Real-time Teleoperation with the Baxter Robot and the Kinect Sensor

Jose Avalos*† and Oscar E. Ramos*
*Department of Electrical Engineering, Universidad de Ingenieria y Tecnologia - UTEC, Lima, Peru.
† Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM, USA.

***Abstract*—Teleoperation includes the control, operation and manipulation of a remote machine, and can be achieved only if the teleoperator transmits sufficient motion information, and the slave reproduces it in a sufficiently articulated and close way. Although there exist different robot models and motion acquisition systems, classical teleoperation techniques still use complex hardware and external sensors limiting the motion of the human operator. One of the reasons is the lack of a proper system integration with new devices. This work proposes a generic real-time teleoperation framework that is independent of the hardware allowing a simple and fast integration with new technologies. This approach is validated by teleoperating a Baxter robot through the motion acquired with a Kinect sensor without the need of external sensors on the operator. The proposed approach can be further extended to different types of robots for different applications.**

Keywords - Baxter robot, Kinect sensor, inverse kinematics, teleoperation.

I. Introduction

The generation of proper motion for redundant robots has long been a challenge in robotics due to the infinite under-constrained solutions that redundancy offers. Thus, programming a redundant robot to execute a specific task can result in a complex, tedious, and very long process. Part of this problem can be attenuated by allowing the robot to imitate the movements of a human user, which is a process known as *Teleoperation*. Mimicking a human reduces the difficulty of programming anthropomorphic robots and has the advantage of combining natural motion with robot speed and precision. It also allows the interaction with objects and the realization of complex tasks in remote and inaccessible environments [1].

There exist different approaches to teleoperate robots for different fields of application. In [2], a robot was built with a human interface to reproduce the human-robot interaction based on a combination of sensors. For [3], an industrial robot was designed with two arms that follows the user motion employing the first version of the Kinect. Finally, [4] developed a teleoperation framework where the right hand controlled the end-effector position and the left hand executed some commands. However, teleoperation need not be restricted to sensors that provide a visual feedback, and special human joint sensors can be used. For example, in [5] the motion information is obtained from one glove designed with acquisition sensors. One of the problems of existing methods is that they are oriented towards a specific robot, or even robots are designed for teleoperation. Also, most current teleoperation relies on intrusive sensors attached to the human body or on expensive motion acquisition systems.

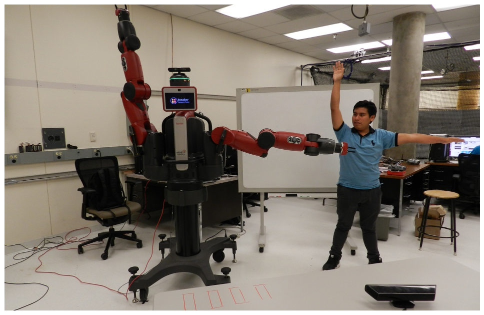


Fig. 1: The Baxter robot being teleoperated by a human in real time. The Kinect sensor obtains the human position and based on inverse kinematics the robot follows the motion.

Contrary to these approaches, this work proposes a generic teleoperation framework where no intrusive physical sensor is used. Therefore, both the user and the robot can execute movements more naturally. Moreover, contrary to other motion imitation techniques, in this work teleoperation is achieved in real time. We based our experiments on the Baxter robot using a Kinect sensor as a motion acquisition system as Fig. 1 shows, but the approach is completely general. This paper is structured as follows: Section II explains the method used to achieve motion imitation, which constitutes the core of teleoperation; Section III presents the details for the implementation of this work; Section IV shows the experimental results, and Section V presents the conclusions of the work.

II. Methodology

Real-time teleoperation of a robot has two steps: first, the motion of the human is captured; and then, the acquired motion undertakes a kinematic re-targeting process that makes it suitable for being reproduced on the robot. This framework, illustrated in Fig. 2 is discussed in this section.

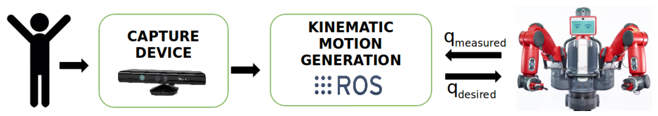


Fig. 2: Scheme showing the parts of the system from the human motion to the humanoid robot execution

A. Data acquisition

The motion data to be used as reference for teleoperation can be acquired with any motion capture system such as infrared cameras, depth sensors (e.g. a Kinect), haptic devices, among others. Our approach deals with the Cartesian coordinates, rather than joint angles, of specific operational points such as the hands and elbows of the human demonstrator. The reason for this choice is twofold: first, it is hard to directly relate human joint angles to the robot joint angles due to kinematic differences; and second, most motion capture systems provide Cartesian coordinates for different points as primary output.

Even if we only consider Cartesian coordinates, we must take into account that the lengths of human bones are different to those of robot links. Therefore, the acquired positions need to be normalized, which is a process that relates the kinematic characteristics of the human to those of the robot. This normalization depends on the specific robot to be used since different robots have different link lengths. The normalized time-varying data will be used as a motion reference to be tracked by the robot.

B. Robot Motion Generation

The processed data coming from the acquisition system needs to be followed as closely as possible, both in time and space, by the robot for the teleoperation to be successful. To this end, a control based on differential inverse kinematics was used. The targets for this kinematic controller are defined as operational space tasks in terms of Cartesian coordinates. Since the robot motion is three dimensional, a full teleoperation can be achieved.

Let the robot joint angles be represented by the joint configuration vector $\mathbf{q} = (q_1, q_2, \cdots, q_n)$, where n is the number of degrees of freedom of the robot. Let the task i be represented by $\mathbf{x}_i$ and its reference by $\mathbf{x}_i^*$. For example, if the task is a Cartesian position, then $\mathbf{x}_i = (x_i, y_i, z_i)$. Using the previous task elements, the task error is defined as

$$\mathbf{e}_i = \mathbf{x}_i - \mathbf{x}_i^*. \tag{1}$$

It should be noted that the task is a function of the joint space such that $\mathbf{x}_i = f_i(\mathbf{q})$, where f_i is the forward kinematics function. The rate of change of the task error is related to the joint angle velocities by the task Jacobian $J_i = \frac{\partial f_i}{\partial \mathbf{q}}$ such that

$$\dot{\mathbf{e}}_i = J_i \dot{\mathbf{q}} - \dot{\mathbf{x}}_i^* \tag{2}$$

which is the time derivative of (1) and represents the definition of a task at the kinematic level. To solve for the joint angular velocities $\dot{\mathbf{q}}$ in (2), given several tasks at the same time, a task weighted approach based on [6] was used. Considering N different tasks, this weighted scheme can be formulated as

$$\min_{\dot{\mathbf{q}}} \left\{ \sum_{i=1}^{N} w_i \| \dot{\mathbf{e}}_i - J_i \dot{\mathbf{q}} + \dot{\mathbf{x}}_i^* \|^2 \right\} \tag{3}$$

where the i^{th} task has an associated weight w_i. This weight is used to set the relative importance for each task. Considering

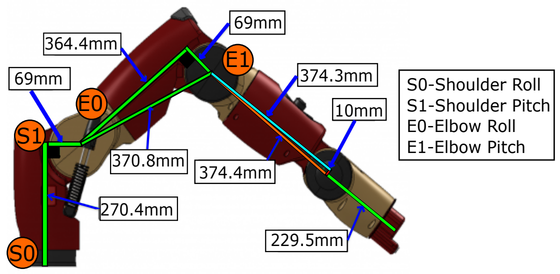


Fig. 3: Baxter arm dimensions

a small reference variation between successive control times (which in practice is true), the term $\dot{\mathbf{x}}_i^*$ can be neglected, and it can be shown that this optimization problem is equivalent to the following quadratic program (QP):

$$\min_{\dot{\mathbf{q}}} \left\{ \dot{\mathbf{q}}^T W \dot{\mathbf{q}} + \dot{\mathbf{q}}^T \mathbf{p} \right\} \tag{4}$$

where

$$W = \sum_{i=1}^{N} w_i J_i^T J_i, \quad \mathbf{p} = -2 \sum_{i=1}^{N} w_i J_i^T \dot{\mathbf{e}}_\mathbf{i}.$$

For the motion to be safely reproduced in a real robot, joint limits need to be taken into account. Provided that the kinematic control is expressed as a QP, the joint limits can easily be added as the following constraint:

$$\max \left\{ \frac{\underline{\mathbf{q}} - \mathbf{q}}{\Delta t}, \underline{\dot{\mathbf{q}}} \right\} \leq \dot{\mathbf{q}} \leq \min \left\{ \frac{\overline{\mathbf{q}} - \mathbf{q}}{\Delta t}, \overline{\dot{\mathbf{q}}} \right\} \tag{5}$$

where $\underline{\mathbf{q}}$, $\overline{\mathbf{q}}$ are the lower and upper joint angular limits, $\underline{\dot{\mathbf{q}}}$, $\overline{\dot{\mathbf{q}}}$ are the lower and upper joint velocity limits, and Δt is the control time between each control signal. The relation between joint velocity and joint position can be found using a first order Taylor expansion.

Using Euler integration, the joint velocities obtained with (4) can be transformed into joint positions as

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \dot{\mathbf{q}}_k \Delta T \tag{6}$$

where $\mathbf{q}_k$ represents the joint configuration at time k. These joint values can then be directly send to the robot.

III. Experimental Setup

A. Hardware for the Experiment

1) Kinect Sensor: The human motion is acquired using a Kinect sensor, which is a low-cost vision controller interface introduced by Microsoft in 2010. It provides a RGB video stream with 30 fps as well as a monochrome in depth map with VGA resolution (640×480 pixels). The depth sensor projects a known IR pattern onto the environment and measures the returned pattern with an IR camera. The differences between the received pattern and the projected one are used to extract the object depth. Some available software for this sensor are *Open NI*, *Libfreenect* (by OpenKinect), and the official *Microsoft Kinect for Windows SDK.*

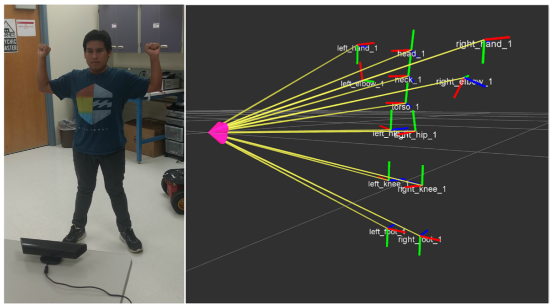


Fig. 4: Example of body representation trough the coordinate frames (TF) using a Kinect sensor directly with ROS.

2) Baxter Robot: This is the robot used to follow the demonstrated movements. It is a compliant dual-arm humanoid torso developed by Rethink Robotics. Each arm has seven degrees of freedom and an installed gripper at the end of the arm. The arm dimensions are shown in Fig. 3. The robot also possesses a head with horizontal motion, but the head was not used for the experimentation. It is worth mentioning that this is a position controlled robot, and therefore no torque information is needed to move the robot and joint angular positions are sufficient.

B. Software Framework

Our approach uses ROS (Robot Operating System), which is becoming a *de facto* standard in robotics, as a middle-ware for interconnecting the proposed system. ROS works under Linux and is the default platform for the Baxter robot. However, the default platform for the Kinect sensor is Microsoft Windows. To solve this problem one possibility is to use the Kinect with its default SDK under Windows and then transmit the data to a machine running Linux, as in [7]. Although there is no official skeleton tracking SDK for Linux, all our system was run on a Linux machine with ROS in order to avoid data transmission delay. To solve this problem and properly acquire the motion information[1], this work uses the following open source libraries: *OpenNI*, *SensorKinect*, and *NITE*. An example of the skeleton acquired with this process is depicted in Fig. 4 where the frames, known as TF in ROS, corresponding to the human parts are shown.

The communication between all the nodes, from the Kinect to the Baxter robot, is done through dedicated ROS topics. This architecture can be implemented in two remote computers to do remote control, depending on the quality of the underlying network. It can be also implemented in a single computer as in our case. The information coming from the user is converted to a TF message which contains position and orientation information for each part of the body. Before being available, this data is multiplied by a rotation matrix and adapted to the dimensions of the robot . The node running the inverse kinematics algorithm reads the published message and computes the desired joint angular positions. These control elements are then published in separated topics that are read by the low-level controllers of each arm of the real Baxter robot. The whole process runs at 100 Hz, which can be considered real time for practical situations involving robotic motion.

[1]We present a tutorial in https:github.com/JoseAvalos/baxter_kinect360

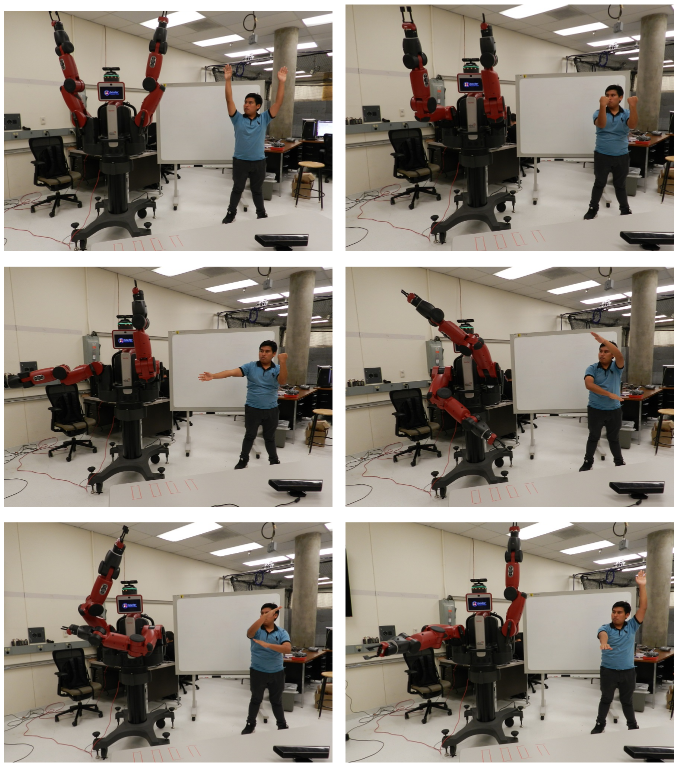


Fig. 5: Example of the results of the three-dimensional telepresence system under different positions

IV. Results

The previously described experimental setup was used to test the validity of the proposed teleoperation approach. Fig. 5 shows the system in different experimental configurations[2]. It can be seen that the human is teleoperating the robot by just moving the arms, without having any attached sensors on the body.

For comparison purposes the motion was performed side by side, but the user can be in a remote location and the robot will still be teleoperated. Fig.6 compares the reference position for one of the robot joints, obtained by rescaling the human motion, with the actual position achieved by the robot. This process shows a short delay due to the inertia of the robot. However, this delay does not cause loss of information from the user and is small for practical robotic demonstrations.

Fig. 7 shows a comparison between the position of the user's left hand and the robot's left end-effector (or, more properly, the robot's left wrist). There is a noticeable offset between both motions as well as a scaling factor. The reason for the scaling factor is that the human arm and the robot arm have

[2] All tests were done in MARHES Lab at *The University of New Mexico*. Videos are available at https://goo.gl/g9epyp

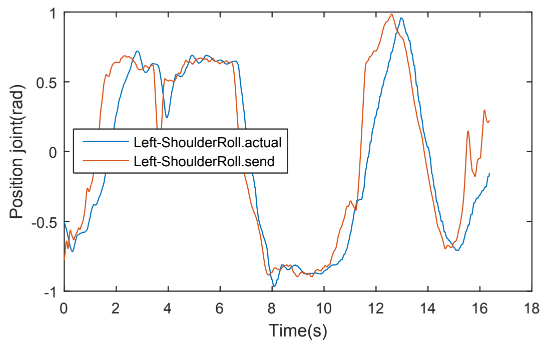


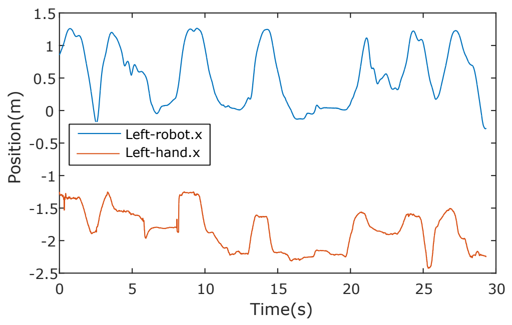
Fig. 6: Comparison between the position sent y the actual position of *Shoulder Roll*.

different dimensions leading to different hand positions for similar motions. The reason for the offset is that the coordinate origins are different for the human and for the robot. In the case of Baxter, the origin is the torso, whereas for the person it is the center of the Kinect frame, which roughly coincides with the center of the sensor. The experimental data shows that the robot follows the motion pattern demonstrated by the user. There is a small tracking delay which, from the experimental data, is less than 0.3 seconds. Since the objective of teleoperation is motion in the sense of human perception, this delay is very small for practical purposes and the system can be considered real time. Note that non real-time motion in robotics usually refers to systems that first acquire the motion, then process it offline (taking minutes or even hours) and finally reproduce them on a robot.

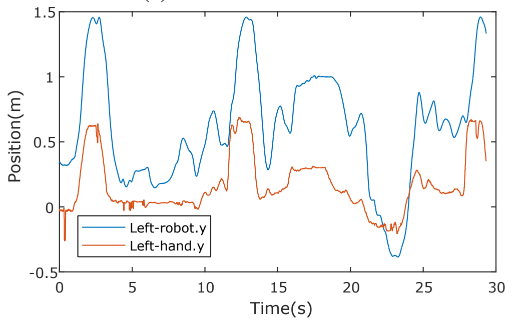
In case of sudden movement (for example in the Z axis between 12 and 20 seconds, where the user abruptly moves the hand), the robot cannot follow the motion with a similar speed due to torque limits, but it attempts to track the motion as closely as possible without affecting its own mechanism. This limitation can be overcome with a different robot design (e.g., [8]). The results for the other robot hand (and elbows) present a similar behavior.

V. Conclusion

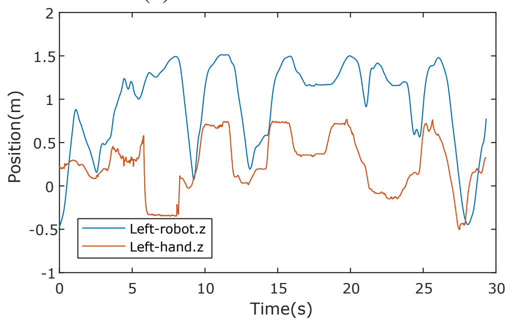
A generic method for the three dimensional teleoperation of an anthropomorphic robot has been presented. The proposed approach was validated using a Baxter robot and Kinect sensor, showing that precise real-time teleoperation can be achieved without the need of complex sensors. This work could be used as a fast implementation tool for emergency cases where we can approach the mechanical characteristic of the robot. This is another step to achieve full telepresence (including video and audio transmission). Future work will involve more body actions in order to better exploit the characteristics of the 7 DOF robot.



(a) Motion in the X axes



(b) Motion in the Y axes



(c) Motion in the Z axes

Fig. 7: Comparison between the motion of the user and the Baxter robot for the left hand.

References

[1] X. Xu, B. Cizmeci, C. Schuwerk, and E. Steinbach, “Model-mediated teleoperation: Toward stable and transparent teleoperation systems,” *IEEE Access*, vol. 4, pp. 425–449, 2016.

[2] Y. Xiao, Z. Zhang, A. Beck, J. Yuan, and D. Thalmann, “Human–robot interaction by understanding upper body gestures,” *Presence: Teleoperators and Virtual Environments*, vol. 23, no. 2, pp. 133–154, 2014.

[3] D. Kruse, J. T. Wen, and R. J. Radke, “A sensor-based dual-arm telerobotic system,” *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, pp. 4–18, 2015.

[4] M. M. Marinho, A. A. Geraldes, A. P. Bó, and G. A. Borges, “Manipulator control based on the dual quaternion framework for intuitive teleoperation using kinect,” in *Robotics Symposium and Latin American Robotics Symposium (SBR-LARS), 2012 Brazilian*. IEEE, 2012, pp. 319–324.

[5] B. Fang, D. Guo, F. Sun, H. Liu, and Y. Wu, “A robotic hand-arm teleoperation system using human arm/hand with a novel data glove,” in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2015, pp. 2483–2488.

[6] O. E. Ramos, “A kinematic whole-body control system for highly redundant robots,” in *IEEE Andean Council International Conference (Andescon)*, 2016.

[7] J. Avalos, S. Cortez, K. Vasquez, V. Murray, and O. E. Ramos, “Telepresence using the kinect sensor and the nao robot,” in *IEEE Latin American Symposium on Circuits & Systems (LASCAS)*, 2016, pp. 303–306.

[8] J. Lanini, T. Tsuji, P. Wolf, R. Riener, and D. Novak, “Teleoperation of two six-degree-of-freedom arm rehabilitation exoskeletons,” in *2015 IEEE International Conference on Rehabilitation Robotics (ICORR)*. IEEE, 2015, pp. 514–519.