Shuya Imajo · Masami Konishi · Tatsushi Nishi · Jun Imai

Application of a neural network to the generation of a robot arm trajectory

Received and accepted: October 25, 2004

Abstract We propose a neural network model generating a robot arm trajectory. The developed neural network model is based on a recurrent-type neural network (RNN) model calculating the proper arm trajectory based on data acquired by evaluation functions of human operations as the training data. A self-learning function has been added to the RNN model. The proposed method is applied to a 2-DOF robot arm, and laboratory experiments were executed to show the effectiveness of the proposed method. Through experiments, it is verified that the proposed model can reproduce the arm trajectory generated by a human. Further, the trajectory of a robot arm is successfully modified to avoid collisions with obstacles by a self-learning function.

Key words Neural network · Trajectory generator · Robot arm · Learning

1 Introduction

Robot systems have been applied in a wide variety of fields, such as nursing care and social welfare, in recent years. The problem in applying robot systems is how to install intelligence on these robots. For example, when the robots are used for load handling, it is necessary to train the robot arms to follow the appropriate trajectory. Several trajectory generation methods for robot arms have been proposed for a robot hand with two links.1,2

Traditionally, trajectory of a robot arm was generated by the teaching-play back method. However, it is not clear how experts can train a robot with the flexible movements of a human. Generally, humans can learn from their environments how to avoid collisions with obstacles. In order to develop this intelligence in robots, it is necessary to estimate the trajectories of movements. A learning function should be developed to automatically generate the appropriate robot trajectory through training. In this article, we propose a recurrent neural network (RNN) model for the automatic generation of robot trajectories. A recurrent neural network was successfully applied to gain tuning for looper control in hot strip rolling.3

The main feature of the proposed RNN model is that the inputs to the RNN are values calculated with estimated functions of robot movements and past locus values of the joint angles of the robot arm. The outputs from the RNN are the desired values for the joint angles of a robot arm in the next instance. In addition to this learning function, the proposed RNN model has an added self-learning function. The weight of the RNN model is updated using the desired value of the evaluation functions for robot movements determined by an analysis of human operations.

Numerical experiments were carried out to confirm the effectiveness of the model using the robot simulator. Finally, laboratory experiments for a 2-DOF robot arm are carried out to check the practicability of the RNN model.

2 Trajectory generators by a recurrent neural network

2.1 Generation of training data

It is necessary to generate teaching data to build the RNN model. A human grasped the robot arm and moved it to the target position from an initial position on several occasions. The time-series data sets of joint angles were gathered. Using the data, an RNN model was identified to cover the wide variety of human operations. As the result, two RNN models were constructed by the data sets corresponding to two joints.

S. Imajo · M. Konishi (✉) · T. Nishi · J. Imai
Department of Electrical and Electronic Engineering, Okayama University, 3-1-1 Tsushima-naka, Okayama 700-8530, Japan
Tel. +81-86-251-8123; Fax +81-86-251-8111
E-mail: konishi@cntr.elec.okayama-u.ac.jp

This work was presented, in part, at the 9th International Symposium on Artificial Life and Robotics, Oita, Japan, January 28–30, 2004
2.2 Recurrent neural network model

The structure of the proposed RNN model is shown in Fig. 1. This network identifies two robot links. The networks have three layers, as shown in Fig. 1. The first layer is the input layer, consisting of two parts. One is the plan layer, comprising four evaluation functions $I_1, I_2, I_3,$ and $I_4$ for robot arm movements. The differential between these evaluation values, $I_i$, and the desired evaluation value, $I_0$, are input to the RNN model. These evaluation functions are given as the following normalized functions.

First, the evaluation function of joint angle variation is expressed by

$$I_1 = \frac{1}{t_f} \int_{0}^{t_f} \left| \frac{\dot{\theta}_i}{\theta_i - \theta_i(0)} \right| \, dt \quad (i = 1 \text{ or } 2)$$

(1)

where $t_i$ is working time, and $\dot{\theta}_i$ is given by

$$\dot{\theta}_i = \frac{\theta_i - \theta_i(t)}{\theta_i - \theta_i(0)} \quad (i = 1 \text{ or } 2)$$

(2)

Secondly, the evaluation function of the angular velocity is written as

$$I_2 = \int_{0}^{t_f} \left| \frac{\dot{x}}{\dot{y}} \right| \, dt$$

(3)

Thirdly, the evaluation function of the normalized distance between the target position and the end position of the robot arm is represented as

$$I_3 = \frac{1}{t_f} \int_{0}^{t_f} \left( \frac{(x_f - x_i(0))^2 + (y_f - y_i(0))^2}{(x_f - x_i(t))^2 + (y_f - y_i(t))^2} \right) \, dt$$

(4)

where $(x_i, y_i)$, $(x_f, y_f)$ are the coordinates of the initial position and the target position, respectively.

Finally, the evaluation function of the deviation of the trajectory of the end effector from a straight trajectory between the start position and the goal position is expressed as

$$I_4 = \frac{1}{t_f} \int_{0}^{t_f} \frac{ax(t) + by(t) + c}{\sqrt{a^2 + b^2}} \, dt$$

(5)

where $a = x_f - x_o$, $b = y_f - y_o$, $c = x_o y_f - x_f y_o$.

The other part of the input layer is the context layer of the previous trajectory. The weighted sum of past output data of the RNN model expressed by Eq. 6 is also inputted to the RNN model.

$$c_k = \sum_{m=1}^{n} d^{m-k} \theta_{NN}[t+(k-1)\Delta t]$$

(6)

where $k$ is number of neurons in the output layer, and $d$ is a positive constant which represents the forgetting factor.

A linear function is used as the input layer of the neurons and the sigmoid function is used in the neurons of the hidden layer and the output layer.

When the model is used as a trajectory generator, the joint angle data output from the RNN every $\Delta t$ times is interpolated by a spline interpolation function.

3 Numerical experiments

In order to check the effectiveness of the proposed RNN model, a control system for the generated trajectory is developed, as shown in Fig. 2. A mathematical model for robot movement is developed based on Lagrangian equations for robot dynamics. In Fig. 2, a trajectory generator is generated by the RNN, and the inverse dynamics proposed by Isobe et al. and the manipulator dynamics are determined by Lagrangian equations.

Simulation results for a single target point when the same training data are used are shown in Fig. 3. Here, $a$, $b$, and $c$ in Fig. 3 represent the joint angle, angular velocity, and angular acceleration at 5 times, respectively. The output of the RNN and manipulator movements are compared with training data created by a human. Figure 3d indicates the trajectory of the end effector from the 1st to the 5th. We compared the data between robot movements and training data. Figure 3e shows the 5th motion of the robot arm in Fig. 2. Block diagram for trajectory control