Recurrent Neural Networks for Hierarchically Mapping Human-Robot Poses

Zainab Al-Qurashi and Brian D. Ziebart

University of Illinois at Chicago, Chicago, USA
{zalqur2, bziebart}@uic.edu

Received (03/15/2021)

Accepted (03/31/2021)

Abstract. To perform many critical manipulation tasks successfully, human-robot mimicking systems should not only accurately copy the position of a human hand, but its orientation as well. Deep learning methods trained from pairs of corresponding human and robot poses offer one promising approach for constructing a human-robot mapping to accomplish this. However, ignoring the spatial and temporal structure of this mapping makes learning it less effective. We propose two different hierarchical architectures that leverage the structural and temporal human-robot mapping. We partially separate the robotic manipulator's end-effector position and orientation while considering the mutual coupling effects between them. This divides the main problem—making the robot match the human's hand position and mimic its orientation accurately along an unknown trajectory—into several sub-problems. We address these using different recurrent neural networks (RNNs) with Long-Short Term Memory (LSTM) that we combine and train hierarchically based on the coupling over the aspects of the robot that each controls. We evaluate our proposed architectures using a virtual reality system to track human table tennis motions and compare with single artificial neural network (ANN) and RNN models. We compare the benefits of using deep learning neural networks with and without our architectures and find smaller errors in position and orientation, along with increased flexibility in wrist movement are obtained by our proposed architectures. Also, we propose a hybrid approach to collect the training dataset. The hybrid training dataset is collected by two approaches when the robot mimics human motions (standard learn from demonstrator LfD) and when the human mimics robot motions (LfDr). We evaluate the hybrid training dataset and show that the performance of the machine learning system trained by the hybrid training dataset is better with less error and faster training time compared to using the collected dataset using standard LfD approach.

Keywords: Deep neural networks; Recurrent neural networks; Long-short term memory; Learn from demonstration; Robot kinematics; Virtual reality.

1 Introduction

In some tasks, such as surgical operations, focusing a camera on an object while moving, carrying a filled cup, or playing table tennis, precisely mapping the position and orientation of the human’s hand—and wrist particularly—to the robot end-effector is very important to enable successful robot imitation. Motion capture, depth cameras, and virtual reality tracking systems provide wrist position and orientation tracking with varying accuracies. However, even with precise tracking data of the human, kinematic differences between human and robot arms complicate this mapping significantly, preventing handcrafted mappings from being effective [20,17].

In this paper we present two hierarchical architectures based on deep learning, which are used to map between two different workspaces (human and robot workspaces). Also, we propose a hybrid method to collect the training data practically, which is required to train the proposed architectures to perform mapping between the human and robot workspaces. The architectures map the human’s hand position and orientation to the Baxter’s robot end-effector. The two workspaces are separated and work independently. The human’s hand position and orientation are sensed and defined in its workspace (human workspace) using HTC-Vive virtual reality system. The Baxter robot system has its own workspace and coordinate system, which is used to define the position and orientation of its end-effector. Therefore, in the human-robot mimicking systems, it is important to map the human’s hand position and orientation defined in the human’s workspace to the robot’s workspace. This is the task of the proposed mapping architectures. It is important to notice that regardless of the strategy that is used to compute the required joint variables of the robot, a mapping system needs to be used to map the two independent coordinate systems. Therefore, even if the robot inverse kinematic (IK) solver is known and can be used to solve the inverse kinematic problem to find the required robot’s joint variables, the mapping system is still required to map the human’s hand position and orientation from the virtual reality coordinate system (or the human workspace) to the robot coordinate system, in which the IK solver can be used to find the required joint variables.

The proposed mapping architectures in this work perform two tasks, they perform the mapping from the human’s workspace to the robot’s workspace and find the required joint variables of the robot system. Thus, with the proposed architectures, there is no need to employ an IK solver, which is sometimes difficult to use, as it generates more than one solution, requiring more effort to decide which solution is better. However, sometimes IK solver even fail to find any solution. In addition, merging the mapping problem and IK problem in a one compact system (the proposed architectures) reduces the transformation error as now the data is transformed directly from the first start point in the human system (human’s hand position and orientation) to the last endpoint in the robot

system (robot’s end-effector position and orientation); while with the IK solver the human information should be transferred to the robot’s workspace and then IK is used to obtain the robot joint variables. Therefore, this extra middle stage is eliminated in the proposed architectures.

One approach for obtaining a more accurate mapping is to collect pairs of human and robot poses that demonstrate points in the desired mapping and then to employ machine learning to generalize beyond those specific points to provide an entire mapping function [5]. Deep neural networks are a promising technique for non-linearly modeling this relationship between human and robot pose descriptors. However, employing these models in a black box manner that ignores spatial and temporal structure can be much less effective, requiring significantly more data to reach a desired level of performance.

To better leverage the temporal and spatial structure of control tasks, we propose human-robot mapping architectures for teleoperating robotic arms using hierarchical deep learning. Our approach partially separates the motion mapping between the human and the robot into a position mapping and an orientation mapping. We do this by dividing the robot’s manipulator structure into two parts: the robot-arm and the robot-wrist. Of key importance, this focuses our attention on the wrist vector (wv) where these parts connect. By doing this, we reduce the main problem into several sub-problems while considering the effects of the mutual coupling among them. Rather than using a mapping for pairs of poses, we employ temporal models that map from a trajectory (i.e., history of poses) to a new robotic pose. Specifically, we use hierarchical deep learning—the recurrent neural network with long-short term memory (RNN-LSTM)—to realize this. By using this technique, we develop mappings for each sub-problem individually and then obtain a solution to the entire human-robot mimicking problem using a hierarchical RNN-LSTM.

We explore two different architectures for mapping from human motion, as tracked by a virtual reality system, to robotic motions. They differ in how they compute the robot-wrist joint variables. Our architectures take into account both the temporal and the spatial structure to provide benefits over networks that ignore the spatial structure or temporal structure. To evaluate the performance of our proposed architectures, we compare the performance of human-robot mapping against basic deep learning models: a single ANN and a single RNN. This comparison is conducted to show empirically that using the neural network alone (by using single RNN or single ANN) is not enough to make the robot mimic the human’s hand position and orientation (especially the human’s wrist) accurately. To train the architectures, we use the HTC-Vive virtual reality to track human motion. In the standard LfD approach the robot is moved manually by the robot-mover who try to position the robot’s arm to mimic the human’s arm (teleoperator or demonstrator). While, in the proposed approach to collect the data, we set the robot motion in advance and ask the demonstrator to follow and mimic the robot’s end-effector. We demonstrate and evaluate our architectures using a Baxter robot’s manipulator (Figure 1).

Fig. 1. The Baxter robot we use in our experiment with each side is divided into two parts: robot-arm and robot-wrist.

2 Background and Related Work

2.1 Recurrent Neural Networks (RNN)

The Recurrent Neural Network is a deep neural network with feedback loops that enable the network to learn sequences and execute temporal processing [14]. The RNN has the ability to capture complex, nonlinear system dynamics [1]. The RNN's time steps are an important component of the RNN's performance. We employ RNNs with Long-Short Term Memory (LSTM) in our proposed architectures due to its power in modeling the dynamics and dependencies in sequential data. The LSTM is a great fit for modeling time-varying information, including that found in robotics applications.

LSTM architecture replaces the nonlinear units in traditional RNNs and avoids the long-term dependency problem. It contains one self-connected memory cell c and three multiplicative units, which are input gate i , forget gate f , and output gate o , that can store and access the long range contextual information of a temporal sequence. The activation of the memory cell and three gates are computed as follows [4]:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (2)$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (3)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \quad (4)$$

$$h_t = o_t \tanh(c_t), \quad (5)$$

where the input of the current LSTM block is x_t , the output of the previous LSTM block is h_{t-1} , c_{t-1} , the memory of the previous block and b is the bias vector. $\sigma(\cdot)$ is the sigmoid function, and W are the weight matrices.

2.2 Related Robotics and Virtual Reality Work

Hierarchical RNNs have previously been employed for skeleton-based action recognition [4] by dividing the human skeleton into parts based on the human physical structure. Each part is separately fed to a sub-network. Extending from this, spatio-temporal LSTMs learn the reliability of the sequential input data and based on the long-term context information stored in memory cells [9]. A Global Context-Aware Attention LSTM (GCA-LSTM) [10] for 3D action recognition enhances the attention capability of LSTM to achieve a reliable attention representation for the action sequence.

In our proposed architectures, we employ the RNN-LSTM to model non-linear relation for our human-robot mimicking system, which makes the robot follow the human’s hand position and mimic its orientation along unknown trajectories.

To capture the spatial structure, we divide the robotic manipulator into two parts: robot-arm and robot-wrist. This decomposition matches our proposed hybrid algorithm to address the inverse kinematic (IK) problem for a high DOF robotic manipulator [2].

Some deep imitation learning approaches for complex manipulation tasks use consumer grade virtual reality headsets and hand tracking hardware to naturally teleoperate robots to perform tasks by collecting high-quality demonstrations suitable for learning [23]. They train deep visuomotor policies that directly map from pixels to actions using behavioral cloning. They show that a simple imitation learning method with high-quality demonstrations is indeed effective. However, a deep visuomotor trained policy can fail when changes occur on the target object or if the background changes dramatically (e.g., altering the brightness of key features). We employ the union of the virtual reality HTC-Vive system, the RNN-LSTM, and the hierarchical architecture to map the position and orientation of the human’s hand (and the wrist in particular), to perform tasks that require flexibility in moving the wrist and copying the orientation of the human’s hand.

Past work [16] presented an interface that allows the user to directly control a robot arm by relaxing the constraint of the direct mapping between hand position, orientation and the end effector configuration. The human interacts with the real robot workspace, rather than the virtual workspace. Thus, the human sees everything in reality which allows them to instruct the robot easily and correct any unwanted robot motion immediately. This is limiting compared to our system.

Alignment of virtual and real coordinate spaces is a challenge in virtual reality research. Varying the offset between the virtual and the physical tracking space may influence control settings that requires high accuracy for mapping the position and the orientation [15,13,11]. An automated alignment and correction for the HTC Vive tracking system has been developed using Vive Trackers arranged to describe the desired axis of origin in the real space [15]. We enhanced the standard HTC Vive tracking system (one headset and two controllers) with

a tracker used to collect more information about the position and the orientation of the human’s wrist.

2.3 Learn from Demonstrator (LfD)

In the Learn from Demonstrator (LfD) data collection setting, in which the robot mimics the human, the demonstrator and robot-mover generate training data from the Baxter robot arm (7-DOF) and the HTC Vive virtual reality system which is used to collect information about the human pose. The demonstrator wears the HTC Vive headset and tracker (fixed on the wrist joint of the right hand), while holding two controllers, one in each hand. The Baxter manipulator is set to zero gravity mode so that all the joints and actuators are disabled. This allows the holder to move Baxter joints freely. The robot-mover holds Baxter’s right arm and mimics the motion of the demonstrator’s right hand. Both demonstrator and Baxter arms start moving from the same starting position at the same time. When the robot-mover copies the hand motions of the demonstrator, all seven of Baxter’s corresponding joint variables and the corresponding position and quaternions of the right-hand of the demonstrator are collected.

Though Learn from Demonstrator (LfD) is a powerful approach to teach the robot, it suffers from some difficulties. In this approach, the robot-mover should move the robot’s arm exactly at the same time with the demonstrator’s hand motion and make the robot’s end-effector follow the demonstrator’s hand position and mimic its orientation perfectly. In practice, this is combined with some difficulties. First, there will certainly be a delay between the demonstrator motion and the corresponding robot motion, as the two motions cannot be perfectly synchronized. This lack in the synchronization between the two motions can exist all the time during the data collection phase. Second, at certain times and positions, the robot-mover cannot mimic and synchronize the demonstrator’s hand position and orientation exactly. Also, there are hidden points where the robot-mover cannot see what the demonstrator’s movement is. Another source of inaccuracy of the collected data is the difference between the demonstrator and the robot-mover in the height and size, which sometimes and at some points makes the robot-mover can not make the robot’s arm mimics the demonstrator’s hand perfectly.

2.4 Hybrid Data Collection

To enhance the quality of the training data and to overcome the difficulties of the standard LfD approach, we propose using hybrid training dataset. We collect the training data by two different approaches and merge them to get the hybrid training dataset. The first approach is the standard LfD where the robot’s arm is moved by a robot-mover who tries to make it follows the demonstrator’s hand as described previously. In the second approach to collect the data, which we call it reverse LfD (LfDr), where instead of making the robot mimic the demonstrator motion as in the standard LfD, we propose having the demonstrator mimic the robot. This can be achieved by setting the robot to move on different pre-defined trajectories and perform different tasks with different end-effector positions and

orientations that are recorded to be watched by the demonstrator before he/she starts mimicking the robot’s end-effector motion to help the demonstrator learn the robot’s motion, which the demonstrator should copy and mimic perfectly as much as possible. To collect the training data, we make the robot move on the recorded trajectories, and we ask the demonstrator who sees the robot via the headset of the vive system to try to copy the motion of the robot’s end-effector exactly and synchronize their motion with the robot motion. The demonstrator will be able to mimic and synchronize their motion much better with the robot than when the robot-mover tries to make the robot’s end-effector mimic the demonstrator’s hand in the standard LfD approach. This is because the demonstrator in LfDr approach has prior knowledge about the trajectories of the robot motion that he/she try to copy during the real time operation to collect the training dataset. It is important to notice that LfDr approach does not require the robot-mover to move the robot’s arm as in standard LfD as the robot’s arm and end-effector move automatically without any external influence. This eliminates the error that can be introduced by the robot-mover in the standard LfD. The hybrid training dataset is constructed by combining the data collected by using both approaches LfD and LfDr. The latter can avoid the two main problems that combine the standard LfD, which could cause the collected data from demonstrator to be asynchronized with the data collected from the robot. The first problem is the variation in height and size between the demonstrator and robot-mover. At times, the latter sometimes is not able to make the robot reach some positions that the demonstrator reaches, because the demonstrator’s height and size are different than the robot-mover’s height and size. This variation could cause mapping mismatch between the corresponding points in the human and robot workspaces. The second problem is the hidden points, which may not be seen by the robot-mover as they are blocked by the demonstrator’s body. These points can prevent the robot-mover from copying the demo-human’s motion accurately.

3 Proposed Mapping Architectures

We propose two architectures for enabling the robotic manipulator’s end-effector to mimic the human’s hand position and orientation accurately. Both employ the same decomposition of the manipulator (Section 3.1) and use as input the same virtual reality tracking data (Section 3.2). They differ in terms of the hierarchical relationships between RNN components, as described in Section 3.3. Our proposed architectures take into account both temporal and spatial structures by using an RNN/LSTM and producing the prediction in a hierarchical manner, respectively.

3.1 Manipulator Decomposition

In order to simplify the entire human-robot mimicking problem and to capture spatial structure, we divide the robotic manipulator structure into two parts [2]. We call the first part, which includes the major or (large) series of links in the manipulator structure, the robot-arm. Due to the large links of this part, it makes

the main contribution to the robot's end-effector position. The second part, the robot-wrist, includes the small series of links in the manipulator structure and contributes mainly to the orientation of the manipulator end-effector. We consider two Cartesian coordinate systems: global and local coordinate systems. The global coordinate system is located at a fixed point at the central structure of the first joint in the robot. It describes the position of the robot's end-effector and the position of the robot-arm endpoint. The local Cartesian coordinate system lies on the robot-arm at the intersection of the two parts (robot-arm and robot-wrist) and has the same orientation as the global coordinate system at any position or time. The intersection point J between the robot-arm and the robot-wrist structures is computed using (6),

$$J = P_d - wv, \quad (6)$$

where P_d is the desired robotic manipulator's end-effector position based on the global coordinate system and wv is the wrist vector, which is the position of the robot's end-effector with respect to the local coordinate system (fixed on the robot-arm at point J).

The robotic manipulator needs at least 6 DOF to reach a desired point in its workspace with the desired orientation (reaches the desired position from any angle) [7]. Robotic manipulators with higher DOF have a kinematically redundant degree [22]. Any arbitrary rotation for the robotic manipulator's end-effector can be obtained by three individual rotations around the three axes in the wrist structure of the robotic manipulator. Thus, if a set of rectangular coordinate axes are fixed to the robot's end-effector, then its orientation may be expressed in a succession of rotations about each axis: the end-effector is first rotated about the z-axis by an angle ϕ , then about the rotated y-axis by an angle θ , and then again about the rotated z-axis by an angle ψ . The orientation of this robot's end-effector is described by the set of Euler angles ϕ , θ , and ψ [7]. We divide the 7 DOF Baxter robotic manipulator into two parts. The first part, the robot-arm has 4 DOF, corresponding to the major joints in the robotic manipulator: shoulder roll, shoulder pitch, elbow roll and elbow pitch. The second part, the robot-wrist has 3 DOF: wrist roll-1, wrist pitch, and wrist roll-2. These three degrees of freedom in the Baxter wrist structure provide the robot-wrist orientation as represented by Euler angles for the robot-wrist structure [6,22]. The main idea behind our decomposition takes the following steps:

1. It is possible to make the orientation of the robot-wrist part similar to the desired orientation of the whole robot's end-effector, since the robot-wrist part has three degrees of freedom (Euler's angles) that can be used to obtain the desired orientation of the robot-wrist.
2. Now it is important to compute the coordinate of the robot-wrist part (which we call wrist vector wv) in terms of a local coordinate system which is located at the beginning of the robot-wrist structure.
3. If we subtract wv from the desired position P_d of the robot's end-effector (which is the position of the human's hand in the human-robot mapping

- system provided by the HTC Vive), we can get the coordinate of the start point of the robot-wrist in terms of the global coordinate system (point J).
4. If the end-point of the robot-arm reaches the point computed in step 3, the robotic manipulator’s end-effector reaches the desired position and orientation.

The above steps explain briefly how we separate partially the position and orientation of the robot’s end-effector, while considering the coupling effect between them in our solution. We employ Hierarchical RNN to achieve our proposed mapping model. We apply our architectures on the Baxter robot’s right arm, though the same architectures and procedures could also be applied to the left side.

3.2 Virtual Reality System

Learning from Demonstration (LFD) has been a powerful paradigm for teaching robots behavior that they then mimic [23]. Obtaining high-quality demonstrations for mapping human motion to robot motion is a challenge especially for tasks like object manipulation with high DOF robotic arms. This is because human and robot arms do not have the same kinematics (structure). Virtual reality teleportation immerses the user in a 3D virtual environment and tracks the user’s hands, letting them specify locations and trajectories by simply moving their arm, as they would in the real world [20]. VR provides detailed data about the trajectories (i.e., human poses at each frame with fine-grained articulated hand poses) [8].

In this paper, we use the Vive virtual reality system from HTC to provide the operator with a virtual view of the robot’s workspace. We simulate the virtual environment in Unity and use ROS Reality package to connect the virtual reality HTC Vive to a ROS network [21,19]. The HTC Vive components, which we used in our experiments, are one headset, two controllers, and one tracker. The human wears the HTC Vive headset, fixes the tracker on the right wrist and holds one controller in each hand. The headset, controller and tracker provide the position (x, y, z) and quaternion (q_0, q_1, q_2, q_3) of the head, hand and wrist of the human pose, respectively.

3.3 Hierarchal RNN-LSTM Architectures

To capture temporal structure, we use RNN-LSTMs in our architectures. Each of our proposed architectures, A and B, consist of three RNN-LSTMs (see Figs. 2 and 3). Two of them are used in both architectures A and B (wv-RNN and arm-RNN), while the third RNN (wrist-RNN1 and wrist-RNN2) differs for the two architectures. Thus, there are four different RNN-LSTMs used and trained using the collected data as shown below. All of the training is performed off-line prior to the robotic operation.

Both architectures utilize the same inputs and generate the same outputs. The inputs are actual position of the human’s right hand P_R , quaternion of the human’s right hand Q_R , quaternion of the human’s head Q_H and quaternion of the human’s tracker Q_T . The outputs of each architecture are the robot joint

variables (for Baxter $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7$) that make the robot's end-effector follow the human's hand position and mimic its orientation.

Architecture A: It includes *wv-RNN*, *arm-RNN*, and *wrist-RNN1* as shown in Figure 2. The inputs to *wv-RNN* are Q_R , Q_H and Q_T , defined above. This RNN-LSTM is trained to generate as its output the corresponding value of the wrist vector wv (x_{wv} , y_{wv} and z_{wv}). The latter is subtracted from the human's right-hand position to generate the x_J , y_J , and z_J components of point J as given in (6). This represents the input to the *arm-RNN* which is trained to generate the arm joint variables (for Baxter $\theta_1, \theta_2, \theta_3, \theta_4$) as its output. The third RNN-LSTM, the *wrist-RNN1*, takes the *wv* components (output of *wv-RNN*) and the arm joint variables (output of *arm-RNN*) as inputs, and outputs the wrist joint variables (for Baxter $\theta_5, \theta_6, \theta_7$).

Architecture B: Figure 3 shows the schematic diagram of architecture B. It is similar to architecture A except that *wrist-RNN1* is replaced by *wrist-RNN2*. The latter RNN-LSTM takes inputs Q_R , Q_H and Q_T and is trained to generate the corresponding wrist joint variables ($\theta_5, \theta_6, \theta_7$) at its output. The *wv-RNN* and *arm-RNN* are trained and used exactly as they are in architecture A. Architectures A and B differ only in the input data that are used to compute

Fig. 2. Schematic diagram of architecture A.

the robot-wrist joint variables, which is the task of *wrist-RNN1* and *wrist-RNN2*, respectively. As shown in figure 2, the inputs to the *wrist-RNN1* in architecture A are related to position information. It takes as input the wrist vector and the robot-arm joint variables. On the other hand, as shown in figure 3, all the inputs of the *wrist-RNN2* (in architecture B) are abstracted from human orientation collected from the human directly, without position information.

All of the hierarchical RNN-LSTMs in Architecture A and Architecture B have 15 time-steps, 5 hidden layers and 101 neurons per hidden layer. The optimal number of time-steps, hidden layers and neurons were found empirically. All of the deep networks were trained off-line before using the training data for real time operation. The choice of architecture inputs were determined empirically.

Fig. 3. Schematic diagram of architecture B.

4 Experimental Setup

4.1 Collection of the Training Data

For our experiment, we first collected data from 13 pairs of demonstrators using the standard LfD approach. The collected data comprises about 55,000 data vectors, mapping the first demonstrator's head, right hand, left hand, and right hand tracker, and Baxter's corresponding seven joint variables. Each vector contains 28 data points (7 from each component):

$$\Theta = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7), \quad (7)$$

$$P_R = (x_R, y_R, z_R), P_L = (x_L, y_L, z_L), \quad (8)$$

$$P_H = (x_H, y_H, z_H), P_T = (x_T, y_T, z_T), \quad (9)$$

$$Q_R = (q_{0R}, q_{1R}, q_{2R}, q_{3R}), Q_L = (q_{0L}, q_{1L}, q_{2L}, q_{3L}), \quad (10)$$

$$Q_H = (q_{0H}, q_{1H}, q_{2H}, q_{3H}) \text{ and } Q_T = (q_{0T}, q_{1T}, q_{2T}, q_{3T}), \quad (11)$$

where Θ are the seven joint variables of the Baxter's right side; P_R , P_L , P_H and P_T are the position of the human right hand, left hand, head and tracker, respectively; and Q_R , Q_L , Q_H and Q_T are the quaternion of the human right hand, left hand, head, and tracker, respectively.

We collected different data by using the proposed approach LfDr. The size of the collected data is like its size in LfD, which is about 55,000 data vectors, mapping the demonstrator's head, right hand, left hand, and right-hand tracker, and Baxter's corresponding seven joint variables. Half from each training dataset (LfD and LfDr) is selected randomly, then they are merged randomly in one dataset, which has same size as the dataset collected by the LfD, but half of the data in the new dataset are collected from the standard LfD dataset and the other half is collected using the proposed LfDr approach. The new training dataset is the hybrid dataset, which used to train the recurrent neural networks.

4.2 Training Components of the Architectures

We now discuss how each of the component RNNs of architecture A and B is trained.

wv-RNN: The input data Q_R, Q_H, Q_T to train wv-RNN, are obtained from the HTC Vive directly. The wrist vector wv can be computed from the local coordinate system using Baxter robot-wrist's forward kinematic [7]. Then the computed wv is multiplied by the rotation matrix of the robot-arm structure to consider the effect of the robot-arm joint variables and make the orientation of the local coordinates coincide with the orientation of the global coordinate system.

arm-RNN: arm-RNN takes x_J, y_J , and z_J (components of point J) as input and is trained to generate the robot-arm joint variables (for Baxter $\theta_1, \theta_2, \theta_3, \theta_4$) that move the robot-arm to reach point J with any orientation. Thus, to train arm-RNN, the following training data are required $x_J, y_J, z_J, \theta_1, \theta_2, \theta_3$, and θ_4 . The input data x_J, y_J , and z_J are computed using Baxter robot-arm forward kinematics, which is derived based on D-H method; [7] while arm-RNN output $\theta_1, \theta_2, \theta_3, \theta_4$ are obtained from the collected training data.

wrist-RNN1: The wrist-RNN1 is used in architecture A. It takes the wrist vector (output of wv-RNN) and the robot-arm joint variables (output of arm-RNN) as inputs and is trained to generate the robot-wrist joint variables ($\theta_5, \theta_6, \theta_7$) that make the quaternion of the robot-wrist similar to the quaternion of the human's hand. The training vector of wrist-RNN1 has the following data:

$$x_{wv}, y_{wv}, z_{wv}, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7.$$

The x_{wv}, y_{wv} , and z_{wv} are computed in a similar way as they were for training wv-RNN; while $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6$ and θ_7 are obtained directly from the collected training data.

wrist-RNN2: In architecture B, wrist-RNN2 is used instead of wrist-RNN1. It takes the quaternion of the human's right hand Q_R , head Q_H , and tracker Q_T , and is trained to generate the robot wrist joint variables ($\theta_5, \theta_6, \theta_7$), which makes the quaternion of the robot-wrist similar to that of the human's hand. All of the elements of the wrist-RNN2 training vector (i.e., $Q_R, Q_H, Q_T, \theta_5, \theta_6$, and θ_7) are obtained directly from the collected training data.

4.3 Single ANN and RNN Architectures

In addition to the two proposed architectures A and B, we consider two basic deep learning architectures: a single RNN and a single ANN to perform the human robot mapping. During the training phase, each input vector consists of the position and quaternion of the human's head, right hand, left hand and right wrist with the corresponding robot joint variables as the output vector. Thus, the input to the single ANN and the single RNN are 28 data points and the output are the 7 joint angles as shown in figure 4. The single ANN and single RNN use the same number of hidden layers (5) and neurons per hidden layer

(101), and the single RNN similarly uses 15 time-steps for training. The inputs of the single RNN and the single ANN are chosen empirically. Table 1 shows the achieved average error in the training phase for the four different deep learning architectures when they trained by using the dataset generated by LfD and LfDr.

Fig. 4. Schematic diagram of single ANN or single RNN.

Table 1. Achieved average error in the training phase of the different architectures when they trained using dataset generated by the standard LfD.

Architecture	Average error (rad)
Single ANN	0.00150
Single RNN	0.0011
Arch. A	wv-RNN= 8×10^{-5} , arm-RNN= 8.5×10^{-4} , wrist-RNN1= 3.1×10^{-4}
Arch. B	wv-RNN= 8×10^{-5} , arm-RNN= 8.5×10^{-4} , wrist-RNN2= 8.5×10^{-4}

5 Experimental Results and Discussion

We conduct different experiments to test and evaluate the human-robot pose mapping performance for our proposed architectures A and B. In addition to the two architectures A and B, we evaluated the single RNN and the single ANN to compare the response of the robot mimicking the human when performing certain tasks using deep learning alone and deep learning when employed with our proposed architectures.

We evaluate the proposed hybrid dataset by using it to train architecture A and compare the performance of the human-robot mimicking system when architecture A is used and trained by the data generated by the standard LfD and the hybrid dataset.

5.1 Mapping Evaluation Results

Figure 5 shows images of the Baxter robot’s performance under the four different architectures. With our proposed architectures A or B, the human-robot mimicking system can move and change the robot’s end-effector freely to copy the human’s hand (specifically the wrist’s) position and orientation. This is because we trained a separated subsystem (wrist-RNN1 and wrist-RNN2) to map the orientation of the human’s wrist to the robot’s end-effector directly. In contrast, the single RNN or single ANN cannot move and change the robot’s end-effector freely to copy the orientation of the human’s hand, because there is no partial separation between the position and orientation since the whole single RNN or single ANN is trained to perform mapping between the human and the robot.

Figure 6 shows the human’s right-hand (Ground Truth) and Baxter’s end-effector trajectories in the 3D workspace when the robot tries to mimic the table tennis player using each one of the four mapping architectures. Figure 7 shows the Euclidean distance between the human’s right-hand position and Baxter’s end-effector using the four architectures. The minimum average error in position between the human’s hand and Baxter’s end-effector is obtained when architecture A is used with a position error of 0.0178 m as shown in table 2. Architecture B has a slightly more average error in position than architecture A, while the average error is significantly higher for both position and orientation when the single RNN or single ANN is used. The worst human-robot mimicking response is obtained with a single ANN, with an average position error of 0.0658 m.

Fig. 5. Performance of the human-robot mimicking system using the four different architectures: (a) Architecture A; (b) Architecture B; (c) Single RNN; and (d) Single ANN. The human carries the Vive controller in her right-hand while the Baxter mimics the position and orientation of the human’s hand (wrist) when architectures A or B is used; while it cannot mimic them accurately when single RNN or single ANN is used.

The orientation of the human’s hand and the robot’s end-effector are described using quaternions [3,12,18]. The best orientation mimicking is obtained with architecture B as shown in table 2. Architecture A has a slightly higher error in the orientation than architecture B, whereas single RNN and single ANN have higher quaternion errors. It is obvious from table 2 that the proposed architectures, A and B, have small average error between the Baxter robot and the human’s hand position and orientation compared to single RNN and single ANN. The best average position error is obtained with architecture A as the input to the wrist-RNN1 are related to position information. The best orientation mimicking is obtained with architecture B as the robot-wrist joint variables are

Fig. 6. Baxter's workspace shows the Baxter's end-effector trajectory to mimic a human table tennis player using architecture A, architecture B, single ANN and single RNN and human right hand position plotted as ground truth.

Fig. 7. Euclidean distance between the (ground truth) human hand position and the Baxter end-effector using architecture A, architecture B, single ANN, single RNN.

computed directly from the quaternion of the human head, hand, and tracker using wrist-RNN2. The worst average end-effector error in position and quaternion are obtained by the single ANN.

Based on the experimental results, architecture A and B provide approximately the same performance in the human-robot mapping system; both are able to transfer the human pose to the robot with minor errors in the position and orientation. When a single RNN is used, the human-robot mimicking has a higher error, especially in terms of the orientation. The Baxter's end-effector gets close to the human's hand position, but it fails to mimic its orientation correctly. The worst human-robot mimicking performance is obtained by a single ANN. The position average error in table 2 is computed as shown in equation 12,

$$\left[\frac{1}{n} \sum_{i=1}^n \left((x_{h_i} - x_{r_i})^2 + (y_{h_i} - y_{r_i})^2 + (z_{h_i} - z_{r_i})^2 \right) \right]^{0.5}, \quad (12)$$

where n is the number of points in the robot trajectory, and $(x_{h_i}, y_{h_i}, z_{h_i})$ and $(x_{r_i}, y_{r_i}, z_{r_i})$ are the position of the human's right-hand and Baxter's end-effector at point i , respectively. The average quaternion error in table 2 is computed by equation 13,

$$\left[\frac{1}{n} \sum_{i=1}^n |Q_{R_i} - Q_{B_i}| \right]^{0.5}, \quad (13)$$

where $|x|$, Q_{R_i} , and Q_{B_i} are the magnitude of x , the quaternion of the human's right-hand and Baxter's end-effector at point i respectively. To evaluate the quaternions error, it is more convenient to compute the angle θ , which indicates the difference between two quaternions as given in (14),

$$\theta = 2 \cos^{-1}(Q_{R_i} \cdot Q_{k_i}^*), \quad (14)$$

where ' $\cdot$ ' is the vector dot product and '*' is the complex conjugate. Figure 8, illustrates the variation of θ during the task execution when the four human-robot mapping systems are used. The minimum difference in the quaternion between the human and Baxter is obtained when our proposed architectures B and A are used, then with single RNN, while the quaternion difference is very large when single ANN is used.

Table 2. Average error in position and quaternion of the Baxter's end-effector with the four architectures.

Architecture	Ave. error in position (m)	Ave. error in Quaternion (rad)
Single ANN	0.0651	0.405
Single RNN	0.02987	0.211
Arch. A	0.01782	0.053
Arch. B	0.02214	0.041

5.2 Task-based Evaluation Results

To evaluate the efficiency and the effectiveness of our approach, we conduct another experiment. In this experiment, the demonstrator (human player) tries to hit a ball that is viewed via the HTC-Vive headset, and assess the capabilities of the robot to copy the player's hand movement and wrist orientation in order to hit the ball.

Figure 9 demonstrates how the Baxter robot accurately mimics the human's hand position and orientation using our proposed architecture B. Figure 9(a) shows the moment when the Baxter robot hits the table tennis ball, mimicking the human player. Figure 9(b) shows the human and Baxter shortly after hitting the ball. The position and orientation of the racket carried by Baxter's end-effector are similar to the position and orientation of the Vive controller carried

Fig. 8. Quaternion difference between human's right hand and Baxter's end-effector when the four architectures are used.

Fig. 9. The Baxter mimics the table tennis player movement accurately and hits the ball when architecture B is used as shown (a) at the moment of hitting the ball and (b) shortly after hitting the ball.

by the human's hand and wrist. The Baxter robot is even able to hit the ball as it moves through the air.

Table 3. Start-mission time S_t , end-mission time E_t , traveling time T_t , and traveling distance T_d with each mapping architecture.

	S_t (sec)	E_t (sec)	T_t (sec)	T_d (m)
Human	0.0	3.4	3.4	3.2893
Baxter with Single ANN	0.075	3.6	3.525	5.0530
Baxter with Single RNN	0.175	3.6	3.425	3.5434
Baxter with Arch. A	0.20	3.6	3.4	3.9254
Baxter with Arch. B	0.25	3.65	3.4	3.9775

Table 3 gives the start-mission time, end-mission time, traveling time, and traveling distance of the human's hand and the Baxter's end-effector with each mapping architecture. The human starts moving the right-hand at time $t=0$

sec; while the mission is completed (reaches the desired position and orientation with minimum error) and stops moving at $t=3.4$ sec.

If we compare the motion of the Baxter’s arm when the four architectures are used, we can see clearly from table 3 and figure 6 that with the single ANN, the Baxter’s arm starts moving first compared to the cases when other architectures are used. However, the Baxter robot almost completes the mission at the same time. This is not surprising as the single ANN has the simplest architecture and relatively small computation rate compared to the other three architectures. Thus it generates output before the others do, which makes the Baxter start moving first. On the other hand, when a single ANN is used, the distance traveled is the largest and its high velocity causes Baxter to overshoot its mark, as seen in figure 6.

A recurrent neural network has a more complex structure than a single ANN, since it can model sequences of data (time series), so that each sample is assumed to be dependent on previous ones. Therefore, it incorporates a time delay to generate its output that cannot be ignored. Thus, the Baxter is delayed compared to the case when a single ANN is used. On the other hand, RNN has a greater ability to learn non-linear and complex relations. The Baxter’s response with RNNs is much better and it completes its mission relatively faster and with smaller error than when a single ANN is used.

Baxter’s performance with both proposed architectures, A and B, is comparable: the response has a relatively large start-mission time compared to the cases when single ANN or single RNN are used. This is because both architectures A and B have three RNNs in their internal structure which make the response needs more time-consuming to generate, delaying the start-mission time. Nevertheless, they learn the human-robot mapping more accurately, which generates a good response from Baxter. In addition, Baxter’s average velocity is higher with these architectures than with a single RNN, which makes the traveling time shorter (better) as shown in Table 3.

5.3 Hybrid Training Dataset

We practically evaluate the quality of the proposed hybrid training dataset when it is used to train architecture A to learn the human-robot mimicking system. Then the trained architecture is used in the human-robot mimicking system to mimic human pose using new test trajectory, which was not used in the training dataset. Figure 10(a) shows the variation in the Euclidean distance and figure 10(b) shows the quaternion difference θ between the human’s hand and the Baxter’s end-effector poses during certain task execution when architecture A is used, which is trained using the dataset collected by the standard LfD and the hybrid training dataset. It is clear from figures 10(a) and (b) that the performance of the human-robot mimicking system for both position and orientation which is obtained when we use the hybrid training dataset to train the architecture is better than the mimicking performance when the training dataset that is collected by standard LfD is used in the training phase. Table 4 shows the Baxter’s end-effector average error in position and orientation when architecture A is used and trained by dataset generated by LfD and the hybrid dataset. The

average error in position and orientation are less when architecture A is trained using the hybrid dataset compared to the errors when the dataset generated by LfD is used in the training phase.

Fig. 10. The Euclidean distance and the quaternion difference θ between the human’s hand and the Baxter’s end-effector poses during certain task execution using architecture A, which is trained using the dataset collected by the standard LfD and the hybrid training dataset.

Table 4. Baxter’s end-effector average error in position and orientation when architecture A is used and trained by dataset generated by LfD and the hybrid dataset.

	Position ave. err. (m)	Quaternion ave. err. (rad)
Training by LfD data	0.0213	0.0434
Training by hybrid data	0.01681	0.0264

6 Conclusion

In this paper, we developed and evaluated two different architectures for human-robot mapping systems based on the RNN-LSTM deep learning concept. In contrast to “black box” applications of deep learning for this task, our approach leverages the temporal and spatial structure of the control task. Specifically, we partially separate the position and orientation in the human-robot mimicking system to divide the mimicking problem into several sub-problems and address each of them using RNN-LSTMs, which incorporates the relevant history of movements in its mapping.

Our experimental results obtained from a deployment on the Baxter robot arm (7 DOF) demonstrate improved performance in mimicking the human hand, and the wrist in particular, with very small error in position and orientation when compared to the case when the single RNN or single ANN is used. The implications for task performance are significant since providing an accurate orientation mapping is highly important for some tasks, such as object manipulation.

We also proposed and investigated an approach to collect a hybrid training dataset for the human-robot mimicking system. The hybrid dataset is generated by merging two datasets, the first dataset is collected using learning from demonstration based on the human mimicking the robot LfDr, while the second dataset is collected using the standard LfD approach, which is based on the robot mimicking the human. We evaluated the efficiency of the training dataset generated by using LfD and the hybrid datasets to train the human-robot mimicking system implemented by the proposed architecture A to learn mapping between the human and the robot poses. Based on our empirical results, the performance of the mimicking system is better when the hybrid dataset is used to train the architecture compared to the case when the dataset collected by the standard LfD is used in the training phase. The proposed approach LfDr to collect the data overcomes two problems that combine the standard LfD approach, which are the variation in height and size between the demonstrator and robot-mover, and the problem of the hidden points which can not be seen by the robot-mover. Both problems usually cause asynchronization in the collected dataset, which significantly decreases the quality of the training dataset.

Future work should combine both architectures into one structure to obtain high precision in position (from Architecture A) and in orientation (from Architecture B) for the human-robot mapping system. It should also investigate the benefits of elbow information for the mapping task. Specifically, we plan to use an additional Vive tracker on the first demonstrator's elbow for the mapping model. Making the robot's elbow more closely mimic the human's elbow may provide further efficiency for learning a mapping and improving task-level performance. It could also be useful when there are some obstacles in the robot's workspace that should be avoided by the robot's hand during the task execution. An interesting future direction is to employing active learning to select the most useful robot trajectories for the human to mimic. We expect this to improve the performance of the human robot system as well as increase the robustness of the machine learning system.

References

1. Al Mashhadany, Y.I.: Recurrent neural network with human simulator based virtual reality. *Recurrent Neural Networks and Soft Computing* p. 89 (2012)
2. Al-Qurashi, Z., Ziebart, B.: Hybrid algorithm for inverse kinematics using deep learning and coordinate transformation. In: *2019 Third IEEE International Conference on Robotic Computing (IRC)*. pp. 377–380. IEEE (2019)
3. Ben-Ari, M., Mondada, F.: *Elements of robotics*. Springer International Publishing (2018)

4. Du, Y., Wang, W., Wang, L.: Hierarchical recurrent neural network for skeleton based action recognition. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1110–1118 (2015)
5. Gaurav, S., Al-Qurashi, Z., Barapatre, A., Maratos, G.P., Sarma, T., Ziebart, B.: Deep correspondence learning for effective robotic teleoperation using virtual reality. In: IEEE-RAS International Conference on Humanoid Robots (2019)
6. Ju, Z., Yang, C., Ma, H.: Kinematics modeling and experimental verification of baxter robot. In: Control Conference (CCC), 2014 33rd Chinese. pp. 8518–8523. IEEE (2014)
7. Koren, Y., Koren, Y.: Robotics for engineers, vol. 168. McGraw-Hill New York et al (1985)
8. Lin, J., Guo, X., Shao, J., Jiang, C., Zhu, Y., Zhu, S.C.: A virtual reality platform for dynamic human-scene interaction. In: SIGGRAPH ASIA 2016 virtual reality meets physical reality: Modelling and simulating virtual humans and environments. p. 11. ACM (2016)
9. Liu, J., Shahroudy, A., Xu, D., Wang, G.: Spatio-temporal lstm with trust gates for 3d human action recognition. In: European Conference on Computer Vision. pp. 816–833. Springer (2016)
10. Liu, J., Wang, G., Hu, P., Duan, L.Y., Kot, A.C.: Global context-aware attention lstm networks for 3d action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1647–1656 (2017)
11. Luckett, E.: A Quantitative Evaluation of the HTC Vive for Virtual Reality Research. Ph.D. thesis, The University of Mississippi (2018)
12. McDonald, J.: Teaching quaternions is not complex. In: Computer Graphics Forum. vol. 29, pp. 2447–2455. Wiley Online Library (2010)
13. Niehorster, D.C., Li, L., Lappe, M.: The accuracy and precision of position and orientation tracking in the htc vive virtual reality system for scientific research. *i-Perception* **8**(3) (2017)
14. Nielsen, M.A.: Neural networks and deep learning, vol. 25. Determination press San Francisco, CA, USA: (2015)
15. Peer, A., Ullrich, P., Ponto, K.: Vive tracking alignment and correction made easy. In: 2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR). pp. 653–654. IEEE (2018)
16. Rakita, D., Mutlu, B., Gleicher, M.: A motion retargeting method for effective mimicry-based teleoperation of robot arms. In: Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction. pp. 361–370. ACM (2017)
17. Vasileiadis, M., Malassiotis, S., Giakoumis, D., Bouganis, C.S., Tzovaras, D.: Robust human pose tracking for realistic service robot applications. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1363–1372 (2017)
18. Vidaković, J.Z., Lazarević, M.P., Kvrđić, V.M., Dančuo, Z.Z., Ferenc, G.Z.: Advanced quaternion forward kinematics algorithm including overview of different methods for robot kinematics. *FME Transactions* **42**(3), 189–199 (2014)
19. Whitney, D., Rosen, E., Phillips, E., Konidakis, G., Tellex, S.: Comparing robot grasping teleoperation across desktop and virtual reality with ros reality. In: Proceedings of the International Symposium on Robotics Research (2017)
20. Whitney, D., Rosen, E., Tellex, S.: Learning from crowdsourced virtual reality demonstrations. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. p. 3 (2018)

21. Whitney, D., Rosen, E., Ullman, D., Phillips, E., Tellex, S.: Ros reality: A virtual reality framework using consumer-grade hardware for ros-enabled robots. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 1–9. IEEE (2018)
22. Williams II, R.L.: Baxter humanoid robot kinematics© 2017 dr. bob productions robert l. williams ii, ph. d., williar4@ ohio. edu mechanical engineering, ohio university, april 2017 (2017)
23. Zhang, T., McCarthy, Z., Jow, O., Lee, D., Chen, X., Goldberg, K., Abbeel, P.: Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). pp. 1–8. IEEE (2018)