Thesis for Master's Degree

Position Control of Cable-Driven Parallel Robot with Combining Recurrent Neural Network and Feedback Algorithm.

Nguyen Dinh Toan

Department of Mechanical Engineering

Gachon University, Korea

December, 2020

Thesis for Master's Degree

Position Control of Cable-Driven Parallel Robot with Combining Recurrent Neural Network and Feedback Algorithm.

Nguyen Dinh Toan

Department of Mechanical Engineering

Gachon University, Korea

December, 2020

Accepted in Fulfillment of the Requirements for the Degree of Master of Mechanical Engineering

December, 2020

Committee Chairman

Committee Member

Committee Member

Abstract.

Position Control of Cable-Driven Parallel Robot with Combining Recurrent Neural Network and Feedback Algorithm.

Nguyen Dinh Toan

Advised by Professor Kyoung-Su Park

Department of Mechanical Engineering

Gachon University – Seongnam City

Currently, the cable-driven parallel robot (CDPR) have been applied in various industries such as 3D printing, painting and so on. These CDPR have mainly used steel cables or polymer cables. Steel cables have big advantages for high loading performance and high safety, but they have a big bending stiffness and the rigidity can be largely changed by the slippage of cables when they are bended sometimes. Polymer cables can overcome these disadvantage of the steel cables. However, they have many kinds of nonlinear elongation dynamic characteristics such as hysteresis, creep and so on. In this paper, we used polymer cables in CDPR system. In order to compensate the

nonlinear characteristics, we proposed advanced control algorithm using recurrent neural network (RNN) feedback method. To do so, the necessary input and output data was measured while the CDPRs was running and we then trained a deep unidirectional RNN and bidirectional recurrent neural network (Bi-RNN) to build error prediction model. Based on the predicting error model, we made some feedback controller in order to improve the position accuracy. The validity of the approach was demonstrated by the execution of 3D circle trajectory. The accuracy was shown to be significantly improved when compared with simple inverse kinematic controller and baseline error feedback controller.

Key words:

Cable-driven parallel robot, Recurrent neural network, Bidirectional recurrent neural network, Prediction, Compensation.

Contents.

List of Figures.

List of Tables.

Abbreviation.

Cable-driven parallel robot ……..…….... CDPR

Recurrent neural network …….…… RNN

Bidirectional recurrent neural network ……...…...… Bi-RNN

Deep neural networks ……..…….…. DNN

Long short term memory …… LSTM

End-effector ……..……... EE

Six-degree of freedom …….…….....….6-DOF

Inverse kinematic …….…… IK

eXtended Automation Engineering …….……. XAE

Programmable logic controller …….……. PLC

Computer numerical control ……..……... CNC

I. Introduction.

Cable-driven parallel robot (CDPR) are a special type of traditional parallel robots in which the rigid links are replaced by flexible cables. CDPR possess some useful properties such as large workspace, relatively easy design, assembly/disassembly and maintenance. Consequently, CDPR are very suitable for many applications in academia and industry [1] such as very large telescopes [2-3], high speed manipulation [4-5], 3D printing [6], the precision and high sensitivity industries like as surgery, pick and place robot [7-8], sandblasting and painting [9].

Figure 1: 3D printing [6]

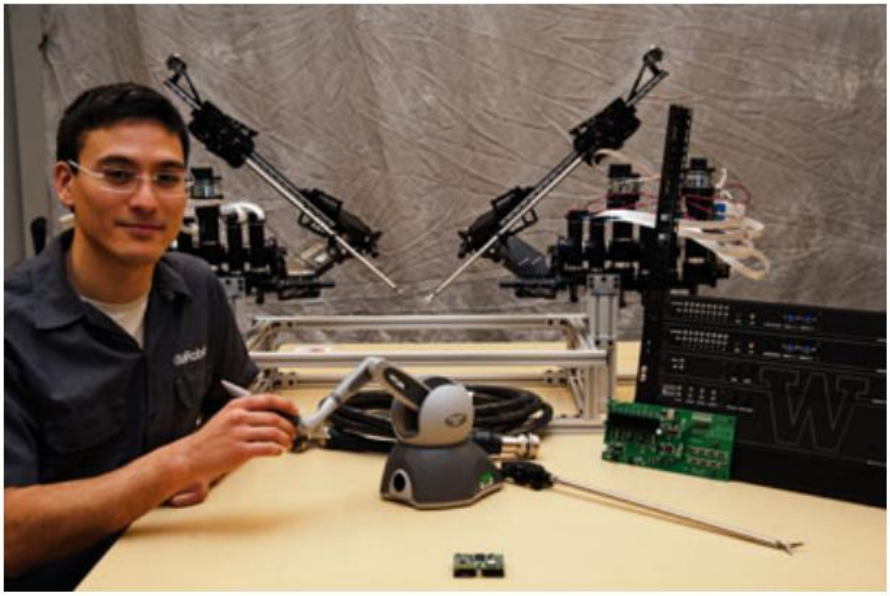


Figure 2: Raven Surgical Robot [8].

Figure 3: Artistic exhibition with 3D printing [9].

There are two mainly types of cable: steel cables and polymer cables. The steel cables have big advantages for high loading and high safety, but they have a big bending stiffness and the rigidity can be largely changed by the

slippage of the cable when they are bended sometimes. In contrast, polymer cables can overcome these disadvantage of the steel cables. are mainly used for the high speed or high precision CDPR because of high energy absorption and lower weight [11]. However, there are a lot of researches to show that polymer cables had more complex elastic behavior. It showed that polymer cables have a changing elastic behavior over time, are sensitive to overload, and show elongation, creep, recovery, hysteresis effects [12]. Thus, many author have been tried to clarify characteristic of cables and improved the tracking performance of CDPR obviously. Huang et al. [13] and Tempel et al. [14] modeled the cable to consider elastic-flexible cable behavior. Miyasaka et al. [15] used Bouc-Wen hysteresis model to develop the longitudinally loaded cables. Kraus et al. [16] increased accuracy of a cable under changing payload, by compensating the error caused by elasticity. Choi et al. [17] proposed the integrated non-linear dynamic cable model based on considering viscoelastic model and all of the non-linear properties of cables. Overall, those approaches are not applicable in real time CDPR control systems because of the complexity of the model or the high computational cost.

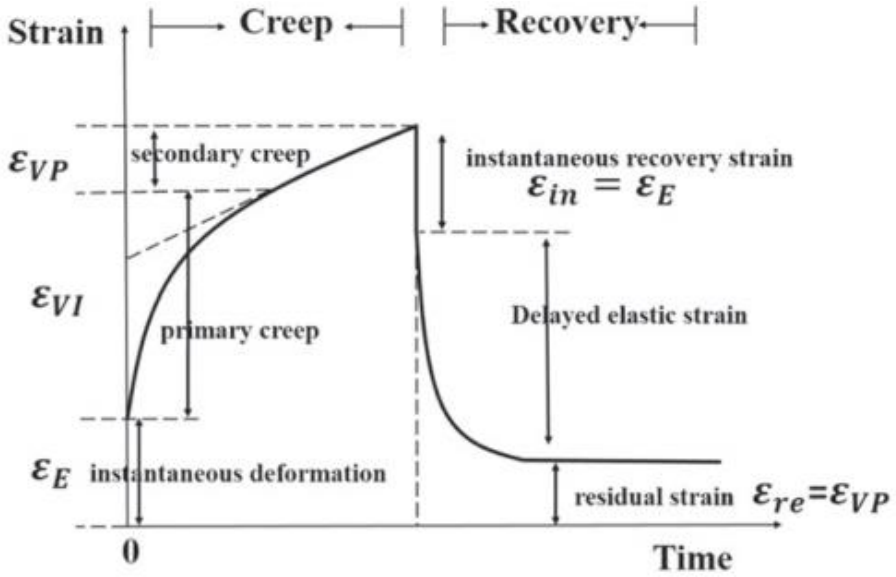


Figure 4: Typical creep and recovery of polymer cable [10].

Recently, a promising approach is to use deep learning techniques to model systems. Indeed, these approaches are generic and do not require knowledge of the specific construction of system, that can be employed to perform prediction and classification operations based on highly complex training data. Due to the unprecedented advances in deep learning, many scientific fields exploit its high accurate performance to build efficient solutions to different kind of problems. Deep neural networks (DNN) showed a superior performance in many other areas of applications such as signal processing, speech recognition and image classification. Therefore, exploring DNN techniques in robotic fields is a highly recommended solution. Li et al. [18] proposed fuzzy neural network to estimate the inverse dynamic of the

process and handle uncertainties in CDPR. Piao et al. [19] presented DNN based on force control to estimate the force distribution of the end-effector. Other researchers have applied various deep learning techniques to time series prediction. Deep recurrent neural network (RNN) is one of the techniques employed by many prediction approaches. These techniques can remember preceding data inputs while using current data to learn network weights and be applied in many studies [20-24]. However, RNN suffer from problem like vanishing and exploding gradients. Long short term memory (LSTM), an extension of RNN, solves the vanishing gradient problem by introducing memory cells with controlling gates and have been used extensively to simulate the time series correlation [25]. Further extensions to RNN have recently become popular where the output is obtained by exploring not only the past but also the future context [26]. These networks connect the outputs from two separate hidden layers (which scan the input sequences in opposite direction) to the same output layer making them bidirectional recurrent neural network (Bi-RNN). Some researchers find it has excellent performance on 1-dimensional time series data prediction, such as stock market prediction [27], energy load forecasting [28] or for traffic speed prediction [29].

In this work, to improve the trajectory tracking control of end-effector (EE) in CDPR, we proposed a neural learning control for cable driven parallel

robot based on combining time series prediction model and feedback algorithm. Firstly, we collected the response position error of end-effector for a large amount of specified trajectories. Then, we trained a deep unidirectional RNN and bidirectional RNN to build error prediction model. Based on the error prediction model, we made some feedback controller in order to improve the position accuracy. In the feedback loop, we inserted the feedback of applied tension into the predicting deep learning model. By comparing with the performance of simple inverse kinematic controller and baseline error feedback controller, we demonstrated that both of two approaches work effectively in reducing the trajectory tracking error.

II. Description of CDPR.

1. Fully constrained CDPR system.

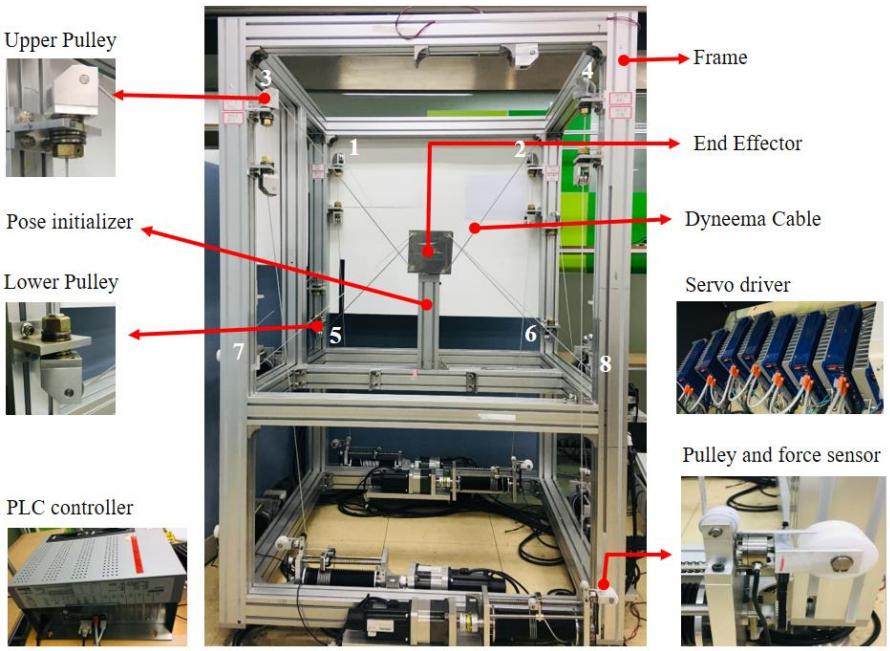


Figure 5: Cable-driven parallel robot (CDPR).

The designed CDPR system is shown in Fig. 5 [30]. The system parameters are listed in Table 1. There are four main parts:

- Winches-servo system: AC servo motor, reducer, drum, …
- Measuring cable tension: Load cell.
- Cables.
- End-effector.

In cable-driven parallel robot, the cable length errors are typically caused by a lack of mechanical actuator accuracy, for instance, the overlapping of cable on the drum as Fig. 6 [31] or the direction of cables changes continuously during operation of cable robot.

Figure 6: Overlapping cable [3].

Therefore, we designed the winches-servo system with ball screw mechanism as Fig. 7.

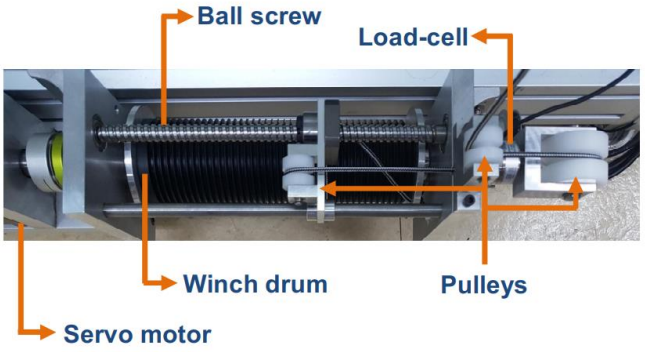


Figure 7: Winches-servo system with ball screw.

TwinCAT3 is used for the real-time control software of the cable robot. The software is used as a programmable logic controller (PLC) and a time controller that supports the computer numerical control (CNC) system for the kinematic transformation and motor control. Specifically, TwinCAT3 provides eXtended Automation Engineering (XAE), based on the widely used Microsoft Visual Studio platform. TwinCAT3 adds support for the IEC61131-3 programming standard into the development environment. TwinCAT System Manager, the configuration tool for connectivity to ERP and peripheral devices such as I/O and Motion, is integrated into the same development environment. This approach allows the integration of additional programming languages or tools, such as MATLAB, Simulink and C/C++.

Table 1: Cable robot: parameters and main components.

Parameter	Value	Unit
Number of cable	8	_
Degree of freedom	6	_
Size of the robot frame	1 x1 x1	m
Size of end-effectors	65 x65 x65	mm
Type of cable	Dyneema SK78	_
Maximum of cable tension	2000	N
Cable diameter	3	mm

Winch diameter	64	mm
Pulley diameter	13	mm
Cable force sensor	Dacell UMM K200	_
Drive	Beckhoff C6650-0030	_
Gear ratio	3:1	_
Type of motor	HIGEN FMACN06-PB10	_

Also, camera vision system is installed to measure the position of end-effector as Fig. 8. The system is implemented using OpenCV library with python programming language in Spyder environment and is capable of tracking moving end-effector by Logitech camera with resolution of 1280x720 and at frame rate of 30 fps.

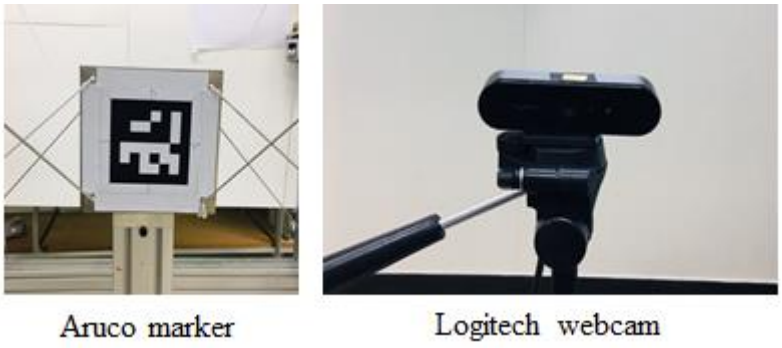


Figure 8: Aruco marker and Logitech camera

Aruco library [32] is used to detect and track markers in images, which attached to end-effector. This library is open source, kept up to date, robust to different lighting conditions, runs in real time and it provides good results in reasonable time.

2. Kinematic modeling.

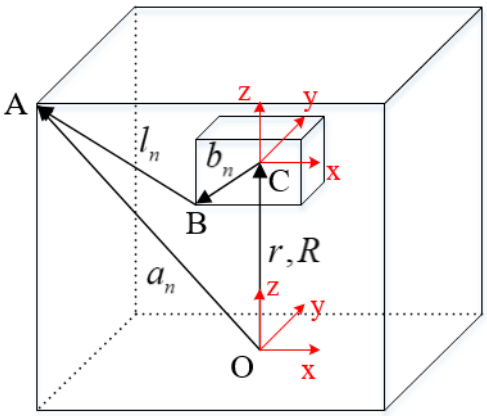


Figure 9: Simple kinematic drawing of the CDPR

For completeness, we briefly describe the simplified kinematic model of cable robot as standard model. The Fig. 9 shows the geometry of the robot in global coordinate frame. We determined two coordinates system: Global frame noted as (*Oxyz*) and moving frame (*Cxyz*) is on end-effector and moved with end-effector. We choose the origin point C of moving frame at center of mass of end-effector. Vector a_n denote the proximal anchor points on the robot base A, vector b_n denote the relative positions of the distal anchor points on the end-effector and l_n denote the vector of the cables. Based on the vector loop, the cable length vector l_n can be described as:

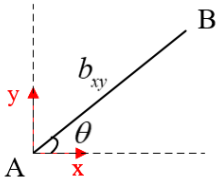
$$l_n = a_n - r - Rb_n \text{ for } n = 1, \ldots, 8 \quad (1)$$

where matrix R and vector r is rotation matrix and position vector of end-effector respectively.

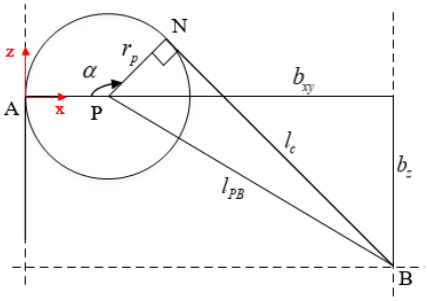
The inverse kinematics solution for the cable robot is simply the length of the cable vector and computed as:

$$l = \| l_n \| \tag{2}$$

(a): Pulley.



(b): Pulley rotated around z-axis



(c): Simple kinematic drawing of the pulley.

Figure 10: Pulley and geometry of pulley kinematic.(a). Pulley. (b): Pulley rotated around z-axis. (c). Simple kinematic drawing of the pulley.

The parameters and coordinate frame used to exactly define the geometry of a guiding pulley are described in Fig. 10 [33]. In this figure, the index n of reference points, frames, angles, lengths are omitted to easy following. There are two types of motion of pulley. The first motion, pulley rotates around the z-axis of local frame (*Axyz*). The second motion, pulley rotates around the center P. Point A and point N denote the start point and end point which cable wrap around pulley. The cable wrap around pulley with starting point A and ending point N. Define α is the angle between point A and point N. Define l_c denotes the length of cable from the anchor points B on the end-effector to the point N.

With $b_{xy} = \sqrt{b_x^2 + b_y^2}$ and b_z are coordinate of point B in the frame (*Axyz*), the distance l_{PB} between the point P and B is calculated by:

$$l_{PB} = \sqrt{(b_{xy} - r_p)^2 + b_z^2} \tag{3}$$

where r_d is pulley radius.

By the Pythagorean theorem, the cable length l_c between anchor point on end-effector and pulley are given by:

$$l_c = \sqrt{l_d^2 - r_p^2} \tag{4}$$

Using elementary trigonometric function yield, the angle α around pulley is calculated by:

$$\alpha = \arccos(\frac{l_c}{l_{PB}}) + \arccos(\frac{b_z}{l_{PB}}) \quad (5)$$

The correct cable length l_d to solve inverse kinematic is calculated by:

$$l_d = l_c + \alpha r_p \quad (6)$$

Another way to calculate equation (2).

III. Methodology.

1. Simple inverse kinematic controller (IKC).

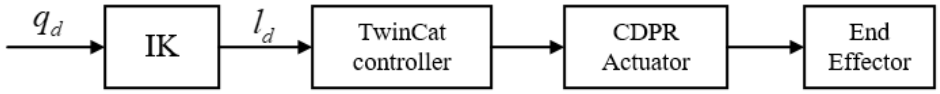


Figure 11: A block diagram of position control CDPR with IK.

For the operation of CDPR, the desired position of the end-effector needs to be controlled by the cable length adjustment. Therefore, we have the relationship between desired position of end-effector (q_d) and desired cable length (l_d) as:

$$l_d = f(q_d) \text{ for } l_d \in \mathbb{R}^{8\times1} \text{ and } q_d \in \mathbb{R}^{3\times1} \tag{7}$$

After computed the cable lengths from Eq. (6), we set its to the CDPR actuator via TwinCat controller, as shown in Fig. 11.

2. Baseline error feedback controller (EFBC).

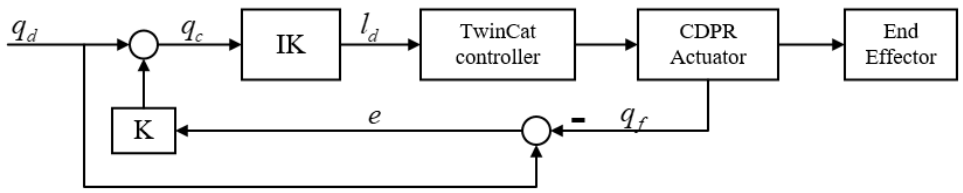


Figure 12: A block diagram of position control CDPRs with IK and error feedback.

Fig. 12 shows the control diagram of system composed of inverse kinematic and error feedback components. The control law is as below:

$$q_c(t+1) = q_d(t+1) + K * e(t) \tag{8}$$

$$l_d = f(q_c) \text{ for } l_d \in \mathbb{R}^{8\times1} \text{ and } q_c \in \mathbb{R}^{3\times1} \tag{9}$$

where $q_c(t+1)$ is the command position at time t+1, $q_d(t+1)$ is the desired position at time t+1, $e(t)$ is the position error feedback and K is proportional coefficient.

3. Proposed controller.

In this section, we proposed the controller composed feedback component as illustrated in Fig. 13. The current pose of end effector ($q_f(t)$) is compared to a known desired pose ($q_d(t)$) and an error vector ($e(t)$) is defined as $e(t) = q_d(t) - q_f(t)$. An error vector ($e(t)$) and cable tension feedback vector ($T(t)$) are processed with time series prediction algorithm, that return an error vector ($\tilde{e}(t+1)$) of end effector at next step time. To

achieve the actual pose of end effector as the desired pose ($q_d(t+1)$), the actual input command ($q_c(t+1)$)to inverse kinematic controller is compensated as below:

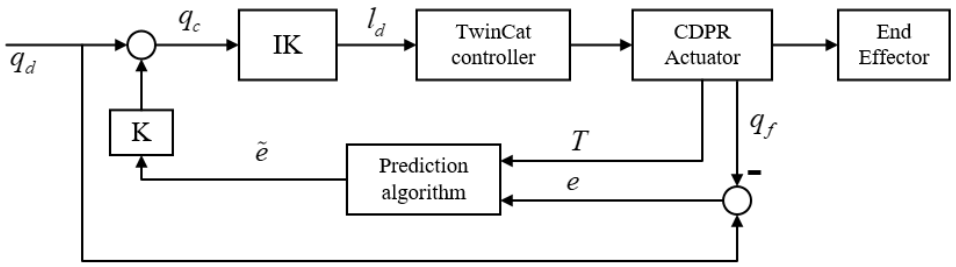


Figure 13: Block diagram of proposed controller.

$$q_c(t+1) = q_d(t+1) + K * \tilde{e}(t+1) \tag{10}$$

$$l_d = f(q_c) \text{ for } l_d \in \mathbb{R}^{8\times1} \text{ and } q_c \in \mathbb{R}^{3\times1} \tag{11}$$

where K is the gain of compensator.

IV. Time series prediction algorithms.

1. The concept of neural network.

The basic idea behind a neural network is to simulate (copy in a simplified but reasonably faithful way) lots of densely interconnected brain cells inside a computer so you can get it to learn things, recognize patterns and make decisions in a humanlike way. Neural networks are typically organized in layers as Fig. 14 [34]. The first layer is the input layer, it picks up the input signals and passes them to the next layer. The next layer does all kinds of calculations and feature extractions, it's called the hidden layer. Often, there will be more than one hidden layer. And finally, there is an output layer, which delivers the final result.

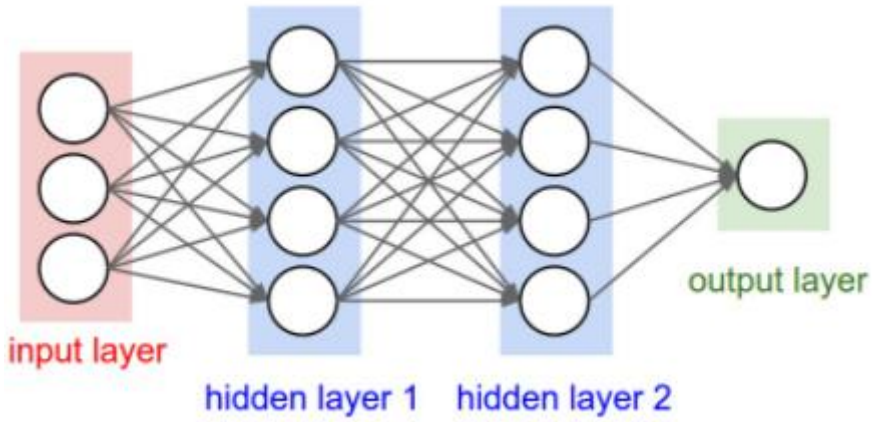


Figure 14: Concept of neural network.

Fig. 15 illustrates a mathematical model of neuron in a neural network. The input multiplied by the connection weight are first summed and then passed through a transfer function to produce the output for that neuron. The activation function is the weight sum of the neuron's inputs and the most commonly used transfer function is the sigmoid function.

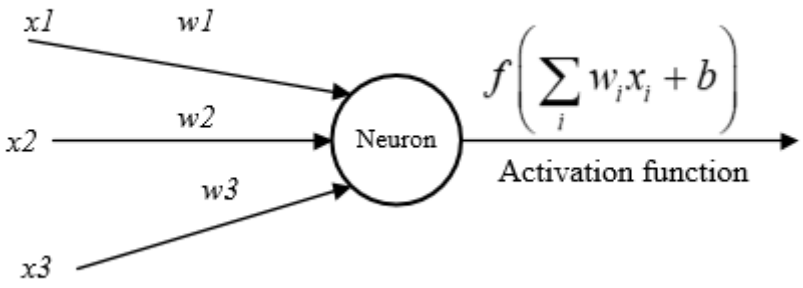


Figure 15: Model of the neuron in a neural network.

These artificial networks may be used for predictive modeling, adaptive control and applications where they can be trained via dataset.

2. Recurrent neural network (RNN).

Recurrent neural network is one of the techniques employed by many prediction approaches. These techniques can remember preceding data inputs and it is decisions are influenced by what it has learnt from the past. Fig. 16

show the schematic of RNN. The formula for current state of hidden layer $h_t \in \mathbb{R}^{n\times h}$ at time *t* can be written as:

$$h_t = \sigma(x_t W_x + H_{t-1} W_h + b) \tag{12}$$

where $x_t \in \mathbb{R}^{n\times d}$ is the current input to the RNN at time t (number of example *n*, number of inputs in each example *d*). $W_x \in \mathbb{R}^{d\times h}$, $W_h \in \mathbb{R}^{h\times h}$, $b \in \mathbb{R}^{n\times h}$ are weight matrix and bias vector for RNN. σ function means the activation function such as sigmoid, tanh and softsign.

Now, once the current state is calculated, we can calculate the output state $y_t \in \mathbb{R}^{n\times q}$ at time *t* as:

$$y_t = W_y h_t + b_y \tag{13}$$

where the weight $W_y \in \mathbb{R}^{h\times q}$ and the bias $b_y \in \mathbb{R}^{n\times q}$ are the model parameters of the output layer.

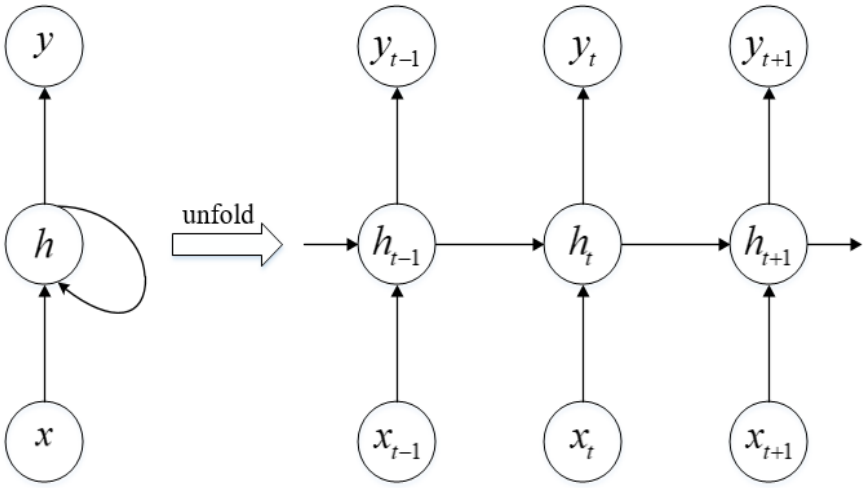


Figure 16: Standard RNN architecture and an unfolded structure.

One issue with RNN in general is known as the vanishing gradients problem. This problem states that, for long input-output sequences, RNN have trouble modeling long term dependencies, that is, the relationship between elements in the sequence that are separated by large periods of time. This problem arises due to the use of the chain rule in the backpropagation algorithm.

3. Long short term memory (LSTM).

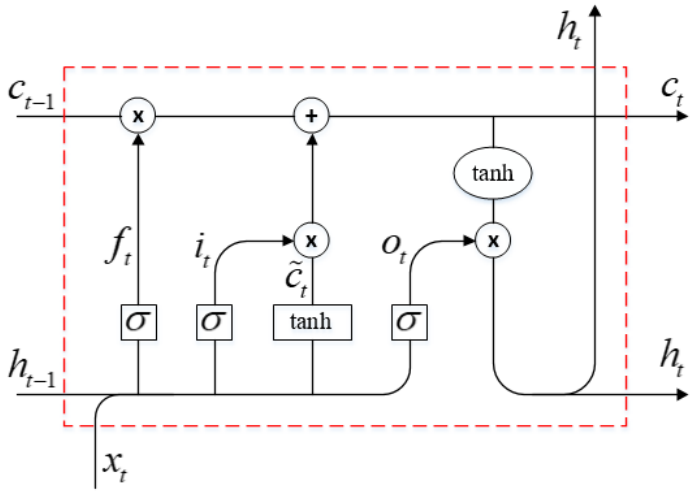


Figure 17: The structure of long short term memory (LSTM).

Due to the vanishing gradient problem, many variations of RNN have been developed to solve this problem, one of them is long short term memory (LSTM) [25], which is proposed by Hochreiter and Schmidhube. It defines gated cell that can act on the received input by blocking or passing information based on the importance of the data element. The learning process through backpropagation estimates the weights that allow the data in the cells either to be stored or deleted. The LSTM transition equations are as follows:

$$f_t = \sigma(x_t U_f + h_{t-1} W_f + b_f) \tag{14}$$

$$i_t = \sigma(x_t U_i + h_{t-1} W_i + b_i) \tag{15}$$

$$o_t = \sigma(x_t U_o + h_{t-1} W_o + b_o) \tag{16}$$

$$\tilde{c}_t = \tanh(x_t U_c + h_{t-1} W_c + b_c) \tag{17}$$

$$c_t = f_t c_{t-1} + i_t \tilde{c}_t \tag{18}$$

$$h_t = o_t \tanh(c_t) \tag{19}$$

where $x_t \in \mathbb{R}^{n \times d}$ is input vector to the LSTM unit (number of example *n*, number of inputs in each example *d*), $c_t \in \mathbb{R}^{n \times h}$ denotes the cell state vector, and $h_t \in \mathbb{R}^{n \times h}$ is the hidden state vector, also known as output vector of the LSTM unit. $i_t \in \mathbb{R}^{n \times h}$, $f_t \in \mathbb{R}^{n \times h}$, and $o_t \in \mathbb{R}^{n \times h}$ are input gates vector, forget gates vector, and output gates vector, respectively, and $\tilde{c}_t \in \mathbb{R}^{n \times h}$ is cell input activation vector.

First, the output of the previous cell and the input of the current cell determine how much of the previous cell state is stored through the forget gate f_t (Eq. (14)). At this time, the output value has a value between 0 and 1 by the sigmoid function, which is multiplied by the cell state c_t. If the cell state is multiplied by 1, the current cell is perfectly remembered, and if it is multiplied by 0, the value is removed. Next, the input gate determines how

much to reflect the input training set (Eq. (15)), and is added to the cell state. Finally, we determine the output from the cell state through the output gate O_t (Eq. (16)). The weight of each gate is determined by back propagation. Through interaction of each gate, LSTM cell can reduce vanishing gradient problem even in long-term data. Therefore, it is mainly used for natural language processing, speech recognition, and stock price analysis.

4. Bidirectional recurrent neural network (Bi-RNN).

Another variation of recurrent neural network is bidirectional recurrent neural network (Bi-RNN), which is developed by Schuster and Paliwal [26] to train the network using input data sequences in the past and future. Fig. 13 illustrates the architecture of a Bi-RNN with a single hidden layer. The principle of Bi-RNN is to split the neurons of a regular RNN into two directions, one for positive time direction (forward states), and another for negative time direction (backward states). Those two states output are not connected to inputs of the opposite direction states. The forward component computes the hidden and cell states similar to a standard unidirectional RNN whereas the backward component computes them by taking the input sequence in a reverse order, starting from time step T to 1. The intuition of using backward component is that we are creating a way where the network sees future data and learns its weights accordingly. This might help the network to capture some dependencies which otherwise wouldn't have been captured by the standard (forward) RNN. In Bi-RNN the forward component's hidden and cell states are different from those of the backward components. So, the hidden and cell states of the forward component have to be concatenated with those of backward component respectively.

For time step *t*, given a mini batch input $x_t \in \mathbb{R}^{n \times d}$ (number of example n, number of inputs in each example d) and let the hidden layer activation function be σ. In the bidirectional architecture, we assume that the forward and backward hidden states for this time step are $\vec{h} \in \mathbb{R}^{n \times h}$ and $\overleftarrow{h} \in \mathbb{R}^{n \times h}$, respectively, where h is the number of hidden units. The forward and backward hidden state updates are as follows:

$$\vec{h}_t = \sigma(x_t W_x^{(f)} + \vec{h}_{t-1} W_h^{(f)} + b^{(f)}) \tag{20}$$

$$\overleftarrow{h}_t = \sigma(x_t W_x^{(b)} + \overleftarrow{h}_{t+1} W_h^{(b)} + b^{(b)}) \tag{21}$$

where the weights $W_x^{(f)} \in \mathbb{R}^{d \times h}$, $W_h^{(f)} \in \mathbb{R}^{h \times h}$, $W_x^{(b)} \in \mathbb{R}^{d \times h}$, $W_h^{(b)} \in \mathbb{R}^{h \times h}$ and biases $b^{(f)} \in \mathbb{R}^{n \times h}$, $b^{(b)} \in \mathbb{R}^{n \times h}$ are all model parameters.

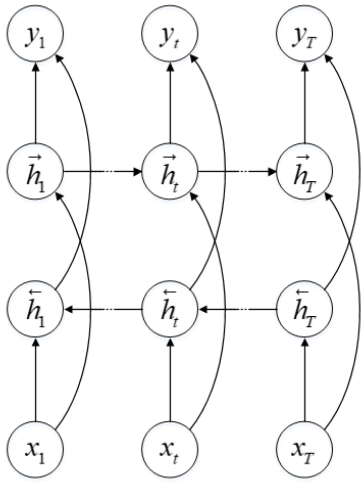


Figure 18: Architecture of a bidirectional RNN.

Next, we concatenate the forward and backward hidden states $\vec{h}_t$ and $\overleftarrow{h}_t$ to obtain the hidden state $h_t \in \mathbb{R}^{n \times 2h}$ to be fed into the output layer. In deep bidirectional RNN with multiple hidden layers, such information is passed on as input to the next bidirectional layer. Last, the output layer computes the output $y_t \in \mathbb{R}^{n \times q}$ (number of outputs q):

$$y_t = h_t W_y + b_y \tag{22}$$

Here, the weight matrix $W_y \in \mathbb{R}^{2h \times q}$ and the bias $b_y \in \mathbb{R}^{n \times q}$ are the model parameters of the output layer.

V. Data collection and training model.

1. Data collection.

To train an RNN with large capacity of memory by supervised learning, we need a large amount of input/output training data. Thus, we operated the robot and collect the necessary raw response data of CDPR from predefined trajectories. Those are three straight and circular orbits (in *XY* plane, *XZ* plane and *YZ* plane) as shown in Fig. 19. At the same time, eight cable tensions T_i (for i = 1, …, 8) and position error e are also collected. A total of 2603 points is collected as Fig. 20. Then, our goal is to obtain an approximation of model by learning from a set of (T, e).

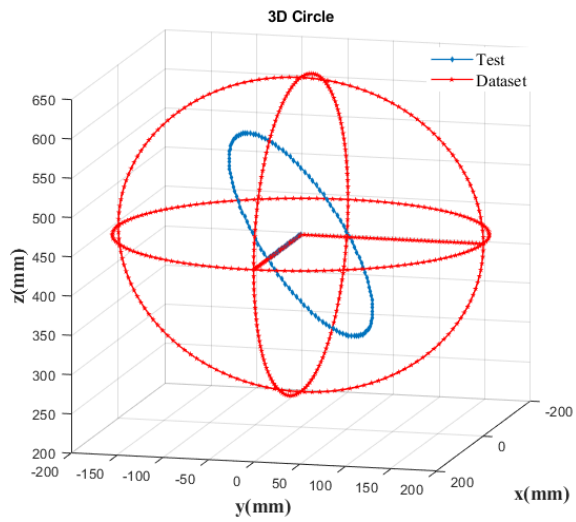


Figure 19: Overall trajectories for training and testing

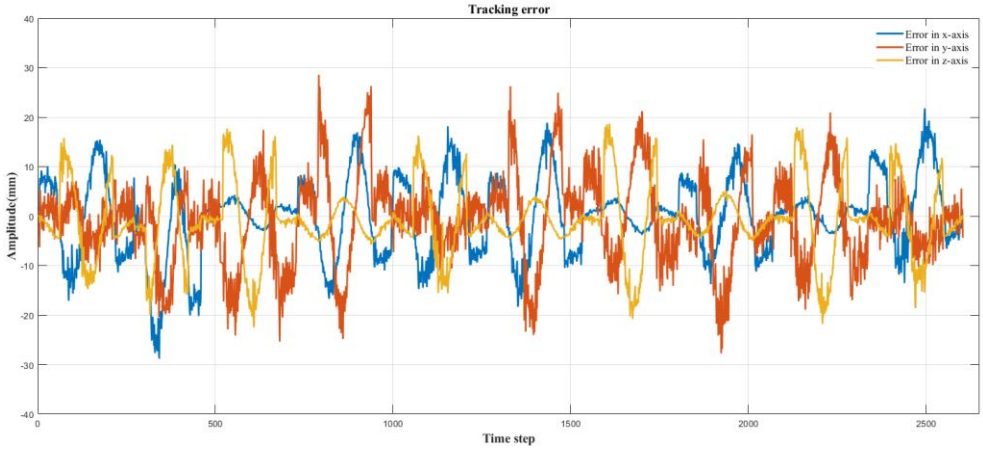


Figure 20: Total datasets

2. Training the network with RNN.

We have trained model in Python using Keras open-source package for deep learning with TensorFlow backend. During training, the input to RNN is a sequence of 8 cable tensions and position error containing T time step: $I(t) := \{T(\tau), e(\tau) : \tau \in [t, t+T-1]\}$ (11 by 1 vector at time τ). The output of the RNN is the position error e at time $t + T$. Fig. 21 shows the input values and expected output. Data is normalized between 0 and 1 using min-max normalization to ease computations and decrease multivariable values divergence. Then, we took 70% of the total samples datasets as training datasets and the remaining 30% as testing datasets. The same data is used to train many networks based on variations of number of epochs and neurons for every applied technique and time scale. We choose T = 15 to guarantee that

the input of RNN contain enough information. We train a 3-layers with LSTM cell and use *AdamOptimizer* to tune the parameter of RNN. After training, the RNN model can provide a mapping from a sequence of 15 time steps cable tension and position error (11 by 15 matrix) to immediately predicted position error in next time step:

$$I(t) := \{T(\tau), e(\tau) : \tau \in [t, t+14]\} \rightarrow e(t+15) \tag{23}$$

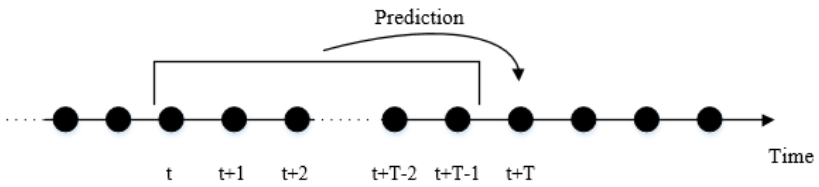


Figure 21: Illustration of prediction.

3. Training the network with Bi-RNN.

To build the Bi-RNN model, the input is also a sequence of 8 cable tension and position error containing T time step: $I(t) := \{T(\tau), e(\tau) : \tau \in [t, t+T-1]\}$ (11 by 1 vector at time τ). The output of the Bi-RNN is the position error e at time $t + T$. Similar to RNN training, data is normalized between 0 and 1 using min-max normalization. Then, we took 70% of the total samples datasets as training dataset and the remaining 30% as testing datasets. The same data is used to train many

networks based on variations of number of epochs and neurons for every applied technique and time scale. We choose $T = 15$ to guarantee that the input of Bi-RNN contain enough information. We train a 3-layers with LSTM cell and use *AdamOptimizer* to tune the parameter of Bi-RNN. After training, the Bi-RNN model can provide a mapping from a sequence of 15 time steps cable tension and position error (11 by 15 matrix) to immediately predicted position error in next time step:

$$I(t) := \{T(\tau), e(\tau) : \tau \in [t, t+14]\} \rightarrow e(t+15) \tag{24}$$

4. Prediction results and analysis.

To assess the performance of the model, root mean square error (RMSE) are calculated on the testing dataset, which are defined as:

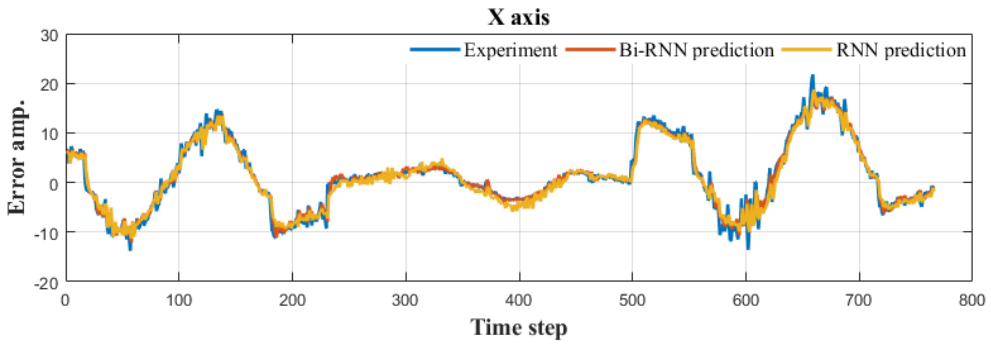
$$RMSE = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(\mathrm{y}_j - \hat{y}_j)^2} \tag{25}$$

where N is the number of testing samples, y_j and $\hat{y}_j$ are the true and predicted output values of the *j-th* testing sample, respectively. RMSE can evaluate the general prediction error by the modeling methods. With the parameter T, number of layers are chosen as mention above, we designed for each model by varying the number of units to be 4, 8, 12, 16 and 32 neurons. The networks are trained based on several run with optimized batch size and epoch. The results are summarized in table 2. We can see that both models

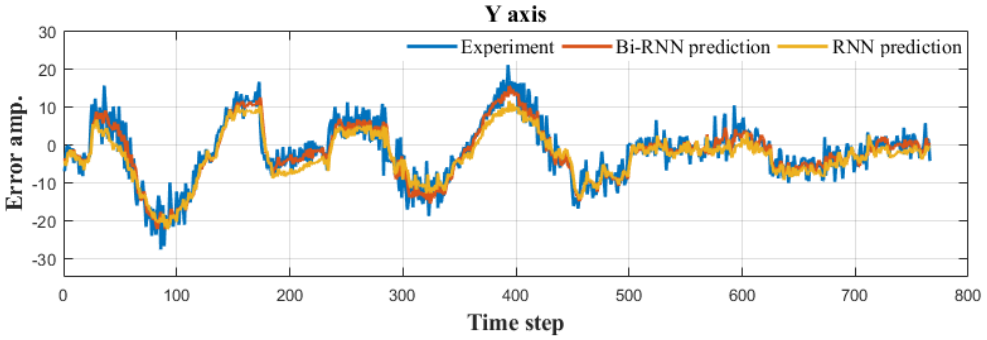
have good performance for position error prediction.

Table 2: Comparison RMSE of RNN and Bi-RNN model for testing sets with various number of units

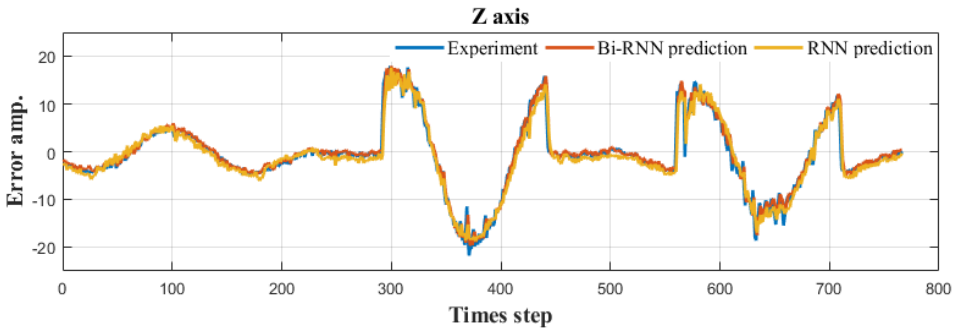
Networks	Bi-RNN			RNN		
	x	y	z	x	y	z
4 neurons network	1.5420	3.9033	1.5796	1.6217	3.9457	1.6778
8 neurons network	1.5222	3.8125	1.5528	1.6809	3.8752	1.6947
12 neurons network	1.4951	3.6633	1.5331	1.6291	3.8727	1.7400
16 neurons network	1.4980	3.6001	1.6572	1.6516	3.8683	1.7250
32 neurons network	1.5098	3.7562	1.5643	1.6671	3.9477	1.7181



(a): X axis.



(b): Y axis.



(c): Z axis.

Figure 22: Learning results in 3 directions. (a) X axis. (b) Y axis. (c) Z axis.

To be clear, we choose the best trained model with optimized parameter and compared them based on RMSE. Table 3 shows the RMSE value of each method with optimized parameter. Fig. 15 illustrates the performance of RNN model and Bi-RNN model with testing datasets. Blue line is real experiment testing set, red line and yellow line are prediction output of RNN model and Bi-RNN model, respectively. Overall, the results showed that both RNN model and Bi-RNN model produced good performance. However, Bi-RNN networks attained slightly higher performance than RNN

networks.

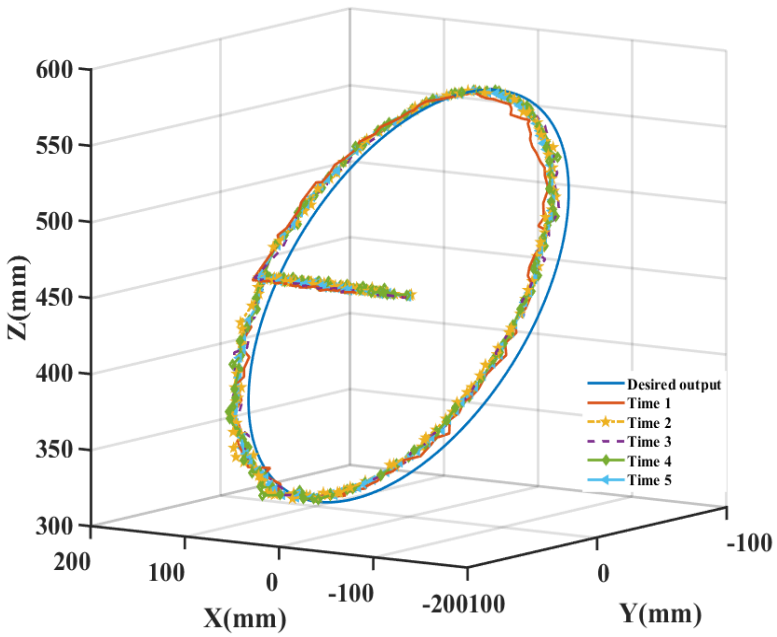
Table 3: RMSE of RNN model and Bi-RNN model with testing datasets.

RMSE (mm)	RNN model	Bi-RNN model	Improvement (%)
x	1.6	1.5	6.3
y	3.9	3.6	7.7
z	1.7	1.5	11.8

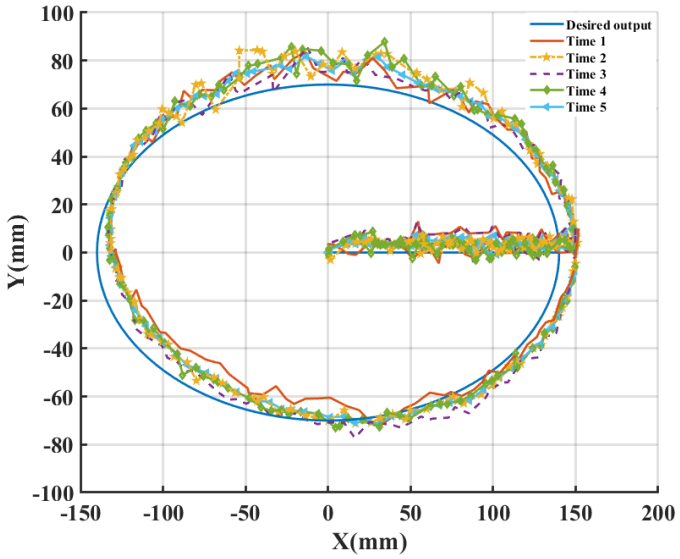
VI. Experimental comparison of algorithm.

To evaluate performance of proposed controller, we carried out tracking 3D circle trajectory with a radius of 140 mm and tilted 30 degrees around the Z-axis, as shown in Fig. 19 (blue line). During the motion, assume the orientation of end effector is fixed.

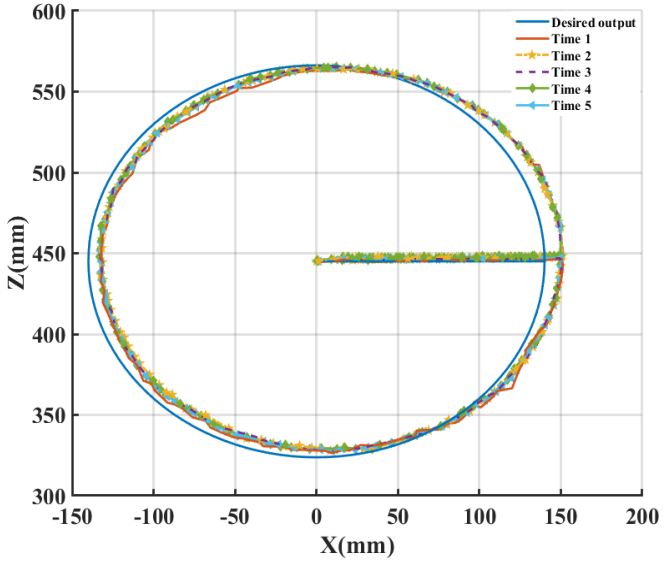
1. Results of simple inverse kinematic controller.



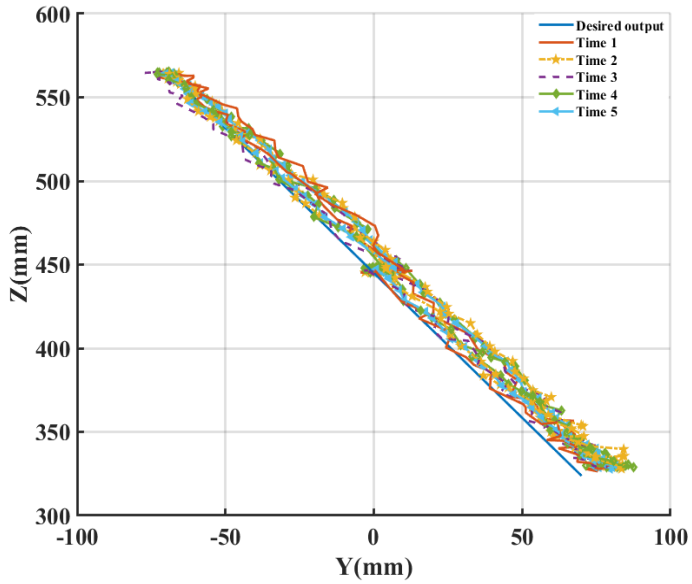
(a): 3D view



(b): *X*-*Y* view



(c): *X*-*Z* view



(c): Y-Z view

Figure 23: : Tracking 3D trajectory with 5 test times. (a) 3D view. (b) *X*-*Y* view. (c) *X*-*Z* view. (d) *Y*-*Z* view.

Fig. 23 shows the tracking results for the 3D circle trajectory in 5 test times. These plots show clearly the end-effector position error occurs due to many factors, which include non-linear characteristic. As Fig. 24, when end-effector are moving, cable tensions changed amplitude, which directly affects the length of cable as previous studies [17, 35, 36]. The RMSE of actual position and desired position along the whole trajectory are in Table 4.

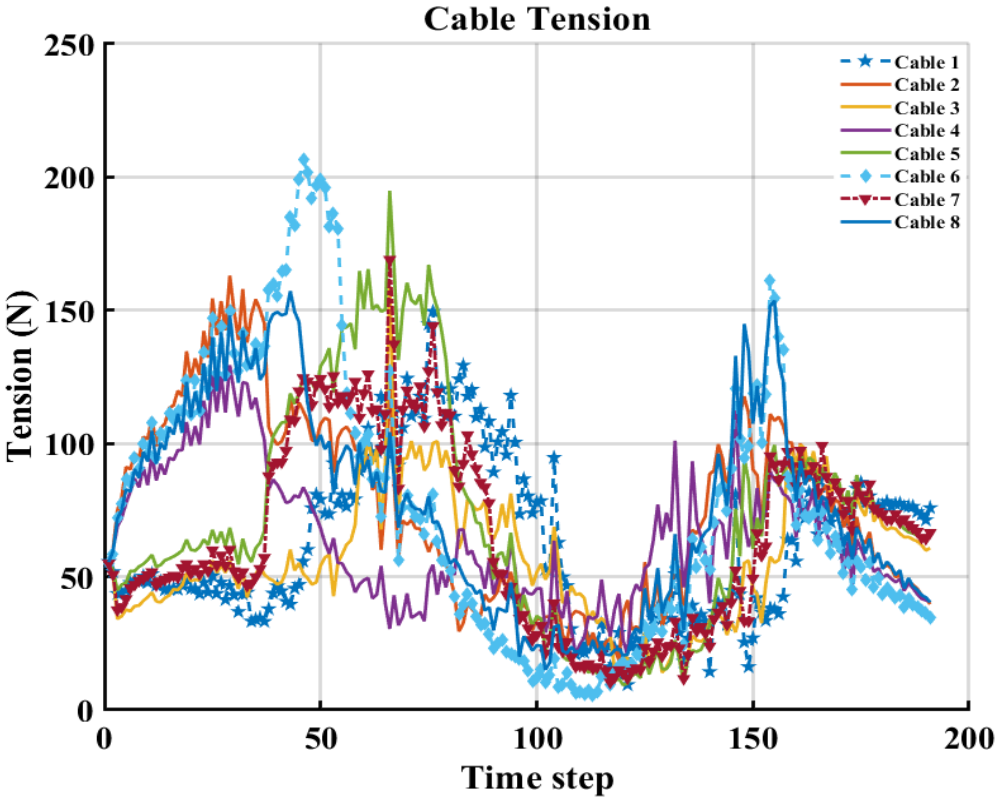
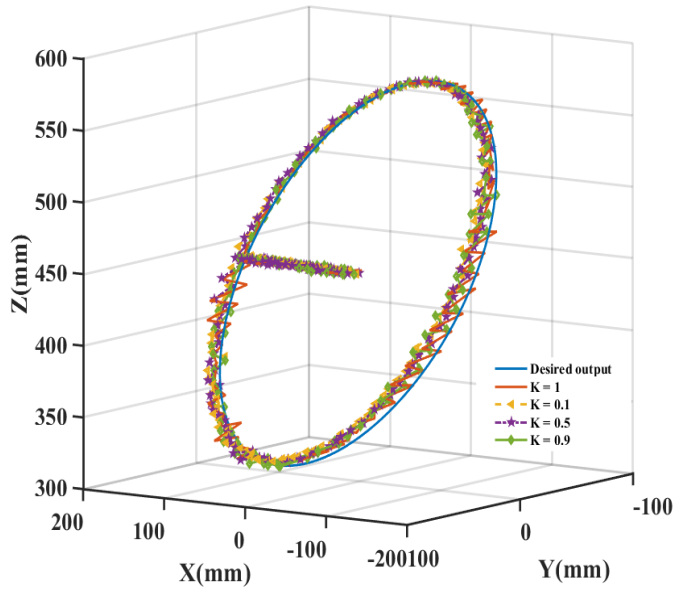


Figure 24: Tension of 8 cable during tracking end-effector.

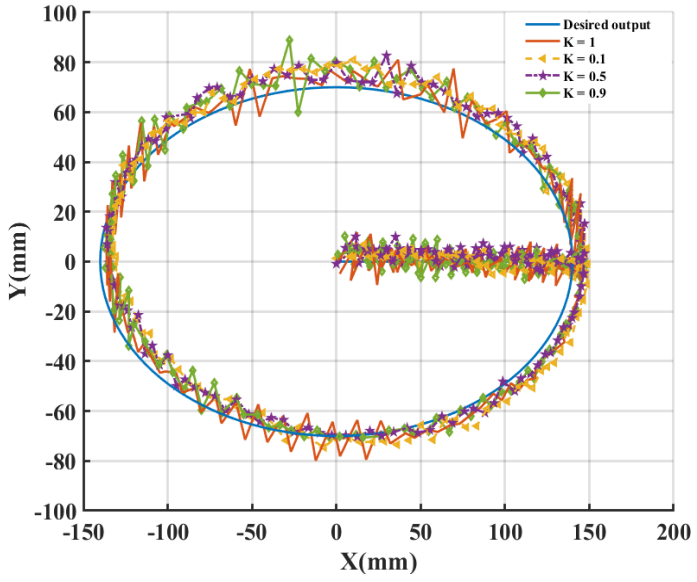
Table 4: RMSE with 5 test times.

RMSE	Time 1	Time 2	Time 3	Time 4	Time 5	Average
x	8.9	9.9	9.3	9.6	9.4	9.4 ±0.5
y	8.6	8.6	7.9	8.1	7.8	8.2±0.4
z	4.3	4.4	3.8	4.1	3.9	4.1±0.3

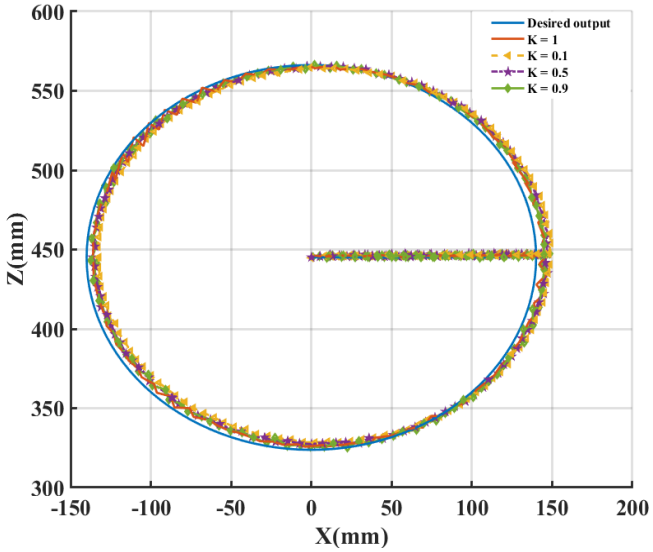
2. Results of baseline error feedback controller.



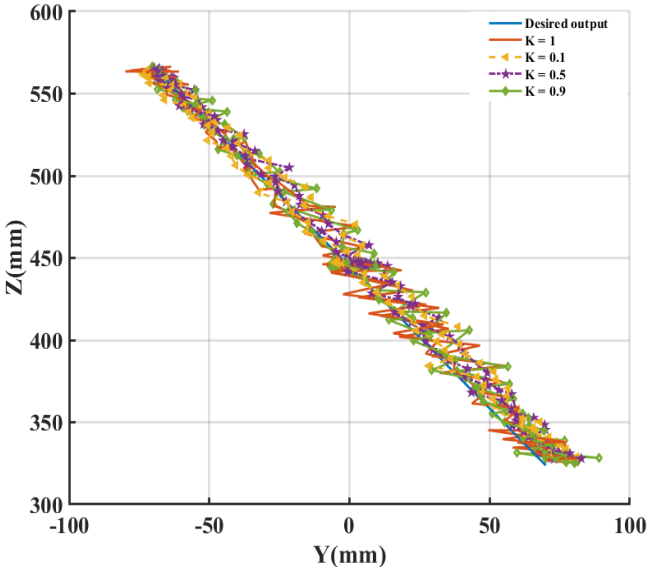
(a): 3D view



(b): *X*-*Y* view



(c): *X*-*Z* view



(d): *Y*-*Z* view

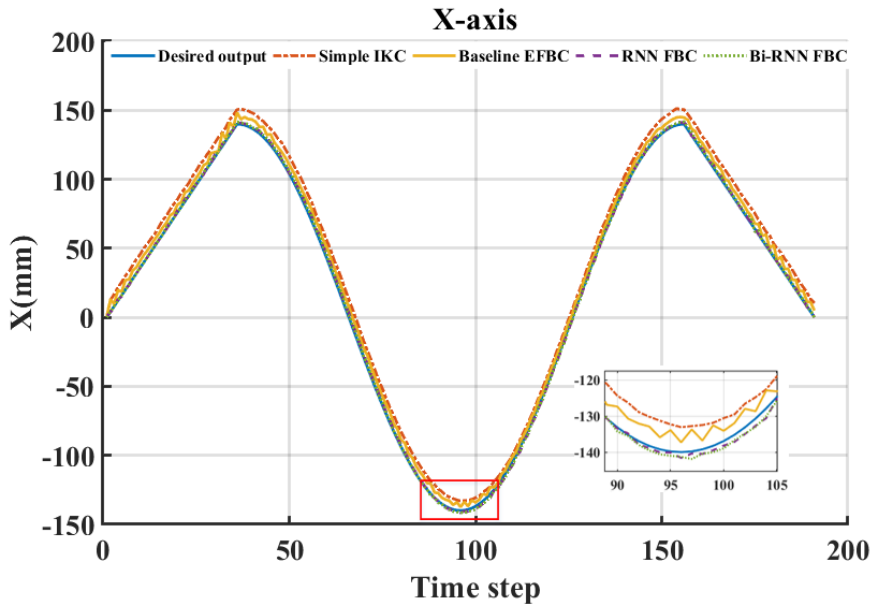
Figure 25: Tracking 3D trajectory with various *K* values. (a). 3D view. (b) *X*-*Y* view. (c) *X*-*Z* view. (d) *Y*-*Z* view.

Fig. 25 shows the tracking results for the 3D circle trajectory with various *K* values (according Eq. 8). And the RMSE of the end-effector position along the whole trajectory are in table 4. It is clear that with baseline error feedback controller, the performance of trajectory tracking control has been improved as compared with simple IK controller. However, these plots also show that because of feedback loop, with the larger *K* value, the controller produces more oscillations with bigger amplitude.

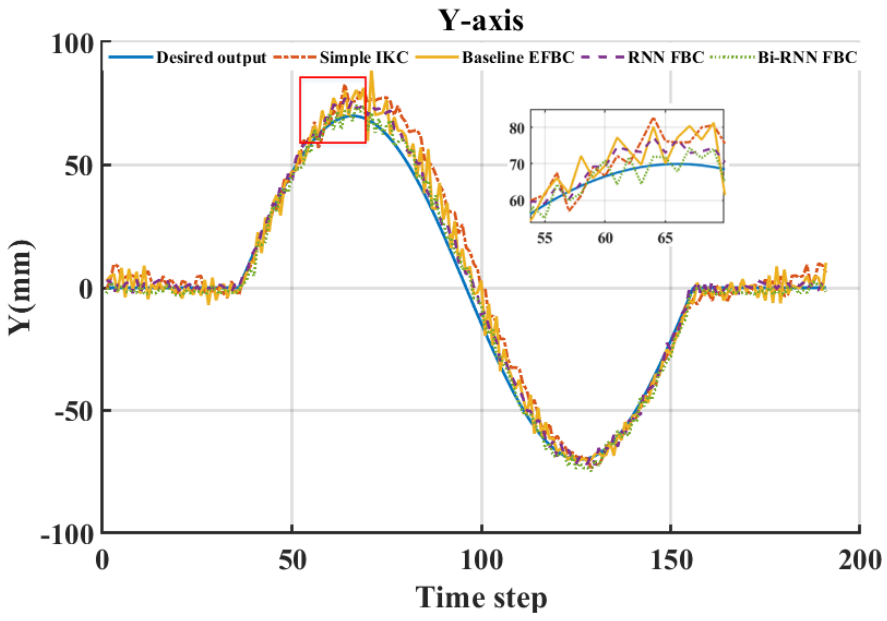
Table 5: RMSE with various *K* values

RMSE (mm)	$K = 1$	$K = 0.1$	$K = 0.5$	$K = 0.9$
x	4.7	7.9	6.2	5.2
y	6.6	7.2	6.9	6.3
z	2.3	2.6	2.4	2.1

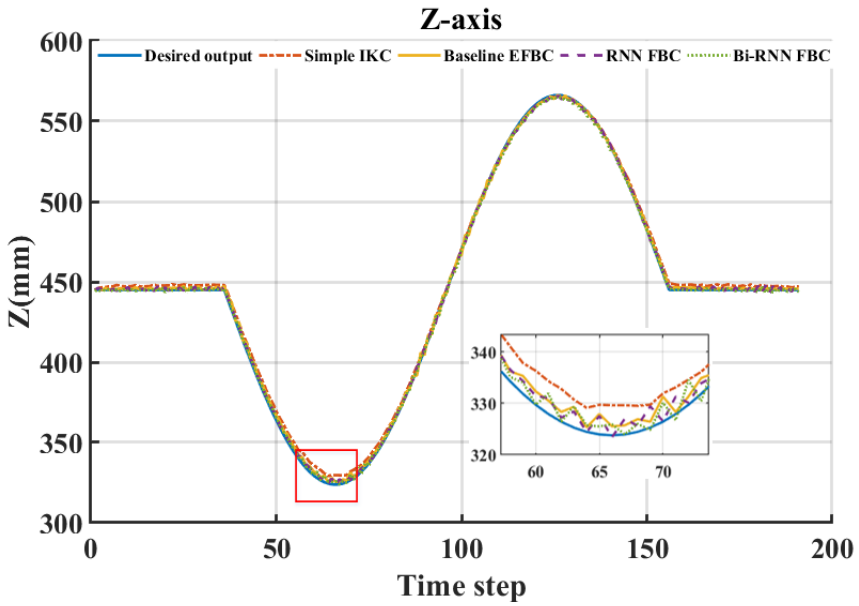
3. Results of proposed controller.



(a): *X* axis.



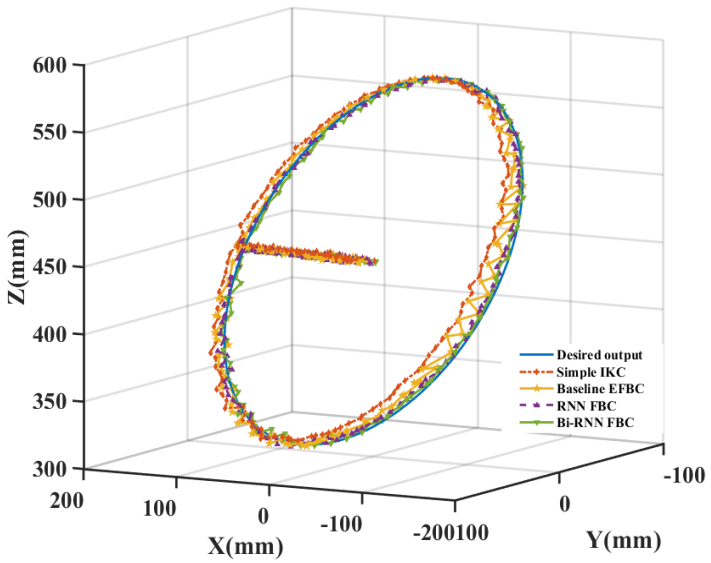
(b): *Y* axis.



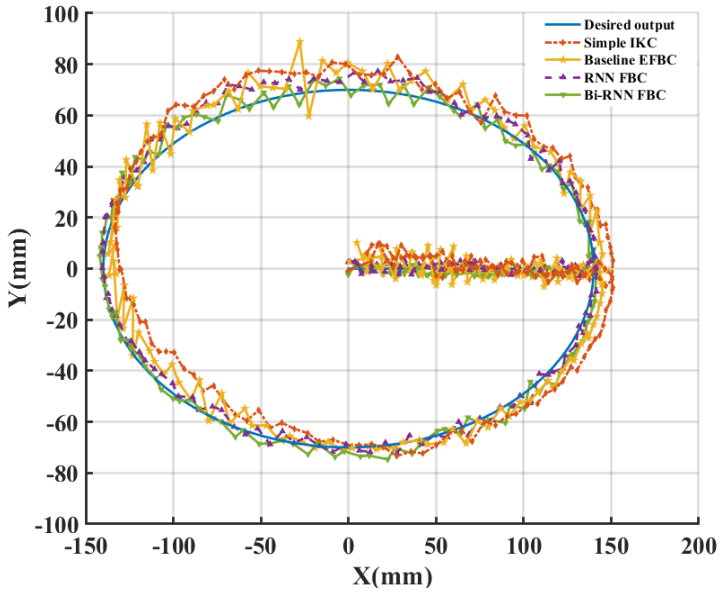
(c): *Z* axis.

Figure 26: Tracking 3D trajectory using different approaches in time domain.

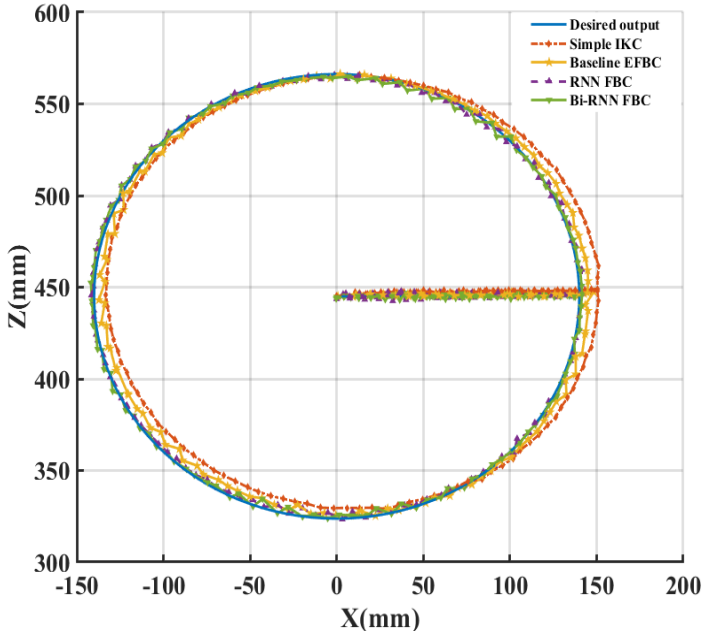
(a) *X* axis. (b) *Y* axis. (c) *Z* axis.



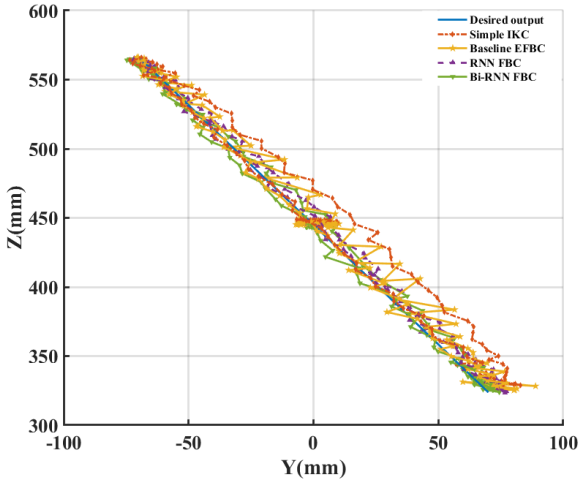
(a): 3D view



(b): *X*-*Y* view



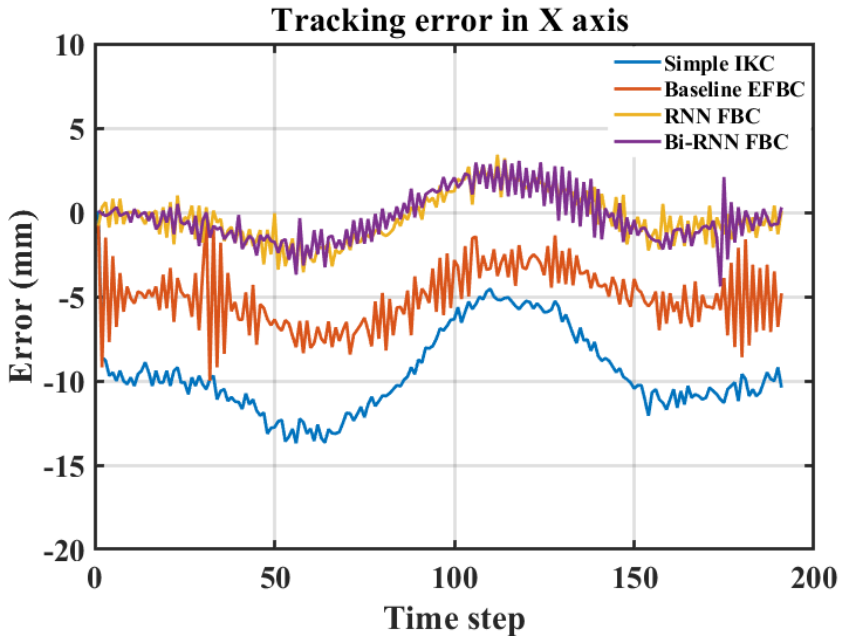
(c): *X*-*Z* view



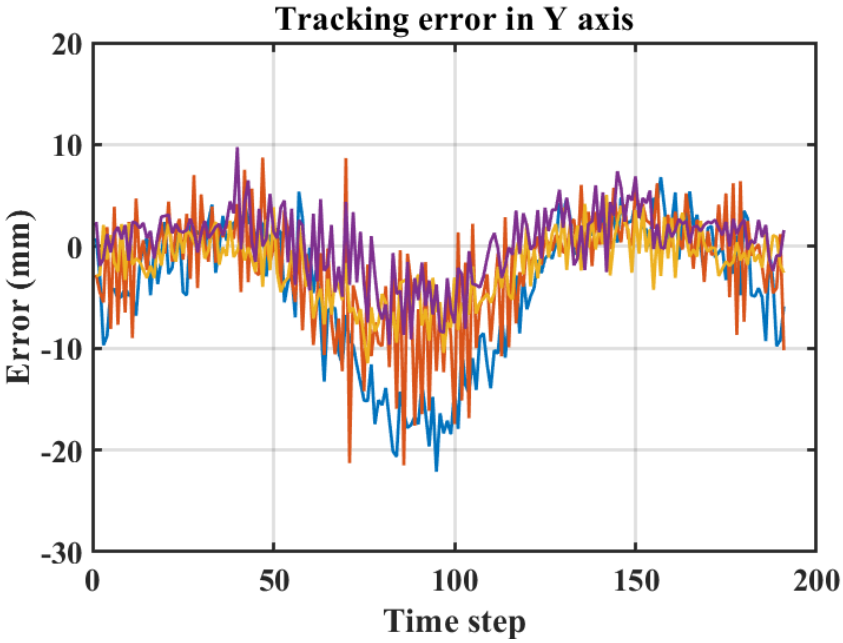
(d): *Y-Z* view

Figure 27: Tracking 3D trajectory using different approaches in the XYZ coordinates space. (a) 3D view. (b) *X-Y* view. (c) *X-Z* view. (d) *Y-Z* view.

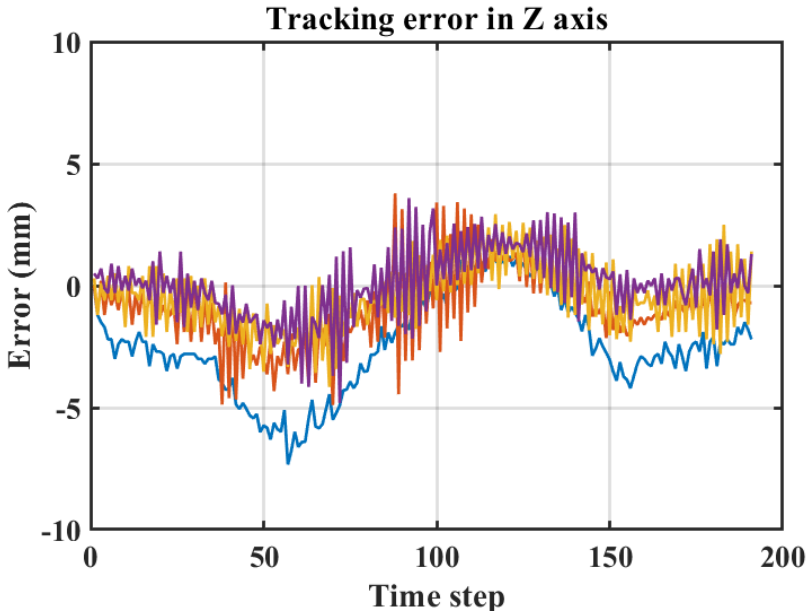
We applied control algorithms with optimal parameters to CDPR and then compared the best results of each method. Fig. 18-20 show the results of tracking control for 3D circle trajectory and the position tracking error in three axes using various approaches. The experimental results are summarized in table 5. From the figure and the table, we show that the tracking performance of proposed controller is significantly improved with proposed controller in two approaches, in average over 60% as compared with simple IK controller and over 22 % as compared with baseline error feedback controller. Also, using Bi-RNN to construct prediction model still works slightly better than using RNN to construct prediction model.



(a): *X* axis.



(b): *Y* axis.



(c): *Z* axis.

Figure 28: Position tracking error of end effector. (a) *X* axis. (b) *Y* axis. (c) *Z* axis.

Table 6: Comparison RMSE among various approaches.

RMSE (mm)	IKC	EFBC	RNN FB	Bi-RNN FB
x	9.4±0.5	5.2±0.6	1.5±0.2	1.5±0.2
y	8.2±0.5	6.3±0.4	4.1±0.3	3.9±0.3
z	4.1±0.3	2.1±0.2	1.6±0.2	1.5±0.2

VII. Conclusions and future works.

In conclusion, the suitable configuration of cable-driven parallel robot was introduced. Due to a lack of mechanical actuator accuracy, winch-motor system is designed with ball screw. Moreover, pulley kinematic is also used to increase accuracy of cable robot. Not only mechanical actuators but non-linear characteristic of cable (creep, recovery, hysteresis, hardening, …) also effect to position error of end-effector. It is difficult to make the numerical modeling of cable and apply it to the control algorithm because of the complexity of the non-linear characteristic. Thus, time series prediction is introduced with RNN, LSTM and Bi-RNN concept. We then proposed the approach for tracking position control of cable-driven parallel robot through compensation by the feedback controller with prediction model. Firstly, we ran the robot and gathered the appropriate signal response data of cable-driven parallel robot from predefined trajectories. Then, we built prediction model in two approaches based on that data to predict position error of end-effector. The first approach is trained data in recurrent neural network methodology. The second one is to construct predictive model by bidirectional recurrent neural network algorithm. We demonstrated the results of proposed controller for the cable-driven parallel robot with tracking 3D circle trajectory. By

comparing results with simple inverse kinematic controller and baseline error feedback controllers, the effectiveness of our proposed strategy is verified.

For future work, the quality of our proposed controller depend on the parameter K and the quality of the prediction model. Thus, development of an adaptive controller for online tuning K is essential and improve the performance of prediction model by advanced method and sophisticated feedback control laws instead of pure P-control. After long time operation of cable robot, cable properties will be changed, performance of position tracking control will be significantly decreased. Therefore, we need to reconstruct prediction model after a certain period time.

Reference.

[1] S. Qian, B. Zi, W. W. Shang, et al. “A review on cable-driven parallel robots”. *Chinese Journal of Mechanical Engineering*, 31(1): 66, 2018.

[2] G. Meunier, B. Boulet and M. Nahon, “Control of an overactuated cable-driven parallel mechanism for a radio telescope application”, *IEEE transactions on control systems technology*, 17(5): 1043-1054, 2009.

[3] R. D. Nan, “Five-hundred-meter aperture spherical radio telescope (FAST) project”, *Science in China*, 49(2): 129-148, 2006.

[4] S. Kawamura, H. Kino and C. Won, “High-speed manipulation by using parallel wire-driven robots”, *Robotica*, 18(3):13-21, 2000.

[5] K. Maeda, S. Tadokoro, T. Takamori, M. Hiller and R. Verhoeven, “On design of a redundant wire-driven parallel robot WARP manipulator”, *In: Proceedings of IEEE international conference on robotic and automation*, pp 895-900, 1999.

[6] S. Qian, K. Bao, B. Zi and N. Wang, “Kinematic calibration of a Cable-Driven Parallel Robot for 3D Printing”, *Sensor*, 18(9):2898, 2018.

[7] P. Miermeister, Potte and Verle, “Dynamic modeling and hardware in the loop simulation for the cable-driven parallel robot IPnema”, *Robotic(ISR),*

41st International symposium, 6th germa conference on robotics (ROBOTIK), pp.1-8, 2010.

[8] B. Hannaford, J. Rosen, D. W. Friedman, H. King, P. Roan, L. Cheng, D. Glozman, J. Ma, S. N. Kosari, L. White and Raven II, "An open platform for surgical robotics research", *IEEE Transaction on biomedical engineering*, 60:954-959, 2013.

[9] J. P. Merlet, Y. Papegay, A. V. Gasc, "The Prince's tears, a large cable driven parallel robot for an artistic exhibition", *2020 IEE International Conference on Robotics and Automation (ICRA)*, Paris, 2020.

[10] S. Qi, M. Yu, J. Fu, P. D. Li and M. Zhu, "Creep and recovery behaviors of magnetorheological elastomer based on polyurethane/epoxy resin IPNs matrix", *Smart material and structures*, 2015.

[11] R. Chattopadhyay, "Textile rope-A review", *Indian journal of fibre & textile research*, pp.360-368, 1997.

[12] S. Qi, M. Yu, J. Fu, P. D. Li and M. Zhu, "Creep and recovery behaviors of magnetorheological elastomer based on polyurethane/epoxy resin IPNs matrix", *Smart material and structures*, 2015.

[13] P. Huang, F. Zhang and Z. Meng, “Adaptive control for space debris removal with uncertain kinematics, dynamics and states”, *Acta Astronautica*, 128:416-430, 2016.

[14] P. Tempel, D. Lee, F. Trautwein and A. Pott, “Modeling of Elastic-Flexible Cables with Time-Varying Length for Cable-Driven Parallel Robots”, *International Conference on Cable-Driven Parallel Robots*, pp.295-306, 2019.

[15] M. Miyasaka, M. Haghighipanah, Y. Li, “Hysteresis model of longitudinally loaded cable for cable driven robots and identification of the parameters”, *In: Proceedings of IEEE international conference on robotic and automation (ICRA)*, Stockholm, pp 4051-4057, 2016.

[16] W. Kraus, V. Schmidt, P. Rajendra, “Load Identification and compensation for a cable-driven parallel robot”, *In: Proceedings of IEEE international conference on robotic and automation (ICRA)*, Karlsruhe, pp 2485-2490, 2013.

[17] S. H. Choi, K. S. Park, “Integrated and nonlinear dynamic model of a polymer cable for low-speed cable-driven parallel robots”, *Microsystem Technology*, 24(11):4677-4687, 2018.

[18] C. D. Li, J. Q. Yi, Y. Yu, D. B. Zhao, “Inverse Control of Cable-driven Parallel Mechanism Using Type-2 Fuzzy Neural Network”, *Acta Automatica Sinica*, 36(3):459-464, 2010.

[19] J. Piao, E. S. Kim, H. Choi, C. B. Moon, E. Choi, J. O. Park, C. S. Kim, “Indirect Force Control of a Cable-Driven Parallel Robot: Tension Estimation using Artificial Neural Network trained by Force Sensor Measurement”, *Sensors*, 19(11):2520, 2019.

[20] C. L. Giles, G. M. Kuhn, and R. J. Williams, “Dynamic recurrent neural networks: Theory and applications,” *IEEE Trans. Neural Networks*, vol. 5, pp. 153–156, Apr. 1994.

[21] B. A. Pearlmutter, “Learning state space trajectories in recurrent neural networks,” *Neural Comput.*, vol. 1, pp. 263–269, 1989.

[22] A. J. Robinson, “An application of recurrent neural nets to phone probability estimation,” *IEEE Trans. Neural Networks*, vol. 5, pp. 298–305, Apr. 1994.

[23] M. H. Hwang, K. Y. Goldberg, “Efficiently Calibrating Cable-Driven Surgical Robots with RGBD Fiducial Sensing and Recurrent Neural Networks”, *IEEE Robotics and Automation Letters*, 2020.

[24] J. M. Kang, S. H. Choi, J. W. Park and K. S. Park, “Position Error Prediction Using Hybrid Recurrent Neural Network Algorithm for Improvement of Pose Accuracy of Cable Driven Parallel Robots”, *Microsystem Technologies,* July 2019.

[25] S. Hochreiter, J. Schmihuber, “Long short-term memory”, *Neural Computation,* pp.9(8): 1735-1780, 1997.

[26] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks”, *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673-2681, 1997.

[27] K. A. Althelaya, E. S. M. El-Alfy and S. Mohammed, “Evaluation of Bidirectional LSTM for Short and Long Term Stock Market Prediction”, *International Conference on Information and Communication Systems (ICICS),* 2018.

[28] L. D. Persio and O. Honchar, “Analysis of recurrent neural networks for short term energy load forcasting”, *AIP Conference Proceeding*, vol. 1906, no.1, p. 190006, 2017.

[29] Z. Cui, R. Ke and Y. Wang, “Deep stacked bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction”, *6th International Workshop on Urban Computing*, 2017.

[30] T. N. Dinh, J. W. Park and K. S. Park, “Design and Evaluation of Disturbance Observer Algorithm for Cable-Driven Parallel Robots”, *Microsystem Technologies*, 2020.

[31] T. D. Hai, “Design of Reconfigurable Cable-Driven Parallel Robot with Base Mobility”, The degree of Master thesis, 2019.

[32] S. G. Jurado, R. M. Salinas, F. J. M. Cuevas and M. J. M. Jimenez, “Automatic generation and detection of highly reliable fiducial markers under occlusion”, *Pattern Recognition*, 47(6):2280-2292, 2014.

[33] A. Pott, “Influence of Pulley Kinematic on Cable-Driven Parallel Robots”, *Latest Advances On Robot Kinematic*, 2012.

[34] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities”, *Proc. Natl. Acad. Sci. U.S.A.*, 79(8):2554-2558, 1982.

[35] Y. C. Chen, L. W. Chen, W. H. Lu, “Power loss characteristics of sensing element based on a polymer optical fiber under cyclic tensile elongation”, Sensors, 11:8741-8750, 2011.

[36] W. N. Findley, F. A. Davis, “Creep and relaxation of nonlinear viscoelastic material”, Courier corporation, North Holland, 2013.

Acknowledgements.

Firstly, I would like to express my gratefulness to my advisor, Professor Kyoung-Su Park for his continuous support and conscientious instruction during my Master course. He provided me not only important knowledge but also the best conditions for my studying and research process. Honestly, this thesis could not be completed without my Professor. He also gave me opportunities to explore the lifestyles of Korea. I feel that I'm so lucky to be your student, Prof. Park. Thank you so much for everything you have done for me.

I would like to say thank you to all of our lab member Ryu Hyeon-gi, Dong-su Park, Sung-ho Park, Byung-gun Kim, Eun-hoo Lee, Jiajun Xu and all of my friends at Gachon University. They helped me a lot during the time I have been stayed here and also in my work. I hope that they will always be happy and successful in their life.

Finally, I would like to express my appreciation to my family for their love and spiritual encouragement during my Master's course.