A Deep-ConvLSTM Collision Prediction Model for Manipulators in Dynamic Environment

Chang Liu, Wansong Liu, Zhu Chen, and Minghui Zheng*

Mechanical and Aerospace Engineering Department, University at Buffalo, Buffalo, NY14260, USA (e-mail: {cliu57, wansongl, zhuchen, mhzheng}@buffalo.edu)

Abstract:
Obstacle avoidance is one of the fundamental problems in human-robot collaboration (HRC) studies. The close proximity between robots and human usually leaves robots a short period of time to re-plan a safe motion, especially when facing non-static obstacles. Therefore, to identify collisions in advance and mitigate the computational efforts regarding robot motion re-planning, this paper proposes a network-based stop-go algorithm that uses only images capturing the states of the robot arm and a non-static obstacle without accessing any robot dynamics. In particular, a deep convolutional long-short-term memory (ConvLSTM) neural network is first developed to learn the spatial features of images, and predict both the robot arm and the non-static obstacle states five steps in advance. Next, the predictions are set back to the robot arm so that the robot arm would halt the current task when a potential future collision is identified. Eventually, the robot arm resumes the task after the non-static obstacle is clear. Extensive numerical studies have been conducted to validate the effectiveness of the proposed trajectory prediction scheme in presence of obstacles.

Keywords: Manipulator, ConvLSTM, Obstacle avoidance, Deep neural network

1. INTRODUCTION

Robot arms have gained growing attention for supporting human workers in collaborative tasks, particularly in the intelligent manufacturing field. The collaborative tasks, e.g., welding (Wang et al. (2018, 2019)), assembling (Roveda et al. (2021)), and disassembling (Lee et al. (2022a,b)), typically require robots and human to share a confined workspace. To ensure the effectiveness and safety of human-robot collaboration (HRC), human workers are usually considered as obstacles to be avoided, and robots must constantly monitor the workspace, assess the collision risk, and, if necessary, re-plan a safe motion.

Studies about static obstacle autonomous avoidance have been well developed in the last two decades, e.g., Rybus (2018); HE et al. (2017). In addition, when executing collaborative tasks, robots also need to be capable of avoiding the sudden entry of human workers (i.e., non-static obstacles) in real-time. Various robot planning algorithms have been developed to find feasible paths or motions for robots from a start position to a target position and inherently embrace obstacle avoidance. Duchoň et al. (2014) and Daniel et al. (2010) applied grid methods to discretize the robot task space into a grid that contains free nodes and obstacle nodes, and only searched the robot path based on free nodes. Additionally, Ratliff et al. (2009) and Schulman et al. (2014) formulated the robot path planning into optimization problems and included the obstacle avoidance as a constraint that needs to be satisfied. Furthermore, Kavraki et al. (1996) and LaValle et al. (1998) developed sampling-based path planning algorithms, i.e, PRM and RRT, to randomly generate robot configurations and only connect the configurations which are safe for the robot. What's more, Karaman and Frazzoli (2010, 2011) proposed RRT* and PRM* to turn the sampled robot trajectories to be asymptotically optimal while they are subject to high computational cost.

Human workers still may face the physical injury risk since the instant collision detection leaves limited time for baseline planners or planning algorithms of robots to re-plan a safe motion. Furthermore, it may be too late for the robot to fully stop in time to protect humans, especially when considering the dynamic constraints of motors (De Luca et al. (2006)). Therefore, it is no surprise that extensive attention has been paid to accurately detecting human presence (Sajedi et al. (2022)) and forecasting human motion in HRC such that the potential future collision can be identified. Liu and Liu (2020) utilized a recurrent neural network (RNN) to predict the human wrist position when conducting tasks and applied inverse kinematics to calculate the full arm's future motion. Dinh et al. (2015) used a minimum-jerk model to predict human motion and integrated the predicted motion into the obstacle avoidance of a robot arm. Liu et al. (2022) employed RNNs and the human arm dynamic model to forecast the human motion of reaching screwdrivers. Li et al. (2021) developed a directed acyclic graph neural network to predict the human states and shown the effectiveness of the prediction in a real human-robot interaction scenario.

Potential collisions between robots and human workers are identified using the prediction of the obstacle. The baseline planner or planning algorithms can re-plan the path or motion of robots based on the obstacle prediction to ensure the task execution and human worker's safety, e.g., Mainprice et al. (2015) and Park et al. (2013). Actually, such a re-planning poses

* * Corresponding author.
** This work was supported by the U.S. National Science Foundation under Grant No.2026533.

the requirement of the computational efficiency since collision-free configurations of robots are changing based on non-static obstacles. To reduce the computational cost of re-planning when facing obstacles, Mainprice and Berenson (2013); Luo and Berenson (2015) first computed a set of robot trajectories as candidates offline, then selected robot trajectories online based on the predicted obstacle's position. Furthermore, Zhao and Pan (2018) provided high-quality initial robot trajectories to the developed online planning framework to reduce the possibility of expensive re-planning.

The re-planning when facing obstacles is an inevitable process in the aforementioned approaches, and robots must adapt to the re-panned motion to prevent collisions from obstacles. In this paper, we propose a stop-go algorithm using networks to detect potential future collisions and avoid the re-planning process when facing a non-static obstacle. First, the workspace states including the robot arm, the non-static obstacle, the start position, and the target position are converted to 2D images. Then, a convolutional LSTM network is developed as a prediction model based on the converted images. In particular, a convolutional filter is employed to extract the positional information of objects in images as feature sequences, and a long-short-term memory with peephole connection structure is applied to learn the temporal dependences of the extracted feature sequences. Next, the collision risk is evaluated based on the position of the future robot arm and obstacle predicted by a convolutional long-short-term memory (ConvLSTM) network. Eventually, the robot arm would freeze if a future collision is identified and resume the original trajectory after the obstacle is clear. The proposed algorithm only uses images capturing the workspace without accessing any robot dynamics, and can detect the potential collision and reserve the originally planned robot trajectory.

The remainder of this paper is organized as follows. Section 2 introduces the ConvLSTM neural network. Section 3 describes the details of the generation of the manipulator configuration using forward kinematics. Section 4 presents the numerical validation details. Section 5 concludes this paper and discusses future plans.

2. NEURAL NETWORKS BASICS

2.1 Long-short-term memory

As introduced in Greff et al. (2016), Long Short-Term Memory (LSTM) is one of the special recurrent neural networks (RNN) architectures, and it is stable, reliable, and performs well on long term sequential data.

Fig. 1 shows the framework of standard LSTM with peephole connection used in Gers et al. (2002) which is also one of the most popular fully connected LSTM structures. The symbol c_t represents the cell in the LSTM structure, and it indicates a collector of the state information. It is controlled by self-parameterized gates. When a new input is injected into the cell, if input gate i_t is open, the state information will be recognized and written. When forget gate f_t is on, the past state status c_{t-1} might be forgotten. The final state is represented by h_t and o_t is the output gate. W indicates the weight matrix, h_{t-1} indicates previous hidden output and x_t is the current input. b_i, b_f, b_c, and b_o are the bias of different gates.

The numerical model of peephole LSTM is illustrated as follows.

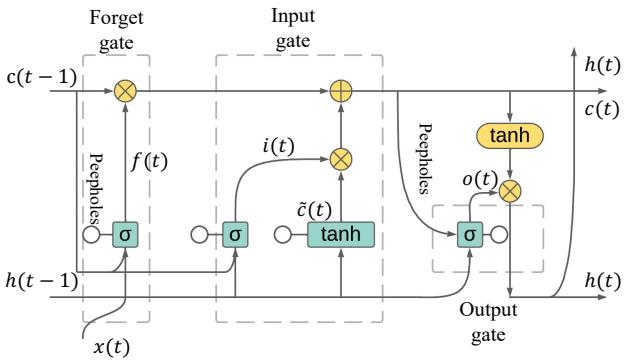

Fig. 1. LSTM architecture with a peephole connection.

$$
\begin{aligned}
i_t &= \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci} \circ c_{t-1} + b_i\right) \\
f_t &= \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf} \circ c_{t-1} + b_f\right) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right) \\
o_t &= \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co} \circ c_t + b_o\right) \\
h_t &= o_t \circ \tanh\left(c_t\right)
\end{aligned} \tag{1}
$$

The symbol $\circ$ indicates Hadamard product, and this product is explained in Eq. (2) where A and B are two matrices with same dimension.

$$
(A \circ B)_{ij} = (A \odot B)_{ij} = (A)_{ij}(B)_{ij} \tag{2}
$$

Traditional LSTM cells used to only depend on previous hidden output h_{t-1} without being directly connected with cell states. It will cause information loss which will decrease the performance of neural networks. In Gers and Schmidhuber (2000), the peephole connection is used to allow the gates to connect the past state status c_{t-1} which can significantly solve this problem and improve the performance.

2.2 ConvLSTM

The standard LSTM with peephole connection structure based on Eq. (1) focuses on sequential data. As mentioned in Kalchbrenner et al. (2015), it contains all the useful sequential data from the input state to the output state transitions, but it doesn't include the spatial information. To incorporate the spatial relationship between the manipulator and obstacle, the workspace including the manipulator and obstacle is converted to 2D images and a convolutional LSTM structure introduced in Shi et al. (2015) has been applied to extract spatial information from images using a convolutional operator $*$. Fig. 2 shows the state relationships in ConvLSTM network. As mentioned inSainath et al. (2015), the convolutional operate is used in the state-state and the input-state transitions. The images describing the movements of the manipulator and obstacle can be transferred to sequential data. ConvLSTM determines the future manipulator joint values by the current input and previous states from its neighbors.

The numerical model of ConvLSTM is introduced as Eq. (3). All the variables in ConvLSTM are the same as standard LSTM.

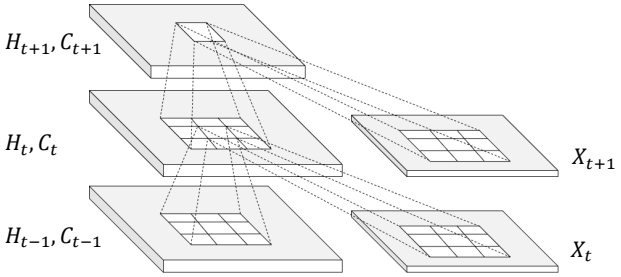

Fig. 2. State relationships in ConvLSTM neural network, ConvLSTM determines the future state by the the current input and previous states from its neighbours.

$$
\begin{aligned}
i_t &= \sigma\left(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \circ C_{t-1} + b_i\right) \\
f_t &= \sigma\left(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \circ C_{t-1} + b_f\right) \\
C_t &= f_t \circ C_{t-1} + i_t \circ \tanh\left(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c\right) \\
fo_t &= \sigma\left(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \circ C_t + b_o\right) \\
H_t &= o_t \circ \tanh\left(C_t\right)
\end{aligned} \tag{3}
$$

Compared to Eq. (3), the convolution operator $*$ replaces a part of the Hadamard operator $\circ$ in Eq. (2) The traditional LSTM layers become ConvLSTM layers. As introduced in Zhang et al. (2017), the 2D images are transformed into sequential data which contains the temporal and spatial information. ConvLSTM is able to deal with sequential image data and predict the future states of this sequence. In addition, we apply two ConvLSTM layers to construct the neural network to deal with the sequential images in this paper.

2.3 Distance calculation

As shown in Fig. 3, there are three possible relationships between the obstacle defined as a point P and robot arm AB. The minimum distances between the obstacle and the robot arm in 3 different situations will be PC, PA, and PB. The projection $\overrightarrow{AC}$ can be calculated with Eq. (4).

$$
\overrightarrow{AC} = \frac{(\overrightarrow{AP} \cdot \overrightarrow{AB})}{|\overrightarrow{AB}|^2} \cdot \overrightarrow{AB} = \frac{(\overrightarrow{AP} \cdot \overrightarrow{AB})}{|\overrightarrow{AB}|} \cdot \frac{\overrightarrow{AB}}{|\overrightarrow{AB}|}. \tag{4}
$$

We set $r = \frac{(\overrightarrow{AP} \cdot \overrightarrow{AB})}{|\overrightarrow{AB}|^2}$, when $0 < r < 1$, and the minimum distance is $|\overrightarrow{CP}|$ in Fig. 3 (a). The distance from the obstacle position P to the robot arm AB can be calculated with Eq. (5).

$$
|\overrightarrow{CP}| = \sqrt{|\overrightarrow{AP}|^2 - |\overrightarrow{AC}|^2} \tag{5}
$$

When $r < 0$, the minimum distance will be $|\overrightarrow{AP}|$ in Fig. 3 (b). The distance can be considered as $|\overrightarrow{BP}|$ in Fig. 3 (c) when $r > 1$.

3. NUMERICAL DATA GENERATION AND NETWORK TRAINING

3.1 Manipulator configuration generation

To train the neural network with sufficient manipulator configuration data, we need all possible end-effector's positions and each end-effector's position needs to have multiple corresponding manipulator configurations. To this end, the Denavit–Hartenberg (DH) transformation matrix shown in Eq. (6)

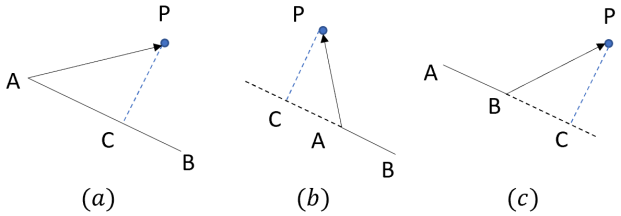

Fig. 3. Three possible C points: point C is the projection of the obstacle P over the robot arm which is denoted by line segment AB.

is applied to calculate all feasible end-effector positions based on forward kinematics.

$$
\mathbf{A}_i = \begin{bmatrix}
\cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & a_i \cos\theta_i \\
\sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & a_i \sin\theta_i \\
0 & \sin\alpha_i & \cos\alpha_i & d_i \\
0 & 0 & 0 & 1
\end{bmatrix} \tag{6}
$$

where the symbol d is the offset along the previous z axis to the common normal. Common normal in robotics is the perpendicular line between two non-intersecting joint axes. The angle θ is about the previous z axis from the old x axis to the new x axis, and α is the angle about common normal which is from the old z axis to the new z axis. And a represents the length of the common normal.

Furthermore, the manipulator end-effector's position is calculated based on the following equation:

$$
\mathbf{T}_e = \prod_{i=1}^{q} \mathbf{A}_i \tag{7}
$$

where the translation part of $\mathbf{T}_e$ is the end-effector's position and q is the number of manipulator joints. The interval of the joint angle θ is defined as ω during the forward kinematics calculation. The joint angle combinations that lead to the same end-effector's position within a task space error tolerance β are stored together into a cell. Multiple end-effector-position-based cells construct a database which could provide multiple manipulator configurations for a single end-effector's position.

3.2 Data generation

We propose a 3-link robot arm with 3-DOF to generate the stop-go data used for training in this paper. The length of the manipulator model links is 0.6m, 0.55m, and 0.5m, separately. The joint change interval ω in database construction is 3^o, and the end-effector's error tolerance β is 0.01m. To regulate the manipulator working pattern, we set the workspace of the robot arm end-effector from -0.8m to 2m in both x and y directions, and we only choose the end-effector positions within this workspace range.

The start and target position of a task sequence is randomly selected from the database constructed by forward kinematics such that we could randomly select the corresponding manipulator configuration of the start and end states. Particularly, the minimum distance between the start and target positions is defined as 0.3m. In addition, we aim to collect not only the manipulator's states but also the manipulator's reactions facing

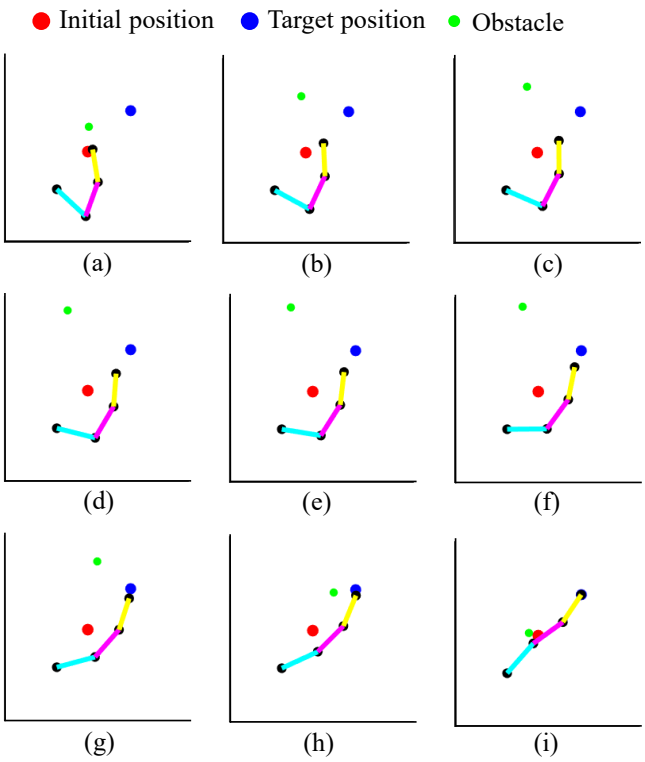

Fig. 4. Sample images of one task sequence: three links of the manipulator model are represented with different colors, and the first link is fix on the origin.

a non-static obstacle during the execution of a task sequence. Therefore, the obstacle is also considered in the data generation process. The obstacle is continuously moving despite the robot arm is moving or not. The obstacle's movement route is defined as a highly non-linear ode function to simulate real-time human activity. The data generation details are illustrated in **Algorithm 1**.

We split one task sequence equally into 20 small steps then execute each step one by one. In this paper, we aim to enable the manipulator to stop 5 steps in advance to avoid the collision with the obstacle, considering only one step leaves too short time for the manipulator to take actions. As shown in Eq. (5), we calculate the minimum distances D_1, D_2, D_3, D_4, D_5 between the robot arm and obstacle in the next 5 steps. If any calculated minimum distance is less than the safety distance that is $0.15m$, the manipulator would stop and stay at the current place. The minimum distances keep changing due to the continuous movement of the obstacle. The manipulator is required to wait until the distance satisfies the minimum distance requirement, then it can resume the designed moving route. In this case, the robot arm would move at least 20 steps to reach the end-effector's target position. To regulate the amount of data, we also set the maximum waiting steps is 40, which means the total steps to finish the whole movement cannot exceed 60.

In summary, firstly, we obtain continuously 550 task sequences totally and the target position of the current task sequence is the start position of the next task sequence. In addition, the start and end positions of each task sequence are different since the end-effector's position is randomly selected. Moreover, for each task sequence, at least 20 continuous workspace states including the manipulator and non-static obstacle are generated, and the manipulator may stop multiple times since the obstacle moves randomly and continuously.

Algorithm 1 Data Generation algorithm

Inputs:
1. Initial a feasible end-effector position P_{init}.
2. Randomly select a corresponding joint angle combination $B_{init} = [I_1, I_2, I_3]$ from the constructed database.
3. Desired number of task moving sequence $N_{ts} = 550$.
4. Initial sequence index $i = 1$.
while $i < N_{ts}$ **do**
  1. Randomly select a feasible target end-effector position as P_{end}.
  2. Randomly select a corresponding joint angle combination $B_{end} = [J_1, J_2, J_3]$ from the constructed database.
  3. Joint varieties in this sequence $B_{se} = B_{end}$ - B_{init}.
  4. Divide the moving sequence into 20 small steps, each moving step joint change interval $\gamma = B_{se}/20$.
  5. Robot arm position set at the current time step P_t, obstacle position at the current time step O_t, joint angle combination at the next time step $B_{t+1} = B_t + \gamma$.
  while $P_{t+1} \neq P_{end}$ **do**
    6. Get the next five steps robot arm position set P_{t+1} to P_{t+5} using forward kinematic and the obstacle position O_{t+1} to O_{t+5}.
    7. Calculate distance between robot arm and obstacle for next 5 steps using Eq. (5), the minimum distance as d.
    while $d \leq 0.15m$ **do**
      8. Robot arm stay at same position, obstacle continuous moving.
    end
    9. Robot arm moves to P_{t+1} based on B_{t+1}.
  end
  10. $P_{init} = P_{end}$
  11. Finish current sequence and $i = i + 1$
end

3.3 Image specification

The generated workspace states based on the stop-go algorithm is transformed into sequential images. We use the transformed images as the inputs of the neural network. We set the image axes scales from -0.8m to 2m in both x and y directions. Fig. 4 shows a few sample images generated using stop-go algorithm, and each sub-figure shows the manipulator state and the obstacle's position. Three lines with different colors indicate the three links of the manipulator model, the red dot represents the initial end-effector's position, the blue dot stands for the target end-effector's position, and the green dot is the moving obstacle. Note that the obstacle does not move continuously in Fig. 4 since we select only 9 samples from one task sequence.

3.4 Network setting

As mentioned before, we aim to utilize the ConvLSTM neural network to predict the future manipulator's states such that a potential future collision could be avoided. For each task sequence, every 5 continuous images will be the input sequence of the network. To reduce the training cost, the resolution of each image is re-sized to $(128, 128)$. We set the output of the ConvLSTM network to be the manipulator's future joint angles

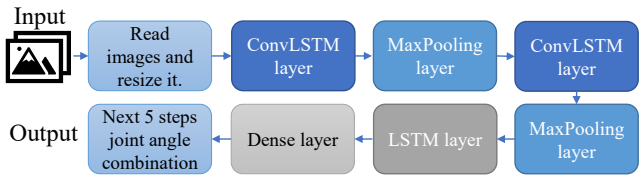

Fig. 5. Neural network structure.

considering as mentioned in Kelly et al. (2006), the control of a real manipulator is based on the joint space. In addition, the size of the network output is (5, 3) where 5 and 3 indicate 5 prediction steps and 3 manipulator joins, separately. Therefore, for one task sequence with K total images, it would generate $K-4$ sets of input and output sequences used for training.

Fig. 5 shows the neural network structure. The first ConvLSTM layer has 16 filters and its kernel size is $(3,3)$. The second ConvLSTM layer is set to have 1 filter and the same kernel size as the first one. Both Maxpooling layers have the same pool size as $(2,2)$. We choose 430 task sequences to train the network, 70 task sequences to validate during the training process, and another new 50 task sequences are used to test the performance of the well-trained network. During the training, the hyperparameters are tuned by adam optimizer, and the mean absolute error is defined as the loss function.

4. NUMERICAL STUDIES

4.1 Prediction results using test dataset

In this subsection, we briefly discuss the prediction accuracy of the trained ConvLSTM network. The total loss was set as the mean absolute error (MAE) function since the accuracy of the prediction depends on how close the predicted data is compared with the real-time data instead of relying on the variance of the prediction result. The total loss when the training is complete for 500 iterations is 0.06. The well-trained network is capable of forecasting the future manipulator's states using the test dataset. Fig. 7, Fig. 8, and Fig. 9 show the comparisons between each predicted joint angle and its ground truth data. We add the prediction results of 50 task sequences together. In addition, we only reserve the first step's prediction when constructing three comparison figures.

In summary, the well-trained neural network model can successfully predict manipulator joint angles' next change sequence. The robot arm's future states can be generated by using forward kinematics. After validating the prediction accuracy, the next step is to see if the stop-go algorithm can be implemented well when the manipulator future states predicted by the neural network are incorporated.

4.2 Obstacle avoidance test results

The main difference between the data generation and the numerical test is that the well-trained neural network is utilized to generate the next 5 steps manipulator states. Moreover, the predicted manipulator states are used to calculate the minimum safe distance from the predicted obstacle's position. Note that when using the neural network, the current manipulator state and obstacle's position need to be converted to a 2D image in real-time such that the observed workspace states represented by images can be squished to the well-trained prediction model and get the prediction results. The prediction process details are introduced in Fig. 6. Furthermore, the observed workspace images are updated recursively. The stop-go test details are illustrated in **Algorithm 2**.

Algorithm 2 Stop-go Test Algorithm

Inputs:
1. 5 observed workspace images describing the manipulator and obstacle states
2. The current end-effector position E_c and target position as P_{end}.
Initialization: Transfer the 5 steps moving data into images as an input sequence.
while $Distance(E_c, P_{end}) \geq 0.01m$ **do**
  1. Put first 5 images as an input sequence into neural networks.
  2. Predict the next five joint angles combination.
  3. Use predicted next 5 steps robot arm's position compare with the future obstacle's position, minimum distance as d.
  if $d \geq 0.15m$ **then**
    4. Robot arm and obstacle move to next position, $P_t = P_{t+1}$, and generate new workspace image.
  else
    5. Robot arm stay at same position, obstacle continuous moving, and generate new workspace image.
  end
  6. Select the last observed 4 images, and combine the new plotted image to generate the new input sequence for next step prediction.
end

Fig. 10 demonstrates the scenario where the robot arm stops moving after detecting a potential future collision. Considering a collision between the predicted robot arm and obstacle states is detected, the robot arm stops in advance shown in Fig. 10 (c). After several steps, the obstacle moves away from the robot arm, there is no risk between the robot arm and the obstacle after Fig. 10 (g), and the robot arm resumes reaching the target point. Note that, to better show the results, some frames are skipped between Fig. 10 (e) and Fig. 10 (f). The neural network can predict the next 5 trajectories of the manipulator and obstacle to successfully avoid the non-static obstacle during the validation using different task sequences from the test dataset.

5. CONCLUSIONS

This paper proposed a ConvLSTM-network-based stop-go algorithm that allows a 3-DOF manipulator model to avoid potential future collisions. In particular, we train a ConvLSTM network to predict the future manipulator joint angles. Since the training data is generated based on stop-go algorithm, the prediction manipulator states from the well-trained ConvLSTM network also consider the obstacle's position. For example, from Fig. 7, Fig. 8, and Fig. 9, we can find that the prediction data matches the real-time world data not simply in the continuous moving condition, it also matches well when the manipulator stops moving due to avoiding obstacle. These actions are shown in the figures as the short horizontal lines within the whole movement sequence. Moreover, by observing 5 sample images, the neural network is capable of providing the future manipulator states during the whole task sequence. The manipulator model successfully stops when a future collision is detected, and resumes the original moving route when the obstacle has no threat.

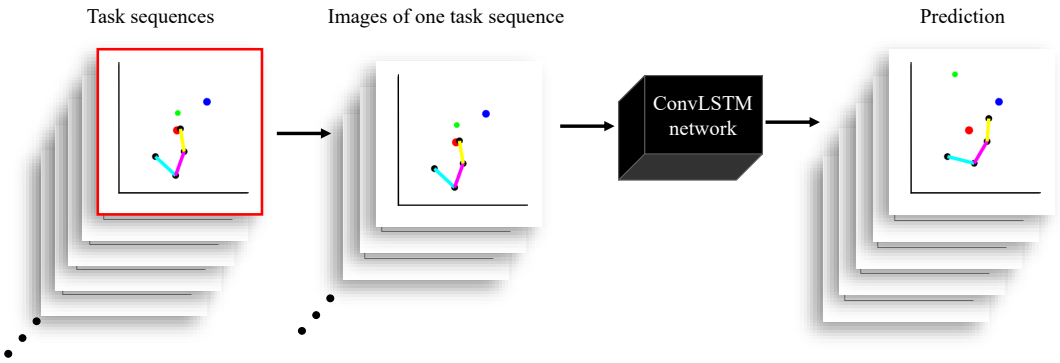

Fig. 6. The prediction process: test dataset includes multiple task sequences, each task sequence includes multiple images describing all the manipulator states from the start position to the target position, ConvLSTM network predicts the future manipulator joint angles for 5 steps, and the prediction is converted to 5 images to give a better illustration.

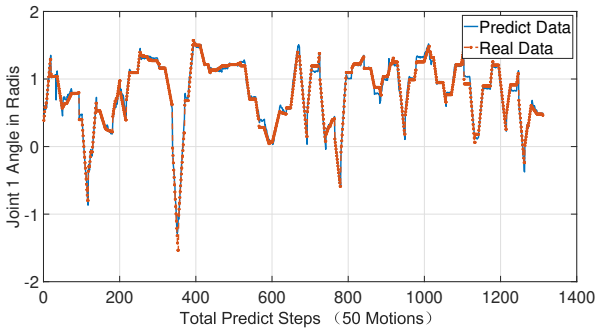

Fig. 7. The real trajectory and its prediction for joint #1.

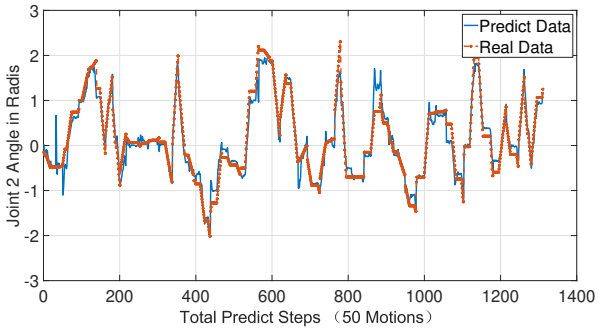

Fig. 8. The real trajectory and its prediction for joint #2.

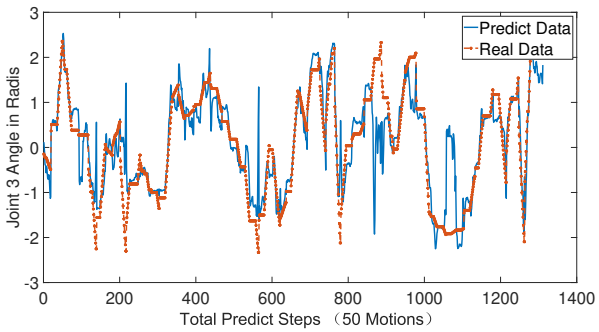

Fig. 9. The real trajectory and its prediction for joint #3.

There are two potential future work directions based on this paper. The first direction is to guarantee human safety in real HRC applications by using the same stop-go algorithm to real 3D industrial robots like 6-DOF universal robots UR series, SCARA robots, and Cartesian robots. By adding 3D cameras or multiple 2D cameras, it is possible to generate the whole manipulator working and moving map. Once we can get this kind of moving map, we can transfer the information generated by images into the neural network and make the network learn. With the development of image processing, especially it can deal with large dimension images. This technology can significantly improve the accuracy of data with the growing of large RAM in a computer.

The second direction is that by the improvement of training data sets amount, the accuracy of manipulator trajectory prediction can be increased to a higher level. When we want to use the neural network as a base of artificial intelligence, it can follow the prediction result as a direction of movement. This is a way to tell the manipulator how to move followed by its mind, but this method has a lot of constraints. The most important is the errors of prediction can be accumulated, which will make the whole manipulator go to an unpredictable position. We need to add more constraints to regulate the manipulator like using more sensors, cameras, and emergency stop functions to make it go back to the right route. But this required a much more intelligent neural network as a core for the manipulator like using reinforcement learning and transfer learning. These deep neural networks can consider more about the working environment and human safety in order to work with a human. Another question is that the manipulator is designed to assist human as a partner. But when it has its own artificial intelligent policy of moving, we cannot guarantee the human can fully trust the manipulator as a partner to finish a task. Human movement can be unpredictable due to a low level of trust when working with the manipulator.

REFERENCES

Daniel, K., Nash, A., Koenig, S., and Felner, A. (2010). Theta*: Any-angle path planning on grids. *Journal of Artificial Intelligence Research*, 39, 533–579.

De Luca, A., Albu-Schaffer, A., Haddadin, S., and Hirzinger, G. (2006). Collision detection and safe reaction with the dlr-iii lightweight manipulator arm. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1623–1630. IEEE.

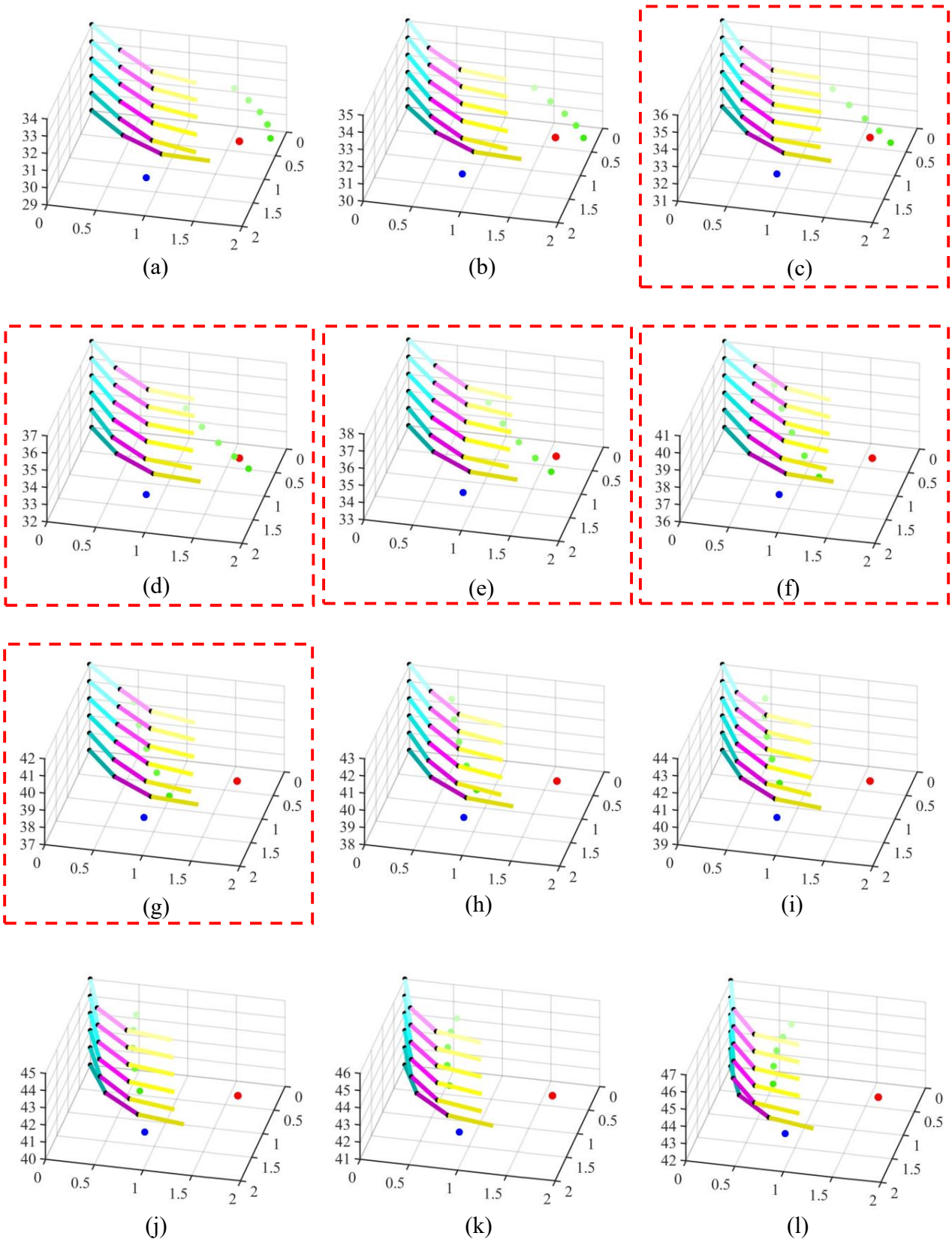

Fig. 10. Real movement test result: the red dot is the start position, the blue dot is the target position, the joints of the robot arm are represented with black dots, the base of the robot arm is fixed on (0,0) point, the current and predicted states of the robot arm are represented by lines with the corresponding gradient colors, the gradient green dots indicate the obstacle, and the frames with red dash squares show that the robot arm stops when the future collision is identified.

Dinh, K.H., Oguz, O., Huber, G., Gabler, V., and Wollherr, D. (2015). An approach to integrate human motion prediction into local obstacle avoidance in close human-robot collaboration. In *2015 IEEE International Workshop on Advanced Robotics and its Social Impacts (ARSO)*, 1–6. IEEE.

Duchoň, F., Babinec, A., Kajan, M., Beňo, P., Florek, M., Fico, T., and Jurišica, L. (2014). Path planning with modified a star algorithm for a mobile robot. *Procedia Engineering*, 96, 59–69.

Gers, F.A. and Schmidhuber, J. (2000). Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 3, 189–194. IEEE.

Gers, F.A., Schraudolph, N.N., and Schmidhuber, J. (2002). Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug), 115–143.

Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., and Schmidhuber, J. (2016). Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10), 2222–2232.

HE, Z., HE, Y., and Zeng, B. (2017). Obstacle avoidence path planning for robot arm based on mixed algorithm of artificial potential field method and rrt. *Industrial Engineering Journal*, 20(2), 56.

Kalchbrenner, N., Danihelka, I., and Graves, A. (2015). Grid long short-term memory. *arXiv preprint arXiv:1507.01526*.

Karaman, S. and Frazzoli, E. (2010). Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems VI*, 104(2).

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7), 846–894.

Kavraki, L.E., Svestka, P., Latombe, J.C., and Overmars, M.H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4), 566–580.

Kelly, R., Davila, V.S., and Perez, J.A.L. (2006). *Control of robot manipulators in joint space*. Springer Science & Business Media.

LaValle, S.M. et al. (1998). Rapidly-exploring random trees: A new tool for path planning.

Lee, M.L., Behdad, S., Liang, X., and Zheng, M. (2022a). Task allocation and planning for product disassembly with human–robot collaboration. *Robotics and Computer-Integrated Manufacturing*, 76, 102306.

Lee, M.L., Liu, W., Behdad, S., Liang, X., and Zheng, M. (2022b). Robot-assisted disassembly sequence planning with real-time human motion prediction. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, to appear.

Li, Q., Chalvatzaki, G., Peters, J., and Wang, Y. (2021). Directed acyclic graph neural network for human motion prediction. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 3197–3204. IEEE.

Liu, R. and Liu, C. (2020). Human motion prediction using adaptable recurrent neural networks and inverse kinematics. *IEEE Control Systems Letters*, 5(5), 1651–1656.

Liu, W., Liang, X., and Zheng, M. (2022). Dynamic model informed human motion prediction based on unscented kalman filter. *IEEE/ASME Transactions on Mechatronics*. doi: 10.1109/TMECH.2022.3173167.

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

Mainprice, J. and Berenson, D. (2013). Human-robot collaborative manipulation planning using early prediction of human motion. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 299–306. IEEE.

Mainprice, J., Hayne, R., and Berenson, D. (2015). Predicting human reaching motion in collaborative tasks using inverse optimal control and iterative re-planning. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 885–892. IEEE.

Park, C., Pan, J., and Manocha, D. (2013). Real-time optimization-based planning in dynamic environments using gpus. In *2013 IEEE International Conference on Robotics and Automation*, 4090–4097. IEEE.

Ratliff, N., Zucker, M., Bagnell, J.A., and Srinivasa, S. (2009). Chomp: Gradient optimization techniques for efficient motion planning. In *2009 IEEE International Conference on Robotics and Automation*, 489–494. IEEE.

Roveda, L., Magni, M., Cantoni, M., Piga, D., and Bucca, G. (2021). Human–robot collaboration in sensorless assembly task learning enhanced by uncertainties adaptation via bayesian optimization. *Robotics and Autonomous Systems*, 136, 103711.

Rybus, T. (2018). Obstacle avoidance in space robotics: Review of major challenges and proposed solutions. *Progress in Aerospace Sciences*, 101, 31–48.

Sainath, T.N., Vinyals, O., Senior, A., and Sak, H. (2015). Convolutional, long short-term memory, fully connected deep neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4580–4584. IEEE.

Sajedi, S., Liu, W., Eltouny, K., Behdad, S., Zheng, M., and Liang, X. (2022). Uncertainty-assisted image-processing for human-robot close collaboration. *IEEE Robotics and Automation Letters*, 7(2), 4236–4243.

Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., Pan, J., Patil, S., Goldberg, K., and Abbeel, P. (2014). Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9), 1251–1270.

Shi, X., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.K., and Woo, W.c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 802–810.

Wang, C., Zheng, M., Wang, Z., Peng, C., and Tomizuka, M. (2018). Robust iterative learning control for vibration suppression of industrial robot manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 140(1), 011003.

Wang, Q., Cheng, Y., Jiao, W., Johnson, M.T., and Zhang, Y. (2019). Virtual reality human-robot collaborative welding: a case study of weaving gas tungsten arc welding. *Journal of Manufacturing Processes*, 48, 210–217.

Zhang, K., Zuo, W., Gu, S., and Zhang, L. (2017). Learning deep cnn denoiser prior for image restoration. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3929–3938.

Zhao, X. and Pan, J. (2018). Considering human behavior in motion planning for smooth human-robot collaboration in close proximity. In *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 985–990. IEEE.