SOFTWARE ARCHITECTURES FOR ROBOTICS

Mapping of upper body human motion to Baxter kinematics

Valentina Pericu

Luca Morando

1/02/2018

1. Abstract

The aim of the project **"Mapping of upper body human motion to Baxter kinematics"** was the online teleoperation of the robot Baxter, in order to make him reproduce human movements. To do this we have collected motion data of the left arm of Baxter (with the use of ROS commands) and of the human arm (with the use of motion capture) making this last follow the robot movements. After having extracted the data of interest through a node in C++ language, created by ourselves, we have been able to collect them on a unique text file. Afterwards we have used all of these information in order to train, in MATLAB environment, a Neural Network based mapping between Baxter and human. After having checked that the training was completed, verifying that the resulting error was acceptable, we have used the network to generate, by putting as inputs the human left arm coordinates, the outputs, corresponding to the joints positions of Baxter. This has made possible the teleoperation of the robot arm. After the creation of a node in MATLAB we have been able to manage the data flow in inputs and outputs.

2. Introduction

The project is focused on the position mapping of human and Baxter in order to make possible the Baxter teleoperation, that is a remote control of the robot. There are several particular types of systems that are often controlled remotely: industrial machinery, remotely operated vehicles (used in hazardous environments, such as radioactive or contaminated environments, deep oceans), remote surgery (that is the ability for a doctor to perform surgery on a patient even though they are not physically in the same location), drones.

3. Data Acquisition

3.1. Baxter Movements

Baxter is an industrial two-armed robot built by Rethink Robotics, each arm is provided with 7 joints (Figure 1: Baxter arm joints) The first step of our project was connecting our laptop with Baxter. We have worked only with the left arm of the robot. After that we have launched the ROS command to record his movements and, at the same time, we have manually moved his left arm making it perform the desired motion. In this way it is created a text file containing the time and the joints positions, both of left and right arms.

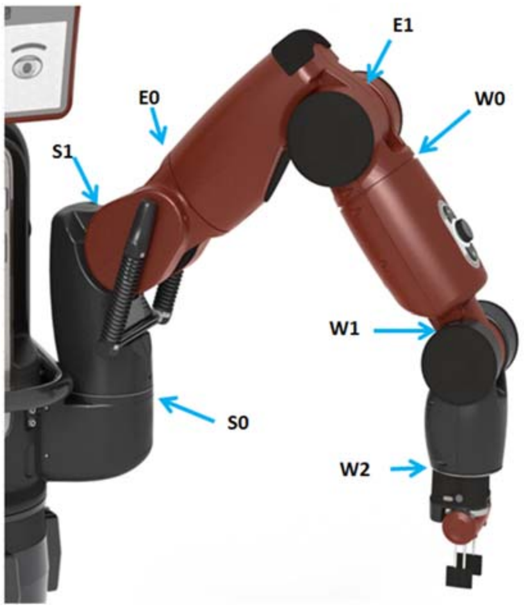


Figure 1: Baxter arm joints

Afterwards, by launching the ROS command for the playback, Baxter reads all the data from the text file previously created and reproduces faithfully the movements. We have made these assumptions:

- joints S0, S1 and E0 correspond to the shoulder
- joints E1 and w0 correspond to the elbow
- joints w1 and w2 correspond to the wrist

3.2. Human Movements

The second step w collecting the human arm motion data. Previously we decided to consider in the human arm 4 rigid bodies: shoulder, elbow, wrist and hand. We have identified each one of them with 4 markers. To acquire the coordinates of the rigid bodies we have used the motion capture, performed with the use of 8 cameras and the software Motive.

3.3. Synchronous Baxter-Human Movements – the sync_node

After having chosen one of the previously recorded motion of Baxter, we have launched the sync_node and the ROS playback command and we have made the human follow the robot movements. As result we have obtained a text file containing both Baxter joints positions both human arm coordinates.

The sync_node is a node situated inside the package Sync. It has several dependencies (type of messages exchanged) and it has the aim of receiving data from different publishers and writing them on a text file. The sync_node is launched simultaneously to the ROS playback command. In this way Baxter starts executing the chosen movement and the human, by remaining in the motion capture area, must replicate in real time. The sync_node has been fundamental for acquiring data simultaneously from motion capture and Baxter left arm joints and for writing them online on the same text file. In this way we have been able to create a matrix in MATLAB environment necessary to train the Neural Network. The sync_node is a C++ script where we have 5 different callbacks to 5 different Topics: 4 for motion capture and 1 for Baxter.

3.3.1. Motion Capture Callback

For acquiring data from the motion capture we have modified the file mocap.yaml in order to have 4 different Topics (Robot_1/pose, Robot_2/pose, Robot_3/pose, Robot_/pose), each one necessary to get the coordinates value from each one of the 4 rigid bodies of human arm. The messages exchanged inside this Topic are geometry_msgs of PoseStamped type. The message PoseStamped is defined by a subsection Pose, subdivided in Position and Orientation. Inside each callback we have a pointer only for the Position information, defined in x, y, z coordinates. The data are stored inside different array previously allocated. We have made this assignment:

- Robot_1 for the shoulder
- Robot_2 for the elbow
- Robot_3 for the wrist
- Robot_4 for the hand

3.3.1. Baxter Callback

We have the necessity of acquiring data from Baxter joint_states in order to obtain the positions of Baxter joints. To do this, we have created the connection at the topic /robot/joint_states. Inside this topic the position joint data are exchanged in form of sensor_msgs/ JointState. This type of messages has different information inside it. We are only interested in extracting Baxter joints name and position (JointState.name and the JointState.position). The acquisition has required some knowledge about vector manipulation in C++ in order to isolate each string of the joint name and each position value for the corresponding joint name inside two arrays (respectively named names and joints). After all the connection has been established, by setting a flag on true for the 4 mocap topics and Baxter topic, it's possible to enter inside the ROS while loop set on rate of 60 Hz in order to acquire all the data correctly. In the while loop it is possible to write on the txt file, previously created, with the correct name, chosen by the user at the creation. We have written on the txt file only the information of our interest (that is the left arm data) stored in the name and position arrays. The different joints name are published only one time at the beginning of the txt file. The second row is occupied by Baxter joints position values and human arm coordinates (x, y, z) for each one of the 4 rigid bodies. The file.txt result organized in 19 columns: the first 7 for Baxter joints data and the next 12 for human coordinates position data. For every row of Baxter joints positions value we have the corresponding human coordinates. When the movement of Baxter was done correctly the sync_node was correctly stopped by ctrl+C.

4. Neural Network

4.1 MATLAB Pre-processing

We have created a MATLAB script in order to prepare and order all the data necessary to the Neural Network. In it we have divided the 19 columns matrix returned by the sync_node into two sub-matrices, the first obtained by extracting the first 7 columns (joint state matrix) and the second obtained by extracting the last 12 columns (human coordinates matrix). After we have chosen the shoulder as referring frame for every rigid body in the human arm.

4.2 Neural Network

The Neural Network used in this project was the MATLAB Neural Network for fitting training given by the NN tool. After the creation on the automatic script.m, instead of using GUI, we have understood and manipulated it in order to define the inputs, the outputs, the number of hidden layers (neurons) and the different size of dataset given for training, validation and testing. The input of the Neural Network is the matrix of 12 columns and tot rows representing the human coordinates, while the desired output is the matrix of 7 columns and tot rows representing the Baxter joint positions value.

The Neural Network has been trained for 2 days, incrementing the size of dataset and the different part of it used for training, validation and testing. The values used for testing the Neural Network are random (we gave the percentages to the dataset), calculated by the MATLAB script. For avoiding an abrupt oscillation between the value used for training, the validation values are used for making an interpolation.

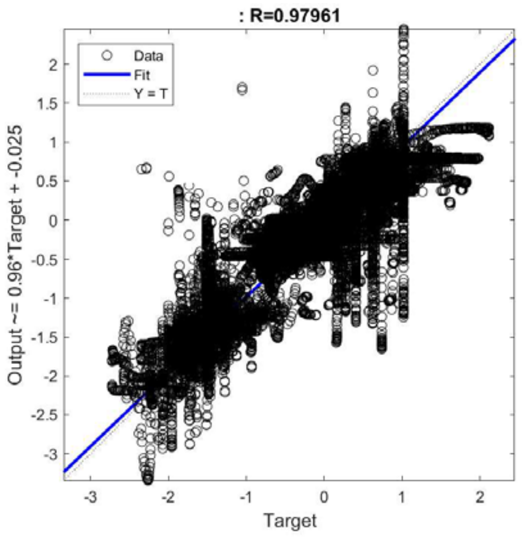
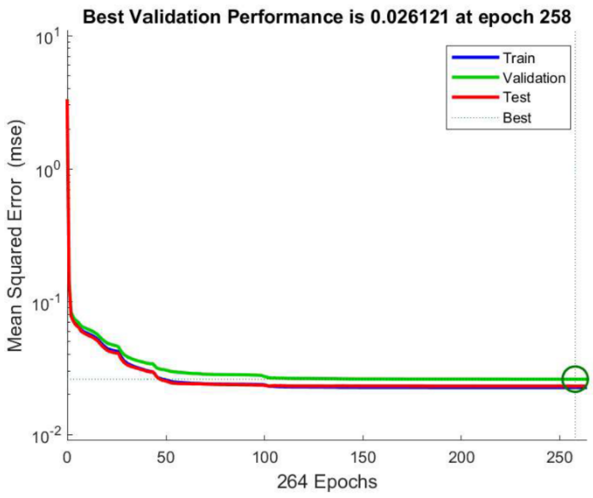


Figure 2: Linear regression of fitting data

The 2 days of training of the neural network were done in order to decrease the error from the actual output and the desired one.



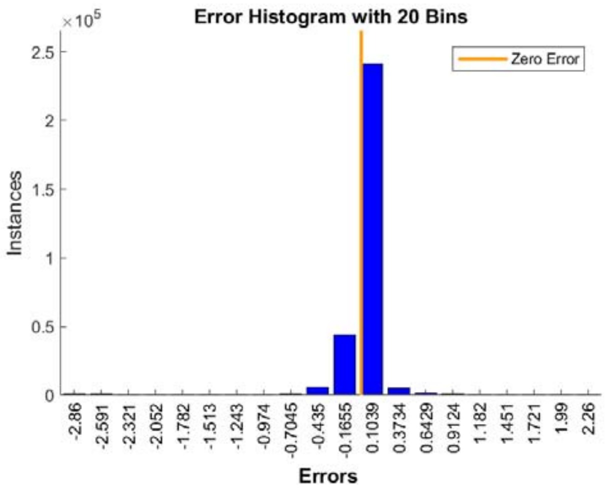


Figure 3: error distance from best validation

Figure 4: error histogram

We have obtained a very small error (an acceptable error is of the order of 10^{-1}) that has offered us the possibility of generating the robot position by starting from the human arm coordinates. Always in MATLAB environment we have created the myNeuralNetworkFunction which provides as output the joint positions, giving as input the human arm coordinates.

Now we have the keys of our work: by starting from a vector of 12 coordinates acquired online by Motion Capture we can generate a vector of 7 joint positions, able to reproduce a movement similar to the human one.

The script Robot_Teleoperation.m is the final achievement of our work: it allows us to connect with all the necessary nodes (ReadMe) to teleoperate online the robot, through a while loop in which the myNeuralNetworkFunction is used. This script represents a subscriber for the 5 topics and a publisher for Baxter, through the topic /robot/limb/left/joint_command by opportunely setting the message information, controlling only the posture (POSITION_MODE), leaving Baxter the task to perform the inverse kinematics and calculate the velocity.

In order to obtain a better result, it is necessary to increment the dataset.