Deep Correspondence Learning for Effective Robotic Teleoperation using Virtual Reality

Sanket Gaurav[1], Zainab Al-Qurashi[1], Amey Barapatre[1], George Maratos[1], Tejas Sarma[1] and Brian D. Ziebart[1]

***Abstract*— By projecting into a 3-D workspace, robotic teleoperation using virtual reality allows for a more intuitive method of control for the operator than using a 2-D view from the robot's visual sensors. This paper investigates a setup that places the teleoperator in a virtual representation of the robot's environment and develops a deep learning based architecture modeling the correspondence between the operator's movements in the virtual space and joint angles for a humanoid robot using data collected from a series of demonstrations. We evaluate the correspondence model's performance in a pick-and-place teleoperation experiment.**

I. Introduction

Tasks like performing experiments with hazardous substances or handling dangerous waste [1], [2] are most safely performed by humanoid robots. However, often these tasks cannot be completed successfully without human guidance. Teleoperation is one of the most popular ways used for a human operator to remotely guide and control humanoid robots [3], [1], [4], [5], [6], [7], [8], [2]. Traditionally, a teleoperator remotely watches the robot's environment projected on a screen (2-D view) from cameras mounted on the robot's head [5], [4] and uses a joystick [6], [7] or controller [8], [9], [10], [1] to operate the robot.

Depth cameras (e.g., the Microsoft Kinect) have been employed as the input sensor to control a Baxter robot in previous work [8], [11], [12], [13], [14], [15]. However, depth cameras often suffer from sensor noise that can produce errors and poor translation for robotic teleoperation when mapping from a tracked teleoperator skeleton to robotic joint positions. Additionally, having only the 2-D view of the robot's workspace often makes it difficult for the operator to precisely control the robot within its environment.

In contrast, recent advances in virtual reality technologies provide an opportunity to address this difficulty. Virtual reality is popular for creating virtual environments of any room or workspace [16], [17]. Virtual reality systems like the HTC Vive typically incorporate two controllers held by an operator that can track the human wrist with very high accuracy and allow the operator to realistically manipulate objects in the virtual environment. Recently, robotic teleoperation has been performed using 3D sensors like the HTC Vive and Oculus Rift [18], [19], [17], [16], [20], [21].

[1] Sanket Gaurav, Zainab Al-Qurashi, Amey Barapatre, George Maratos, Tejas Sarma and Brian D. Ziebart are with the Department of Computer Science, University of Illinois at Chicago, 851 S. Morgan St. (M/C 152) Chicago, IL 60607 {sgaura2, zalqur2, abarap2, gmarat2, tsarma2, bziebart}@uic.edu,

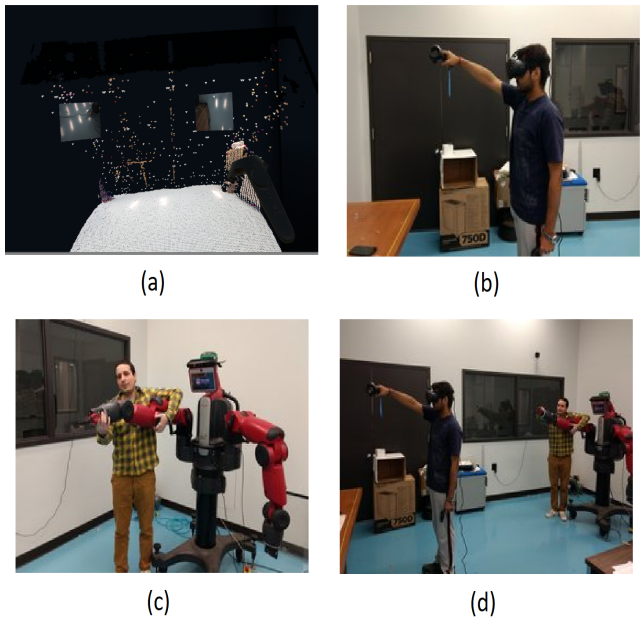

Fig. 1. (a) Visualization of the Baxter robot's workspace captured from a depth camera (Microsoft Kinect) and displayed on a virtual reality headset (HTC Vive); (b) demonstrator holding HTC Vive controller slowly moves his hand from neutral position; (c) trainer moves robot hand manually to follow human trajectory; and (d) demonstrator and trainer with Baxter reaching the same configuration.

The teleoperation correspondence problem of mapping from teleoperator poses or controls to robot poses is a crucial problem for enabling robotic teleoperation using a virtual reality system. Existing methods use linear coordinate transfer from a virtual reality frame of reference to a robot's frame of reference and then perform inverse kinematics to move a robot's arm [18], [19], [17], [16], [20]. This translation mechanism can be slow and erroneous due to multiple joint configuration solutions being provided by inverse kinematics for a single point.

In this paper, we propose a machine learning approach for estimating an appropriate non-linear correspondence for robotic teleoperation from human pose. We consider teleoperating the Baxter robot using a Microsoft Kinect depth camera for perceiving the robot's workspace and an HTC Vive virtual reality system for visualizing the workspace and providing 3-D control, as shown in Figure 1. First, we collect correspondence positions of human end-effectors and Baxter

joint angles by asking a demonstrator holding an HTC Vive controller to move his or her hand while an operator moves a Baxter arm in synchronization with the human demonstrator (Figure 1). Second, we explore different non-linear machine learning regression models as baseline correspondence models. Next, we explore deep learning architectures to learn a non-linear correspondence model for HTC Vive controllers to Baxter Robot joints. We show that our proposed deep correspondence model performs significantly better than linear and non-linear regression baselines and helps to enable more effective robotic teleoperation using virtual reality. To demonstrate the effectiveness of our proposed model, we conduct a simple real-life experiment: picking up a box from the table and placing it at another location. Our proposed deep network enables the teleoperator to perform the task faster and more effectively than the baseline methods.

The paper is organized as follows: we first provide related work on robotic teleoperation using virtual reality. Then, we describe in detail our deep correspondence architecture for robotic teleoperation. Next, we describe the experiments we conducted to compare our correspondence learning approach with baseline methods and discuss the results. Lastly, we conclude the paper and propose future work.

II. Background and Related Work

A. Correspondence Learning in Virtual Reality

Virtual reality (VR) can provide a teleoperator with a first-person perspective from a robot's viewpoint [17]. This enables high-quality demonstrations for robotic manipulation to be collected [22]. Fritsche et al. [17] use the Oculus Rift and Microsoft Kinect camera as teleoperation input and iCub as their humanoid robot. Their correspondence (transfer of human embodiment to the robot embodiment) is accomplished via Kinect camera skeleton tracking. The virtual reality setup is only used to give the first-person perspective for performing the task. As mentioned in the previous section, the Kinect camera can produce erroneous translations, preventing effective teleoperation. In this paper, we directly learn controls from human end-effector to robot embodiment via a deep learning approach.

Previous work uses consumer-grade virtual reality headsets (HTC Vive) supported by hand tracking hardware that can be used to naturally teleoperate robots to perform complex tasks with some delay [18], [16], [20]. Zhang et al. [18] teach the robot via demonstrations collected in virtual reality. These prior works [18], [16], [20] use linear coordinate transfer from virtual reality frame of reference to the robot frame of reference for correspondence learning. Then, they use inverse kinematics to find robot joint angles to move the arm. Unfortunately, inverse kinematics can provide different joint configurations for one end-effector of a robot arm. This may lead to an undesirable joint setting that produces irregular arm movements and ultimately sometimes fails to accomplish a task. Also, the whole process is complicated and time-consuming. We use a linear regression baseline method to emulate these prior work for comparison.

B. Baxter Robot

The Baxter robot is built by Rethink Robotics. It has a torso mounted on a movable pedestal and two arms on the left and right sides of the robot [23]. Each arm has seven degrees of freedom (DOF), i.e., seven joint angles:

$$R_{\text{joints}} = [s_0, s_1, e_0, e_1, w_0, w_1, s_2]. \tag{1}$$

Forward kinematics[1] provides the end-effector:

$$R_{\text{end-effector}} = [x_t, y_t, z_t, x_r, y_r, z_r, w_r], \tag{2}$$

where the first three are translation points describing the position of Baxter's arm end-effector and the last four are Quaternion angles describing the rotational position.

C. Deep Learning Architecture

Deep neural networks been developed to address prediction tasks for which more conventional computation approaches have proven ineffective [24], [25]. They are attractive for computing the inverse kinematics and dynamics of robots because they can be trained for this purpose without explicit programming [26] and can represent complex non-linear relationships. The basic operation carried out at a single neuron is represented as:

$$a_i = f^i(w_i a_{i-1} + b_i), \tag{3}$$

where a_i is the output of layer i, $f^i()$ is the activation function of layer i, w_i is the weight matrix between layer i and layer i-1 and b_i is the bias of layer i. Several algorithms have been developed to train a neural network, including back propagation (with momentum) and Levenberg Marquaidt algorithms [27] [28]. We use deep networks for our correspondence model in this paper.

D. Loss Function

The loss or evaluation functions used to evaluate our correspondence learning models are:

1) Mean Squared Error: The Mean Squared Error (MSE) is calculated by computing the squared difference between actual value (Y) and predicted value ($\hat{Y}$) and averaging over total number of values,

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y_i}\right)^2. \tag{4}$$

We employ this measure to train our deep network model and evaluate the performance of its resulting predictions.

2) Cosine Similarity: Cosine similarity measures the similarity between two non-zero vectors of an inner product space based on the cosine of the angle between them:

$$\cos(\theta) = \frac{\mathbf{AB}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}, \tag{5}$$

where A and B are vectors. We use cosine similarity to compute the loss of rotation angles of the end-effector.

[1] http://sdk.rethinkrobotics.com/wiki/Baxter_PyKDL#baxter_kinematics.py

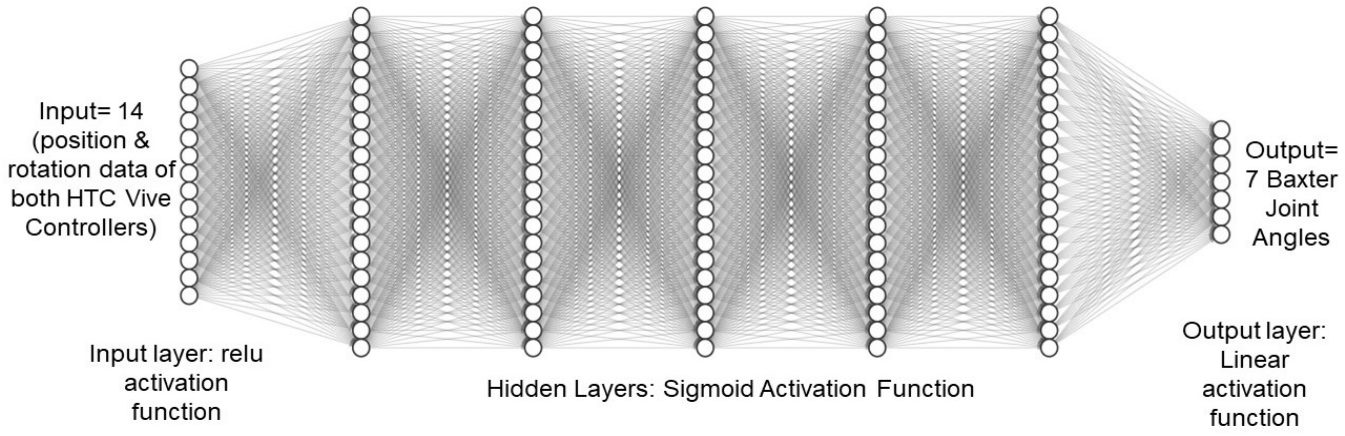

Fig. 2. Training of the deep neural network to learn the correspondence between the Baxter robot joint angles and the human poses where input 14 represents the position and rotation data of HTC Vive controller placed in two respective hand of the human demonstrator and output 7 represents the seven joint angles of one of the Baxter robot's arm.

III. Approach

In this section, we first describe the visualization of the Microsoft Kinect depth data in our virtual reality system. Then, we explain the underlying problem of non-linearity in the correspondence from the HTC Vive controller to the Baxter joint angles. Lastly, we propose in detail our main contribution: a deep correspondence learning architecture.

A. Visualizing Microsoft Kinect depth data in Virtual Reality

We first develop a visualization of the Kinect depth point cloud data in the HTC Vive. In the Unity application, we read the sensor data from a Kinect V1, which has a resolution of 640 x 480 pixels and an 11-bit depth. We use this collected sensor data to render a mesh in Unity in real-time to create a point cloud representation. The color information from the collected data is also maintained in the representation as it makes objects more discernible. We scale and orient this point cloud to elicit realism in the visualization using manually selected parameters. Using the Unity SteamVR plugin, we integrated the HTC Vive in the application to view the point cloud in virtual reality. Figure 1(a) shows the visualization of Baxter's environment in HTC Vive headset.

The 3-D projection of the robot's environment provides teleoperators with a first-person perspective for performing tasks effectively. However, to accomplish a precise grasping task, we stream frames from the Baxter wrist camera (2-D) to the HTC Vive headset. Using the Unity engine, we display the two wrist cameras by streaming them on the top corners, as shown in Figure 1(a). This can help the teleoperator perform a fine-grained task when the robot arm has reached close to the target location.

B. Robotic Teleoperation using Virtual Reality

The crux of any robotic teleoperation system relies on the transformation of the human operator's control inputs to the robot joint values. For a virtual reality system, this involves a transfer of embodiment from the human to the robot that is called a correspondence. We consider the setting in which the human operator holds an HTC Vive controller in each hand and wears the HTC Vive headset to view the Baxter robot's 3D environment from the first-person perspective. The HTC Vive data consists of 21 data points, which comprise positions (translations x_t, y_t, z_t) and orientations (quaternion angles x_r, y_r, z_r, w_r) of the HTC Vive headset and two HTC Vive controllers.

The two HTC Vive Controller positions indicate the two human end-effectors. Mapping each position to the seven joint angles of each Baxter arm is challenging due to non-linearity. To verify this, we compared a linear model with polynomial models to regress each one of the joint angles from the HTC Vive data. The polynomial models outperform the linear models with the result being statistical significant with a P-value $< 10^{-16}$, thus proving that it is indeed a non linear transformation.

C. Deep Correspondence Learning Architecture

Deep learning has proven to be an effective method for non-linear function approximation, as explained in Section II-C. In this paper, we propose a fully connected deep neural network architecture for our correspondence model, as shown in Figure 2. The data points from two HTC Vive Controllers are used as input to the deep network, and the output is the required seven joint angles of the Baxter Robot arm. We choose the position of both hands (14 data points) versus only the controlling hand (seven data points) as input because the location of the other non-controlling hand helps the model better situate the relative position of the controlling hand in the space. We use off-line training for the deep network as shown in Figure 2. The deep network consists of five hidden layers (reason explained in Section IV-E) each with 64 neurons. The motivation for deciding neurons in the hidden layer comes from the statistical difference between human end-effector position to the Baxter joint angles as explained in the previous section.

We use a Rectified Linear Unit (ReLU) activation function (also known as the ramp function) in the input layer, a

Sigmoid response ("S"-shaped curve) in the five hidden layers, and a linear activation function in the output layer in order to obtain an output in radians between -1 to 1.

We use the mean squared error (MSE), as the loss function, $\frac{1}{n}\sum_{i=1}^{n}\left(\theta_{ip}-\theta_{id}\right)^2$, where θ_{ip} is the predicted output joint angles and θ_{id} is the desired joint angles. We use ADAM [29] as the optimizer to converge quickly to small loss.

IV. Experiment

A. Hardware Setup

We evaluate our approach using a Baxter robot as our experimental robot in this paper. We use a Microsoft Kinect Depth V1 camera with frequency up-to 20Hz mounted to the head of the Baxter robot to visualize the robot workspace or environment, and an HTC Vive as our virtual reality platform, which publishes data with a frequency of up to 90Hz. The HTC Vive comprises one headset, two controllers and two base stations for tracking. We use a Windows machines for running the VR platform that has an i7 processor, 16 GB of RAM and a GTX 1080 ROG graphics card. We use ROS (Robot Operating System) to communicate with the hardware. We also use the Baxter wrist camera to stream frames to the HTC Vive headset with a frequency of up to 25Hz.

B. Data Collection

A demonstrator is asked to wear the HTC Vive headset where he/she can view what Baxter sees as shown in Figure 1-a. Also, the same demonstrator is requested to hold the two HTC Vive controllers in respective hands. This demonstrator then moves one of their hands slowly for 3-5 minutes in random directions. The trainer moves the corresponding arm of Baxter in synchronization to the demonstrator's movement, as shown in Figure 1(c). Both the demonstrator and Baxter's arm start moving from the same starting position in space. This data collection process is conducted for both arms separately. The attached video contains more details of the training data collection.

There are three components of the HTC Vive used in this setup, thus 21 data points (7 points for each component) are transferred over the network to the Robot Operating System (ROS) environment. These 21 data points from the HTC Vive and the 7 joint angles of one Baxter arm are recorded at 40 Hz frequency. The collected data are 44,941 data points for the right hand and 36,155 data points for the left hand from 11 demonstrators.

C. Data Pre-processing

The 11 demonstrators who contributed to data collection are of different height and arm length. To generalize our training, we subtract the HTC Vive headset translation positions (x_t, y_t, z_t) from both HTC Vive controllers' translation positions to measure the controllers' movements relative to head positions. We randomly withheld one person's trajectory data for conducting an offline experiment. On the remaining ten demonstrators' data, we apply leave-one-out-cross-validation (LOOCV), i.e., training on nine people's data and testing on the tenth. We repeat the process with each person's data and take the average.

D. Training of Baselines

We compare our deep correspondence model against a linear regression approach [13] with either no, polynomial, or Gaussian expansion of the feature space, and support vector regression with polynomial or radial basis function (RBF) kernels. The linear regression method is used as a state-of-the-art approach to the correspondence learning problem. We observe that the difference in average Euclidean loss between linear regression with and without expansion of the feature space is significant, which is further evidence that the correspondence task is non-linear in nature. The detailed result of each baseline is reported in Table I.

E. Training of Deep Learning Models

Motivated by the deficiencies of the linear and non-linear baseline models, we investigate deep learning models. We started with a simple network having one hidden layer and experimented with different activation functions (e.g., hyperbolic tangent, Sigmoid, rectified linear) for input and output layers. For each configuration, we trained until reasonable validation loss was obtained. We then tested the performance of the correspondence model on the Baxter robot. The configuration of rectified linear activation functions for the input, Sigmoid activation functions for the hidden layers, and linear activation functions for the output layer gave the best correspondence. We then tested different numbers of layers (up to 20 layers) and different numbers of neurons (e.g., in multiples of 16) in each layer. Using the best of numbers of layers and number of neurons, we then trained our deep model to convergence with a difference of training losses for the last ten epochs of less than 0.0001.

F. Experimental Setup

We conducted two types of experiments to evaluate our proposed model against the linear and non-linear baselines.

1) Offline Experiment: We first find the difference between the predicted output and the ground truth for the baselines and our deep model. We randomly selected a demonstrator's trajectory from the Vive-Baxter correspondence dataset comprising of 2500 pairs of poses. The corresponding Baxter's joint angles of this trajectory (actual/desired value) are considered as ground truth. The trajectory is passed to the linear regression correspondence model, the best non-linear regression model, and the deep network correspondence model. The respective output joint angles from both models are recorded. We apply forward kinematics to output Baxter's arm joint angles (ground truth, linear regression, non-linear regression, and deep network) to calculate Baxter's arm end-effectors, as shown in Figure 4. The simultaneous movement of the Baxter robot arm using position control for each of the four models is demonstrated in an attached video.

We compute the Euclidean distance of Baxter's end-effector as provided by the correspondence model with the ground truth position, as shown in Table II. To measure the

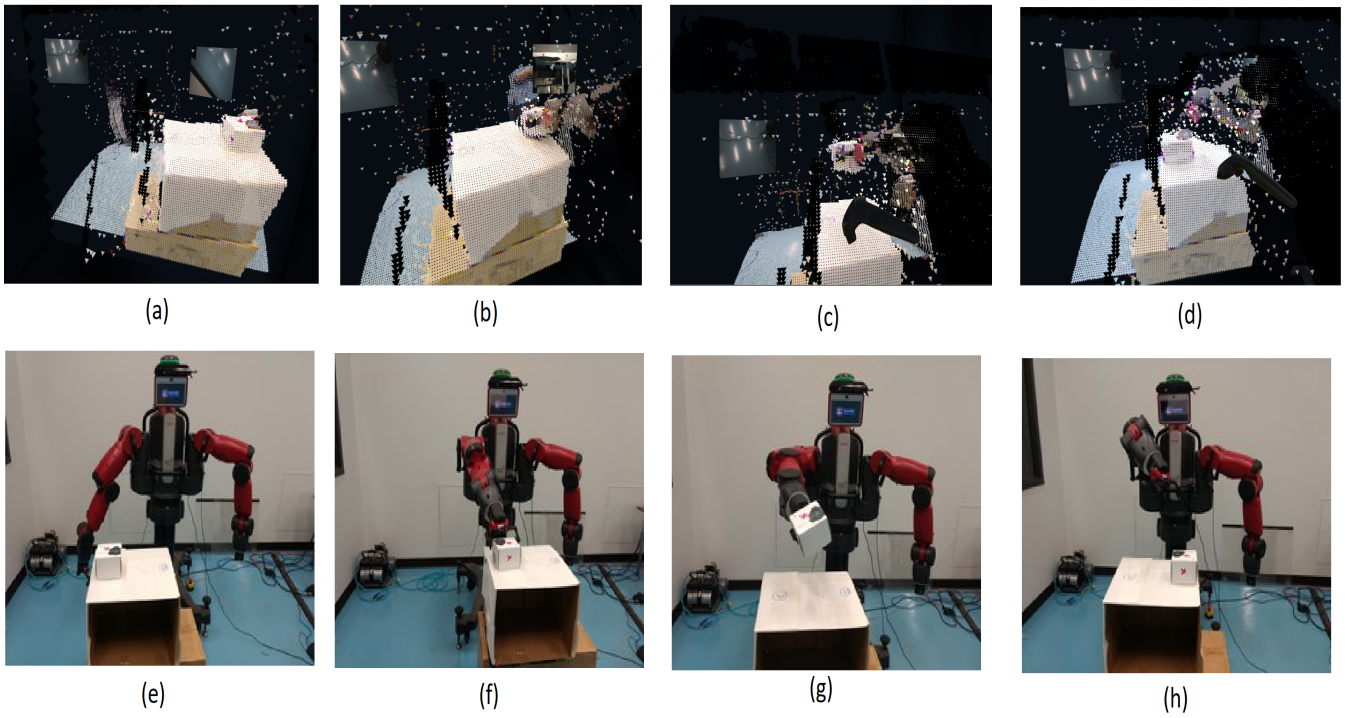

Fig. 3. (a) Pick and place experiment setup in virtual reality, (b) Pick and place experiment setup in reality, (c) Teleoperator reaching the object (white box) kept at position A in virtual reality, (d) Baxter teleoperated to reach the object (white box) kept at position A, (e) Teleoperator picking up the white box from position A and moving to position B in virtual reality, (f) Baxter teleoperated to pick up the white box from position A and moving to position B, (g) Teleoperator placing the box at position B in virtual reality, (h) Baxter teleoperated to place the box at position B.

loss in rotational angle of Baxter's end-effector, we compute the cosine similarity between the predicted angles and the ground truth as shown in Table III. The graph between predicted end-effector from linear regression vs deep network vs ground truth is plotted in Figure 4 for three translation position and four rotational angles respectively.[2]

2) Real-time Robotic Teleoperation: We use a simple pick and place task (Figure 3) for our second set of experiments. A teleoperator is asked to wear the HTC Vive headset where he/she can view the virtual robot environment, as shown in Figure 3(a). The teleoperator must move the Baxter arm from a neutral position to position A where a white box is placed, grasp the box, and move it from position A to position B within one minute to be considered successful. Figures 3(a)-(d) shows the steps conducted by the teloperator in the Virtual reality and Figure 3(e)-(h) demonstrates corresponding steps performed by Baxter robot using the proposed deep correspondence model. This task is repeated by two different teleoperators using linear regression, the best performing non-linear model (SVR-RBF), and our proposed deep architecture. There are five trials for each model and each of them is randomized. Thus, we collected ten samples for each model to compare success rate.

In real-time, the two HTC Vive controller positions (after subtracting the HTC Vive headset's translation position) are passed to the trained model. The output (7 Baxter's arm joint angles) from the trained correspondence model is then passed to the Baxter, which moves its arm using position control, as shown in Figure 3. The average time taken to complete this task and the success rate of each model is reported in Table IV. The detailed video on collection of training data and the pick and place experiment is attached to the paper.

[2]SVR (RBF) does not provide significant improvement, but clutters the differences between deep and linear, so we do not include it.

TABLE I

RESULTS (TEST LOSS IN RMSE) OF HTC VIVE CONTROLLER TO BAXTER CORRESPONDENCE MODELS

Model	**Left Arm (rad)**	**Right Arm (rad)**
Linear Regression	0.6430	0.6699
Kernel Regression (Poly=2)	0.5808	0.5840
Kernel Regression (RBF)	0.5788	0.6196
SVR (Poly=2)	0.6019	0.4863
SVR (Poly=3)	0.5716	0.4758
SVR (RBF)	0.4944	0.4567
Deep Networks	0.0735	0.0659

TABLE II

EUCLIDEAN DISTANCE MEASURE FROM GROUND TRUTH BAXTER'S END-EFFECTOR POSITION TO PREDICTED BAXTER'S END-EFFECTOR

Model	**Euclidean Loss (m)**
Linear Regression	0.3414
SVR (RBF)	0.2173
Deep Network	0.0267

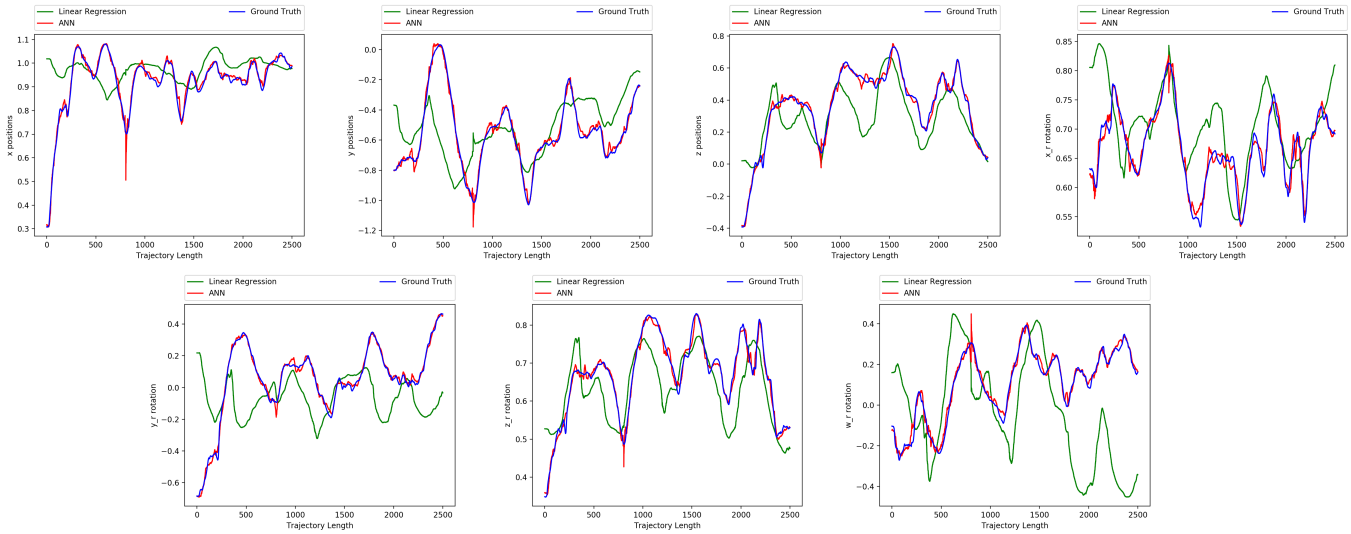

Fig. 4. (a)-(c) Robot end-effector positions (x_t, y_t, z_t) plotted for Ground Truth vs Linear Regression vs Deep Network; (d)-(g) Robot end-effector rotational angles (x_r, y_r, z_r, w_r) plotted for Ground Truth vs Linear Regression vs Deep Network.

TABLE III
COSINE SIMILARITY RESULT

	Linear Regression	**SVR (RBF)**	**Deep Network**
x_r	0.467	0.628	0.986
y_r	0.863	0.876	0.976
z_r	0.560	0.785	0.985
w_r	0.160	0.428	0.959

TABLE IV
SUCCESS RATE OF PICK & PLACE EXPERIMENT

	Success Rate (%)	Average Time (sec)
Linear Regression	60	56
SVR (RBF)	70	50
Deep Network	80	41

V. RESULTS AND DISCUSSION

Table I describes the evaluation results of the HTC Vive controller to Baxter arm joint angle correspondence. As shown in this table, the MSE for the Baxter joint angles using the simple linear regression model is very high (i.e., a poor correspondence). All of the four non-linear models performed better than the linear regression model and Support Vector Regression (SVR) with a Radial Basis Function (RBF) kernel performed best on the LOOCV test. Thus, we selected SVR (RBF) as our non-linear baseline model. The deep network provided substantially lower error, with a decrease in MSE for the left hand by a factor of 8.74 compared to the linear model and a factor of 6.73 to SVR (RBF). A similar decrease in MSE can be seen for the right arm. This demonstrates that our proposed deep architecture more successfully models the non-linearity in the Vive-Baxter correspondence data.

Table II shows the Euclidean distance between the robot's ground truth end-effector and predicted end-effector positions. According to Table II, the Euclidean distance loss between linear regression to ground truth robot's end-effector is 0.3414 meters and SVR (RBF) non-linear model to ground truth is 0.2173 meters. On the other hand, the Euclidean distance loss between ground truth using the deep network is only 0.0267 meters. Also, the attached video clearly demonstrates the difference of performance across the four models (ground truth, linear regression, non-linear regression, and deep network) in the real teleoperated arm movement of the robot. Thus, we have demonstrated significant amounts of improvement in predicted end-effector by our proposed deep network compared to linear and non-linear baselines.

Table III shows the cosine similarity between the rotational angles of the predicted end-effector with the ground truth. For all four rotational angles, our deep model outperforms the baseline linear and non-linear regression by a large margin. Therefore, our deep correspondence model enables more appropriate control of end effector orientation for fine-grained manipulation tasks.

The translation positions (x_t, y_t, z_t) and rotational angles (x_r, y_r, z_r, w_r) are plotted for Baxter's end-effector for ground truth, linear regression and our deep network in Figure 4. We infer from these seven graphs that the deep learning model follows the ground truth very closely whereas linear regression is far away from the ground truth.

Figure 3 shows snapshots of Baxter performing a pick and place task using the deep correspondence model. We report the results of this experiment in Table IV. Success rates and average completions times both improve from the linear regression model to the non-linear regression model (SVR with RBF kernel), and from the non-linear model to our deep learning approach. We find that our approach provides the highest success rate and lowest average completion time. Thus, our offline performance successfully transfers to real-time control.

VI. CONCLUSION AND FUTURE WORK

We have successfully trained correspondence models for HTC Vive Controller to Baxter's arms. Our proposed deep

model achieves better results than linear and non-linear regression baseline models for correspondence-based evaluations. In the real-time experiment, our deep network performed better than baselines model, resulting in faster completed tasks.

In this paper, we have investigated the problem of correspondence learning for teleoperating the hands of a humanoid robot. In future work, we plan to extend our approach to a complete humanoid robot (e.g., Nao) [17], [19], [3], [8], [1], [10]. This will help better collect training data for teaching humanoid robots and also perform effective real-time humanoid robotic teleoperation.

We are planning to evaluate more complex tasks, like "opening a jar" and tasks mentioned by Whitney et al. (2018) [20]. To improve the completion time and smooth teleoperation, we are planning to apply methods for goal predictions [13], [30] using our trained HTC Vive to Baxter arm correspondence model. This will help the teleoperated robot reach the goal in less time and with less distance traveled compared to previous work [13].

In the future, we would like to improve the visualization of the robot's environment in virtual reality using better 3-D cameras and better visualization techniques. We would like to incorporate better data collection techniques involving HTC Vive [22]. Moreover, we are planning to apply other deep learning techniques such as recurrent neural networks that accommodate the sequential nature of points in the trajectory.

ACKNOWLEDGMENT

We thank the demonstrators and trainers who volunteered for data collection. This research is supported as part of the Future of Life Institute (futureoflife.org) FLI-RFP-AI1 program, grant #2016-158710 and NSF grant #1652530.

REFERENCES

[1] J. Ramos, A. Wang, W. Ubellacker, J. Mayo, and S. Kim, "A balance feedback interface for whole-body teleoperation of a humanoid robot and implementation in the hermes system," in *Proceedings of IEEE-RAS 15th International Conference on Humanoid Robots*, Nov 2015, pp. 844–850.

[2] J. Vertut, *Teleoperation and robotics: applications and technology*. Springer Science & Business Media, 2013, vol. 3.

[3] F. Abi-Farrajl, B. Henze, A. Werner, M. Panzirsch, C. Ott, and M. A. Roa, "Humanoid teleoperation using task-relevant haptic feedback," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 5010–5017.

[4] Sokho Chang, Jungtae Kim, Insup Kim, Jin Hwan Borm, Chongwon Lee, and Jong Oh Park, "Kist teleoperation system for humanoid robot," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*, vol. 2, Oct 1999, pp. 1198–1203 vol.2.

[5] Sooyong Lee, Dae-Seong Choi, Munsang Kim, Chong-Won Lee, and Jae-Bok Song, "An unified approach to teleoperation: human and robot integration," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*, Oct 1998, pp. 261–266 vol.1.

[6] N. E. Sian, K. Yokoi, S. Kajita, F. Kanehiro, and K. Tanie, "Whole body teleoperation of a humanoid robot - development of a simple master device using joysticks," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2002, pp. 2569–2574 vol.3.

[7] M. Stilman, Koichi Nishiwaki, and Satoshi Kagami, "Humanoid teleoperation for whole body manipulation," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, May 2008, pp. 3175–3180.

[8] I. Almetwally and M. Mallem, "Real-time tele-operation and tele-walking of humanoid robot nao using kinect depth camera," in *Proceedings of IEEE International Conference on Networking, Sensing and Control (ICNSC)*, April 2013, pp. 463–466.

[9] J. Kofman, Xianghai Wu, T. J. Luu, and S. Verma, "Teleoperation of a robot manipulator using a vision-based human-robot interface," *IEEE Transactions on Industrial Electronics*, vol. 52, no. 5, pp. 1206–1219, Oct 2005.

[10] I. Rodriguez, A. Astigarraga, E. Jauregi, T. Ruiz, and E. Lazkano, "Humanizing nao robot teleoperation using ros," in *Proceedings of IEEE International Conference on Humanoid Robots*, Nov 2014, pp. 179–186.

[11] H. B. Suay and S. Chernova, "Humanoid robot control using depth camera," in *Proceedings of ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, March 2011, pp. 401–401.

[12] C. Schultz, "Goal predictive infused robot teleoperation with kinect depth camera (ms thesis)," 2016.

[13] C. Schultz, S. Gaurav, M. Monfort, L. Zhang, and B. D. Ziebart, "Goal-predictive robotic teleoperation from noisy sensors," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5377–5383.

[14] S. Gaurav, "Goal predictive robot teleoperation using predictive filtering and goal change modeling (ms thesis)," 2017.

[15] A. Sripada, H. Asokan, A. Warrier, A. Kapoor, H. Gaur, R. Patel, and S. R, "Teleoperation of a humanoid robot with motion imitation and legged locomotion," in *3rd International Conference on Advanced Robotics and Mechatronics (ICARM)*, July 2018, pp. 375–379.

[16] D. Whitney, E. Rosen, E. Phillips, G. Konidaris, and S. Tellex, "Comparing Robot Grasping Teleoperation across Desktop and Virtual Reality with ROS Reality," in *International Symposium on Robotics Research*, 2017 in press.

[17] L. Fritsche, F. Unverzag, J. Peters, and R. Calandra, "First-person tele-operation of a humanoid robot," in *Proceedings of IEEE-RAS International Conference on Humanoid Robots*, Nov 2015, pp. 997–1002.

[18] T. Zhang, Z. McCarthy, O. Jow, D. Lee, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," *arXiv preprint arXiv:1710.04615*, 2017.

[19] K. Theofilis, J. Orlosky, Y. Nagai, and K. Kiyokawa, "Panoramic view reconstruction for stereoscopic teleoperation of a humanoid robot," in *Proceedings of IEEE International Conference on Humanoid Robots*, Nov 2016, pp. 242–248.

[20] D. Whitney, E. Rosen, D. Ullman, E. Phillips, and S. Tellex, "ROS Reality: A Virtual Reality Framework Using Consumer-Grade Hardware for ROS-Enabled Robots," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[21] B. Xi, S. Wang, X. Ye, Y. Cai, T. Lu, and R. Wang, "A robotic shared control teleoperation method based on learning from demonstrations," *International Journal of Advanced Robotic Systems*, vol. 16, no. 4, p. 1729881419857428, 2019.

[22] M. Borges, A. Symington, B. Coltin, T. Smith, and R. Ventura, "Htc vive: Analysis and accuracy improvement," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2610–2615.

[23] Z. Ju, C. Yang, and H. Ma, "Kinematics modeling and experimental verification of baxter robot," in *Chinese Control Conference (CCC)*. IEEE, 2014, pp. 8518–8523.

[24] I. J. Goodfellow, Y. Bengio, and A. C. Courville, *Deep Learning*, ser. Adaptive computation and machine learning. MIT Press, 2016.

[25] D. Psaltis, A. Sideris, and A. A. Yamamura, "A multilayered neural network controller," *IEEE control systems magazine*, vol. 8, no. 2, pp. 17–21, 1988.

[26] G. A. Bekey and K. Y. Goldberg, *Neural Networks in robotics*. Springer Science & Business Media, 2012, vol. 202.

[27] M. A. Nielsen, *Neural networks and deep learning*. Determination Press, 2015.

[28] H. Demuth and M. Beale, "Neural network toolbox," *For Use with MATLAB. The MathWorks Inc*, vol. 2000, 1992.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[30] S. Gaurav and B. Ziebart, "Discriminatively learning inverse optimal control models for predicting human intentions," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '19, 2019, pp. 1368–1376.