Mapping Human Motion to a Humanoid Robot

by Jiang Chen He

Supervisors: Professor Goldie Nejat & Shane Saunderson

April 2020

Mapping Human Motion to a Humanoid Robot

by Jiang Chen He

Supervisors: Professor Goldie Nejat & Shane Saunderson

April 2020

Abstract

In this paper we describe a human to robot motion mapping system that controls a humanoid robot named Pepper to mirror a human demonstrator, for the purpose of having the robot learn anthropomorphic movement through learning from demonstration. The system will try to address the challenges caused by the limited degrees of freedom and motion range of Pepper joints compared to the joints of average humans. The motion mapping encompasses the arms and partial head joints. We use a skeleton-tracking software named Nuitrack to gather the human motion data and a deep learning model to map the desired joint positions to Pepper joint angles, in which the model achieved a mean absolute error of a worst case 0.15 radians. We assess the quality of the motion mapping by conducting a survey on emotion recognition based on Pepper's body gestures, as Pepper imitates a human demonstrator conveying one of Ekman's six basic emotions through body movement.

Keywords — Learning from Demonstration, Humanoid Robots, Motion Mapping, Emotion Recognition

Acknowledgement

I would like to express my gratitude towards Professor Goldie Nejat for supervising the project and allowing me the opportunity to conduct research in her lab. I would like to thank Shane Saunderson for his invaluable guidance and support throughout my thesis project and allowing me the opportunity to work with him in this project. Without their help, this project would not have been possible. I would also like to thank the members of the ASB lab, Mingyang Shao, Chris Mohamed, and Daniel Dworakowski, for their advice and support. I also thank the people that participated in the video matching survey. Finally, I would like to thank the MIE undergraduate department for providing the opportunity to conduct research as part of the undergraduate curriculum.

Table of Contents

List of Figures

List of Tables

1. Introduction

Humanoid robots are slowly emerging in commonplace human environments such as hospitals [1], where they will help and interact with patients and staff. For humanoid robots to better engage with humans, there is a need for humanoid robots to achieve anthropomorphism in the expression of emotional states through human-like facial features and body gestures. It is important for humanoid robots to be more anthropomorphic to facilitate social interactions and favor cooperation between humans and robots [2]. For our work, we focus on robots achieving anthropomorphic movement.

A favorable method for robots to learn anthropomorphic movement is through learning from demonstration (LfD). In LfD, a human or another machine can show the robot how to perform a task or sequences of motion [3]. The robot learns to generalize a task by observing several demonstrations. In our case, the humanoid robot learns anthropomorphic movement by observing several demonstrations of human movement at different emotional states. Compared to the traditional method of programming the robot, which could involve breaking down a task into hundreds of steps, LfD is more advantageous as it is less intensive and time consuming. There are three common interfaces for demonstration: immersive demonstration such as using a joystick to control the robot, kinesthetic teaching where a human manually moves the robot, and tracking human motion and mapping the human joints to the robot joints [3]. For our work, we focus on the last interface, as it allows the demonstrator to move freely so we can capture the human movement in its natural state.

We developed a system to map human motion to Pepper, a humanoid social robot produced by Softbank [4]. Pepper is used to mirror the movements of a pre-recorded human demonstrator that embodies Paul Ekman’s six basic emotions: anger, disgust, fear, happiness, sadness, and surprise [5]. In this work, we will describe the architecture of the system and the algorithms designed for human to Pepper motion mapping. The goal is not to design a novel human to robot motion mapping system, but a functional system that is able to control a humanoid robot, specifically Pepper, to smoothly mirror human motion. The significance of this work is that the motion mapping system is a crucial component for LfD, which we can use to have Pepper learn anthropomorphic movement to achieve better human engagement.

The system will also try to address the correspondence problem of having different physical structure between Pepper and a human demonstrator. Even though Pepper has a considerable motion range, it still is very limited compared to the motion range of the average human. The Pepper joints also have limited degrees of freedom (DoFs). For example, the human shoulder joint has 3 DoFs whereas the Pepper shoulder joint has 2 DoFs. This prevents us from simply extracting the roll, pitch, yaw angles from the skeleton links and applying it directly to the Pepper joint angles.

Different methods to convert the skeleton motion to Pepper joint angles were explored, including using inverse kinematic (IK) solver and a deep learning algorithm. The motion is recorded using a third-party skeleton-tracking software that requires only a single RGB-D sensor and a CPU computing platform. No extra devices, such as a GPU, were required to capture the human motion. Due to time constraints, only the upper body movements of the wrist and elbows and the head pitch are tracked and mirrored.

2. Literature Review

This section shows the literature review conducted for robots imitating human movement for various robots and motion capturing systems.

Pepper has been used in various research for mirroring human movement. In [6], Pepper mirrored a human demonstrator performing flag semaphores in real time. The researchers used a 2D video stream using the cameras in the Pepper robot, the skeleton-tracking software OpenPose, and a cloud-based framework to perform gesture recognition on the human. Because OpenPose required state-of-the-art GPU servers to perform in real time, the researchers decided to stream the data to the cloud rather than process the data locally. This study focused on robot gesture mirroring in real time, and in order to mitigate network latency from using the cloud, the researchers used precomputed motion trajectories to get Pepper to mirror the semaphore gestures. It did not use a human to robot motion mapping system.

A separate study used a motion-tracking system of wearable motion sensors to enable participants to control the Pepper robot in real time [7]. Specifically, the researchers used a HTC Vive headset and two HTC Vive handheld controllers. The headset allowed the control of two DoFs in the robot's head. The researchers processed the arm movements using an IK solver, although the exact positioning of the elbow was not tracked.

Humanoid robots other than Pepper have been used in motion mapping studies. At Carnegie Mellon University, Sarcos humanoid robot was used to mirror pre-recorded motion of human actors performing "I'm a little teapot" [8]. For the motion-tracking system, the researchers used a commercially available system from Vicon which involves eight cameras and 35 14mm markers placed on the actors. The raw marker data is mapped onto a skeleton using Vicon's Bodybuilder software. The skeleton is then mapped to the Sarcos robot. The head, shoulder, elbow, and wrist joints could be matched directly, since both the skeleton and Sarcos robot contained the same DoF for each joint. The other joints were mapped to the DoF of the robot by inverse kinematics on individual limbs. The Sarcos robot managed to retain the individual movement style of each of the seven actors through its motion mapping system.

Another study proposed a quadratic programming IK framework to enable the humanoid robot HUBO2+ developed by the KAIST Humanoid Research Center to efficiently imitate human motion [9]. The study focused on the lower body movement of the bipedal legs rather than the upper body movement. To obtain the six DoF information of each leg, a data acquisition device developed by KAIST Humanoid Research Center and Rainbow Robotics was utilized.

A paper proposed a methodology for human to robot motion mapping [10], using the Isotrak II motion capture system which uses magnetic position sensors placed on the wrist and elbows. It suggests finding all inverse kinematic solutions of the robot arm with the desired end-effector position and choosing the best one that minimizes the difference between desired elbow and end-effector position and the actual elbow and end-effector position. This methodology worked well for the case of Mitsubishi PA 10 - DLR/HIT II robotic arm hand system.

In [11], the humanoid robot NAO was used to copy the upper-body pose of a human subject. The poses were captured with a Microsoft Kinect sensor and the skeleton tracking capability of Microsoft Kinect SDK. The researchers presented an analytic solution to the correspondence problem that is general enough to be applicable to most available humanoid robots. They tested their solution against the IK solver for NAO provided by the NAOqi library, and found their solution to be far superior with an error value of 0.01 compared to 0.412 for the IK solver. However, the researchers computed the error in a way that harshly penalized the IK solver when it failed to find a solution, which it did often, magnifying the error value.

One study described a system that allows the controlling of robots through real time motion stream [12]. The system accommodates different motion capture systems and uses a motion mapping library called HUMAN, which is typically used for mapping human motion to virtual avatars but can also be used for humanoid robots. HUMAN uses an XML based Orientation File (XOF) and an avatar skeleton file (XSF) to map positions and orientations of the corresponding bones and joints. The system was tested on NAO and Robothespian robots. Both successfully imitated the human demonstrator.

Many studies were conducted on humanoid robots imitating human movement, with many successful results. A summary of the literature review can be seen in Table 1. However, no literature or open source solution was found for human to robot motion mapping for Pepper using video stream motion capturing.

Authors	Year	Motion Capture System	Motion Mapping Method	Robot
N. Tian et al. [6]	2018	Pepper's camera and OpenPose	N/A	Pepper
S. Thellman, J. Lundberg, M. Arvola, and T. Ziemke [7]	2017	HTC Vive headset and handheld controllers	IK solver	Pepper
N. S. Pollard, J. K. Hodgins, M. J. Riley, and C. G. Atkeson [8]	2002	Vicon, multi camera marker system	Directly Applied	Sarcos
J. Oh, I. Lee, H. Jeong, and J.-H. Oh [9]	2019	proprietary data acquisition device	IK solver	HUBO2+
M. V. Liarokapis, P. K. Artemiadis, and K. J. Kyriakopoulos [10]	2012	Isotrak II wearable motion sensor	IK solver	Mitsubishi PA 10 - DLR/HIT II robotic arm
Y. Mohammad and T. Nishida [11]	2013	Kinect sensor and Kinect SDK skeleton-tracker	Analytical solution	NAO
B. Spanlang, X. Navarro, J.-M. Normand, S. Kishore, R. Pizarro, and M. Slater [12]	2013	N/A	HUMAN library	NAO, Robothespian

Table 1. Summary of literature review

3. System Architecture

The high-level controller is based on reactive control and follows a SENSE-ACT pattern. The motion capture unit allows the robot to “sense” its surroundings and subjects. In the motion capture unit, a Microsoft Kinect sensor is used to stream images of the human demonstrator. A skeleton-tracking software receives the stream of images and tracks the 3D coordinates of the human demonstrator’s joints. The skeleton data feeds into the action unit, which determines the robot’s movement and allows the robot to “act”. The action unit is composed of three modules: skeleton data processing, deep learning model, and a motion controller. The skeleton data processing module scales the skeleton data from human perspective to Pepper perspective to obtain the desired Pepper joint positions. The desired joint positions are fed into a model which outputs the corresponding Pepper joint angles. Finally, the motion controller provided by NAOqi API [13] directly sends the joint angles to Pepper and interpolates the motion trajectory. Once Pepper reaches the desired joint positions, the robot repeats the SENSE-ACT behaviour, taking the most recent skeleton data, obtaining the desired Pepper joint angles, and controlling Pepper to reach the next joint position target. The software was written in Python (for ease of development) using the Robotic Operating System (ROS). The entire architecture is shown in Figure 1.

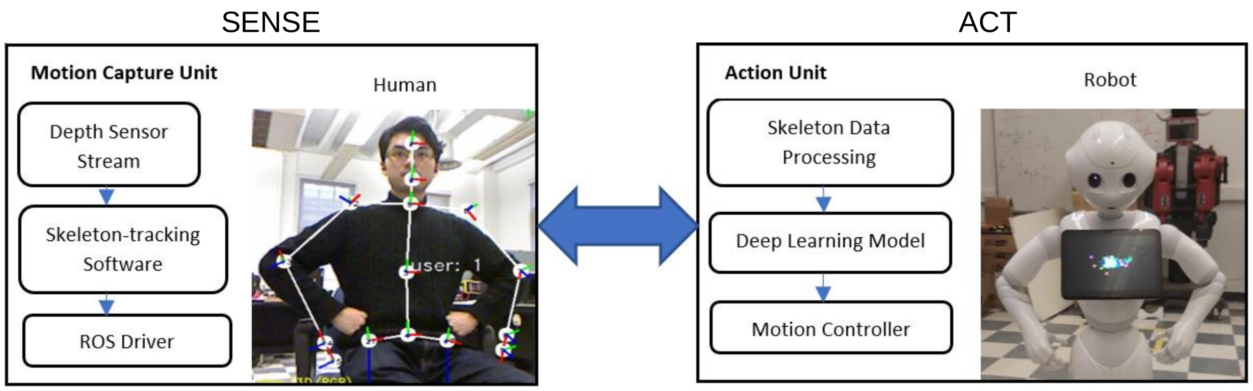

Figure 1. System Overview

There could be an argument to be made that from the perspective of a low-level controller, the controller type is a deliberative control and follows a SENSE-PLAN-ACT pattern, where the planning portion is the skeleton data processing and deep learning modules. One could say the controller plans the joint angles of the Pepper robot based on the actions of the human demonstrator the robot sees. But from the perspective of a high-level controller, Pepper simply sees the human demonstrator and mirrors their action without future consideration, making the controller type reactive in design. Thus, one could argue that the controller is reactive or deliberative depending on one’s perspective of the controller as a high-level or low-level controller.

3.1. ROS architecture

This section describes the architecture of the ROS nodes and topics. A diagram of the full ROS architecture can be seen in Figure 2. A ROS driver for Nuitrack publishes the skeleton data to the topic /body_tracker/skeleton_v2. Two ROS nodes were written to map the skeleton to Pepper. The /ros_joint_converter node makes up the skeleton data processing module and the deep learning module. It gets the skeleton data and generates the Pepper joint angles, which gets published to the topic /pepper_joint_angles. It also visualizes the skeleton in RVIZ, a 3D visualization tool for ROS, through the /pose_visualization topic. The /ros_move_pepper node makes up the motion controller module. It grabs the Pepper joint angles and sends it to the NAOqi API motion controller which controls the movement of Pepper. It also publishes the Pepper joint angles to /joint_states topic, which /robot_state_publisher node uses to control the movement of a simulated Pepper in RVIZ, as shown in Figure 3.

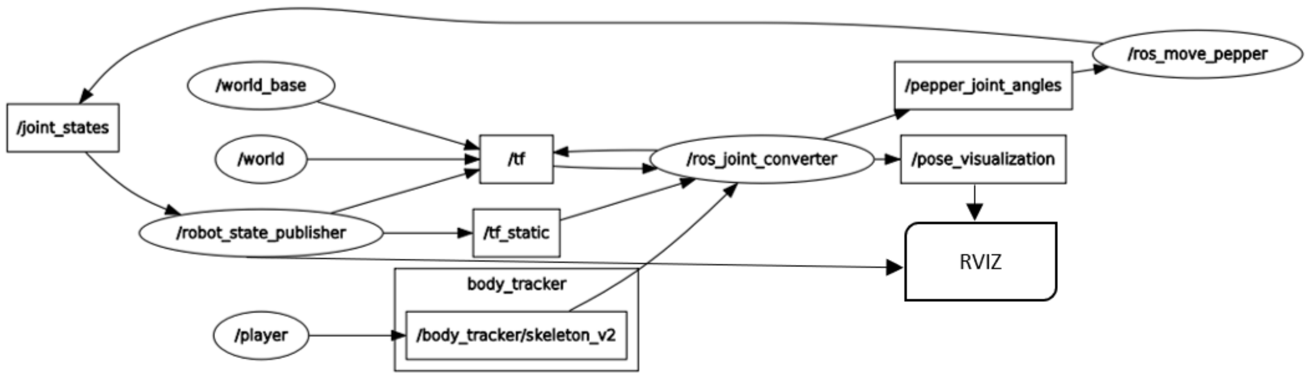

Figure 2. ROS Architecture of the Motion-Mapping System

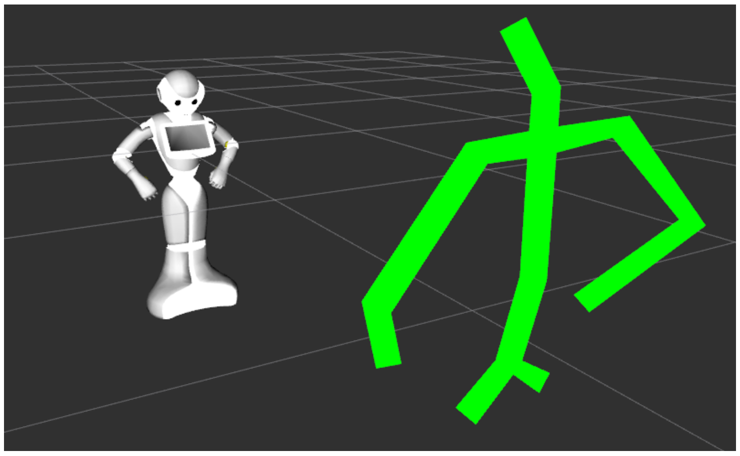

Figure 3. A Simulated Pepper in RVIZ Mirroring a Skeleton

4. Software and Hardware

This section details the software and hardware tools chosen for this system as well as the evaluation behind selecting those particular tools among others.

4.1 Humanoid Robot Pepper

Pepper is a humanoid social robot built by Softbank. Other humanoid robots exist in the ASB lab such as NAO [14], Tangy [15], and Casper [16], but Pepper was the optimal choice with a body frame designed to be as human-like as possible and a wide motion range compared to that of other humanoid robots in the ASB lab. Pepper has two arms with two DoFs for each shoulder and elbow joints and one DoF for each wrist. In addition, Pepper has two DoFs for the head, two DoFs for the hip, and one DoF for the knee. Figure 4 shows a diagram of Pepper with all of its DoFs. For this work, we will only be controlling the head pitch and each arm’s shoulder pitch, shoulder roll, elbow yaw, and elbow roll. Pepper has its own CPU but due to its limited capabilities we used an external computer.

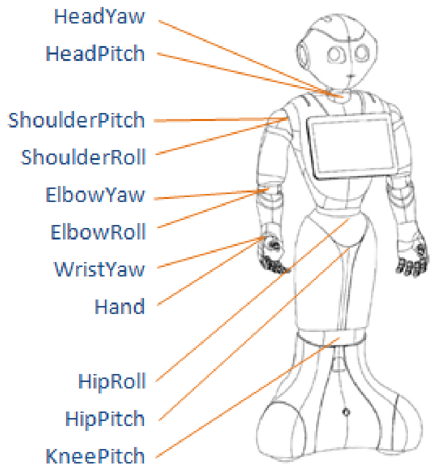

Figure 4. Degree of Freedoms of Pepper Robot

4.2 Motion Tracking System

There exist various motion tracking systems based on vision (with and without markers), exoskeleton, or wearable motion sensors. However, considering the high financial cost of acquiring an exoskeleton or wearable motion sensors, a vision based markerless motion tracking system was determined to be the optimal choice, as it was the easiest for demonstrators to use and was the most economical. For this system, only an RGB-D sensor and a skeleton-tracking software was required.

4.2.1 RGB-D Sensor

We compared three main candidates for the RGB-D sensor: the Microsoft Kinect v1, ASUS Xtion Pro Live, and Pepper’s own 3D camera, which is an ASUS Xtion located behind the eyes.

Pepper's sensor provided an image resolution of 320x240, whereas the other two sensors provided an image resolution of 640x480. With a lower image resolution, Pepper's 3D sensor was rejected. That left the Kinect and Xtion Pro Live. Although Xtion Pro Live has slightly better specifications in terms of wider field of view and greater distance of use [17, Tab. 1], the Kinect sensor was chosen as it was easily available in the lab and there was no significant differences between the two devices in terms of critical functionalities and depth resolution since they are both based on PrimeSense infrared technology.

a) ASUS Xtion Pro Live

b) Microsoft Kinect v1

Figure 5. Images of the 3D sensors

4.2.2 Skeleton Tracking Software

We compared different skeleton-tracking software to determine which one would be the most appropriate for our application. The main requirements include maintainability, as in the software is actively supported and maintained by some organization or community, and quality, as in the skeleton should be stable and consistent. Four candidates were chosen: Nuitrack, PoseNet, OpenPose, and NITE [18][19][20][21]. A comparison summary can be seen in Table 2.

	GPU required	Cost	Skeleton Quality	Maintained
Nuitrack [18]	No	Free trial	Good	Yes
PoseNet [19]	No	Open Source	Poor	Yes
OpenPose [20]	Yes	Open Source	N/A	Yes
NITE [21]	No	Open Source	N/A	No

Table 2. Comparison of 4 different skeleton-tracking software

NITE was neglected since it is no longer maintained. We attempted to use OpenPose but encountered issues with setting up CUDA on the lab computer. Due to limited time and hardware, we had to move on. Then we attempted to use PoseNet but the skeleton quality was poor and unstable and it only captured 2D coordinates. We tried converting the 2D pixel coordinates to 3D coordinates using the depth map created by the Kinect sensor, but the 2D pixel coordinate did not match the depth map, giving infinite distance at points where there was supposed to be a joint. Next we tried the commercially available skeleton-tracking software Nuitrack. The skeleton quality was stable and smooth but with a slight offset and occasional glitches in which the software lost track of the actual arm positions. It captured 3D and 2D coordinates without requiring a server with GPU. With everything considered, we settled on

Nuitrack as the optimal choice. It provided a free trial which was used throughout the development of the motion mapping system. A diagram of the Nuitrack skeleton can be seen in Figure 6.

A ROS driver exists for Nuitrack [22], which publishes the skeleton data to a ROS topic. However, the naming of the topic message was incorrect and confusing. Thus, a forked repository containing the correct naming was used instead [23].

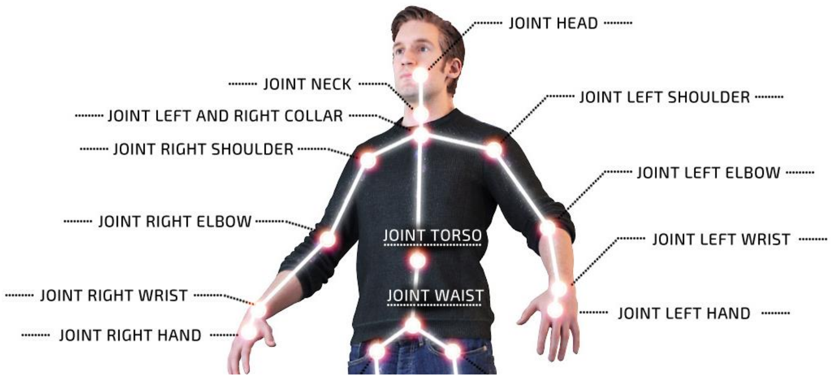

Figure 6. Nuitrack Skeleton Joints

5. Module Design

The action unit is the main function that processes the skeleton data, computes the joint angles for the Pepper robot, and controls the Pepper robot movement. In this section, a more in-depth explanation of each module of the action unit is provided.

5.1 Skeleton data processing

In order to compute the desired elbow and wrist coordinate with respect to Pepper’s frame of reference (near the spine), the skeleton joint coordinates were transformed with respect to the torso joint as the origin. Then the wrist and elbow coordinates were scaled by the ratio of skeleton arm length and Pepper arm length. It resulted in a generally accurate representation of where the joints should move to.

To convert the desired coordinates to the corresponding joint angles, couple methods were explored. Simply retrieving the roll, pitch, yaw angles from the skeleton’s upper arm and forearm link and applying it directly to Pepper joint angles was not possible due to the different DoFs for Pepper arms and human arms. We then tried to use IK solvers such as the one provided by [24] and [25] to determine the joint angles. However, the output was not reliable and often failed to find adequate angles to position the robot arm in the desired configurations. We suspect the desired coordinates had a slight offset due to the imperfect scaling solution, making the joints unable to reach that exact location, which the IK solver could not solve for. Although there were other IK solvers we could have tried, due to time constraints we moved on to using a deep learning technique, which will be explained more thoroughly in the next section. The deep learning model had a much better result in finding the optimal joint angles for the desired elbow and wrist positions.

Pepper has limited motion range which correlates to limited joint angles. The model sometimes outputs a joint angle that is outside the joint angle range (the range of each joint angle can be found in [26]). To address this issue, each joint angle was automatically restricted to its range, in which if the angle was over the upper limit, it would instantly fall back to the upper bound, and likewise for below the lower limit. Letting α represent a joint angle that the model outputs, the equation used was:

$$\alpha = \begin{cases} upper\_limit, & \alpha > upper\_limit \\ lower\_limit, & \alpha < lower\_limit \\ \alpha, & otherwise \end{cases}$$

The head pitch was the easiest to map. Letting α represent the head pitch, Δy represent the difference between the head and neck joint in the vertical axis, and Δz represent the difference between the head and neck joint in the axis the Kinect sensor faces, the equation used was

$$\alpha = \begin{cases} \tan^{-1}\left(\dfrac{\Delta y}{\Delta z}\right), & |\Delta z| > 0.01 \\ 0, & otherwise \end{cases}$$

In addition, the head pitch was amplified to make the motion more obvious and was automatically restricted to its angle limits of -0.7068 radians to 0.6371 radians.

The only constraint is that the human demonstrator must be directly facing the Kinect sensor without leaning forward or backward. If they did, the absolute value of Δz would be enlarged, since the head would be closer/farther from the Kinect sensor than the neck, making the head pitch inaccurate. It also assumes that the hip and knee joints are immobile.

5.2. Deep learning model

An artificial neural network (ANN) was trained to find a relation between the elbow and wrist coordinates and the shoulder and elbow joint angles. Each arm was trained individually. The ANN consists of 2 fully connected hidden layers of size 200 and 100 units. It outputs 4 values: elbow yaw, elbow roll, shoulder roll, and shoulder pitch. It takes in an input of 6 values: the desired robot XYZ coordinate of the elbow and wrist joint. The architecture of the model is shown in Figure 7 and was inspired by the article in [27].

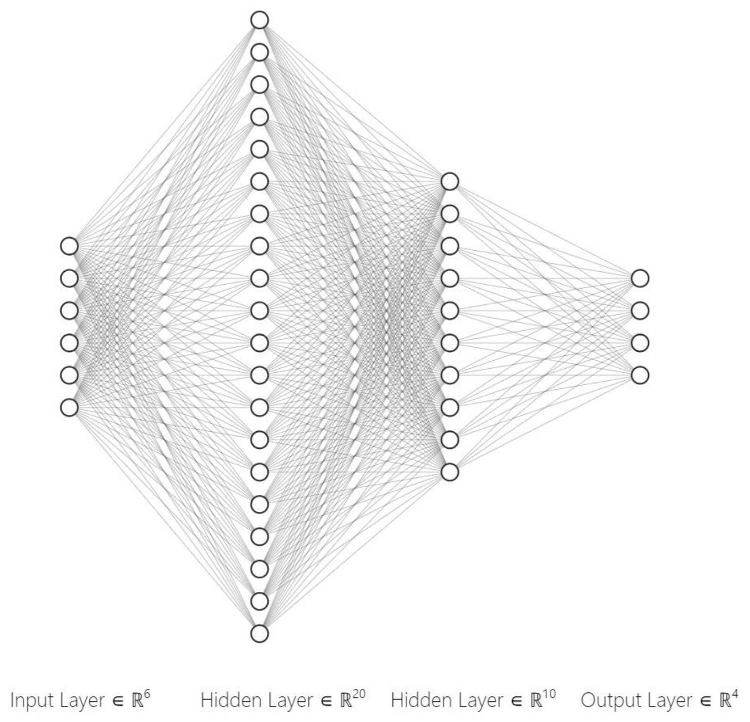

Figure 7. Artificial Neural Network Architecture. The 2 hidden layers are of size 200 and 100, however for visualization purposes they are condensed to 20 and 10, respectively.

Regarding its hyperparameters, it used a ReLU activation function, an Adam optimizer, mean square error as its loss function, batch size of 10000, and a learning rate of 0.01.

Some experimentation was conducted for the hidden layer structure of the ANN, with the rest of the hyperparameters remaining the same. A new model was trained with 2 fully connected hidden layers of size 20 and 10 units. Another model was trained with 3 fully connected hidden layers of size 200, 100, and 100 units. However, both models performed worse than the original ANN.

The dataset to train the model was collected using the simulated Pepper. 2 million data points were collected by sending random shoulder and elbow joint angles to the simulated robot. Using TF transform lookup function provided by ROS [28], the wrist and elbow coordinates were obtained.

The model used the sum of the mean absolute error of each joint angle as its accuracy marker. The model achieved its best result at epoch 82 with a total validation error of 0.31 radians. Individually, the errors for each angle were 0.1 radians for shoulder pitch, 0.017 radians for shoulder roll, 0.15 radians for elbow yaw, and 0.03 radians for elbow roll. After the 100th epoch the training error would gradually taper off and slowly start to overfit. The performance of the model can be seen in Figure 8. The loss function has a similar graph, only squared, thus was not included.

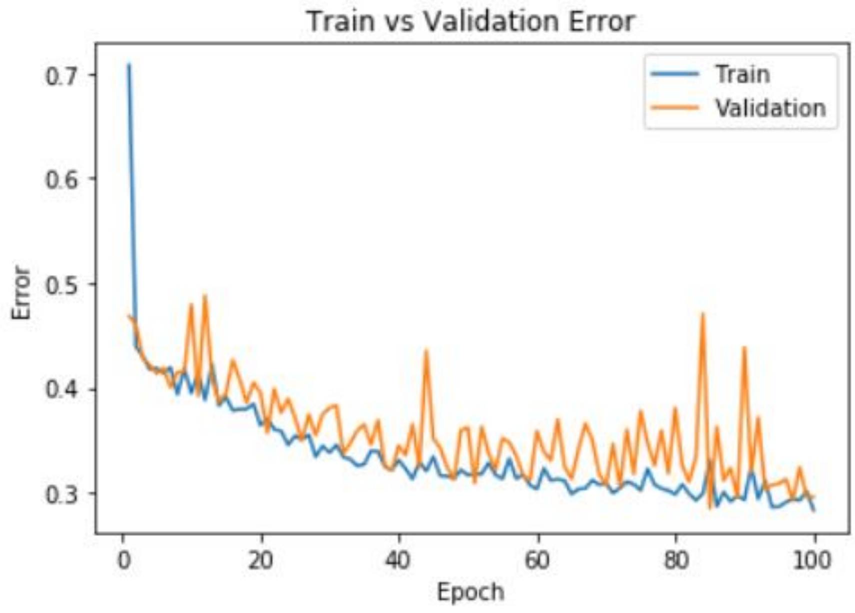

Figure 8. Model Error Over 100 Epochs

The validation curve is very volatile with many spikes and jumps. That is because for certain arm configurations, there exists infinite solutions when only considering the elbow and wrist coordinates. When the arm is outstretched to the side, the shoulder pitch can turn at any angle and the elbow and wrist coordinate would still be the same, and likewise for the elbow yaw. This confused the model as there existed data points with similar elbow and wrist coordinates with vastly different joint angles. That is why the shoulder pitch and elbow yaw have much higher errors than the shoulder roll and elbow roll.

Another drawback of this method is that it is not scalable. If we decided to incorporate the hip and knee movements, the model would fail to output the shoulder and elbow joint angles since it assumes the hip and knee are stationary. The model would have to be retrained with the hip and knee joint angles as input nodes.

Despite these issues, the model performed well in real life and was able to output appropriate shoulder and elbow joint angles for the majority of arm configurations.

5.3. Motion Controller

The NAOqi API provides a module that facilitates robot movement, controlling directly the position of the robot joints. It also provides interpolation between the current joint angles and target joint angles along a timed trajectory to achieve a smoother behavior. The module takes in an input of 6 arrays: the joint names we wish to control, the desired joint angles for each joint name, and the desired times (in seconds) it takes for each joint to reach the desired angle. Due to velocity limits of up to 0.56 m/s for the joints [4], the time could not be too fast. Each joint can be controlled in parallel with other joints.

6. Demonstration Study

An assessment of our motion mapping system is conducted. In this section, the human demonstration, the setup of the study, and results of the study are explained.

6.1. Demonstration

For the demonstration of Pepper to mirror human movement, the human demonstrator was required to stand in front of the camera about 1 meter away so all its upper body movement could be captured in the camera's field of view. Also, the human demonstrator refrained from moving in certain ways, such as hugging themselves or swinging their arms behind their back, motions which Pepper could not physically achieve.

The motion mapping system allows for two types of demonstration: real time mimicry and recorded motion. Some issues with real time mimicry is that it is delayed by network latency, as Pepper movement lags behind the human movement by a second. In addition, when the skeleton-tracking software loses track of the human arm positions during real time demonstration, Pepper's arm movement becomes chaotic. For this demonstration study, recorded motion was used since it could be played back multiple times, and we could re-record the human motion if the skeleton-tracking software lost track of the human arm positions.

When the recording is played back, the action unit retrieves the latest skeleton data. If a motion is too fast, Pepper misses that motion while joints move to target angles. For example, it could not fully capture the arms shaking in fear. In cases such as those, it was necessary to adjust the playback speed of the recording such that it could capture the quick and subtle motions.

6.2. Study Setup

To assess the quality of the motion mapping, we recorded a human demonstrator performing Ekman's six basic emotions: happy, sad, angry, fear, surprise, and disgust. Those emotions were selected because they are universal across all cultures [5]. The recordings were played back for Pepper to mirror the human demonstrator. The human movements were deliberate and exaggerated to make the motion more apparent. A snapshot of each motion can be seen in Figure 9, and a description of each motion is shown in Table 3. We took videos of Pepper enacting the emotions and asked survey participants to match the video of Pepper motion to one of Ekman's six emotions. If participants can identify which emotion Pepper is trying to convey better than chance, the motion mapping is deemed successful in capturing and mirroring human movement.

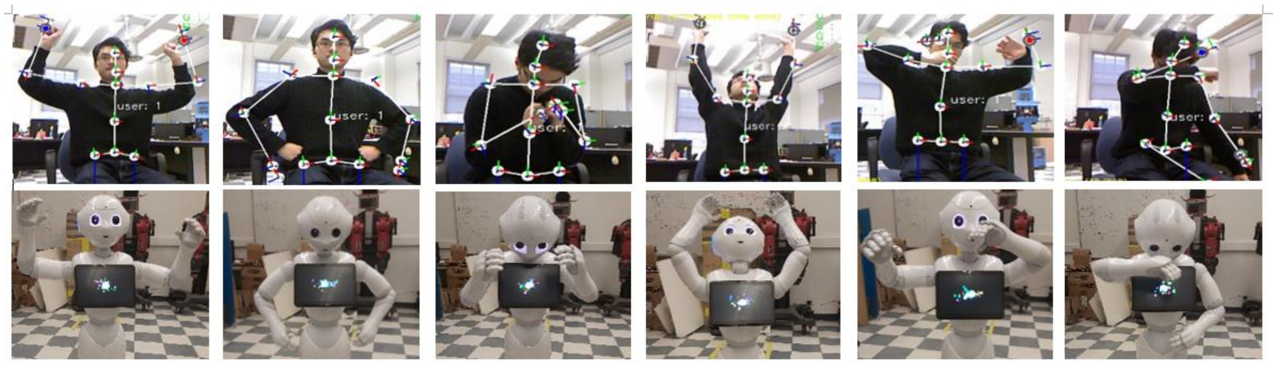
Figure 9. Snapshot of a human demonstrator and Pepper performing Ekman's six basic emotions, which are from left to right: happy, angry, fear, surprise, disgust, and sad.

Emotion	Motion Description
Happy	Pepper raises its arms multiple times enthusiastically
Angry	Pepper places its hands on its hips and tilts its head slightly down
Fear	Pepper looks down and covers its head with its shaking arm
Surprised	Pepper suddenly swings its arm forward and looks up, as if being blown away
Disgust	Pepper covers its mouth with one arm and whips its other arm sideways to signal get away
Sad	Pepper looks down and drags its arm across its eyes

Table 3. Description of the human motion representing Ekman's six emotion

12 participants were asked to do the survey. The 12 participants were all undergraduate university students in their fourth year of study, consisting of 7 females and 5 males, at the age of 21 or 22. The survey was conducted using the Google Forms online questionnaire tool. It displayed the videos in a row and the emotions in a column and forced a one-to-one matching between the videos and emotion.

6.3. Study Result

Overall, participants were able to match the right emotion with the right videos at a rate much better than chance of 17%. The highest accuracy rate among the six emotions were sadness, happiness, and surprise, with at least 75% of the participants getting them correct. The lowest were disgust and fear at less than 60% of participants getting them correct. The full survey result can be seen in Table 4.

	Happy	**Anger**	**Fear**	**Surprise**	**Disgust**	**Sad**
P1	Happy	Anger	Fear	Surprise	Disgust	Sad
P2	Happy	Anger	Fear	Surprise	Disgust	Sad
P3	Happy	Anger	Surprise	Fear	Disgust	Sad
P4	Happy	Anger	Disgust	Surprise	Fear	Sad
P5	Happy	Disgust	Anger	Surprise	Fear	Sad
P6	Happy	Anger	Fear	Surprise	Sad	Disgust
P7	Fear	Happy	Anger	Surprise	Disgust	Sad
P8	Happy	Anger	Fear	Surprise	Disgust	Sad
P9	Happy	Anger	Disgust	Surprise	Fear	Sad
P10	Surprise	Happy	Fear	Anger	Disgust	Sad
P11	Happy	Anger	Sad	Surprise	Disgust	Fear
P12	Happy	Disgust	Anger	Surprise	Sad	Fear
%	83	66	42	83	58	75

Table 4. Survey result

The survey link was sent out to each participant individually through the Facebook Messenger platform. After the survey was completed, participants were encouraged to share any of their thoughts or comments, with feedback immediately collected through the Messenger platform. From participant feedback, all the participants found it difficult to decipher the emotions through only body movement and had to take educated guesses, as they are more accustomed to looking at facial features for clues. Certainly, some body movements can be interpreted as one or the other. For example, fear and disgust were often mistaken for each other.

Indeed, other analyses could have been performed to assess the motion mapping quality, however due to external circumstances leading to lab closure, other analyses could not be performed at this time.

7. Discussion

The results showed that the motion mapping system achieved its goal of adapting human movement to Pepper robot by having participants match Pepper motion to Ekman's six emotions better than chance. However, it was not flawless as many improvements could be made to the current system. The results of the study also showed some emotions were harder to match than others. In the following sections, the discrepancy between each emotion's matching accuracy is discussed, technical improvements in the existing design is suggested, and the hardware limitations of the current system is examined.

7.1. Discrepancy in Study Result

Happiness and surprise had the most success with 83% matching accuracy rate. For happiness and surprise, the movement was faster than for the other emotions. Happiness also had much more movements than suprise which only had a single swinging motion of the arm. Both motions kept an open body language in which the arms were not covering the torso or face. Based on those motion characteristics, participants were able to match happiness and surprise more easily. Another emotion that had great success was sadness with 75% matching accuracy rate. Majority of participants recognized the motion of dragging the arm across the face with the head tilted down as indicative of the wiping of tears away after crying.

Fear and disgust had the worst accuracy with 42% and 58% respectively. For fear, the shaking of the arms was interpreted in many different ways to different people. Some thought Pepper was shaking its arms in anger, or in disgust, or in a surprised shock. The motion was not distinctive enough for participants to see that Pepper was conveying fear. For disgust, the motion of covering the mouth and whipping the arm to signal go away was interpreted many times as fear. Some also thought the motion indicated sadness. People usually express disgust through facial expressions and not through exaggerated movements, thus some participants found it difficult to match that emotion correctly. It is not uncommon that disgust is confused in these types of surveys for emotion categorization through body movements, as other research with similar surveys also had the same result [29][30][31].

7.2. Technical Improvements

Many improvements could be made to the current system. Starting with the skeleton-tracking software, there are occasional glitches when the software loses track of the arm. OpenPose is another skeleton-tracking software supported by CMU that could potentially solve the issue. It also tracks facial features and fingers, which could be integrated to the system in the future. OpenPose does require greater computational power and would have to rely on a GPU.

For the deep learning model data collection process, because it sent randomized joint angles, we collected some arm configurations that humans could not achieve. A filter could have been implemented to discard data points that resulted in arm configurations that are not humanly possible, thus improving the quality of the data collected.

For the deep learning model itself, more hyperparameter tuning could have been performed, such as experimenting with weight decay and dropout. However, the majority of the issue comes from having infinite solutions of joint angles for certain arm configurations. We could have integrated a hand-tracking software that determines the orientation of the hand (e.g. which way the palm is facing) and use that data with the desired elbow and wrist coordinate to determine a single solution for that arm configuration. We could have also combined that data with the previous joint angles to make sure the motion (change in joint angles) was smooth.

For the next steps, more joint angles can be integrated to the system, such as the head yaw, knee pitch, and hip joint. For the head yaw, which is the turning of the head, a facial tracking system would be required. Also, by integrating the hip or knee joint, we would have to retrain the model, and use a different system for mapping the head pitch, the nodding of the head, as well, since the current system assumes the hip and knee joint are stationary.

7.3. Hardware Limitations

As mentioned before, hardware limitations include Pepper's limited motion range and DoFs, restricting Pepper from certain movements that humans can achieve. The velocity limit of the Pepper joints prevented the human demonstrator from moving too fast, or the recording had to be played back slowly, which inhibited the robot from mirroring human movement at its natural speed. Network latency also had an effect in Pepper mirroring human motion in real time. Quicker movements could not be mirrored as Pepper lagged behind the human demonstrator. Regarding the Kinect sensor, its field of view and distance of use could be improved by using an ASUS Xtion Pro Live instead.

8. Conclusion

To enable better human-robot interaction, robots need to learn how to move like humans. For robots to learn anthropomorphic movement through learning from demonstration, a motion-mapping system is required. In this paper, we described our approach to designing a human to robot motion mapping system for the humanoid robot Pepper, focusing mainly on the arm and head motion. We used a video-based motion capture system using Microsoft Kinect sensor and Nuitrack skeleton-tracking software. We demonstrated that a deep learning solution can be used to find the Pepper joint angles given a desired arm configuration, with worst case error of 0.15 radians. The system was written as a ROS package, which can be integrated easily into other ROS projects. To test the motion-mapping system, we took videos of Pepper mirroring a human demonstrator performing Ekman's six basic emotions and asked participants to match the video of Pepper to the emotion Pepper is trying to convey. Participants were able to match the right videos and emotion better than chance, with the worst accuracy at 42% with fear. The system can be improved with a more robust skeleton-tracking software, collecting better data for the model, and tuning the hyperparameters of the model. For future work, more joint angles can be incorporated to the system, such as hip joint, knee pitch, and head yaw. Also, we could track the orientation of the hand through a hand-tracking software and use that data to solve the issue of having infinite solutions for the same arm configuration.

9.0 References

[1] A. Joseph, B. Christian, A. A. Abiodun, and F. Oyawale, “A review on humanoid robotics in healthcare,” MATEC Web of Conferences, vol. 153, p. 02004, 2018.

[2] L. Damiano and P. Dumouchel, “Anthropomorphism in Human–Robot Co-evolution,” Frontiers in Psychology, vol. 9, Mar. 2018.

[3] A. Billard and D. Grollman, “Robot learning by demonstration,” Scholarpedia, vol. 8, no. 12, p. 3824, 2013.

[4] S. Robotics, “pepper”, Pepper Datasheet 1.8a, 2017. [Online]. Available: https://www.generationrobots.com/pepper/Pepper%20Datasheet%201.8a%2020170116%20EMEA.pdf

[5] P. Ekman and W. V. Friesen, “Constants across cultures in the face and emotion.,” Journal of Personality and Social Psychology, vol. 17, no. 2, pp. 124–129, 1971.

[6] N. Tian et al., “A Cloud-Based Robust Semaphore Mirroring System for Social Robots,” in 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE), 2018.

[7] S. Thellman, J. Lundberg, M. Arvola, and T. Ziemke, “What Is It Like to Be a Bot?,” in Proceedings of the 5th International Conference on Human Agent Interaction - HAI ’17, 2017.

[8] N. S. Pollard, J. K. Hodgins, M. J. Riley, and C. G. Atkeson, “Adapting human motion for the control of a humanoid robot,” in Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292).

[9] J. Oh, I. Lee, H. Jeong, and J.-H. Oh, “Real-time humanoid whole-body remote control framework for imitating human motion based on kinematic mapping and motion constraints,” Advanced Robotics, vol. 33, no. 6, pp. 293–305, Mar. 2019.

[10] M. V. Liarokapis, P. K. Artemiadis, and K. J. Kyriakopoulos, “Functional Anthropomorphism for human to robot motion mapping,” in 2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication, 2012.

[11] Y. Mohammad and T. Nishida, “Tackling the Correspondence Problem,” in Active Media Technology, Springer International Publishing, 2013, pp. 84–95.

[12] B. Spanlang, X. Navarro, J.-M. Normand, S. Kishore, R. Pizarro, and M. Slater, “Real time whole body motion mapping for avatars and robots,” in Proceedings of the 19th ACM Symposium on Virtual Reality Software and Technology - VRST ’13, 2013.

[13] Aldebaran, “NAOqi APIs”. [Online]. Available: http://doc.aldebaran.com/2-4/naoqi/index.html

[14] S. Saunderson and G. Nejat, “It Would Make Me Happy if You Used My Guess: Comparing Robot Persuasive Strategies in Social Human–Robot Interaction,” IEEE Robotics and Automation Letters, vol. 4, no. 2, pp. 1707–1714, Apr. 2019.

[15] C. Thompson, S. Mohamed, W.-Y. G. Louie, J. C. He, J. Li, and G. Nejat, “The robot Tangy facilitating Trivia games: A team-based user-study with long-term care residents,” in 2017 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS), 2017.

[16] P. Bovbel and G. Nejat, “Casper: An Assistive Kitchen Robot to Promote Aging in Place1,” Journal of Medical Devices, vol. 8, no. 3, Jul. 2014.

[17] C. Nock, O. Taugourdeau, S. Delagrange, and C. Messier, “Assessing the Potential of Low-Cost 3D Cameras for the Rapid Measurement of Plant Woody Structure,” Sensors, vol. 13, no. 12, pp. 16216–16233, Nov. 2013.

[18] Nuitrack. Nuitrack. [Online]. Available: https://nuitrack.com/

[19] Tensorflow. PoseNet. [Online]. Available: https://github.com/tensorflow/tfjs-models/tree/master/posenet

[20] CMU Perceptual Computing Lab. OpenPose. [Online]. Available: https://github.com/CMU-Perceptual-Computing-Lab/openpose

[21] PrimeSense. NITE. [Online]. Available: http://wiki.ros.org/nite

[22] D. Shinsel. Nuitrack Body Tracker. [Online]. Available: https://github.com/shinselrobots/nuitrack_body_tracker

[23] A. Laurens. Nuitrack Body Tracker. [Online]. Available: https://github.com/alaurens/nuitrack_body_tracker

[24] Y. Suga. Pepper Kinematics. [Online]. Available: https://github.com/sugarsweetrobotics/python_pepper_kinematics

[25] V. Berenz. Playful Kinematics. [Online]. Available: https://github.com/vincentberenz/playful_kinematics

[26] Aldeberan, “Joints”. [Online]. Available: http://doc.aldebaran.com/2-4/family/pepper_technical/joints_pep.html

[27] B. Philips, “Regression with Neural Networks in PyTorch”, Medium, Dec. 2018. [Online]. Available: https://medium.com/@benjamin.phillips22/simple-regression-with-neural-networks-in-pytorch-313f06910379

[28] ROS. TF. [Online]. Available: http://wiki.ros.org/tf

[29] R. Calvo, S. D’Mello, J. Gratch, A. Kappas, M. Lhommet, and S. C. Marsella, “Expressing Emotion Through Posture and Gesture,” in The Oxford Handbook of Affective Computing, Oxford University Press, 2015.

[30] E. P. Volkova, B. J. Mohler, T. J. Dodds, J. Tesch, and H. H. Bülthoff, “Emotion categorization of body expressions in narrative scenarios,” Frontiers in Psychology, vol. 5, Jun. 2014.

[31] T. Sapiński, D. Kamińska, A. Pelikant, and G. Anbarjafari, “Emotion Recognition from Skeletal Movements,” Entropy, vol. 21, no. 7, p. 646, Jun. 2019.