robot will be refreshed with continuously rotating motors with encoders rather than servos, and its physiognomy will be modified in order to become more similar to human's. Wheels could be substituted by legs, but it's a discussing point because they cause balance problem and speed decrease.

As we mentioned in Section 4 the robot is connected to the computer via cables. In the future the robot will be equipped with an "Arduino wireless shield" and the webcam will be replaced with a radio model. This won't require any software modification.

In order to extend the operating range of the robot, we decided to make the software work through internet. On the first computer will be connected the sensor for the recognition and the specific client software, the second computer will be connected to the robot and will run the specific server software. The serial communication protocol adopted will be modified for TCP-IP communication. It should also be included the video stream.

## 6. Conclusions

Learning from demonstration is the scientific field which studies one of the easier ways a human have to deal with a humanoid robot: mimicking the particular task the subject wants to see reproduced by the robot. To achieve this a gesture recognition system is required.
In this paper we presented a novel and cheap humanoid robot implementation along with a visual, gesture-based interface, which enable users to deal with it. To catch and control subject's gestures we employed the Microsoft Kinect RGB-D along with OpenNI and NITE frameworks.Users are allowed to control the robot just by mimicking the gestures they want to be performed by the robot in front of the depth camera.
This should be seen as preliminary work, where we are providing elementary interaction tools, and should be extended in many different fashions, depending on the tasks the robot should perform.

## References

[ACVB09] ARGALL B. D., CHERNOVA S., VELOSO M., BROWNING B.: A survey of robot learning from demonstration. *Robotics and Autonomous Systems 57*, 5 (2009), 469 – 483. 1

[Ard] http://www.arduino.cc/. 2

[AWU*08] ASFOUR T., WELKE K., UDE A., AZAD P., DILLMANN R.: Perceiving objects and movements to generate actions on a humanoid robot. *Lecture Notes in Electrical Engineering 8* (2008), 41–55. 2

[GS07] GOODRICH M. A., SCHULTZ A. C.: Human-robot interaction: a survey. *Found. Trends Hum.-Comput. Interact. 1* (January 2007), 203–275. 1

[Kin] http://www.xbox.com/kinect. 1, 3, 5

[MA07] MITRA S., ACHARYA T.: Gesture recognition: A survey. *IEEE Transactions on Systems, Man and Cybernetics - Part C 37*, 3 (2007), 311–324. 1

[NIT] http://kinect.dashhacks.com/primesense-nite. 1, 3

[Ope] http://www.openni.org/. 1, 3

[RGPS11] RAMEY A., GONZÁLEZ-PACHECO V., SALICHS M. A.: Integration of a low-cost rgb-d sensor in a social robot for gesture recognition. In *Proceedings of the 6th international conference on Human-robot interaction* (New York, NY, USA, 2011), HRI '11, ACM, pp. 229–230. 2

[SC11] SUAY H. B., CHERNOVA S.: Humanoid robot control using depth camera. In *Proceedings of the 6th international conference on Human-robot interaction* (New York, NY, USA, 2011), HRI '11, ACM, pp. 401–402. 2

[VJiCGP09] VARONA J., JAUME-I CAPÓ A., GONZÍLEZ J., PERALES F. J.: Toward natural interaction through visual recognition of body gestures in real-time. *Interact. Comput. 21* (January 2009), 3–10. 2

[Wil10] WILSON A. D.: Using a depth camera as a touch sensor. In *ACM International Conference on Interactive Tabletops and Surfaces* (New York, NY, USA, 2010), ITS '10, ACM, pp. 69–72. 2