Collaborative Human-Robot Motion Generation using LSTM-RNN

Xuan Zhao, Sakmongkon Chumkamon, Shuanda Duan, Juan Rojas, and Jia Pan†

***Abstract*— We propose a deep learning based method for fast and responsive human-robot handovers that generate robot motion according to human motion observations. Our method learns an offline human-robot interaction model through a Recurrent Neural Network with Long Short-Term Memory units (LSTM-RNN). The robot uses the learned network to respond appropriately to novel online human motions. Our method is tested both on pre-recorded data and real-world human-robot handover experiments. Our method achieves robot motion accuracies that outperform the baseline. In addition, our method demonstrates a strong ability to adapt to changes in velocity of human motions.**

I. Introduction

In human-robot collaboration (HRC), robots must have the ability to physically interact with humans safely and synergistically. Specific collaborative patterns provide a common language by which both humans and robots interact and mutually aid each other efficiently. Pre-programming continues to be the main medium by which robots are taught; however, teaching appropriate collaborative patterns for different tasks this way is tedious, especially for complex tasks. For this reason, we propose an interactive learning: a data-driven approach based on imitation learning that involves the human-robot pair to accomplish a smooth handover [1].

Much work has been done in generating appropriate robot motions in response to observed human motions. In [2], Maeda *et al.* first considered a joint distribution of human and robot movement primitives over an entire set of trajectories. Then, the robot conditioned on a partial human observation of the joint distribution to select the most likely trajectory pattern. The latter in turn was used to regress a corresponding robot manipulator trajectory. This system was further integrated with Electromyography (EMG) signals in [3] to improve the robot's ability to recognize motions that looked similar but handled different objects. These works, however, did not update their observations in real-time and thus generated robotic trajectories that are rigid and inaccurate. That is, they lack the ability to be responsive to changes in the human's actual motion as well as they suffer from poor predictive accuracy as their reach location tends to be distant from a human's actual goal position. Another challenge in HRC is that human motions may occur at different speeds, leading to different time scales for different trajectories and making it difficult for the robot to respond appropriately. To resolve such temporal scaling issues, previous works estimated the trajectory phase through probabilistic models [2], [3] or dynamic time warping [4]. However, the estimated phase or alignment may not be accurate resulting in error propagation during the trajectory generation step. Recently, learning-based approaches like Recurrent Neural Networks (RNNs) have been widely used in robotics to learn how robots should move to accomplish a task. This includes goal-directed tasks [5], manipulation tasks [6], and physical interactions between a humanoid and other physical robots [7]. These works demonstrated that RNN models with multiple time scales are able to generate robot motion successfully for complex tasks. Recently, Ochi use a Recurrent Neural Network with Long Short-Term Memory units (LSTM-RNN) to generate online robot scooping motions in a changing environment [8].

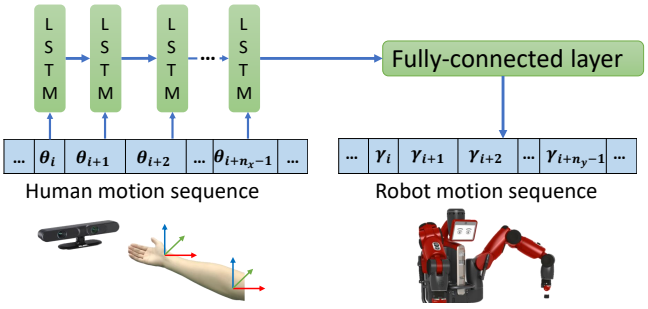


Fig. 1. Our system generates robot motion based on observed human motions. We input a human motion sequence that is processed by an LSTM layer and a fully-connected layer. Finally a robot motion sequence is output.

This paper contributes an LSTM-RNN model in human-robot collaboration scenarios to generate appropriate robot motions in response to observed human actions. In particular, we first learn an LSTM-RNN model from interactive training patterns of human and robot handover demonstrations. At runtime, the LSTM-RNN acts as an online robot controller, which inputs human motion observations from the wrist and the elbow and then outputs reactive robot trajectories. We compare the learned LSTM-RNN model's performance with a baseline method based on the probabilistic motion primitives [3] on both the pre-recorded data and the real-world human-robot handover experiments. Results demonstrate that

X. Zhao and J. Pan are with the Department of Mechanical and Biomedical Engineering, City University of Hong Kong, Hong Kong e-mail: xuan.zhao@my.cityu.edu.hk; jiapan@cityu.edu.hk Sakmongkon Chumkamon, Shuanda Duan, and Juan Rojas are with the Guangdong University of Technology, Guangzhou, China e-mail: juan.rojas@gdut.edu.cn. † denotes the corresponding author.

S. Chumkamon, S. Duan and J. Rojas were supported by "Major Project of the Guangdong Province Department for Science and Technology (2014B090919002), (2016B0911006) and by the National Science Foundation of China (61750110521)." X. Zhao and J. Pan were supported by HKSAR General Research Fund (GRF) CityU 21203216, NSFC/RGC Joint Research Scheme (CityU103/16-NSFC61631166002, and Innovation and Technology Fund ITS/4571/17FP.

our method outperforms the baseline approach in terms of the position accuracies achieved by the generated robot motion. In addition, our method enables robot adaptive behaviors to changes in velocity of human motions. Our approach further improves the efficiency and synergy of HRC by helping robots achieve fast and responsive motion generation that is robust to a wide variety of time scales.

Section II presents the novel robotic motion generation approach using LSTM-RNNs. Section III, Section IV and Section V present the experiments as well as comparisons with our baseline [3]. Finally, we highlight key findings in Section VI.

II. Approach

A. Introduction to LSTM-RNN

RNN is the feed-backward version of the conventional feed-forward neural network. It allows the output of one neuron at time step t_i to be the input of the same neuron at time step t_{i+1}. Standard RNN methods suffer from the vanishing gradient problem [9]. To overcome this problem, [10] developed the LSTM unit. LSTM units are fit to store and access information over long periods of time. LSTMs achieve this through their 3-gate architecture, which consists of an input, an output, and a forget gate. Furthermore, RNNs with LSTM units have been effective in sequence learning while also being scalable. In this work, we leverage the strengths of RNN-LSTMs to model correlations in human-robot motions that then serve to generate accurate and responsive robot motions in response to human observations.

B. Network architecture

We now introduce the architecture and the training process of our neural network. The network, which is shown in Figure 2, is composed of two layers. The first layer takes an input sequence of length n_x. The sequence enters into each of the n LSTM units contained in the LSTM layer. The LSTM layer outputs then enters a fully-connected layer to align the dimensionality of the output. The range of n is decided by the complexity of the dataset and the specific values set empirically (see Section III for details). During training, we use the Adam optimizer [11] and set the loss function as the the mean square error as defined below:

$$L(Y, f(X)) = L(Y, \widetilde{Y}) = \left[\frac{1}{P}\sum_{i=1}^{P}(Y_i - \widetilde{Y}_i)^2\right]^{\frac{1}{2}}, \quad (1)$$

where P is the feature dimensionality, X is the input sequence, Y is the output ground truth, and $\widetilde{Y}$ is the generated sequence.

C. Data collection

For experimentation, a Baxter humanoid robot uses its left arm to collaborate with humans. The robot is equipped with a standard Baxter electric pinching gripper and one ASUS Xtion RGB-D camera for sensing. The RGB-D camera is mounted at the base of the robot to observe human motions. The Openni tracker ROS package is used to identify human joint positions. ROS Kinetic in Ubuntu 16.04 is run on

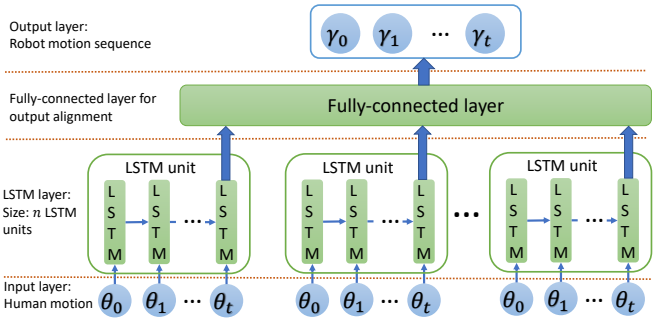


Fig. 2. The architecture of our robot motion generation network. θ_t is the sample of a human sequence at time step t and n is the number of hidden LSTM cells. The output of each LSTM unit is connected to a fully-connected layer to align the output. The robot motion sequence is the output of this fully-connected layer.

TABLE I
Collected features from demonstrations.

Features	Human	Robot
Input features (6 dimensions)	Palm position Elbow position	None
Output features (7 dimensions)	None	Joint angles

several computers to control all aspects of the experiment. As for the human collaborator, two human participants assist with data collection under two roles: (i) a robot collaborator, and (ii) a human teacher. Figure 3(a) show how a human collaborator performs the interactive task with Baxter. The human teacher uses kinesthetic guidance to teach appropriate collaborative motions. Human and robot movements are recorded at 50 Hz and are parsed into a multi-dimensional feature vector described in Table I. These features are then used as the input-output pairs to train a LSTM-RNN. During training, the fixed length human motion segments along with the corresponding robot trajectory segments are used to train the network. In this way, during testing, the robotic joint angles are generated to achieve appropriate collaborations.

D. Data pre-processing

For the purpose of training the network, it is necessary to scale the input-output data such that the ordered pairs share the same data range. In our experiment, both human arm positions and robot joint configurations are scaled to lie in the interval $[0, 1]$.

Additionally, we filter the normalized data through a Gaussian filter $g(x) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{x^2}{2\sigma^2}}$ with $\sigma = 0.2$ to reduce the impact of noise. Furthermore, to improve the robot's ability to respond to motions of different time scales, features are linearly interpolated as a function of speed values. In particular, we study five different temporal scales, with $0.5\times$, $0.75\times$, $1.0\times$, $1.25\times$ and $1.5\times$ speeds, respectively. To test the network's ability to adapt to the time scales, the testing trajectories were interpolated to six different temporal scales; namely: $0.4\times$, $0.6\times$, $0.8\times$, $1.0\times$, $1.2\times$, $1.4\times$ and $1.6\times$, respectively.

For each collaborative task, the number of demonstrations

is limited, which is not enough for training the LSTM-RNN. Thus the features of an entire task are separated into several sub-sequences, in order to increase the amount of training data and to help reduce the overfitting. We denote the features of the input and the output trajectories as $F_x^{\mathcal{H}}$ and $F_y^{\mathcal{R}}$, the time steps of the input and the output as n_x, n_y, the total number of time steps in a demonstration as T, and the index of a time step as t. The sub-sequences can then be generated as follows:

$$\begin{aligned} x &= \left\{F_x(t, ..., t+n_x-1)\right\}, \\ y &= \left\{F_y(t, ..., t+n_y-1)\right\}, \end{aligned} \tag{2}$$

where $t = 0, 1, 2, ..., T - n_x + 1$. In this way, one demonstration can be separated to $T - n_x + 1$ sequences.

III. Simulated Experiments and Results

A. Experimental setup

In Section II-C, we described the human-robot movement data collection process. In our experiments, we recorded three handover tasks that have movements ending in different positions. Each task has 30 pairs of human and robot trajectories—a total of 90 trajectory pairs—as shown in Figure 3. To train the network, we randomly selected 70% of the recorded dataset for training, 10% for validation (training and validation sets are pre-processed as described in Section II-D), and 20% for testing.

B. Selection of network parameters and sequence length

Since our data is relatively low dimensional and the dataset size is small, we empirically select a set of appropriate network parameters to avoid overfitting. Specifically, the number of LSTM units n is set to 50 and the validation result shows that the LSTM-RNN converged at around 10K iterations without overfitting. These parameters are then used in later experiments. To provide a fast response to the human motion, the robot should not stop to observe the human for a long time. This implies that the robot should only consider a brief sequence of human observations. As a result, we empirically selected the observation duration to be about 5-10% of the trajectory length for the input sequence. The average length of our demonstrations is around 130 time steps; so we test three different (n_x, n_y) setups: 10-5, 5-5, and 10-10 as the input-output sequence lengths of the LSTM-RNN.

Figure 4 shows the average mean squared error of the generated joint states with different choices of (n_x, n_y). From the figure, it can be observed that there is no obvious difference between these choices in terms of the generation error. We choose the input and output sequences to be the equal length to avoid the trouble of sequence alignments at the start and the end of the trajectory. In addition, to resolve the observation missing in the real-world experiment due to hardware limitations, we choose longer sequences to improve the robustness toward the missing data. Based on above considerations, the lengths of the input-output sequences are set to $n_x = n_y = 10$ and the time step duration in all the experiments is set to 20 ms.

C. Performance on pre-recorded data

In this section, we compare the generated robot motion using our methods and a baseline method, respectively. In order to test the trajectory generator's adaptivity to different time scales, we use two types of training sets: one is the set with only original trajectories; and the other has trajectories with five different time scales (as described in Section II-D). For convenience, we call the LSTM-RNN model learned from the first training set as the 1-speed model, and the LSTM-RNN model learned from the second training set as the 5-speed model. The baseline method is a probabilistic motion primitive based model for early motion prediction and generation [2]. Our method generates the full trajectory only after 100% observation, because it works in a batch-based manner.A s a result, we also set the baseline method at 100% observation for a fair comparison.

In order to achieve a smooth robotic motion, linear regression is used to post-process the output sequence. In addition, to evaluate the motion quality in a given task, we need to compose output sequences into a trajectory. If the generator works efficiently, there is overlap across different sequences, and such overlap is averaged as waypoint values in the trajectory. Figure 5 shows an example of the generated trajectories in joint space by the 1-speed model, and the end-effector position calculated from the generated robot joint configurations.

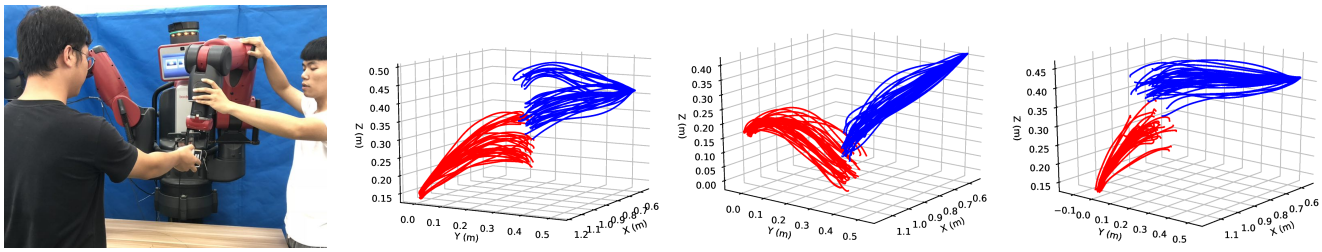
We first evaluate the performance of our system by calculating the error of the generated robot trajectories in the test set. The error between two trajectories is calculated as follows:

$$err(\gamma, \widetilde{\gamma}) = \frac{1}{T}\sum_{t=1}^{T}\left[\sum_{i=1}^{P}(\gamma_i(t) - \widetilde{\gamma}_i(t))^2\right]^{\frac{1}{2}}, \tag{3}$$

where γ and $\widetilde{\gamma}$ represents the ground truth and generated robot trajectory respectively, so $\gamma(t)$ is a waypoint at time step t of the trajectory. P is the dimension of the waypoint and T is the length of the trajectory. Because different trajectories have different length, all trajectories are re-sampled to $T = 100$.

Figure 6 shows the average trajectory errors generated by our method and by the baseline method. The average joint configuration space error for the 5-speed model, 1-speed model, and baseline methods are 0.17 ± 0.06, 0.17 ± 0.05, and 0.36 ± 0.25 radians respectively. The average end-effector error for these three methods are 5.9 ± 2.4, 5.7 ± 2.4, and 10.4 ± 5.3 cm respectively. These results demonstrate that both LSTM-RNN models outperform the baseline in all three tasks. In addition, our methods shows a smaller variance, which implies that our system works more stably. It is also interesting to see that both the 1-speed and 5-speed models are robust to different time scales in the human motion, since the test set has data with six different time scales. Therefore, the result implies that even when the training dataset did not contain data with a wide variety of timescales, the LSTM-RNN model still adapted to speed variations.

Additionally, given that we want the robot to learn to have



(a) Collect data (b) Handover task 1 (c) Handover task 2 (d) Handover task 3

Fig. 3. (a): The data collection procedure. Two human participants are working on the data collection. They are playing different roles in the experiment: the one to the left is the robot collaborator interacting with the robot, and the one to the right is the teacher who is teaching the robot to generate appropriate response to the human collaborator. (b, c, d): Collected human-robot trajectories for three different collaborative tasks in Cartesian space. We plot the x-y-z position of the human hand (red line) and the robot end-effector (blue line) to provide an intuitive illustration of the human and robot movements. From left to right, we label the tasks as Task 1, Task 2, and Task 3.

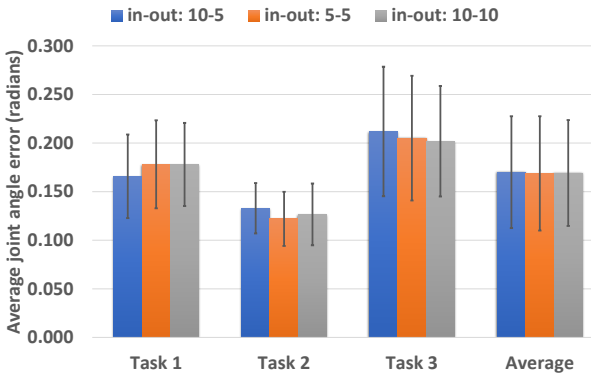


Fig. 4. The average trajectory error of joint angles from the three experimental handovers.

appropriate responses to human motion rather than reactively servoing to a target region, the quality of each part of the trajectory is also important. Figure 7 shows the average error along the trajectory. From the results, we can observe that both of our LSTM-RNN methods have a smaller error than the baseline in most part of the trajectory, especially in the middle of the motion. Though trajectories generated by the baseline method are slightly better at the end point, their error in the middle can be as large as 20 cm. This comparison of handover errors indicates that our methods learn the collaborative pattern along the entire task better than the baseline method. And the higher accuracy in the middle part of the handover shows that our methods are more robust in generating reactive motions.

IV. Real World Collaboration Experiments

TABLE II

Distance between the robot end-effector and the human hand at the human preferred handover position (cm).

Model name	Task 1	Task 2	Task 3	Average
5-speed model	5.08±1.84	7.82±1.27	22.05±0.55	11.65±7.56
1-speed model	3.77±1.36	5.56±0.41	9.54 ± 0.76	6.29 ± 2.58
baseline [3]	6.95±0.70	5.83±2.52	10.16±2.83	7.65 ± 2.88

As a more comprehensive evaluation, the LSTM-RNN models trained in the simulated experiment are then tested in a real human-robot collaborative experiment. In this experiment, a human subject is asked to perform the 3 tasks illustrated in Section III-A and Figure 3. Each task is repeated 10 times to evaluate the performance.

Given that the system continuously updates the robot motion online, the system must be sufficiently fast to capture human motion and then generate the robot motion. Our experiment learned that the time required to generate the motion sequence is around 10 ms to 15 ms with the graphics card of model Nvidia GTX 970M , and thus our system can work at 50 Hz. To evaluate the model's ability to complete the tasks, we compute the distance between the human hand and the robot end-effector at the human's preference handover point. In addition, because the human has the ability to adapt to the robot's motion, the success rates of three tasks are also recorded to evaluate the robot's ability to accomplish the task.

Table II shows the distance between the human hand and the center of the robot end-effector. All the models obtained a better result in tasks 1 and 2, and a worse result in task 3. In particular, the 1-speed model outperformed other models in all tasks. It is interesting that the 5-speed model obtained poor results in task 3. The latter indicates that there might be over-fitting in the model. In the success-rate experiment, we count a task as a success if the human successfully grabs the object handed over by the robot at the human's preferred point. In the baseline method, the model does not update the robot motion after the first motion generation, so the human can move to the location where the robot stops to grab the object. In our method, the robot continues responding to the human motion so the human can adjust his motion to make the handover possible. For the three tasks, the 5-speed model has the success rates of 90%, 100%, and 0% respectively; while the 1-speed model's success rate is 100%, 100% and 60% respectively. The 0% success rate of task 3 in the 5-speed model occurs because the robot moved to locations very far away from the human. However, the handover finally succeeds at the handover position in task 1 or 2 after the human adjusts his motion. Other failure modes are due to the robot slightly crashing into the human arm, at which

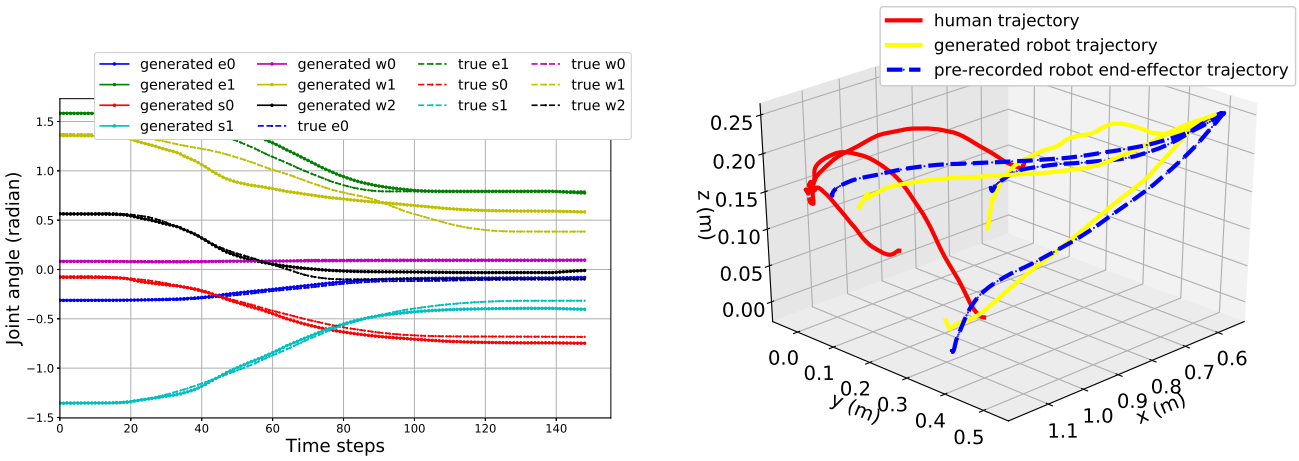


Fig. 5. Left: examples of the generated robot trajectories and the pre-recorded ground truth in the joint space, where e_0, e_1, s_0, s_1, w_0, w_1 and w_2 are the seven joints of the robot. Right: examples of human trajectories, and the robot end-effector position calculated from generated robot joints and the ground truth.

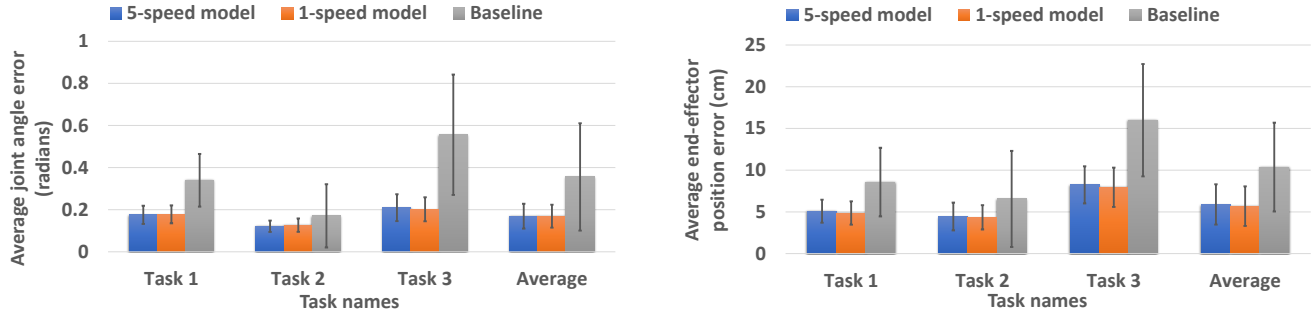


Fig. 6. The average error of the generated trajectory in the joint space (left) and in the Cartesian space (right), for three tasks implemented by three approaches. It shows that both of our LSTM-RNN models outperform the baseline method in all three tasks. In addition, our methods provides a smaller variance.

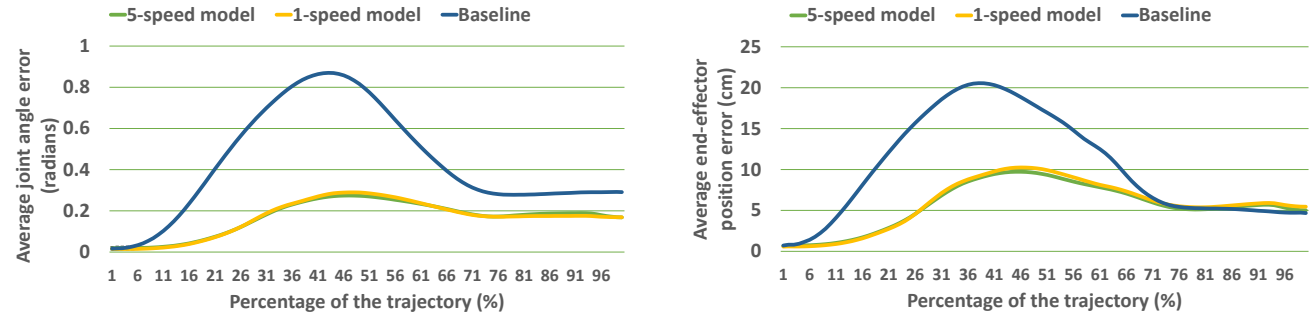


Fig. 7. The error along the generated trajectory, both in the joint space (left) and in the Cartesian space (right). It shows that our LSTM-RNN models have smaller errors along the whole trajectory in the joint space, and most of the configuration space. All methods have a small position error with the value about $5\,\mathrm{cm}$ at the end of the trajectory.

TABLE III

THE AVERAGE ERROR OF EACH JOINT ANGLE BY 3 METHODS.

Joint name (radians*10^{-2})	e_0	e_1	s_0	s_1	w_0	w_1	w_2
5-speed method	1.5 ± 1.1	8.6 ± 5.8	7.0 ± 3.7	9.2 ± 4.6	5.0 ± 1.3	12 ± 5.4	12 ± 4.6
1-speed method	1.5 ± 1.1	8.2 ± 5.8	7.4 ± 3.9	9.3 ± 4.9	5.0 ± 1.3	12 ± 5.5	12 ± 4.6
Baseline [3]	5.0 ± 4.6	21 ± 20	14 ± 15	20 ± 21	6.7 ± 3.8	23 ± 32	24 ± 31

point the human draw his hand back to avoid the collision.

Figure 8 illustrates two examples of the real motion sequences, where the robot keeps responding to the human's motion while the latter is moving. In the second sequence, the human observes that the robot may not reach his initial expected area, so she adjusts her motion to adapt to the robot.

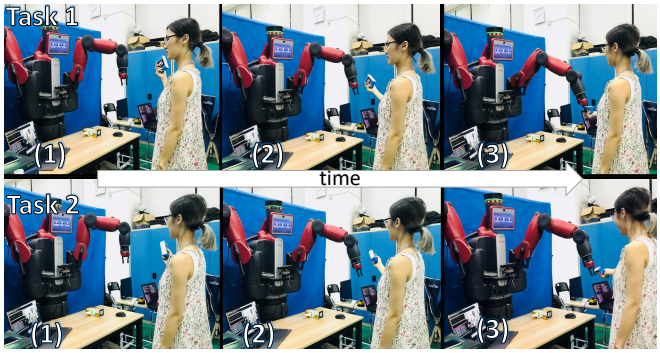


Fig. 8. Two examples of cooperative tasks sequences. The robot responses to the human motion while the human is moving. In the second sequence, the human also observes the robot motion, and then adjusts her motion to adapt to the robot.

V. Discussion

From all the experiments, results show that our models generate trajectories which are more in line with human's preferences. Especially, the 1-speed model shows good responsiveness to human motion even if the motion has multiple temporal scales. Besides, we observe that our models perform task 1 and 2 much better than task 3. To find out the reason of this kind of difference, we inspect the following factors. It is found that in the training set the robot motion of task 1 and task 2 are both at the left side of the robot, and the task 3 starts at the left side and ends at the right of the robot. In addition, when the robot moves to the middle area, it always blocks the camera and the position of human hand is not accurate. Therefore, the robot does not learn how to go right very well due to the lack of data. These above factors cause the performance difference among tasks.

Considering that the robot motion consists of the motions of its seven joints, we further calculate the average error between the generated joint angles and the pre-recorded data, and analyze the average error of each joint separately to investigate whether there is any joint better or worse than others. As shown in Table III, s_0 and s_1 represent the two shoulder joints of the left arm of the humanoid robot; e_0 and e_1 are two elbow joints; w_0, w_1 and w_2 represent three wrist joints. From the table, we can observe that w_1 and w_2 have the largest errors. Some reasons about these largest errors are as follows. Firstly, the wrist joints have more relevance to the orientation of the end-effector. Secondly, the robot end-effector orientation depends to human hand's orientation which has not been recorded, and so our models does not learn how to move the wrists well without training data. In addition, though the error of shoulder joints is smaller, a slight difference of shoulder joints may cause a big difference of the end-effector position. Therefore, we could further explore to set different weights to different joint angles during training in the future work.

VI. Conclusion

This paper proposes an LSTM-RNN based method to generate a robotic motion from the observations of the human movements for achieving fast and responsive human-robot collaborative tasks, including the human-robot handover. The robot learns the collaboration behavior directly from demonstrations without pre-programming, and shows a strong ability to adapt to movements with multiple time scales, which are common for human motions. In addition, the LSTM-RNN model is able to directly generate the motion in the joint space, and thus avoids the trouble of solving an inverse kinematics or motion planning problem.

Our paper demonstrates the possibility to apply deep learning approaches on HRC problems. However, currently we need to train separate models for different tasks and cannot switch smoothly between tasks. In particular, some collaborative tasks can be similar in the human's motion but can be very different in robot's movements. Thus for future work, we hope to combine the task recognition and the motion generation in our deep learning framework.

References

[1] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[2] G. Maeda, M. Ewerton, G. Neumann, R. Lioutikov, and J. Peters, "Phase estimation for fast action recognition and trajectory generation in human–robot collaboration," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1579–1594, 2017.

[3] L. Chen, H. Wu, S. Duan, Y. Guan, and J. Rojas, "Learning human-robot collaboration insights through the integration of muscle activity in interaction motion models," in *IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 491–496.

[4] R. Luo and D. Berenson, "A framework for unsupervised online human reaching motion recognition and early prediction," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 2426–2433.

[5] H. Arie, T. Endo, T. Arakaki, S. Sugano, and J. Tani, "Creating novel goal-directed actions at criticality: A neuro-robotic experiment," *New Mathematics and Natural Computation*, vol. 5, no. 01, pp. 307–334, 2009.

[6] S. Jeong, H. Arie, M. Lee, and J. Tani, "Neuro-robotics study on integrative learning of proactive visual attention and motor behaviors," *Cognitive neurodynamics*, vol. 6, no. 1, pp. 43–59, 2012.

[7] Y. Yamashita and J. Tani, "Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment," *PLoS computational biology*, vol. 4, no. 11, p. e1000220, 2008.

[8] H. Ochi, W. Wan, Y. Yang, N. Yamanobe, J. Pan, and K. Harada, "Deep learning scooping motion using bilateral teleoperations," *IEEE International Conference on Advanced Robotics and Mechatronics (ICARM)*, 2018.

[9] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber *et al.*, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

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