Proceedings of Australasian Conference on Robotics and Automation, 3-5 Dec 2012, Victoria University of Wellington, New Zealand.

Teleoperation of a humanoid robot using full-body motion capture, example movements, and machine learning

Christopher Stanton[1], **Anton Bogdanovych**[1] and **Edward Ratanasena**[2]
[1]MARCS Institute, University of Western Sydney
c.stanton@uws.edu.au, a.bogdanovych@uws.edu.au
[2]University of Technology, Sydney
edward.ratanasena@student.uts.edu.au

Abstract

In this paper we present and evaluate a novel method for teleoperating a humanoid robot via a full-body motion capture suit. Our method does not use any *a priori* analytical or mathematical modeling (e.g. forward or inverse kinematics) of the robot, and thus this approach could be applied to the calibration of any human-robot pairing, regardless of differences in physical embodiment. Our approach involves training a feed-forward neural network for each DOF on the robot to learn a mapping between sensor data from the motion capture suit and the angular position of the robot actuator to which each neural network is allocated. To collect data for the learning process, the robot leads the human operator through a series of paired synchronised movements which capture both the operator's motion capture data and the robot's actuator data. Particle swarm optimisation is then used to train each of the neural networks. The results of our experiments demonstrate that this approach provides a fast, effective and flexible method for teleoperation of a humanoid robot.

1 Introduction

Teleoperation is the operation of a machine at distance. An early example of teleoperation was in 1951 when Goertz [1] developed a mechanical master-slave manipulator arm for work with radioactive material. Recent examples include the control of wheeled robots for tasks such as bomb disposal and the Mars Rovers[1]. Advances in robotic hardware have seen the emergence of sophisticated humanoid robots, such as Honda Asimo[2], HRP-4C[3], and Robonaut[4]. Humanoid robots have advantages over robots with non-human morphologies (e.g. wheeled robots) in that their human-like form allows for the robot to take advantage of urban environments designed by people for people. For example, humanoid robots are well suited to using human tools, opening doors, climbing staircases, and so forth. The unstructured nature of the real world, coupled with limitations in artificial intelligence and autonomous robot behaviour, means that teleoperation of humanoid robots is a more suitable approach to robot control for some situations where dexterous and complex movements are required in environments too dangerous for humans (e.g. mining disasters, war zones, chemical spills, nuclear accidents, space exploration, etc). Furthermore, giving a robot a humanoid form allows intuitive teleoperational control due to the similarities in embodiment between the human master and the robot slave. Initiatives such as DARPA's[5] robotic challenge in which humanoid robots are used in disaster and rescue situations highlight the potential benefits of teleoperation of humanoid robots.

In this paper we present a novel system for teleoperating a humanoid robot using a full-body motion capture suit. Our immediate aim is to teleoperate a humanoid robot using full-body motion capture, but without the use of any *a priori* analytical or mathematical modeling (e.g. forward or inverse kinematics). Our longer-term vision is to develop a seamless method for calibrating any human-robot teleoperation pairing, regardless of differences in physical embodiment and independent of the motion capture device and the robot's morphology. Ideally such a system would allow individual users to quickly and easily tailor their idiosyncratic/chosen movements and gestures for accurate and intuitive control of any robotic system.

Traditional approaches to humanoid teleoperation require explicit kinematic modeling of both the robot slave

[1]http://marsrovers.jpl.nasa.gov
[2]http://world.honda.com/ASIMO/
[3]http://en.wikipedia.org/wiki/HRP-4C
[4]http://robonaut.jsc.nasa.gov/default.asp
[5]http://www.darpa.mil/

and master device to provide mappings between human motion capture data and robot actuator commands. As modeling and solving kinematics problems can be laborious, hardware specific and sometimes computationally expensive, our aim is provide a general method of mapping human motions to robots, regardless of the robot's form or the device used for capturing human motion. The novelty of our approach is the use of a machine learning process to calibrate the human master with the robot slave, thus eliminating the need for explicit kinematic modeling of both the robot and the relationship between motion capture data and robot actuator commands. The key benefits of our approach are flexibility and adaptability as calibrating the system with different hardware would simply require a brief retraining period, rather than solving inverse kinematic solutions for the new robot. Furthermore, our approach allows for the human operator to choose movements and poses that they wish to correspond to robot movements and poses.

This paper is structured as follows: Section 2 outlines the problem domain of designing and building user interfaces for teleoperation of humanoid robots. Section 3 presents related literature in the use of motion capture and machine learning for teleoperation tasks. In Section 4, our approach is described. Section 5 highlights our results, and in Section 6 we discuss the implications of our work, and possible future avenues of research and development.

2 Problem Domain

The development of humanoid robots has brought increased research interest in teleoperation. As humanoid robots are bipedal and have a large number of degrees of freedom, the main challenges in teleoperating humanoid robots concern how to best satisfy the operator's desired behaviour for the robot given the (dimensional) differences between the input capture device and the robot, while also maintaining the robot's stability.

Early approaches to teleoperating humanoid robots captured user intention through the use of graphical user interfaces (GUIs), joysticks, buttons and keyboards [2; 3]. Another approach is through the use of purpose-built "cockpits" which feature exoskeleton robot master arms [4; 5; 7] and feet [6] with force feedback. An alternative is to employ a "marionette" system, consisting of a small scale puppet version of the humanoid robot [8] for manipulation by the operator. A drawback of these approaches is the need for purpose built hardware, which in some cases can be large, cumbersome and difficult to transport.

2.1 Motion Capture

Motion capture has had a significant impact on robotics, being used for not only teleoperation but also for improving humanoid locomotion [18] and robot learning from human demonstration [9; 12; 13]. Early approaches used inertial measurement units [10], flex sensors and photo detectors [19], and shape tape [11]. In recent years, many new motion capture products have come onto the market, ranging from the cheap but somewhat limited (e.g. Microsoft Kinect, Nintendo Wii) to the (relatively) expensive but highly accurate Xsens MVN full-body motion capture suit (described in Section 4.1).

2.2 Robot Kinematics

Forward kinematics involves calculating the position of an end-effector in three dimensional space from joint angles (e.g. calculating the position of the hand in XYZ space based upon the values of shoulder, elbow, and wrist joints). Conversely, inverse kinematics refers to the use of the kinematics equations of a robot to determine the joint parameters that provide a desired position of the end-effector.

Teleoperation requires finding kinematic solutions to the physical asymmetries between robot and master. Furthermore, often multiple solutions exist for the robot's inverse kinematic calculation if multiple configurations of the robot's joints can result in the same end-effector position. One approach to solving inverse kinematics is through trigonometry, while another is to use Jacobian based iterative approaches [27]. Both approaches are laborious and changes in robotic hardware require new analysis and kinematic calculations.

2.3 Neural networks

Artificial neural networks are capable of approximating highly complex nonlinear functions. As such, they are suitable for learning the mapping between the data produced by sensor devices used to teleoperate robots (such as joysticks and motion capture devices) and the actuator command values required to control the robot. Tejomurtula and Kak [20] demonstrated how neural networks could be used to solve a variety of inverse kinematics problems in robotics, arguing that the benefits of using neural networks to solve these problems include reducing software development labour costs, reducing computational requirements, and that they can approximate solutions to problems where algorithms or rules are not known and cannot be derived.

3 Related Work

Artificial neural networks have been trained to control robotic devices, though often in simulation and for robots with small degrees-of-freedom. Smagt and Schulten [21] train a neural network to control a rubber robotic arm (designed to resemble a skeletal muscle system). Jung and Hsia [22] use neural networks to fine tune

robot input trajectories in simulation. Larsen and Ferrier [23] train a neural network using visual information to map the relationship between the motor position and the pixel location of a large deflection, planar, flexible robot manipulator. Wang and Bai [24] use feed-forward neural networks with back-propagation to improve the positional accuracy of an industrial robot by finding the inverse kinematics of a simulated 2 and 3 link manipulator arms in different configurations, and then using a lookup table to select the appropriate weights at runtime. Neto et al. [14] employ low cost accelerometers attached to a human arm, and use artificial neural networks to recognise gestures for control of an industrial robotic arm. Later, Neto et al. [15] use a similar approach with a Nintendo Wii controller and an industrial robot arm. Morris and Mansor [16] use neural networks to find the inverse kinematics of a two-link planar and three-link manipulator arms in simulation.

Setapen et al. [17] use motion capture to teleoperate a Nao humanoid robot (with the aim of teaching the robot new skills and motions), using inverse kinematic calculations for finding the mapping between motion capture data and robot actuator commands. Matsui et al. [18] 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 movement on the robot. Song et al. [19] 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.

The most similar approach to the one presented in this paper is that of Aleotti *et al.* [11], who use neural networks to learn a mapping between the positions of a human arm and an industrial robot arm. In their approach the human operator wears ShapeTape[6] sensors on an arm and copies a series of pre-programmed robot movements, and a neural network for each DOF is trained using back-propagation to find a mapping between the operator's arm positions and the robot's arm positions. Mixed results were reported, with performance described as being "not yet satisfactory on complex tasks". A possible explanation for the error in their system was the use of absolute "bend and twist" angles, meaning that the user was required to place their arm in the exact same position during the repeated trials of the learning phase.

4 Our Approach

We teleoperate a humanoid robot by training a feed-forward neural network for each DOF on the robot to learn a mapping between motion capture sensor data and the angular position of the robot actuator to which each neural network is allocated. To collect data for the learning process, the robot leads the human operator through a series of repeated paired synchronised movements which capture both the operator's motion capture data and the robot's actuator data. Our example movements were designed to capture the full range of motion of each of the robot's motors (rather than using a series of static poses as per [11]). Another important distinction between our work and [11] is that we perform a data pre-processing step in which absolute motion capture data is transformed to relative rotations (see Section 4.2.), which allows for the kinematic mapping functions learned by the neural networks to not be affected by the user's location or orientation in absolute coordinate space. Particle swarm optimisation [28] is used to train each of the neural networks. The system is tested using an Aldebaran Nao and an Xsens MVN full-body motion capture suit. Empirical results detailing the neural networks' kinematics approximations are presented.

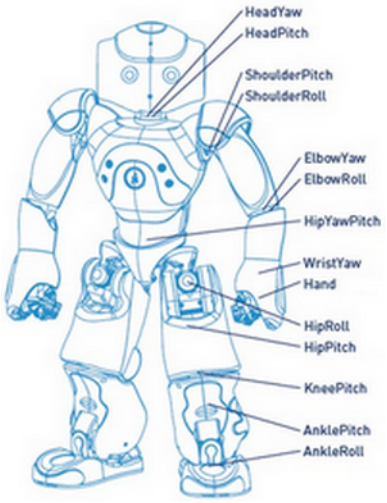

Figure 1: The Nao humanoid robot. The Nao is approximately 58cm high, with 25 degrees of freedom. Picture source: http://www.aldebaran-robotics.com/

4.1 Equipment, Hardware and System Architecture

Robot

We use an Aldebaran Nao humanoid robot[7]. The Nao is approximately 58cm high, with 25 DOF (see Figure 1). In this experiment we used the "H23" version of the Nao robot, which is specifically designed for robot soccer and does not have working wrists or hands (thus it has 23 DOF).

[6] http://www.measurand.com/shapetape.htm

[7] http://www.aldebaran-robotics.com/

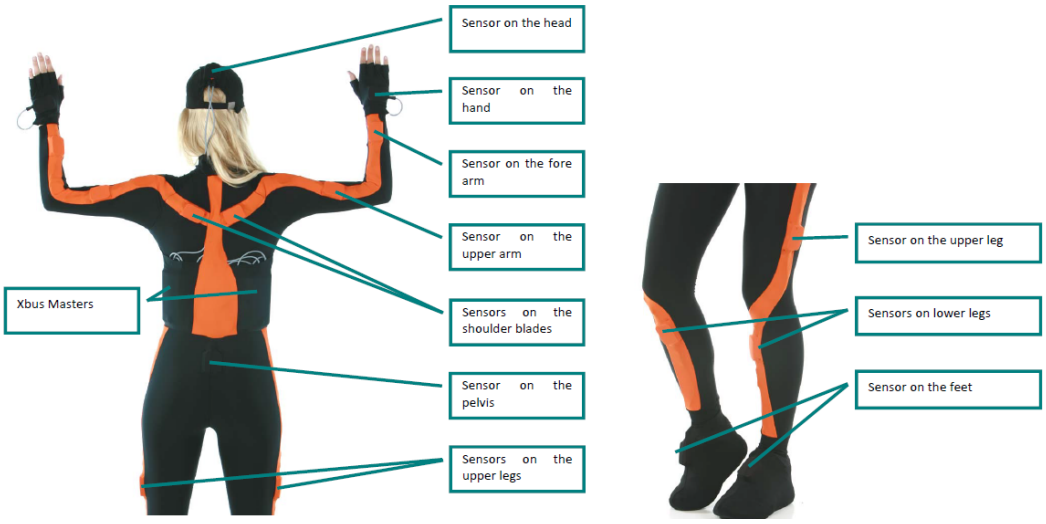

Figure 2: The Xsens MVN full-body motion capture suit. Picture source: htttp://www.xsens.com

Motion Capture Suit

As an interface to control the Nao robot we employ a high precision full-body motion capture suit, Xsens MVN[8]. Only recently motion capture suits similar to Xsens MVN reached the level of precision when they can correctly capture real-time motion of a human body with no significant data errors. This equipment comes in a form of a Lycra suit with 17 embedded motion sensors, illustrated in Figure 2. The suit is supplied with MVN Studio software that processes raw sensor data and corrects it. MVN studio is capable of sending real-time motion capture data of 23 body segments using the UDP protocol with the frequency of up to 120 motion frames per second. The key elements of the data being transmitted are absolute (X,Y,Z) position of each segment and its absolute (X,Y,Z) rotation.

Figure 2 shows a person wearing the XSENS MVN suit and displays the locations of inertial sensors. Figure 3 outlines the resulting joints that are being computed by the MVN Studio software and used in our experiments.

Architecture

The motion capture suit transfers via wireless communication sensor data to a software application running on a personal/laptop computer. A custom built C++ software application running on the computer performs data preprocessing (detailed in Section 4.2), the training of the neural networks, and the operation of the neural networks during teleoperation mode[9].

[8]For more information, visit the manufacturer's website http://www.xsens.com/en/general/mvn

[9]The neural network code could just as easily run on board the robot, as demonstrated by our previous work [25]. For the purposes of rapid prototyping an off board training system was implemented for this research.

4.2 Experimental Setup

Data Collection

To find a mapping between human movement and robot movement, motion data for both robot and human needed to be collected and synchronized. To do this, the robot was preprogrammed to perform a number of slow, symmetric repetitive motions. The human watches the robot, and is asked by the robot's text-to-speech system to copy the robot's movements. Thus the robot, much like a gym or fitness instructor, leads the human through a series of movements designed to demonstrate the robot's range of physical motion. The human imitates the robot's motion, and both the robot's motors' angular position data and the human's motion capture data is streamed to an off-board computer software application, where it is logged for use in machine learning. An example robot motion can be viewed online[10].

The robot's motions were simple to implement, and required programming a start position, an end position, and a speed of movement. The motions were designed to capture the the full range motion of all of the robot's motors. Many of the motions resembled exercises a person would perform during sports like weight training or aerobics. For example, to collect data for the elbow a "bicep curl" is performed, while to collect knee and hip data "squats" are performed. Figure 4 illustrates the start and end points of some of these motions. The motion would be demonstrated by the robot to the user before the data logging process began, which allowed the user the time to find a comfortable, repeatable corresponding motion. When user is ready to imitate the robot, each

[10]http://youtu.be/RgIQPtMIOXg

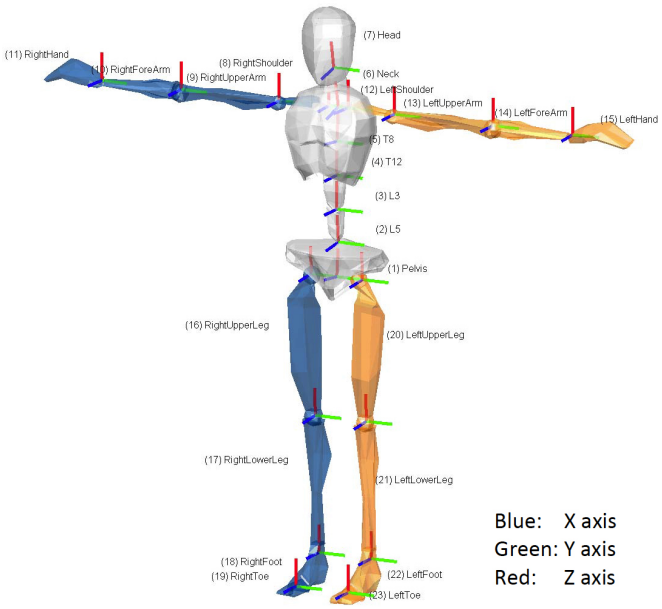

Figure 3: Human body joints used in motion capture. Picture source: XSENS MVN User Manual.

motion is repeated by the robot for approximately 5 to 10 repetitions (an arbitrary number chosen to balance the competing goals of collecting numerous examples while minimising the data collection time).

During the paired movements between human and robot, the robot's sensor data was logged at 5 frames per second for the entire duration of each repeated motion. To ensure motion capture data and robot sensor data logs are synchronised, a "3, 2, 1" countdown is used before commencing each repetitive motion. A slow speed was used to allow the user to closely follow the robot's example movements. As with the robot sensor data, motion capture data is logged at 5 frames per second. After the final repetition of each motion is completed, the data from both robot and human is merged into a single log file containing a time series of matched robot motor position sensor data and human motion capture data, the format of which is described in the next section.

Motion capture data preprocessing and transformation to "relative rotations".

The Xsens MVN motion capture suit produces raw data in absolute coordinate space. That is, its output values are specified in relation to a global origin coordinate in 3-dimensional physical space. As a consequence if the user is standing in two different locations with the exact same posture, the output values of the suit will be different due to the two different physical locations. Similarly for rotations, orienting the suit operator's body differently in space while holding the same pose (e.g. standing up versus lying down) would result in two different sets of absolute rotational values.

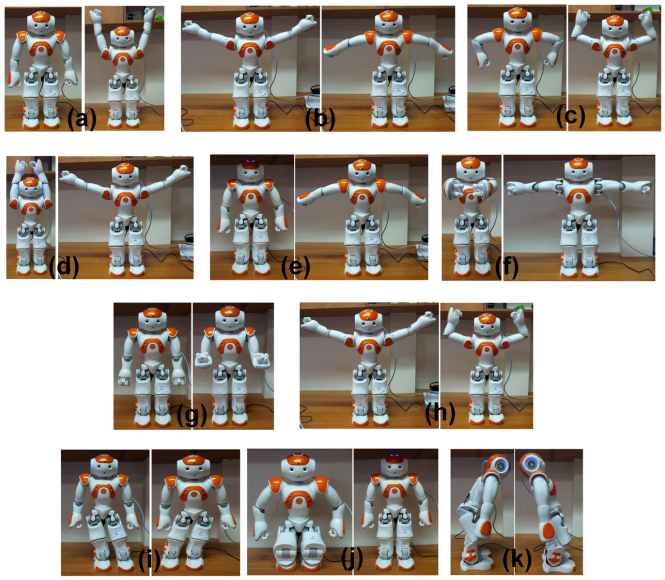

Figure 4: Snapshots of the start and end poses of robot motions which the human would mimic during the training process. The motors used during the transition from start pose to end pose for each of the motions are as follows: (a), (b) and (c) shoulder pitch; (d), (e), and (f) shoulder roll; (g) and (h) elbow roll; (i) ankle roll, hip roll; (j) ankle pitch, knee pitch, hip pitch (k) hip pitch.

Table 1: Relative 3-Dimensional Angular Rotations derived from motion capture data.

Relative Rotations	Sensor 1	Sensor 2
Left Shoulder	LeftUpperArm	Chest
Right Shoulder	RightUpperArm	Chest
Left Elbow	LeftUpperArm	LeftForearm
Right Elbow	RightUpperArm	RightForearm
Left Knee	LeftLowerLeg	LeftUpperLeg
Right Knee	RightLowerLeg	RightUpperLeg
Left Hip	LeftUpperLeg	Pelvis
Right Hip	RightUpperLeg	Pelvis
Left Ankle	LeftFoot	LeftLowerLeg
Right Ankle	RightFoot	RightLowerLeg
Neck	Head	Chest

To reduce the complexity of the learning problem, to allow our teleoperation system to be used regardless of the user's location and orientation, and to avoid the problems described by Aleotti *et al.* [11] from using absolute data, data pre-processing was performed which involved involved calculating rotational angles in three dimensional space of the user's body joints relative to each other. These "relative rotations" were dervied by

comparing pairs of sensors from the motion capture suit. For example, if we consider the robot's elbow roll motor and want to use motion capture input for training this motor, there is no exact corresponding sensor on the motion capture suit. To approximate an "elbow" sensor value, we compared the relative rotational movements of the forearm sensor and the upper-arm sensor (see Figure 2). Likewise the motion capture's "knee" is derived by comparing the upper leg with lower leg. Figure 3 describes the inputs from the motion capture suit, while Table 1 describes the relative rotations calculated from these input values.

In order to compute relative rotations, we use the following quaternion [26] equation:

$$Qrot_{relative}(a,b) = \frac{Qrot_b}{Qrot_a} \quad (1)$$

$Qrot_{relative}(a,b)$ corresponds to the resulting relative rotation (in the quaternion form) of the two different body segments represented by $Qrot_a$ and $Qrot_b$. The resulting quaternion rotation can be represented as:

$$Qrot_{relative}(a,b) = [q_0, q_1, q_2, q_3]^T \quad (2)$$

The values of this matrix are then converted to Euler angles (x,y,z) and used as the training input to each neural network.

Ensuring Robot Stability

In this initial study, no balance controller was used. The robot was in a standing position, and capable of movements such as squatting, shifting weight from one foot to the other, leaning forwards or backwards, and so forth. To ensure robot stability, the robot's ankle pitch and ankle roll motors did not employ neural networks for calculating their actuator commands[11]. Instead, the position of the robot's ankles was calculated relative to the position of the robot's hips and knees, so that the robot's feet always remained parallel with the ground at all times. In particular, the value of the ankle pitch motor was determined by the value of the hip pitch and knee pitch values of that particular leg. Similarly, ankle roll was determined by the value of the hip roll motor on the same leg of the robot.

4.3 Learning

The output of the data collection process is a series of datasets containing the actuator values for every motor on the robot, and a set of relative rotational angles for different parts of the human body, as described in Table 1. Initially there is a dataset for every motion performed by the robot, but one new large dataset was formed by merging the datasets. One feed forward neural network is allocated for each actuator on the robot (except those that have been disabled or overridden, as described in previous sections). Each neural network has one hidden layer of 20 neurons[12]. The neural networks are trained to approximate a function which outputs the actuator value (a) based on the relative rotations (x, y, z) of the corresponding sensors on the motion capture suit, i.e. *a = f(x,y,z)*. The function is represented by a matrix of values by the weights of each node. These weights are optimised to approximate the function (training) as closely as possible using particle swarm optimisation (PSO) [28] (50 particles and 50 generations). For this initial work, the neural networks were trained manually offline, rather than on board the robot. The training process for each network is fast (a matter of a few seconds on a standard personal computer).

5 Results

Neural network approximations of the relationship between motion capture data for a particular human joint and its corresponding robot motor joint are displayed in Figures 5 to 8. Figures 5 and 6 display different leg joints, all trained from the same 3 minutes of training data - motions (i), (j) and (k) as per Figure 3. Figures 7 and 8 display results for arms. The data presented in this figures is typical of data trained for the other robot joints. As can be seen in all data, the trained neural networks approximate the mapping between mocap data and robot actuator.

Error rates were analysed for the trained neural networks. Error rates were measured as a percentage of each motor's range of movement. An average mean error of 5.55% across all motors was achieved. Repeated training of the neural networks could further marginally improve accuracy, but with diminishing returns. Possible reasons for error in the system is that the user's performance of the individual repetitions of each motion varies considerably over time, creating multiple mappings between motion capture inputs and robot actuator outputs. Also, as can be seen in Figures 5 and 6 there are movements in the motion capture data while the robot's equivalent joint is stationary. This may suggest that in approximating the robot's movement, other articulation points on the human's body are being used, and this information is not being provided to the neural networks in our current model. It also reflects that while it is easy for

[11] We attempted to use neural networks to calculate the robot's ankle motor positions, but this resulted in instability of the robot. It may be possible to use neural networks for controlling the robot's legs for tasks such as standing and walking, but this requires further investigation.

[12] The number of hidden neurons can be easily modified by changing a parameter in the neural network training tool. Through numerous trials twenty hidden neurons was found to be a suitable number, capable of producing accurate output while keeping training time low.

a robot to remain perfectly still, it is very difficult for a human being.

5.1 Experiments

The trained neural network weights are used to control the robot in real-time. A video of the results can be found online[13]. Figure 9 shows the teleoperated robot and the human operator testing the system. During our experiments the user tried to perform a number of complex movements, such as picking up cardboard boxes, using the robot's fist to touch and hit objects, and by making dynamic movements such as "shadow boxing". The user had sufficient control of the robot to perform these tasks.

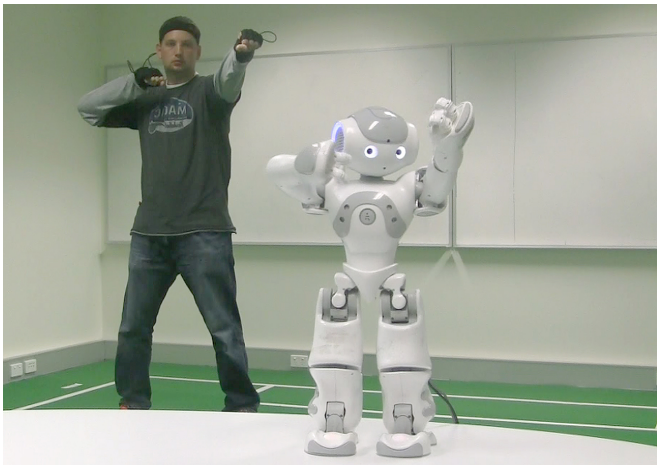

Figure 9: Teleoperating a robot in real time

To evaluate empirically the level of control for novel motions attempted by the user, trained neural network weights were tested against robot motions which did not form part of the training dataset. Figure 10 illustrates results for the robot's right arm from a sequence of motions that occur over a 48 second duration.

6 Discussion

We have constructed a teleoperation system using machine learning to find relationships between paired human and robot example motions. From approximately 10 minutes worth of example paired movements, a time series of motion capture inputs and robot actuator outputs is used by neural networks and particle swarm optimisation to find kinematic mapping functions between the physical pose of the user and the physical pose of the robot. Our results suggest that this is a promising general approach to building intuitive and flexible teleoperation systems. Benefits of this approach include lack of robot-specific kinematic modeling, the ability to adapt to the idiosyncrasies of individual users, and that minimal work is required to use the system with new motion capture sensors or new robots. Possible applications include teleoperation of any form, teleoperation of reconfigurable robots, or teleoperation of systems by people with disabilities or unusual human form.

6.1 Contribution

We have demonstrated a systematic method for finding kinematic mappings between humanoid robots and human operators for teleoperation. This method involves creating a set of simple robot motions that demonstrate the robot's physical capabilities, having the human operator imitate these motions, logging the robot's actuators angular position data and human's motion capture data, and then using machine learning to find a functional approximation of robot actuator commands from motion capture sensor data. We applied this method to successfully control sixteen[14] degrees of freedom on a Nao humanoid robot. The data collection process took approximately 10 minutes, and total learning time is in the order of seconds (rather than minutes).

In contrast to previous related work [11], our work differs in the following ways. Firstly, we employ a data preprocessing step in which motion capture values are transformed to relative rotational angles betwen different parts of the human body. This allows for the kinematic mappings learned during training to be generalisable regardless of the operator's absolute position or orientation in three dimensional space. Secondly, our example motions involve fluid movements that demonstrate the robot's entire range of motion, rather than a series of static poses. Capturing this data allows the kinematic mappings for each motor's full range of movement to be learned. Furthermore, by using fluid motions we obtain a large and rich set of examples which cover nearly every possible robot actuator value. Lastly, we have implemented the system on a humanoid robot with 25 degrees of freedom, rather than a 3 degree of freedom industrial robot arm [11; 14; 15; 16], demonstrating that our approach is suitable for robotic systems with large numbers of degrees of freedom.

6.2 Future Work

There are a number of immediate improvements to the research platform that could be easily implemented. For example, the machine learning process should be encapsulated in a user-friendly way, so that the user can easily retrain the neural networks at any time[15] (e.g in a sim-

[13] http://www.youtube.com/watch?v=ggLge1Rw2z4

[14] We had neural networks operating on the following joints of the Nao robot: head (2-DoF), shoulders (4-DoF), elbows (4-DoF), hips (4-DoF), knees (2-DoF).

[15] Our current approach requires us to manually open the data files, choose the inputs and outputs to each neural net-

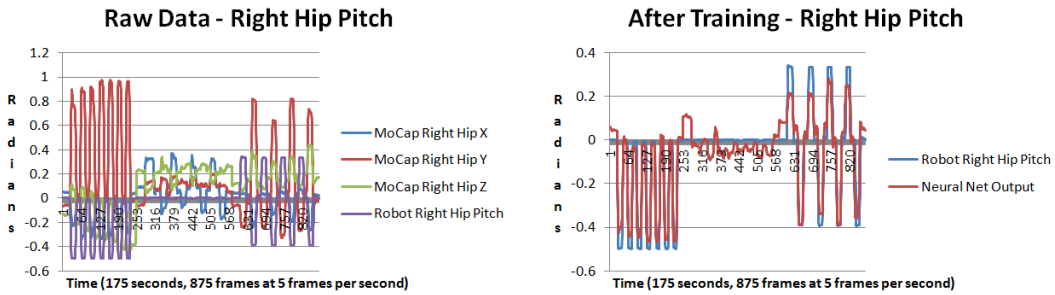

Figure 5: Right Hip Pitch training data and neural net approximation for a dataset comprised of 3 different motions, with 6 repetitions of each motion (see Figure 4, motions "j", "i" and "k").

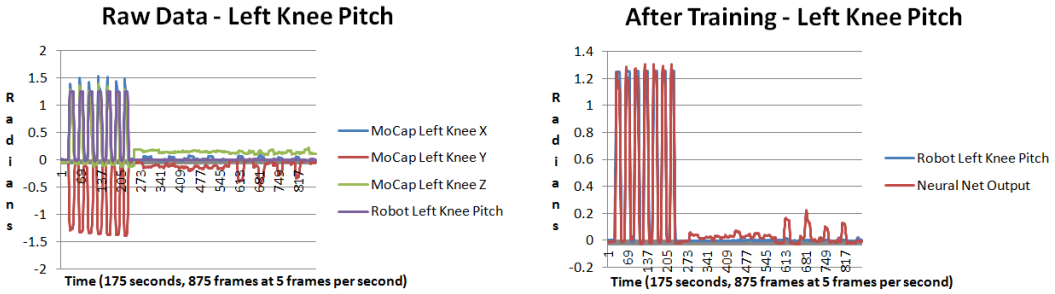

Figure 6: Left knee pitch training data and neural net approximation for a dataset comprised of 3 different motions, with 6 repetitions of each motion (see Figure 4, motions "j", "i" and "k").

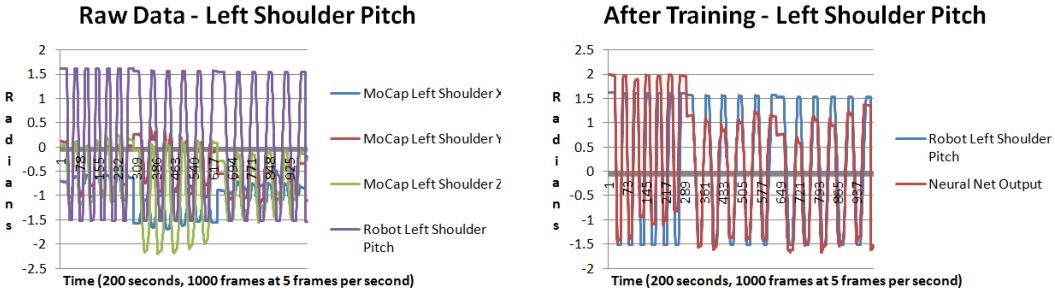

Figure 7: Left Shoulder Pitch training data and neural net approximation for a dataset comprised of 3 different motions, with 6 repetitions of each motion (see Figure 4, motions "a", "b" and "c").

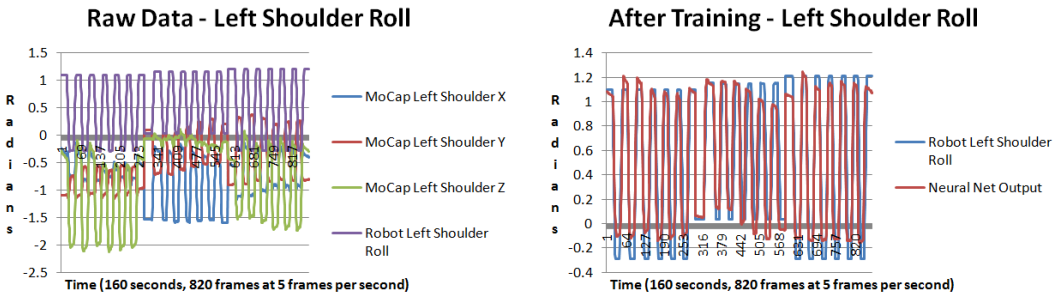

Figure 8: Left Shoulder Roll training data and neural net approximation for a dataset comprised of 3 different motions, with 6 repetitions of each motion (see Figure 4, motions "d", "e" and "f").

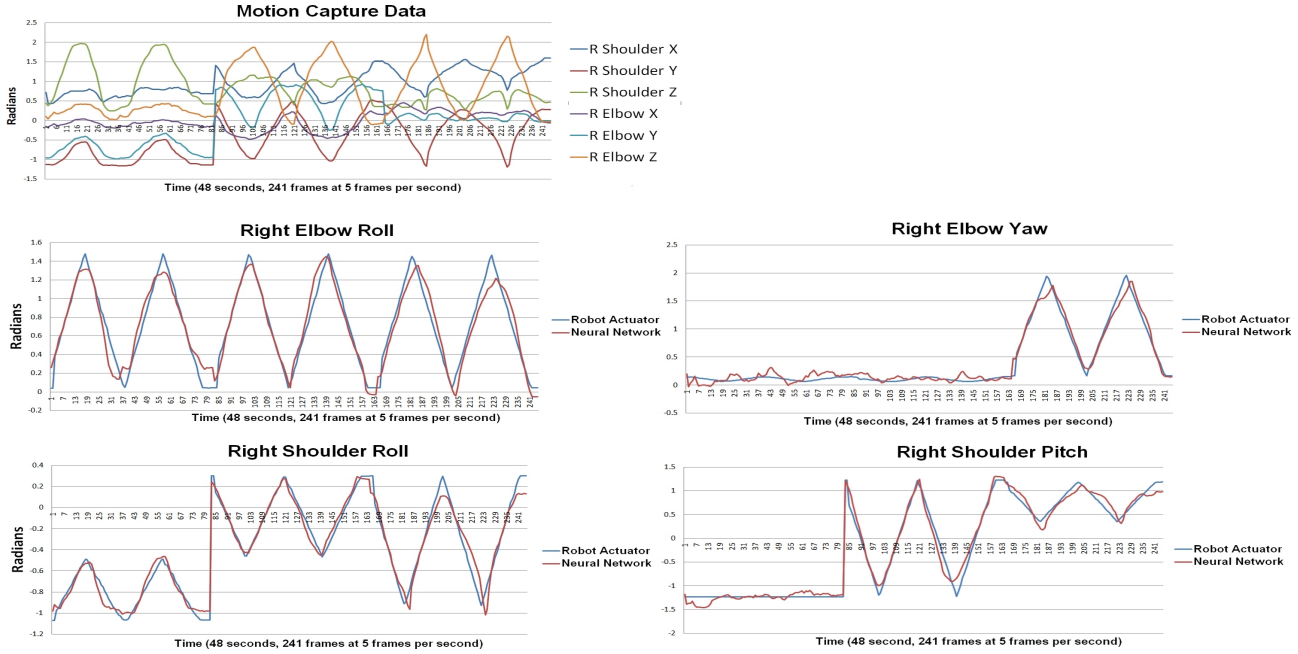

Figure 10: Motion capture data for the operator's right arm (top). The data is collected when the human user is imitating three distinct robot motions that occur over a period of 48 seconds. The four graphs below the motion capture data display the robot's actuator values, and the neural net approximation of these actuator values. It is important to note that the neural network weights were not trained using this data, but with data collected from other movements, demonstrating that the kinematic mapping learned during the training process generalises to novel movements.

ilar way to the button-press system presented in [11], or using the Nao's speech recognition capabilities). Another is to test the flexibility of the system by using a different motion capture input device and/or a different robot.

It is possible that different machine learning techniques and fitness functions may yield more accurate kinematic approximations. We used neural networks with particle swarm optimisation, simply because we achieved good results with this machine learning technique in previous work [25]. The inputs to each neural network could also be varied. Our current approach involved isolating corresponding parts of the human's body relative to the robot's body. An alternative approach would be to use a larger set of relative rotations as inputs to each neural network. This would allow the user to explore training unusual patterns of robot control, such as using their arms to control the robot's legs.

Acknowledgments

Much of this research was conducted while the first author was an employee of the University of Technology, Sydney (UTS). Most of the data and experiments presented in this paper were done using UTS's H23 Nao robots.

work, and then manually select each neural network's weights for the teleoperation run-time system.

References

[1] R. Goertz and R. Thompson. Electronically controlled manipulator. Nucleonics, Vol.12, No.11, pp.46-47, 1954.

[2] H. Takanobu, H. Tabayashi, S. Narita and A. Takanishi. Remote Interaction between Human and Humanoid Robot. Intelligent and Robotic Systems, 25: 371–385, 1999.

[3] N. Sian, K. Yokoi, S. Kajita, F. Kanehiro, K. Tanie. Whole Body Teleoperation of a Humanoid Robot - Development of a Simple Master Device using Joysticks. IEEE/RSJ Intl Conf on Intelligent Robots and Systems, pp. 2569–2574, 2002.

[4] H. Hasunuma, M. Kobayashi, H. Moriyama, T. Itoko, Y. Yanagihara, T. Ueno, K. Ohya, and K. Yokoi. A tele-operated Humanoid Robot Drives a Lift Truck. IEEE Intl Conf on Robotics and Automation, pp. 2246–2252, 2002.

[5] H. Hasunuma, K. Nakashima, M. Kobayashi, F. Mifune, Y. Yanagihara, T. Ueno, K. Ohya, K. Yokoi. A tele-operated humanoid robot drives a backhoe.

Proc IEEE Intl Conf on Robotics and Automation (ICRA), pp. 2998–3004, 2003.

[6] K. Yokoi, K. Nakashima, M. Kobayashi, H. Mihume, H. Hasunuma, Y. Yanagihara, T. Ueno, T. Gokyuu, and K. Endou. A tele-operated humanoid operator. The Intl. Journal of Robotics Research, 2006, 25: 593-602.

[7] L. Zhang, Q. Huang, T. Liu, and Y. Lu. A teleoperation system for a humanoid robot with multiple information feedback and operational modes. IEEE Intl Conf on Robotics and Biometrics (ROBIO), pp. 290–294, 2005.

[8] T. Takubo, K. Inoue, T. Arai and K. Nishii. Wholebody Teleoperation for Humanoid Robot by Marionette System. Proc. IEEE/RSJ Intl Conf on Intelligent Robots and Systems (IROS), p.4459-4465, 2006.

[9] B. Dariush, M. Gienger, B. Jian, C. Goerick, and K. Fujimura. Whole Body Humanoid Control From Human Motion Descriptors. Proc. IEEE Intl Conf on Robotics and Automation (ICRA), p. 2677-2684, 2008.

[10] N. Miller, O. C. Jenkins, M. Kallman, M. J. Mataric. Motion capture from intertial sensing for untethered humanoid teleoperation. Proc. of the 4th IEEE-RAS Intl Conf on Humanoid Robotics, p.547-565, 2004.

[11] J. Aleotti, A. Skoglund and T. Duckett. Position Teaching of a Robot Arm by Demonstration with a Wearable Input Device. Proc Intl Conf on Intelligent Manipulation and Grasping, (IMG04), p.459-464, 2004.

[12] R. A. Peters II, C. L. Campbell, W. J. Bluethmann and E. Huber. Robonaut Task Learning through Teleoperation. Proc of the 2003 IEEE Intl Conf on Robotics and Automation, p.2806-2811, 2003.

[13] B. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. Robotics and Autonomous Systems. Vol 57, Issue 5, Pages: 469-483, 2009.

[14] P. Neto, J. Norberto Pires, and A. Paulo Moreira. Accelerometer-Based Control of an Industrial Robotic Arm. The 18th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), p.1192-1197, 2009.

[15] P. Neto, J. Norberto Pires, and A. Paulo Moreira. High-level programming and control for industrial robotics: using a hand-held accelerometer-based input device for gesture and posture recognition. Industrial Robot: An International Journal, 37/2 (2010) 137–147.

[16] A. S. Morris and A. Mansor. Finding the inverse kinematics of manipulator arm using artificial neural network with lookup table. Robotica (1997) vol 15, pp 617–625.

[17] A. Setapen, M. Quinlan, and P. Stone. Beyond Teleoperation: Exploiting Human Motor Skills with MARIOnET. AAMAS 2010 Workshop on Agents Learning Interactively from Human Teachers (ALIHT).

[18] Matsui, D., Minato, T., MacDorman, K.F. and Ishiguro, H. Generating Natural Motion in an Android by Mapping Human Motion. Proc. of Intelligent Robots and Systems (IROS), p.3301-3308, 2005.

[19] H. Song, D. Kim, M. Park and J. Park. Teleoperation between Human and Robot Arm using Wearable Electronic Device. Proceedings of the 17th IFAC World Congress, Seoul, Korea, p.2430-2435, 2008.

[20] S. Tejomurtula and S. Kak. Inverse kinematics in robotics using neural networks. Information Sciences 116 (1999) 147-164.

[21] Smagt, P.V.D. and Schulten, K., Control of pneumatic robot arm dynamics by a neural network. Proceedings of the World Congress on Neural Networks, 3, pp. 180–183, 1994.

[22] S. Jung and T. C. Hsia. Neural network reference compensation technique for position control of robot manipulators. IEEE Intl. Conf. on Neural Networks, pp. 1765–1770, 1996.

[23] Larsen, J.C. and Ferrier, N.J. A case study in vision based neural network training for control of a planar, large deflection, flexible robot manipulator. Proc. IEEE Intelligent Robots and Systems, pp. 2924–2929, 2004.

[24] D. Wang and Y. Bai. Improving Position Accuracy of Robot Manipulators Using Neural Networks. Proc. of 2005 IEEE Instrumentation and Measurement Technology Conf., pp. 1524–1526, 2005.

[25] C. Stanton, E. Ratanasena, S. Haider, and M-A Williams. Perceiving forces, bumps and touches from proprioceptive expectations. In Proc of RoboCup 2011, LNCS 7416, pp. 377–389.

[26] Kuipers, J.B. Quaternions and Rotation Sequences. Princeton University Press, Princeton, 1999.

[27] Baillieul, J. Kinematic Programming Alternatives for Redundant Manipulators. In IEEE Intl. Conf. on Robotics and Automation, pp. 722–728, 1985.

[28] Kennedy, J. and Eberhart, R. (1995). Particle Swarm Optimization. Proc. of IEEE Intl. Conf. on Neural Networks. pp. 1942–1948.