Human interface to a humanoid robot

The authors (Removed for Blind Review)

Abstract— **This study involves creating a human interface for a humanoid robot by a combination of several interface components. Movements of the human operator are translated into robot movements with the aid of inverse kinematics. This translation cannot be done one-on-one, due to the difference in build between operator and robot and the limits of the interface components. The robot that is used is the Aldebaran Nao. Interface components that are used are the Xbox 360 Kinect, Vuzix virtual reality glasses and Wii remotes. The robot is controlled through a server/client-connection, where the server receives commands and synchronizes the commands from the different channels. The result is a humanoid robot, which can be controlled in a natural way to handle objects, demonstrated with a Towers of Hanoi inspired task.**

I. INTRODUCTION

The goal of the study is to let a human operator control a robot, without using a joystick, i.e. using gestures or motions. The robot mimics the motions of the user, through the use of a depth camera, the Microsoft Kinect. Software that works with the Kinect can extract a wireframe of a human body when someone is in front of the Kinect. This wireframe is then used to find angles of the various joints of the user and map these to the joint angles of the robot through inverse kinematics. Visual information from the robots cameras is sent to virtual reality glasses. These glasses have a gyroscope giving the user the opportunity to control the head of the robot simply by looking around.

The study will be successful when the Nao can mimic the users motions and visual information can be seen through the virtual reality glasses. The goal can thus be rephrased as 'Controlling a robot using a human interface'. The implemented system is a basic version of the human interface: Controlling of robots or other machinery through motion instead of buttons. The Nao is relatively small which means that it cannot be used for all house-holding tasks. In addition, its hands are not strong and cannot be used to lift heavy objects. As the used robot is a biped humanoid robot, care has to be taken to not bring the robot out of balance during a manipulation task. The great benefit of this robot is its resemblance to a human; the joints can easily be indentified with their human counterparts. Commands for the robot are intuitive and are easily replaced by commands for another humanoid robot. All programs connected to the system will be able to control the robot as long as protocol for the commands is followed. This system is easily universally applicable.

The institute (Removed for Blind Review)

II. RELATED WORK

Telepresence allows a user with remote presence, which can be useful in applications like teleconferencing, bomb disposal and remote surgery. Telepresence lets the user experience a remote environment and act upon it. One way of reaching this, is raising the situation awareness of the human operator [1], [2]. It has been demonstrated that the use of virtual reality glasses enhances the situation awareness [3], [4]. The immersion into the virtual reality can be quite complete, as illustrated with the anecdote of [3]: operators tend to jerk their feet away when the robot drops something. Another example of the immersion is the motion sickness-phenomenon reported by [4]. Using multiple interface components to control the robot is a prerequisite for [2], because their Robonaut robot has 45 degrees of freedom. The humanoid robot used in this study, Aldebaran's Nao, has 25 degrees of freedom.

Controlling a humanoid robot with human movements, including gestures and facial movements is an active field of research [5], [6]. Suay and Chernova [5] use specific gestures to start certain behaviours on a Nao robot. One of the behaviours is the controlling of the gaze direction, another is the mimicking of the movements of the human operator. Those behaviours are not combined, because the depth camera cannot detect the gaze direction of the human. The approach used in this article this is solved using the sensors inside the VR glasses. Broccia, Lisevu and Scateni [6] build a humanoid torso on top of a wheeled platform. The head cannot be turned, so the gaze direction is no issue here. The torso of the robot is controlled by mimicking the movements of the human upper body; the wheeled platform is controlled by gestures of the human lower body. Their method to determine the angles between the human joints is equivalent to our approach, although they use distances, while our method uses 3D-vectors.

The Tower of Hanoi puzzle has been used as benchmark for HRI experiments before [7], [8]. Wainer *et. al.* [7] used a wheeled robot to investigate the state of the puzzle and gave instructions to a human to manipulate the disks. Koenig *et. al.* [8] uses the Tower of Hanoi puzzle with a Willows Garage PR2 robot doing the manipulation, based on instructions from a human operator. A result of this study is the difference in completion time when the operator can see the robot motions directly and when the operator can only monitor the movements via the user interface.

III. PLATFORM

In this study several different hardware components are used, these are listed below.

Fig. 1. The components our system uses, from left to right and top to bottom: The Aldebaran Nao, the Xbox Kinect, the Vuzix VR glasses and the Wii remotes.

A. The robot: The Aldebaran Nao

The robot that is used is the Aldebaran Nao, Academic version (see Figure III). The RoboCup edition Version 3.3 was used as well. The difference between the two being the first has control over its wrists and fingers, which were needed for the mimicking of the hands. The driver software for the Naos was Naoqi 1.10.10.

B. The Xbox Kinect

In order to track the body of the human user an Xbox Kinect (see Figure III) was used. When a user is calibrated the Kinect maps a wireframe to the different joints of the user. Useful output can be derived from this wireframe, such as vectors pointing from one joint to another. The drivers used for the Kinect were in the OpenNI library[1]. The Kinect is able to track a whole body, however, it is only used to track the arms (shoulders and elbows). The output of the Kinect can be a rotation matrix of the joints in respect to the Kinect. This means the Kinect knows the orientation of each joint (in respect to its own coordinate system). The Kinect is also able to estimate the position of all the joints (if they are visible, otherwise it will try to make an educated guess. If even that fails, the position is unknown). This position is relative to the position of the Kinect.

C. The VR-Glasses

Due to the fact that tracking the head is difficult for the Kinect and the orientation of the head is not provided, virtual reality glasses with accelerometers were used to get the orientation of the head. The glasses used are the Vuzix VR920 (see Figure III). Moreover, images can be sent from the cameras of the Nao to the glasses, enabling the user to see what the robot sees. The SDK from Vuzix (Version 3.0.7) was used to retrieve the orientation of the users head.

D. The Wii Remotes

The Kinect is performs poorly at determining the lower part of the arms, namely the wrists and hands. In order to be able to mimic the hands another piece of hardware was needed, the Wii Remotes (see Figure III). The Wii Remote, or informally Wiimote, has several buttons and a built-in accelerometer. The orientation of the hands can be derived from this. Another possibility is to use buttons on the remote to control the Nao (for example, open or close hands). Other sequences are also initiated through the Wiimotes, such as walking or bending the knees. The software used for the Wiimotes is 'WiiYourself!' version 1.15.

[1]Stable version 1.3, available at http://www.openni.org/

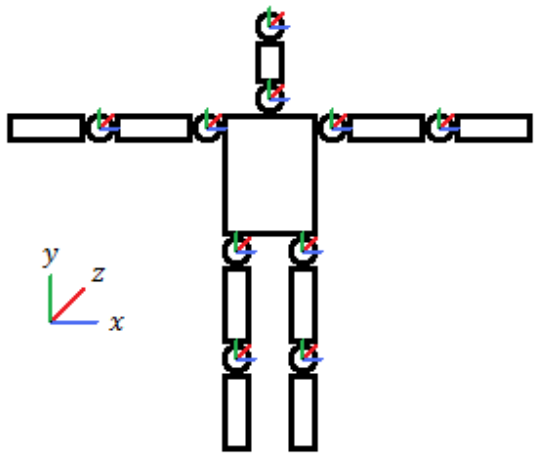

Fig. 2. The joints as the Kinect sees them. This pose is the zero-orientation, i.e. the rotation of all joints is zero. Courtesy NITE project.

IV. DESIGN ISSUES

The task was divided into several subtasks. These subtasks could all be solved independently.

A. Mimicking

One of the main challenges is the mimicking of the joints by the Nao. Firstly, the context of this task will be explained in more detail, then, the tried approaches are described.

1) The task: Each pose the user makes involves angles for the different joints of the user. These angles can be found using the Kinect and the gyrometer in the VR glasses. However, there is a complication: the proportions of the Nao are somewhat different than those of humans. Moreover, the Nao has less degrees of freedom. For example: while humans have three degrees of freedom in their shoulder (yaw, pitch and roll), the Nao only has two degrees of freedom (yaw and pitch). This means that different angles for the joints of the Nao have to be used to reach the same pose of the hand. Finding those joint angles is known as inverse kinematics.

2) Approach: Several approaches are studied, although three approaches were studied in detail.

Euler Angles

The first approach was to calculate the Euler angles from the rotation matrix the Kinect returns. The rotation matrices from the Kinect are in respect to the coordinate system of the Kinect itself, so it was necessary to rotate these back, in order to make all rotations in respect to the Nao. After this transformation, the rotation matrices of the joints were known, all in respect to the coordinate system of the Nao. Now, the Euler angles can be derived from the rotation matrix. As it turns out, there are always multiple solutions to the problem, i.e. one rotation matrix can lead to several

Euler angle combinations. In the worst case there is a infinite number of solutions, this situation is referred to as a gimbal lock [9]. Luckily, there are several solutions to this problem [10]. Unfortunately, the gimbal lock is not the only problem. Another problem is that the Nao does not have the same number of degrees of freedom as humans. Therefore, a rotation that is quite easy for a human might be impossible for the Nao. For the Nao the rotation matrix of the shoulder reduces to the following matrix, because an rotation around the x-axis is not possible ($\sin\psi = 0$):

$$\begin{bmatrix} \cos\theta\cos\phi & -\sin\phi & -\sin\theta\cos\phi \\ \sin\theta\cos\phi & \cos\phi & -\sin\theta\sin\phi \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \quad (1)$$

The human shoulder is a ball-and-socket joint with three degrees of freedom [11], while the Nao has a swing-and-twist joint with only two degrees of freedom. The rotation around the y-axis can be determined from the third row of the rotation matrix (i.e. $\theta = -\sin^{-1}(R_{31})$ or $\theta = \cos^{-1}(R_{33})$). The rotation around the z-axis can be determined from the second column of the rotation matrix (i.e. $\phi = -\sin^{-1}(R_{12})$ or $\phi = \cos^{-1}(R_{22})$). Yet, when the element $R_{32} \neq 0$, the Nao cannot perform any rotation which is defined by this rotation matrix. As an easy way to solve this problem was not found, resulting in the need for a different approach.

'Forward backward'

The forward backward approach also uses rotation matrices, only in a quite different way. The idea is that the position of the end effector (i.e. the hands) is calculated first, then the Nao's built-in functions are used to solve the resulting inverse kinematics problem. The name 'forward backward' is derived from how the values for these angles are found: First the observed human rotation matrices are used to calculate the hands position (starting at the shoulders position). Then the inverse kinematics calculates the angles in the Nao kinematic chain using the hands position. In short: go forward from the shoulder to the hand and then backwards from the hand to the shoulder. This method did work, however, since its only the end-effectors position that is mimicked and not the joints in between, the pose of the arms sometimes does not resemble the users pose. Thus, there was need for yet another approach.

Trigonometry/dotproduct

The method that is currently used does not involve rotation matrices. The positions of the joints are known because of the Kinect. From these positions, vectors are derived and the dot product of these vectors is used to get the angle of the joints.

To illustrate the approach, an example: Assume the roll of the right shoulder is currently unknown (see figure 3). To calculate the roll, firstly, take the positions of the right elbow(p_E), the right hip(p_H) and the right shoulder(p_S) in order to calculate $\angle p_E p_S p_H$. Regular trigonometry could be used to solve this, however, for some joints, it is not possible to create a triangle of joint positions lying in a single plane. For example, the elbow yaw cannot be calculated in a triangular fashion. The elbow yaw is the joint that makes you able to turn you lower arm around the axis of your upper arm. To create a triangle that calculates the elbow yaw, the positions of the elbow, the hand and another joint in that plane that is perpendicular to the upper arms axis are needed. There is no such joint, therefore the angle cannot be calculated through trigonometry. The solution to this problem is the use of vectors, which do not need to have a common point in order to calculate the angle between them. Now for the example, to calculate the elbow roll, the vector pointing from the shoulder to the hip and the vector pointing from the elbow to the hand are needed. The dot product between the two can be used to determine the angle even if the vectors are not in the same plane (where θ is the angle between the two vectors):

$$\vec{a} \cdot \vec{b} = ||\vec{a}|| \cdot ||\vec{b}|| \cos\theta$$

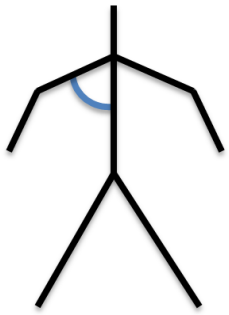

Fig. 3. A stick figure showing the shoulder roll in the Nao

B. Realtime datatransfer to/from Nao

The Kinect, the VR glasses and Wiimote all need to send data to the Nao. This has to be done as fast as possible so the Nao can immediately change the angles of its joints.

1) The task: The idea was to use configuration files or text files to store data locally and then send commands to the Nao based on this data. However, using a TCP connection is a more general solution. Through this connection, information can be sent from server to client or vice versa, in real time. The benefit of using this format is that another problem is solved as well: The two programming languages that are used to control the Nao (Python) and simultaneously get data from the glasses and Kinect (C++) do not have a direct correspondence but both can handle strings. Limiting the messages to string format does not only keep messages relatively small but also makes it possible to send any message from one program to another, given the right protocol.

2) Approach: A program C++ that reads data from the gyroscope in the VR glasses calculates the corresponding angle for the Naos neck joints, which are then sent to it in the format <Jointname><Angle>. The same goes for the program that reads data from the Kinect. Depending on the number of connections, other data is appended to the first incoming string, which is then decomposed in the Python program that also started the server. First, it is checked if

the given angle is possible for the current joint (if the angle is in range of the Naos joint) and if that is true, the Naos joint is set in that angle. The fluency of the movements of the Nao depends on the frequency data is sent.

C. Leg control

Control of the legs of the robot makes it possible to do simple tasks, such as walking around or reaching the ground by bending the knees.

1) The task: Using the Kinect to control the legs through mimicking seemed to be an impossible task due to balance issues. The Nao has less degrees of freedom and the weight/length-ratio of the limbs is fundamentally different from that of a human. Another problem is that humans make use of momentum and falling during both walking and maintaining balance. The Nao cannot do this without being at risk of damaging itself. Although the Nao has two built-in gyroscopes, it is hard to predict the consequences of certain motions without a working simulation. Real-time simulation of motions before performing them is simply not possible for every command the clients send.

2) Approach: Two different approaches have been performed, with varying results.

Gesture

The first idea was to control the legs through gestures. The signal for walking in the direction of the head was pointing forward with the right arm. Since the head is steered by the glasses, the Nao walks in the direction the user looks. A benefit of this approach is that the user always sees where the Nao is going. However, accidental activation is possible. When picking up something in front of the Nao, it can occur that the arms need to be stretched, thus involuntarily satisfying the walking condition. It is also possible that due to strong arm movements and a shifted center of gravity the Nao loses balance, especially while walking.

Wiimote

A slightly better approach would be to use an external device. Apart from making it possible to mimic hands and wrists and opening/closing the hands, there are several buttons on the Wii remote that can be used to steer the Nao. Movements such as walking or bending the knees are useful when wanting to pick something up. A downside is that the use of the remotes does not contribute to the idea of a human interface, since it involves button use. The standard protocol also has to be changed, since the Naos walking movement is steered by a different function entirely. Our protocol has a simple addition of the name of the button followed by a boolean (e.g. LBButton 1 means that the B-button on the left Wii remote is pressed). The server program checks if such a message is sent and if so, executes the corresponding action.

V. SOLUTION ARCHITECTURE

The TCP server that runs on the Nao is the core of the system (see figure V). It can connect to a various number of clients, which can all send data as long as they satisfy the protocol. The messages should always be strings so there is no limit to the possible programming languages, as long as a connection to the server can be made. Duplicate messages are not a problem either, since the program only executes the first command in a message containing duplicate (and thus possibly contradicting) joints and angles. The only constraint is that the buffer for the messages is not of infinite size, a message from a client can, in the current program, not be larger than 4 kilobytes. The slowest client determines how fast the Nao is able to execute given commands, since the server waits for one message from every client before executing the commands.

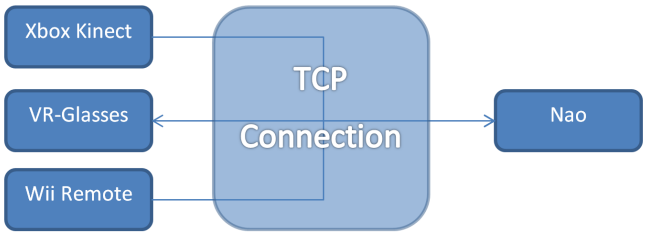

Fig. 4. A visual representation of the TCP Connection. All hardware devices send data to the Nao.

After initiating the program with a number of connections as input for the start-function, clients can connect to the TCP server. Once all three interface components have connected, the Nao will start executing commands from the clients. The first client that was implemented is a program collecting data from the Xbox Kinect. It calculates the angles for the joints of the Nao that correspond to the angles of the users body, as described in section IV.

The second client collects data from the gyroscope in the Vuzix virtual reality glasses. This data is used to determine two angles, a yaw and pitch, for the head of the Nao (the Nao does not have a neck joint for the roll). It also provides visual input for the user, acting as a second screen for the computer it connects to. To create this visual input, we use Choreographe, a program by Aldebaran robotics. The benefit of this program is that it can simultaneously show the camera images and the pose of the Nao in simulation, the latter because the real Naos body is not visible to users with the glasses on.

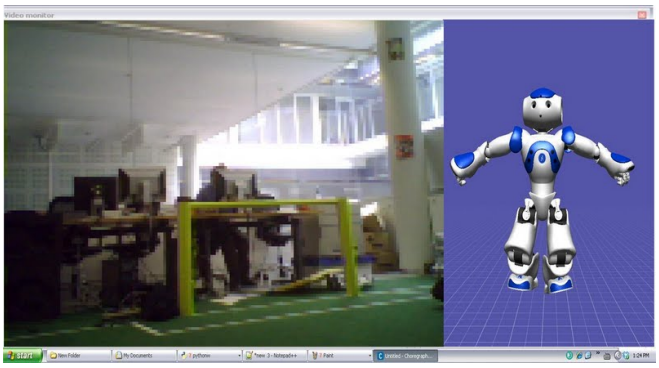

Fig. 5. The image, as seen through the virtual reality glasses.

VI. RESULTS

In the approach several components were enabled to control the Nao.

A. Our results

The components that are currently used (the Xbox Kinect, the Vuzix virtual reality glasses, the Wii remotes, the Aldebaran Nao and Choreograph), the program returns the user a camera view and the pose of the Nao in simulation through the glasses. The Nao itself will mimic the users upper body, which is possible through the use of the Kinect and Wiimotes, though this can only be seen in simulation if the user is wearing the VR glasses. Gestures or buttons on the Wii remotes activate walking of the Nao. This enables the user to execute simple remote operations, such as lifting a box or pushing a button. It should be noted that calibration of the VR glasses is needed before use of the system with all its current components.

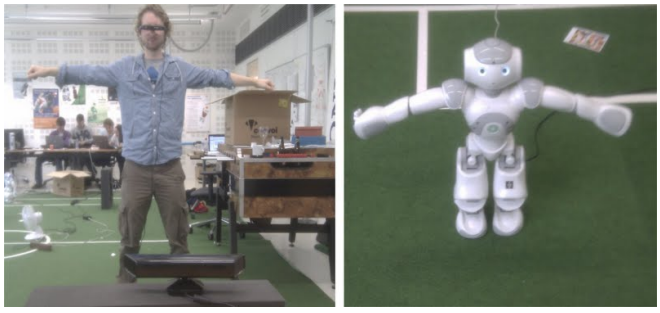

Fig. 6. An example of the system when using the VR glasses and Kinect.

B. Use of this system

The user starts by initiating the server and client programs. The user then puts on the Vuzix virtual reality glasses, holds the Wii remotes and stands in front of the Xbox Kinect in the calibration pose. When calibration is complete, the server program will start receiving commands from the clients. The client program that controls data transfer from Kinect to Nao sends commands for the upper body. The client reading values from the VR glasses gyroscope sends commands for the Naos neck joints. The client sending data from the Wii remotes controls the lower body and enables closing and opening of the hands. The Nao thus starts mimicking the user as soon as commands are sent. If a single client program stops, all clients and the server program will stop as well. Once the server has stopped, clients will stop sending data. An example of the system in action can be seen in figure 5 and 6.

C. General use

The implemented system allows a human user to control humanoid robots through body motions. It does not require the user to lift the legs or walk, only the upper body is mimicked. Requirements for use are, of course, practice but no knowledge of robotics is needed. It is therefore applicable in any area. A few examples of possible use are teleoperation in dangerous areas (e.g. after natural disasters or in warzones), house-holding tasks but also industrial processes. The human interface makes it relatively easy to learn the robot a new task since it can simply mimic the user executing the task.

VII. CONCLUSION

Conclusively, the result of this study is the creation of a human interface to a humanoid robot. This can only be done by solving several issues: maintaining balance, the coupling of human kinematics with the humanoid kinematics and synchronizing the different interface components controlling the robot. Balancing the robot is still an issue because of the differences between human and Nao (humans behave fundamentally different in walking and balancing tasks). This problem was not solved, instead, time was spent on implementing a different method for controlling the Nao's legs.

The control system, consisting of one server program and a number of interface clients (3 clients were chosen in this case because 3 interface components were needed to control all relevant degrees of freedom of the robot), allows smooth teleoperation of the Nao. The server program is exclusively designed for the Aldebaran Nao but this can easily be altered to an equivalent humanoid robot, as long as the control functions are intuitive (e.g. for the Nao setAngles sets angles for given joints, openHand opens the given hand). While searching for information about the used components, it was clear that there were several students and freelancers who had attempted similar approaches [12] [13], though this is the first study which uses VR glasses for gaze control.

REFERENCES

[1] T. Sheridan, *Telerobotics, automation, and human supervisory control*. MIT Press, 1992.

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

[3] S. M. Goza, R. O. Ambrose, M. A. Diftler, and I. M. Spain, "Telepresence control of the nasa/darpa robonaut on a mobility platform," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, ser. CHI '04. New York, NY, USA: ACM, 2004, pp. 623–629.

[4] J. B. van Erp, M. Duistermaat, C. Jansen, E. Groen, and M. Hoedemaeker, "Tele-presence: Bringing the operator back in the loop," in *Virtual Media for Military Applications*. Meeting Proceedings RTO-MP-HFM-136, Paper 9, 2006, pp. 1–18.

[5] H. B. Suay and S. Chernova, "Humanoid robot control using depth camera," in *Proceedings of the 6th international conference on Human-robot interaction (HRI'11)*. New York, NY, USA: ACM, 2011, pp. 401–402, note the video submission at http://dl.acm.org.

[6] G. Broccia, M. Livesu, and R. Scateni, "Gestural interaction for robot motion control," in *Eurographics Italian Chapter Conference*, 2011, pp. 61–66.

[7] J. Wainer, D. Feil-Seifer, D. A. Shell, and M. J. Matarić, "Embodiment and human-robot interaction: A task-based perspective," in *Proceedings, 16th IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN 2007)*, Augustus 2007, pp. 26–29.

[8] N. Koenig, L. Takayama, and M. J. Matarić, "Learning from demonstration: A study of visual and auditory communication and influence diagrams," in *International Symposium on Experimental Robotics (ISER)*, December 2010.

[9] K. Shoemake, “Animating rotation with quaternion curves,” *SIGGRAPH Comput. Graph.*, vol. 19, no. 3, pp. 245–254, July 1985.

[10] G. G. Slabaugh, “Computing euler angles from a rotation matrix,” Georgia Institute of Technology, Tech. Rep., August 1999. [Online]. Available: http://www.gregslabaugh.name/publications/euler.pdf

[11] W. E. Woodson, B. Tillman, and P. Tillman, *Human Factors Design Handbook*, 2nd ed. McGraw Hill, 1992.

[12] E. Berger and H. B. Amor, “Advanced kinect controller for humanized robots,” student project similar to our project. [Online]. Available: http://naoforge.net/naovideoblog/2011/04/28/advanced-kinect-controller-humanoid-robots/

[13] T. Veltrop, “Humanoid navigation teleoperation using nao and kinect,” freelancer working with the Aldebaran Nao and Kinect. [Online]. Available: Seehttp://taylor.veltrop.com/robotics.php