2014 14th IEEE-RAS International Conference on
Humanoid Robots (Humanoids)
November 18-20, 2014. Madrid, Spain

Humanizing *NAO* robot teleoperation using ROS

Igor Rodriguez[1], A. Astigarraga[1], E. Jauregi[1], T. Ruiz[1], E. Lazkano[1]

***Abstract*— The work presented here proposes two different ROS packages to enrich the teleoperation of the robot *NAO*: speech-based teleoperation (in Basque) and gesture-based teleoperation together with arm control. These packages have been used and evaluated in a human mimicking experiment. The tools offered can serve as a base for many applications.**

I. INTRODUCTION

Human-robot interaction (HRI) is the study of interactions between humans and robots. HRI is a multidisciplinary field with contributions from human-computer interaction, artificial intelligence, robotics, natural language understanding, design, and social sciences.

A requirement for natural HRI is to endow the robot with the ability to capture, process and accurately and robustly understand human requests. Therefore it is important to analyse the natural ways by which a human can interact and communicate with a robot. A considerable number of robotic systems have been developed in the last decade showing HRI capabilities [5][7].

In recent years, the robotics field has seen the emergence of sophisticated humanoid robots, including Honda Asimo and *NAO*. Due to their human-like morphologhy, humanoids are well-suited to operate in shared environments with humans. Therefore, they are used by many researchers to investigate fields like navigation in unstructured environments, full body motions and human-robot interaction.

Real-time teleoperation of humanoid robots by detecting and tracking human motion is an active research area. This type of teleoperation can be considered as a particular way of interaction between a person and a robot, allowing an intuitive teleoperational control due to similarities in embodiment between the human master and the robot slave. It is an interesting research topic, the related work is abundant. To mention some, Setapen et al. [15] use motion capture to teleoperate a *NAO* humanoid robot, using inverse kinematic calculations for finding the mapping between motion capture data and robot actuator commands. Matsui et al. [12] use motion capture to measure the motion of both, a humanoid robot and a human, and then adjust the robot's motions to minimise the differences, with the aim of creating more naturalistic movements on the robot. Song et al. [17] use a custom-built wearable motion capture system, consisting of flex sensors and photo detectors. To convert motion capture data to joint angles, an approximation model is developed by curve fitting of 3rd order polynomials. Koenemann et al. [10] present a system that enables a humanoid robot to imitate complex whole-body motions of humans in real time, ensuring static stability when the motions are executed and capturing the human data with an Xsens MVN motion capture system consisting of inertial sensors attached to the body.

The above mentioned methods are limited in the sense that the human needs to wear different type of sensors in order to interact with the robot. This can be avoided with the Kinect sensor. Moreover, the cost of the equipment is declined. Song et al. [18] propose a new approach for robot control in order to give autonomy to the robot and people's subjective initiative. Suay et al. [20] present a new humanoid robot control and interaction interface that uses depth images and skeletal tracking software to control the navigation, gaze and arm gestures of a humanoid robot. The work described here follows these criteria and develops a Kinect sensor-based gesture teleoperation system.

This paper proposes two new software packages to enrich the teleoperation of *NAO*. The software provides functionality to fully teleoperate the robot in real-time, allowing speech-based guidance, gesture-based teleoperation that includes arm motion control. With the mentioned modules, the robot *NAO* is able to maintain speech-based communication with the user, copy the motion of the arms and walk and rotate in all directions. Experiments in the laboratory as well as public performances have been conducted to evaluate the usefulness of the proposed software modules.

II. THE ROBOT *NAO*, THE KINECT SENSOR AND SPEECH RECOGNITION WITH ROS

A. ROS

ROS (Robot Operating System) [13] is a framework for robot software development, and it also provides operating system-like functionality on an heterogeneous computer cluster. The aim of ROS is to be a system that combines some useful elements. These elements are drivers and algorithms (such as navigation algorithms, control algorithms for robotic arms, etc.). The system is based on a modular concept. Modules are named nodes in ROS and nodes communicates via *topics* or *services* following a publisher/subscriber protocol.

B. ROS for NAO

One of the robots for which ROS is available, although in a rather limited way, is *NAO*. *NAO*'s human like shaped body is about 58 cm tall and weights about 4,8 kg. It is built in polycarbonate and abs (a common thermoplastic) materials

*This work was supported by the Basque Government Research Team Grant (IT313-10), SAIOTEK Project SA- 2013/00334 and the University of the Basque Country UPV/EHU (Grant UFI11/45 (BAILab))

[1]Authors are with Faculty of Informatics, Computer Science and Artificial Intelligence, University Basque Country (UPV/EHU), San Sebastian, igor.rodriguez@ehu.es

that allow better resistance against falls and it has a lithium battery with which it can get an autonomy of 90 minutes approximately. Its heart is composed by a 1.6 GHz Intel Atom processor running Linux. 25 servos enable to control the 25 degrees of freedom of the robot. Regarding to robot motion, *NAO* can move in any direction (omnidirectional walking), it uses a simple dynamic model (linear inverse pendulum) and quadratic programming. It is stabilized using feedback from joint sensors. This makes walking robust and resistant to small disturbances, and torso oscillations in the frontal and lateral planes are absorbed. It can walk on a variety of floor surfaces, such as tiled and wooden floors, and he can transition between these surfaces while walking.

`nao_robot` is the name of the metapackage used to control the robot. It was developed by the University of Freiburg and it is available on the ROS web. ROS can be run on the robot, or remotely, sending appropriate actions and reading information from *NAO* through a wifi. The last option is the alternative used in this work.

C. Kinect and ROS OpenNI tracker

The Kinect sensor is a horizontal bar connected to a small base with a motorized pivot, and is designed to be positioned lengthwise above or below the video display. The Kinect consists of three different sensors (a RGB camera, a depth sensor and a multiarray microphone) that work together to create the experience of a natural user interface [9].

ROS has several packages that allow to work with the Kinect. The `openni_tracker` package detects when a person gets into the scene captured by the Kinect and tracks the position of his/her head and limbs afterwards. This package makes use of the OpenNI (*Open Natural Interaction* framework.

D. Speech recognition and ROS `gspeech`

Google Speech [19] is an automatic speech recognizer (ASR), freely available, used in several Google applications such as "Google Voice" or "Google Now". It is a cloud based service in which a user submits audio data using a HTML POST request and receives as reply the ASR output in the form of an n-best list. The user only can customize the number of hypotheses returned by the ASR, specify the language used and enable a filter to remove profanities from the output text. The service returns only the final hypothesis and their corresponding confidence level. Google's speech engine has the advantages of being speaker independent and that no training is required.

ROS provides a package based on the *Google Speech* tool and dependant on SOX[1], named `gspeech`. It has a single node that is in charge of:

1) Sound capture: when the user starts speaking, the sound captured from the microphone is recorded by SOX and saved into an audio file.
2) Speech to text conversion: the audio is sent to the Google's server, which in turn will process it and return the speech content and the confidence value of the recognition.

[1]A sound processing program that can convert various formats of audio, apply various effects to those sound files, and also, can play and record audio files on most platforms

III. Speech based teleoperation in Basque

Verbal communication should be a natural way of human-robot interaction. It is a type of communication that allows the exchange of information with the robot.

To serve a human being, it is necessary to develop an active auditory perception system for the robot that can execute various tasks in everyday environments obeying spoken orders given by a human and answering accordingly. Several systems have been recently developed that permit natural-language human-robot interaction. Foster et al. [6] propose a human-robot dialogue system for the robot JAST, where the user and the robot work together to assemble wooden construction toys on a common workspace, coordinating their actions through speech, gestures, and facial displays. Wang et al. [22] introduce a human-robot speech system for teleoperating a humanoid mobile robot.

A speech based teleoperation interface should provide the user the possibility to teleoperate *NAO* giving predefined orders. The system also should give feedback to the operator when an instruction is not understood and this feedback should also be verbal. Three elements are identified in an architecture for speech-based teleoperation:

1) The speech recognition system (SR)
2) The text to speech (TTS) system
3) The robot control system

Within the available ROS packages for *NAO*, the `nao_speech` package provides the necessary tools for making *NAO* understand and speak in English. But this package is of no use when another language is required, as it is the case. Our *NAO* robot is supposed to interact in **Euskara** (Basque, a minority language spoken in the Basque Country) and thus, a tool adapted to this requirement was needed.

A. Implementation: The `nao_teleop_speech_eus` package

A new package (`nao_teleop_speech_eus`) has been developed with three different nodes, one for each of the identified elements:

a) `nao_gspeech`: As mentioned before, ROS *gspeech* package gives ASR capabilities to the robot and can be configured for many languages, including Basque. But this package needs some modifications in order to be useful in a real-time teleoperation scenario. These are the introduced changes:

1) When the native `gspeech` runs the Google's speech service, it is executed only once, i.e, when the user starts speaking, the audio is captured and sent to Google. There it is analysed and the text "corresponding" to the received audio is returned with a confidence level; then, the program ends. It could be tedious for the user to run the speech recognition node each time she/he wants to order something to the robot, or each

time she/he receives an error message. Hence, the new node now runs iteratively avoiding the problem of having to launch the node each time the user wants to talk.

2) When the Google's speech service does not recognize the spoken words, it returns an error message and then the node is forced to quit. Now, error messages received from Google's speech service are specially treated. If an error message is received, `nao_gspeech` publishes a `Repeat` message in the `google_speech` topic to advertise the user that his spoken words are not being recognized.
3) The original `gspeech` node only prints the response received, it does not publish any messages or services, so it can not communicate with other nodes. After the modifications, the confidence level of the hypothesis received from *Google Speech* is processed and, if it is lower than a predefined threshold (0.15 for the performed experiments), the response is declined and treated as an error message.

b) `nao_tts_eus`: This is the node in charge of converting the text into speech using the *AhoTTS* tool [11]. That system, developed by the Aholab group in the University of the Basque Country, is a modular text to speech synthesis system with multithread and multilingual architecture. It has been developed for both, Euskara and Spanish languages. The TTS is structured into two main blocks: the linguistic processing module and the synthesis engine. The first one generates a list of sounds, according to the Basque SAMPA code [14], which consists of the phonetic transcription of the expanded text, together with prosodic information for each sound. The synthesis engine gets this information to produce the appropriate sounds, by selecting units and then concatenating them and post-processing the result to reduce the distortion that appears due to the concatenation process. This tool is required for communicating in Basque Language, but it would not be required for English interlocution.

`nao_tts_eus` has a subscriber that receives messages from `nao_teleop_speech` (see bellow) in the `text_speech` topic. When a text message is received, this node converts it into speech (an audio file) and plays the audio over *NAO*'s speakers.

c) `nao_teleop_speech`: This node allows the user to control *NAO*'s movements using several voice commands. The operator, situated in the teleoperation cab (the place where the remote PC is located), gives orders to the robot using a microphone. The robot is able to perform these movements: *Stand up*, *Sit down*, *Move forward*, *Move backward*, *Move left*, *Move right*, *Turn left*, *Turn right* and *Stop*

As we previously said, commands are given in Euskara. With the intention to communicate with the robot in a more natural way, the user has more than one choice for each possible command. That is, if the operator wants the robot to stand up, he can say: "Altxatu", "Tente jarri", "Zutik jarri", etc. Therefore a dictionary of some predefined words has been created, including several synonymous for each command. When the user gives a voice command (it can be a long sentence), the voice is converted to text, and processed afterwards. The system tries to find matches between the dictionary and the text received from Google's speech service. If a match is found, the robot performs the movement corresponding to the received command, otherwise the robot says that it could not understand the order and asks the user to repeat it.

`nao_teleop_speech` is the node in charge of receiving messages from `nao_gspeech`, finding any matches in the predefined commands dictionary and deciding what is the action that *NAO* must perform. It has a subscriber to receive messages that `nao_gspeech` publishes on the `google_speech` topic, and two publishers; one to set *NAO*'s walking velocity according to the given command, and another one to publish the text messages that *NAO* has to say.

In order to test the speech capabilities in a HRI context, we integrated the ASR and TTS modules in our Bertsobot project [1]. The Bertsobot project was showed to the general public in a live performance entitled "Minstrel robot: science or fiction" [2], in wich *NAO* robot showed his verse-improvisation and speech-based communication capabilities.

It must be said that the system, and as a consequence, the speech based teleoperation works when the google's voice server gives the correct answer. The drawback is that Google can disable the service without notice.

IV. Gesture based teleoperation and arm movement

We want to control the robot's walking motion and also manipulate his arms based on the movements the operator makes. Hence, the gesture teleoperation proposed in this work uses all body gestures. Arm movements are used to control the robot arms, i.e. *NAO* replicates the operator's arm movements. Besides, the operator's position (of her/his body) defines the action she/he wants to perform, like move forward or turn left.

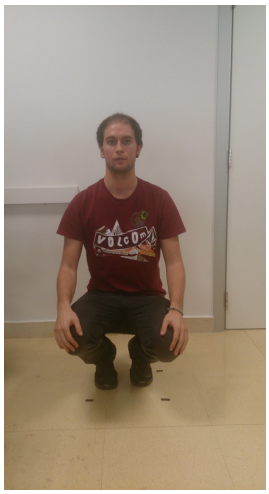
A. Gesture-based teleoperation

The *NAO* teleoperation process has two stages; the operator's calibration stage and the robot control stage. First of all, the operator must perform the calibration pose in front of the Kinect view, this is a necessary step because it verifies that the Kinect is seeing a person. The calibration process is done by `openni_tracker`. When the calibration process is successful, the system is ready to receive gesture commands.

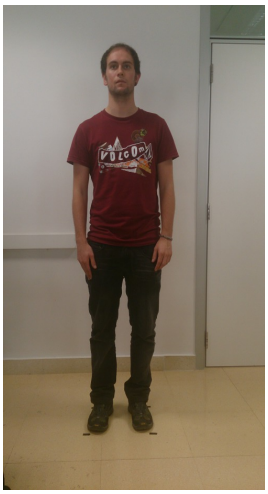
The commands that the robot understands are: *Stand up*, *Crouch*, *Stop*, *Move forward/backward*, *Move left/right* and *Turn left/right*. Figure 1 shows the gestures required to the user in order to command the robot.

Those gestures must be performed from a particular position, since the system is configured for specific user positions and distances (see the black marks on the floor in figure 1). The teleoperation process starts with the operator in crouch

[2]http://www.alhondigabilbao.com/web/guest/detalle-evento/-/journal_content/56_INSTANCE_aJV5/10140/4092781?last-page=/programacion/badu-bada



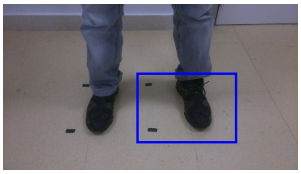
(a) Crouch



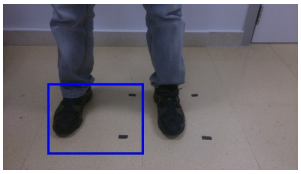
(b) Stand up

(c) Turn left

(d) Turn right



(e) Move left



(f) Move right

(g) Move forward

(h) Stop

(i) Move backward

Fig. 1. Gesture teleoperation commands

position, because this is the position *NAO* rests when it is not being operated. When the operator stands up, the robot stands up too and it is ready to receive moving orders.

B. Arm control

The goal of this process is to give the operator the option to move the robot arms remotely. This can be very useful if the robot is involved in a task where it must carry objects. Human arms have 7 degrees of freedom (DOG): three at the shoulder and wrist, and one at the elbow (yaw). On the contrary, *NAO*'s arms have five DOGs, two at the shoulder (pitch and yaw) and elbow (yaw and roll), and one at the wrist (yaw) (figure 2). Thereby, the movement configurations of human and robot arms differ. For sake of simplicity, we will limit the robot arm movement to raise and extend the arms, imitating the operator's arm motions.

Imitation starts with the perception of the demonstrator; the operator must perform the calibration pose and when it succeeds, the system is ready to perform the imitation process.

For the purpose of imitation, first human movement data have to be acquired, then data are suitably transformed to *NAO*'s coordinate space, and finally, the gesture is executed by *NAO*. We will name this problem as the correspondence problem. This is the main problem of this task.

In order to solve the correspondence problem, the arm is modelled as a stick. The data are acquired by using `openni_tracker` and provide Cartesian coordinates of twenty joints on the human body. The joints tracked by `openni_tracker` are: *Head, Neck, Torso, LeftShoulder, LeftElbow, LeftHand, RightShoulder, RightElbow, RightHand, LeftHip, LeftKnee, LeftFoot, RightHip, RightKnee and RightFoot*. The Kinect treats itself as the origin when providing the Cartesian coordinates, but *NAO* has its own reference system with a different origin. For this reason, Kinect's joint information must be translated.

To transform the Cartesian coordinates obtained from the Kinect into *NAO*'s coordinate space a joint control approach was employed. In this approach, we calculate the human joint angles at shoulders and elbows from Cartesian coordinates using inverse kinematics (as described bellow), and command the robot to set the arms' position at the calculated angles.

Transformation from Kinect's coordinate space to *NAO*'s space: We will focus the explanation on the left arm. The analysis of the right arm is similar and it will be omitted here.

NAO's left arm has four joints (see figure 2): *LShoulderRoll, LShoulderPitch, LElbowRoll, LElbowYaw and LWristYaw*. *LElbowYaw and LWristYaw* joints have not been taken into consideration for the joint control proposed in this work, because the `openni_tracker` package can not detect the operator's arms yaw motion.

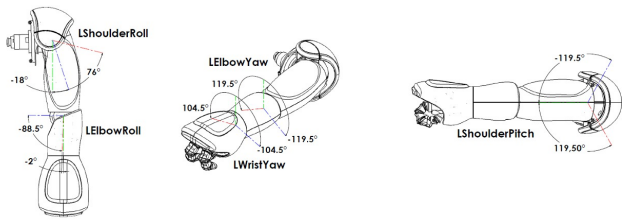


Fig. 2. Left arm motion

In order to calculate the *LShoulderRoll* motion angle ($\alpha_{ShoulderRoll}$) we use the dot product between *LRShoulder* and *LShoulderElbow* vectors, where *LRShoulder* is the vector between the right and left shoulder joints, and *LShoulderElbow* is the vector between the left elbow and left shoulder joints.

Since *LRShoulder* and *LShoulderElbow* vectors are known, we can obtain the angle between these two vectors using the dot product.Note that before applying the equation, *LRShoulder* and *LShoulderElbow* vectors must be normalized. The $\alpha_{ShoulderRoll}$ angle can be calculated as follows:

$$\alpha_{ShoulderRoll} = arccos(LRShoulder \cdot LElbowShoulder) \quad (1)$$

The $\alpha_{ShoulderRoll}$ angle is calculated in the Kinect's coordinate space, therefore, it must be transformed into *NAO*'s coordinate space by rotating it $\frac{-\pi}{2}$ radians. Figure 3 shows both coordinate frames and the transformation needed.

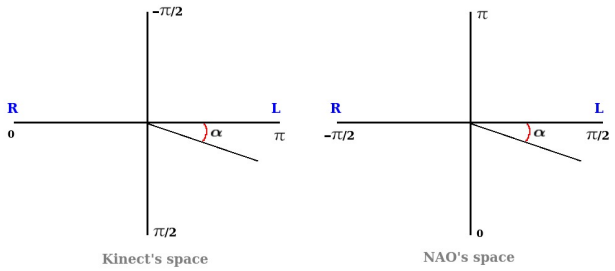


Fig. 3. Left shoulder roll motion in Kinect and *NAO* spaces

LElbowRoll motion angle is calculated in the same way as *LShoulderRoll* motion angle. But now, to calculate the $\alpha_{ElbowRoll}$ angle the vectors we need to consider are *LShoulderElbow* and *LHandElbow*, where *LShoulderElbow* is the vector between the left shoulder joint and the elbow and *LHandElbow* is the vector between the left hand joint and the elbow.

Again, $\alpha_{ElbowRoll}$ angle must be transformed to *NAO*'s space, in this case by rotating it $-\pi$ radians.

And finally, the *LShoulderPitch* motion angle, which is calculated taking into account only the height the *LShoulderElbow* vector takes with respect to z axis. When the arm is extended making an angle of 90 degrees with the torso, it is considered on the z=0 axis. Therefore, if you raise the arm from this position the angle will be positive, and if you lower it, negative (see figure 4).

After normalizing the *LShoulderElbow* vector, the $\alpha_{ShoulderPitch}$ angle for *LShoulderPitch* motion can be defined as:

$$sin(\alpha_{ShoulderPitch}) = \frac{\|A\|}{\|LShoulderElbow\|} = \frac{\|A\|}{1} \quad (2)$$

$$\|A\| = z_{LShoulderElbow} \quad \text{(by definition)} \quad (3)$$

$$\alpha_{ShoulderPitch} = arcsin(z_{LShoulderElbow}) \quad (4)$$

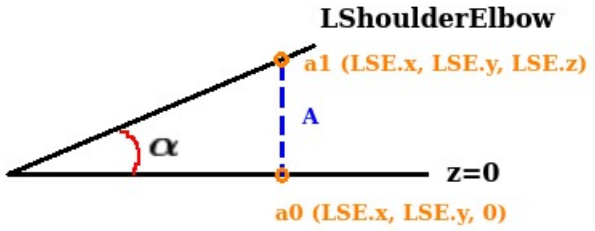


Fig. 4. Left shoulder pitch motion in the Kinect space

where $z_{LShoulderElbow}$ is the Z coordinate of `LShoulderElbow`, represented as $LSE.z$ in figure 4. To transform the $\alpha_{ShoulderPitch}$ angle to *NAO*'s space it must be rotated $2\pi - \alpha_{ShoulderPitch}$ radians.

C. Implementation: The `nao_teleop_gesture` package

A new package named `nao_teleop_gesture` has been developed in order to achieve the gesture-based teleoperation system. This new package contains two nodes:

- `nao_motion_control`: it has to perform the following two main tasks:
  1) Receive messages published by the `openni_tracker` package.
  2) Publish *NAO*'s walking velocities.

  So for, the `nao_motion_control` node contains a subscriber that receives messages published by `openni_tracker` in the `skeleton` topic. When a message is received, the operator's position is analyzed. And according to that position the robot performs the corresponding walking motion. For example, if the message received indicates that the user took a step forward, the robot starts walking forward. Although each gesture indicates a movement order, different gestures can be combined. i.e. combining move forward and turn left gestures the robot can move forward rotating to the left at the same time. Only walking action movements (forward, backward, left, right) with rotational motions can be combined.
  On the other hand, the `nao_motion_control` node has a publisher that publishes the omnidirectional velocity (x, y, and theta) for the walking engine. The velocity with which the robot moves has been set to a constant value. If the walking velocity is too high the robot starts to swing. Thus, the linear velocity and the angular velocities have been assigned low values. The walking velocity of the robot is published in the `cmd_vel` topic.
- `nao_arm_control`: This node is in charge of sending to the robot the necessary motion commands to replicate the operator's arms motion. The node performs tasks by:
  1) Receiving the messages published by the `openni_tracker` package.

2) Publishing *NAO*'s joint angles with speed.

Therefore, `nao_arm_control` is subscribed to the `skeleton` topic in order to receive the operator's skeleton messages published by `openni_tracker`.

On the other hand, the `nao_arm_control` node has a publisher that publishes the joint angles with speed in the `joint_angles` topic, which allows the communication with the `nao_controller` node. The *NAO*'s joints motion speed is set to a constant value appropriate for the robot to mimic the operator arms motion in "real" time.

The robot imitates human actions in real-time with a slight delay of less than 30 milliseconds. This delay is approximately the time the system needs to capture the operator's arms motion, calculate the angles that make up the operator's arm joints, and send motion commands to the robot via Wi-Fi.

It must be mentioned that the native `nao_robot` package has been enriched to provide ultrasound information from *NAO*'s chest and `nao_teleop_gesture` now can use that information, to alert the user when *NAO* gets too close to obstacles in front.

V. System evaluation: Mimicking behavior

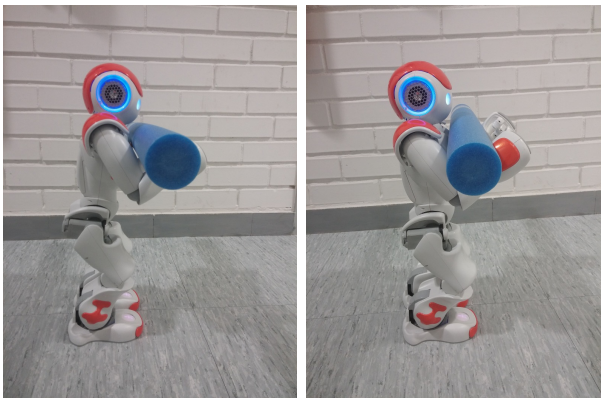
Imitation is an important way of skill transfer in biological agents. Many animals imitate their parents in order to learn how to survive. It is also a way of social interaction. A sociable robot must have the capability to imitate the agents around it. In a human society, people generally teach new skills to other people by demonstration. We do not learn to dance by programming, instead we see other dancers and try to imitate them. Hence, our artificial partners should be able to learn from us by watching what we do.

In order to evaluate the developed system we propose a set of experiments based on imitation. The speech-based teleoperation was not included in the experiments due to some problems during the transition to version 2 of the *Google Speech Server*.

The mimicking behavior consists of the arm control behavior together with the teleoperation behavior. The walking behavior of the robot has been modified for those cases when the robot has something on its arms. The center of gravity (COG) of the walking robot has been lowered and backwarded so that the COG is maintained within the support polygon (see figure5).

The process runs as follows. *NAO* starts in crouching position and when the operator enters the Kinect's view, the calibration process starts. *NAO* tells the operator that the calibration ended successfully saying "Kinect control enabled" in Basque and then, the operator can control the robot with his/her body.

A GUI has been created (with existing *rqt* plugins) in order to help the operator to know the system state. It can be divided into two main parts (figure 6). The top side is composed by the *Topic Monitor*and the *Rviz* interface. The *Topic Monitor* shows all the topics and messages sent by the nodes that are in execution. *Rviz* shows the *NAO* 3D model moving in real-time. The bottom side shows visual information from the cameras; *Image View* shows the image received from *NAO*'s top camera and the right window shows the image captured by the Kinect together with the skeleton of the tracked body.



(a) Original (b) Modified

Fig. 5. Modified walking position

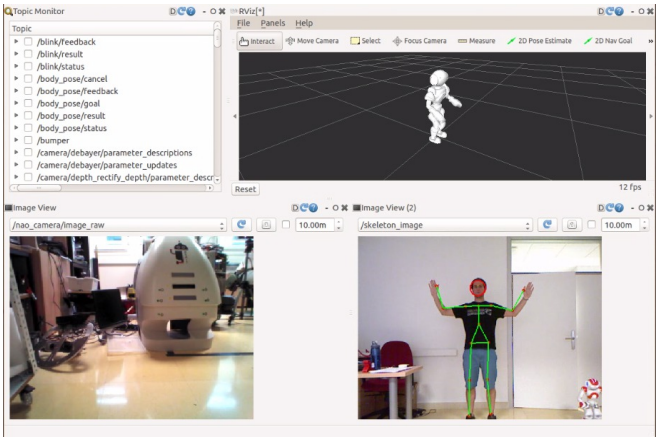


Fig. 6. Teleoperation display

Two experiments were defined. Those experiments involved several people that should give qualitative measures of the system performance by means of a questionnaire that participants completed after carrying out each experiment. Experiments were performed until each participant achieved the aim of the experiment at least once. To run the experiments a package named `nao_imitation` has been created with just a launch file that run both nodes in the `nao_teleop_gesture` package and the `rqt` GUI interface. All the code is available at `github.com/rsait/NAO_ROS`.

Experiment #1: In this first experiment, the operator has to drive the robot in the circuit formed by a series of obstacles using a predefined sequence of movements that included all the movements available in the system: walking, lateral displacement and rotations. Figure 7 shows the circuit proposed, together with the movements that must be performed.

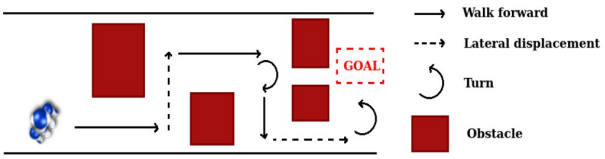


Fig. 7. Test environment for experiment #1

The aim of this experiment was to test if the control of the robot could be carried out with accuracy in reduced spaces, and the adequateness of the robot's view about the environment for the operator to properly teleoperate the robot. It must be emphasized that during the experiments users were not in the same environment as the robot, but in a different lab with a monitor where the GUI displayed the scene view of the robot. Moreover, the only knowledge about the scenario given to the user was the schema shown in figure 7.

Table I shows the results of questionnaires completed by the participants.

User	Goal reached	Attempts	Precision of movements	Difficulty
1	Yes	2	3	4
2	Yes	2	3	3
3	Yes	2	4	4
4	Yes	3	5	3
5	Yes	1	5	2
Total	5/5	2	4/5	3.2/5

TABLE I
RESULTS OF THE FIRST EXPERIMENT

As shown in table I, all participants successfully reached the goal. Most participants needed more than one attempt to complete the experiment. Some of them failed in the first attempt because the perception of the environment was not good and they were not able to avoid all the obstacles. More specifically, the lack of side view difficults the guidance of the robot.

On the other hand, despite the difficulty of the experiment, most participants believed that the selection of gestures is correct (natural) and the movements are quite precise. We can conclude that the system requires a short period of training for the operator to get used to the distances.

Experiment #2: In the second experiment, the operator must operate the robot to transport an object from one place to another. To do it, the operator must drive the robot towards the person who will give it the object that has to take in its arms. Once the robot has taken the object, the operator must send the robot towards the other person who will collect the object. The aim of the experiment was to test the movements of the robot arms while performing a simple task in cooperation with humans. Figure 8 shows the test environment of this experiment.

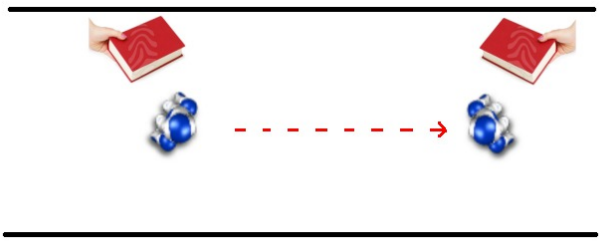


Fig. 8. Test environment for experiment #2

Table II shows the results of the second experiment obtained in the questionnaire completed by the participants.

User	Goal reached	Attempts	Precision of movements	Precision of arms	Difficulty
1	Yes	2	5	5	2
2	Yes	4	5	2	3
3	Yes	2	4	5	2
4	Yes	1	5	5	1
5	Yes	1	4	5	1
Total	5/5	2	4.6/5	4.4/5	1.8/5

TABLE II
RESULTS OF THE SECOND EXPERIMENT

Even though according to table II all participants successfully reached the goal, some needed more than one attempt to complete the aim of the experiment; two of them failed at least once because the robot lost balance when walking with the arms raised, and the other one failed because the Kinect did not detect well (for unknown reasons) the participant's left arm movements.

VI. CONCLUSIONS AND FURTHER WORK

The main contribution of this paper is a set of freely available ROS packages that serve for teleoperating a humanoid robot. A speech based teleoperation package has been developed for Basque, but it can be easily available to other languages as long as *Google Speech Service* provides speech recognition for it. Right now it supports about 35 languages[3]. A new node has been created taking as reference the native `gspeech` ROS package that allows to command the robot verbally repeatedly in spite of network errors or misunderstood commands. Even though the used TTS is not as general as the ASR system, the node can be adapted by changing the code to use a different TTS system's API.

On the other hand, the gesture teleoperation system together with the user interface allows the robot to imitate arm movements and act accordingly to predefined body movements made by the operator at the same time. We are now working on some improvements such as letting the operator to dynamically adjust the velocity of the walking commands (till now, the velocities were constant and fixed

[3]http://en.wikipedia.org/wiki/Google_Voice_Search

experimentally). New movements are also being studied such as that of the head. Allowing the user to control the head position would give the user the chance to take lateral views without moving the robot, relieving the effect of the narrow view of the camera (video of the ongoing progress at http://www.sc.ehu.es/ccwrobot).

Something that remains to be done in the arm control is the yaw movement of *NAO*'s elbows and hands. *NAO* can rotate its arms from the elbows, also its hands from the wrists. A future solution to this problem could be based on the work done by Cole, Grimmes and Rao [3]. They propose a system able to learn full-body motions from monocular vision. In order to detect body motions they use different colors to identify different body parts. Considering the idea of identifying body parts with different colors, the top and bottom sides of the arms and hands could be marked with different colors, enabling to distinguish the palm and the back of the hands.

With respect to the usefulness of the system, we are offering a tool that can have many applications beyond pure teleoperation. Within the area of socially assistive robotics, i.e. robots that assist through social interaction [4], many authors agree that humanoid robots can help assisting patients with disabilities[16], or in pediatric therapy [8][2] because children tend to accept robots in a more natural way than adults. In [21] authors present an experience using the Nao humanoid robot in a role of a physiotherapist for rehabilitation and prevention of scoliosis in children. Children are motivated to replicate robots' movements as a therapy. The developed software could help the therapists without programming skills to record the exercises on the robot. Our next step will be to provide a tool for recording or learning sequences of poses from the mimicked movements.

REFERENCES

[1] A. Astigarraga, M. Agirrezabal, E. Lazkano, E. Jauregi, and B. Sierra. Bertsobot: the first minstrel robot. In *Human System Interaction*, pages 129–136, 2013.

[2] T. Belpaeme, P. Baxter, R. Read, R. Wood, H. Cuayhuitl, B. Kiefer, S. Racioppa, I. Kruijff-Korbayov, G. Athanasopoulos, V. Enescu, R. Looije, M. Neerincx, Y. Demiris, R. Ros-Espinoza, A. Beck, L. Caamero, A. Hiolle, M. Lewis, I. Baroni, M. Nalin, P. Cosi, G. Paci, F. Tesser, G. Sommavilla, and R. Humbert. Multimidal child-robot interaction: Building social bonds. *Human-Robot Interaction*, 1(2):33–53, 2012.

[3] J. C. Cole, D. B. Grimes, and R. P.N. Rao. Learning full-body motions from monocular vision: Dynamic imitation in a humanoid robot. In *Intelligent Robots and Systems (IROS)*, pages 240–246, 2007.

[4] D. Feil-Seifer and M-J. Matarc. Defining socially assistive robotics. In *9th International Conference on Rehabilitation Robotics*, pages 465–468, 2005.

[5] T. Fong, I. Nourbakhsh, and K. Dautenhahn. A survey of socially interactive robots. *Robotics and autonomous systems*, 42(3):143–166, 2003.

[6] M. E. Foster, M. Giuliani, A. Isard, C. Matheson, J. Oberlander, and A. Knoll. Evaluating description and reference strategies in a cooperative human-robot dialogue system. In *IJCAI*, pages 1818–1823, 2009.

[7] M. A. Goodrich and A. C. Schultz. Human-robot interaction: a survey. *Foundations and trends in human-computer interaction*, 1(3):203–275, 2007.

[8] Ayanna M. Howard. Robots learn to play: Robots emerging role in pediatric therapy. In *26th International Florida Artificial Intelligence Research Society Conference*, 2013.

[9] S. Kean, J. Hall, and P. Perry. *Meet the Kinect: An introduction to programming natural user interfaces*. Technology in action, 2011.

[10] J. Koenemann and M. Bennewitz. Whole-body imitation of human motions with a nao humanoid. In *HRI*, pages 425–426, 2012.

[11] I. Leturia, A. Del Pozo, K. Arrieta, U. Iturraspe, K. Sarasola, A. Ilarraza, E. Navas, and I. Odriozola. Development and evaluation of anhitz, a prototype of a basque-speaking virtual 3d expert on science and technology. In *Computer Science and Information Technology, 2009. IMCSIT'09*, pages 235–242, 2009.

[12] D. Matsui, T. Minato, K. F. MacDorman, and H. Ishiguro. Generating natural motion in an android by mapping human motion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3301–3308. IEEE, 2005.

[13] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *Open-Source Software workshop of the International Conference on Robotics and Automation (ICRA)*, 2009.

[14] SAMPA. http://bips.bi.ehu.es/sampa.html.

[15] A. Setapen, M. Quinlan, and P. Stone. Beyond teleoperation: Exploiting human motor skills with marionet. In *AAMAS 2010 Workshop on Agents Learning Interactively from Human Teachers (ALIHT)*, 2010.

[16] M. Simonov and G. Delconte. Assesment of rehabilitative exercises by humanoid robot. In *7th Conference on Pervasive Computing Technologies for Healthcare and Workshops*, pages 331–334, 2013.

[17] H. Song, D. Kim, M. Park, and J. Park. Tele-operation between human and robot arm using wearable electronic device. In *17th IFAC World Congress*, pages 2430–2435. IFAC, 2008.

[18] W. Song, X. Guo, F. Jiang, S. Yang, G. Jiang, and Y. Shi. Teleoperation humanoid robot control system based on kinect sensor. In *4th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, volume 2, pages 264–267. IEEE, 2012.

[19] Google Speech. https://www.google.com/intl/es/chrome/demos/speech.html.

[20] H. B. Suay and S. Chernova. Humanoid robot control using depth camera. In *6th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 401–401. IEEE, 2011.

[21] M. Vircikova and P. Sincak. Experience with the children-humanoid interaction in rehabilitation therapy for spinal disorders. In J. H. Kim, E. T. Matson, and H. Myung an P. Xu, editors, *Robot Intelligence Technology and Aplications*. Springer, 2013.

[22] B. Wang, Z. Li, and N. Ding. Speech control of a teleoperated mobile humanoid robot. In *IEEE international conference on automation and logistics*, pages 339–344. IEEE, 2011.