Trajectory Generation Using RNN with Context Information for Mobile Robots

You-Min Lee and Jong-Hwan Kim

Abstract Intelligent behaviors generally mean actions showing their objectives and proper sequences. For robot, to complete a given task properly, an intelligent computational model is necessary. Recurrent Neural Network (RNN) is one of the plausible computational models because the RNN can learn from previous experiences and memorize those experiences represented by inner state within the RNN. There are other computational models like hidden Markov model (HMM) and Support Vector Machine, but they are absent of continuity and inner state. In this paper, we tested several intelligent capabilities of the RNN, especially for memorization and generalization even under kidnapped situations, by simulating mobile robot in the experiments.

1 Introduction

In the robotics and artificial intelligent society, how to implement powerful computational model like human brain has been a big issue. The artificial neural network (ANN) has been studied at least 50 years for the purpose of making intelligent system comparable to brain [1–3]. Recently, among those ANNs related models, deep learning has shown amazing performance in the visual and voice recognition problems [4, 5]. Those recent successes of the ANN seem to give promising futures for realization of a real intelligent system. For the behavioral level, however, a conventional feed forward neural network (FFNN) model is not suitable for generation of intelligent actions because the FFNN has no context node memorizing previous action sequences. The RNN is a neural network model with feedback connections. Because of the feedback connections, the RNN can memorize actions

Y.-M. Lee (✉) · J.-H. Kim
School of Electrical Engineering, KAIST, 291 Daehak-ro, Yuseong-gu,
Daejeon 305-701, Republic of Korea
e-mail: ymlee@rit.kaist.ac.kr

J.-H. Kim
e-mail: johkim@rit.kaist.ac.kr

it experienced, which can lead to generation of intelligent actions. Moreover, the RNN has its own advantageous in generalization capability compared to other behavioral computational models.

In this paper, computational ability of the RNN is investigated. The purpose of the task is to reach the several goal targets and way points with a proper sequence and trajectory and the robot used in this paper is a differential wheeled mobile robot. In the task, there are two goal targets, one is a red color point and the other is a green color point. In the training phase, the mobile robot moves to the red point first and after closer to the red point to some allowable error, turns and moves to the green point. To succeed this task, context information should be memorized by the RNN. The positions of robot starting and target points were generated randomly from 0 to 1 on the normalized x-y plane. In the test phase, using obtained training data, the relation between current goal positions and desired robot movement is learned. All the process related to experiments is computer simulation programed by the MATLAB. In the following sections, RNN simulation results and some analysis are presented.

2 Artificial Neural Networks

2.1 The Recurrent Neural Network

The RNN is an artificial intelligence computational model. In contrast to the FFNN, the RNN has feedback connections by which temporal memory can be stored. In the robotics, this RNN model has been used for generating variety of robot trajectories such as mobile robot and robot manipulators [6, 7] (Fig. 1).

In general, the RNN can be categorized into two models, one is a continuous time RNN (CTRNN) and the other is a discrete time RNN (DTRNN). In this paper, the DTRNN is used for our robot simulation experiments.

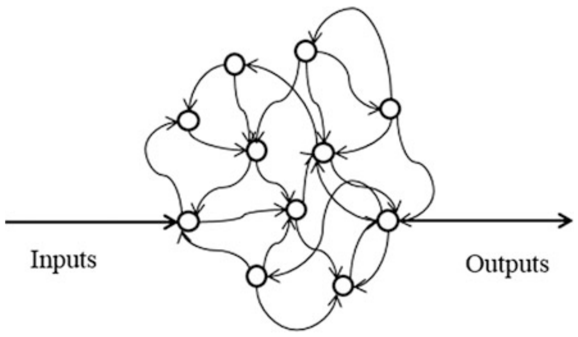

Fig. 1 Recurrent neural network

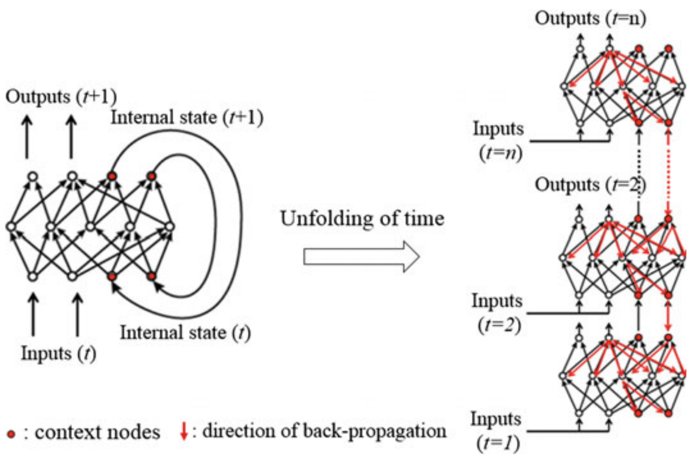

Fig. 2 Cascaded recurrent neural network and back propagation through time

2.2 *Training Algorithm*

There are various algorithms for training the RNN such as BPTT (Back Propagation-Through Time), EKF (Extended Kalman Filtering) algorithm and optimization training [8, 9]. In contrast to the FFNN, its training algorithm in most case is conventional BP (Back Propagation) algorithm; there is no clear winner in training the RNN. In this paper, the BPTT algorithm is used for training the RNN. The BP algorithm was invented for the purpose of training the conventional FFNN and the learning rule behind the BP algorithm utilizes a gradient descent method. The BP algorithm cannot be applied directly to the RNN because the RNN has feedback loops. To apply the BP algorithm to the RNN, it is needed to cascade the RNN through its training time. Figure 2 shows cascaded RNN architecture.

Since all connections inside the RNN is cascaded through time, there exist no closing loops and the conventional BP algorithm can be applied.

2.3 *Mobile Robot Kinematics*

In this section, mobile robot kinematics for moving to a target point is described. The mobile robot used in this paper is differential wheeled robot. For given velocity ν and angular velocity ω, next position and heading angle of the mobile robot can be calculated by equation below:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (1)$$

where, θ is the heading angle value of the mobile robot with respect to the x axis.

2.4 Feed Forward Neural Network for Mobile Robot

Before conducting main experiments with multiple way points by RNN, the FFNN is tested for a mobile robot to get to a single goal point. The neural network has two input nodes and two output nodes. Two input nodes consist of the current angle value between the robot and a goal and the distance value from the robot to a goal position. Two output nodes consist of the velocity and angular velocity values for the next movement. Also the network has two hidden layer each with 40 nodes. The training data was collected through simulation in which the mobile robot reached the goal position according to the following equation:

$$v = k_1 d \quad \omega = k_2 \theta \quad (2)$$

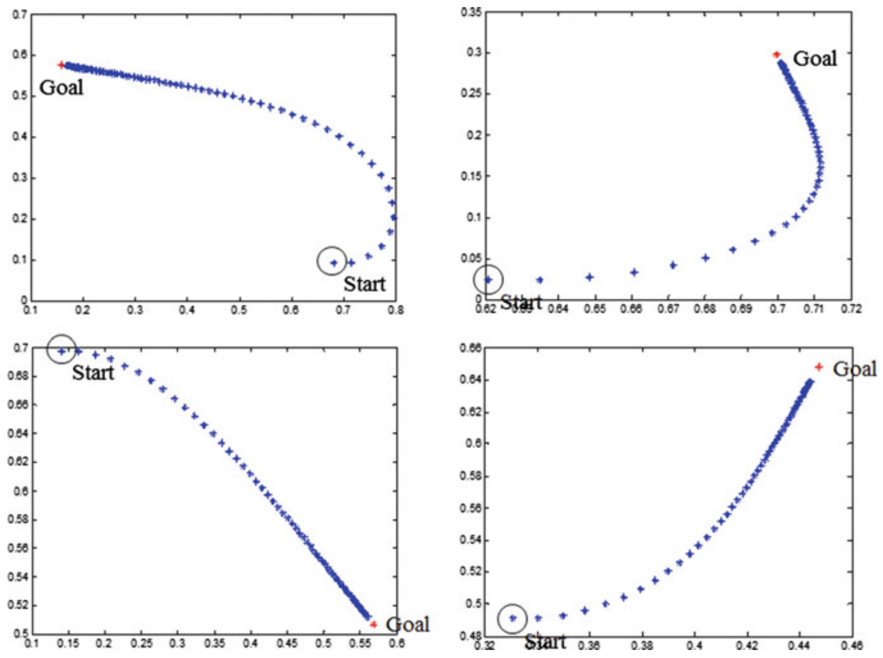

Fig. 3 Mobile robot trajectories when there is only one goal positions

where constant k_i were 0.05 and 0.2, respectively. Robot starting and goal positions were generated randomly on the x-y plane from 0 to 1. The simulation results are plotted in Fig. 3.

As shown in Fig. 3, the mobile robot could reach the goal position with proper sequences. Definitely, however, this simple task does not require the robot to be aware of the context information. The FFNN's output values solely depend on the current input values. Therefore, when there are more than two goals, the FFNN gets confused whether or not it has already reached the goal with higher priority.

3 Implementation

3.1 Architecture

To have the mobile robot to reach several target positions in a proper sequence, the RNN is required. In this paper, the RNN has four input nodes and two output nodes. Four input nodes consist of the current heading angle values respectively between the mobile robot and the goals 1 and 2 and the distance values respectively from the mobile robot to the goals 1 and 2. Two output nodes consist of the velocity and angular velocity values for the next movement. The network has one hidden layer with 50 nodes. Also the network has 10 context node with feedback connections. Activation function used in the network node is a sigmoid function by which only one directional rotation is permitted to the mobile robot because the sigmoid function can generate a positive value only. In the training phase, initially all weight values exist of the RNN were set to random based on the Gaussian distribution. And the learning rate for training the network was fixed to 0.1 during the entire learning procedure (Figs. 4 and 5).

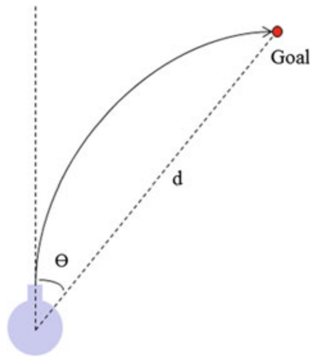

Fig. 4 Angle and distance between the robot and goal position

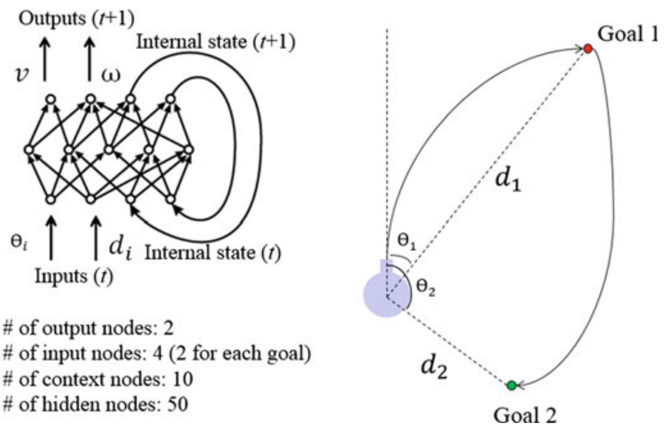

Fig. 5 The recurrent neural network architecture for the mobile robot

4 Experiments

After training the RNN, in the test phase, starting and target point positions were generated with the same method used in collecting the training data. Total training time was about 5 h by the MATLAB programming simulator. The test results are plotted in Fig. 6.

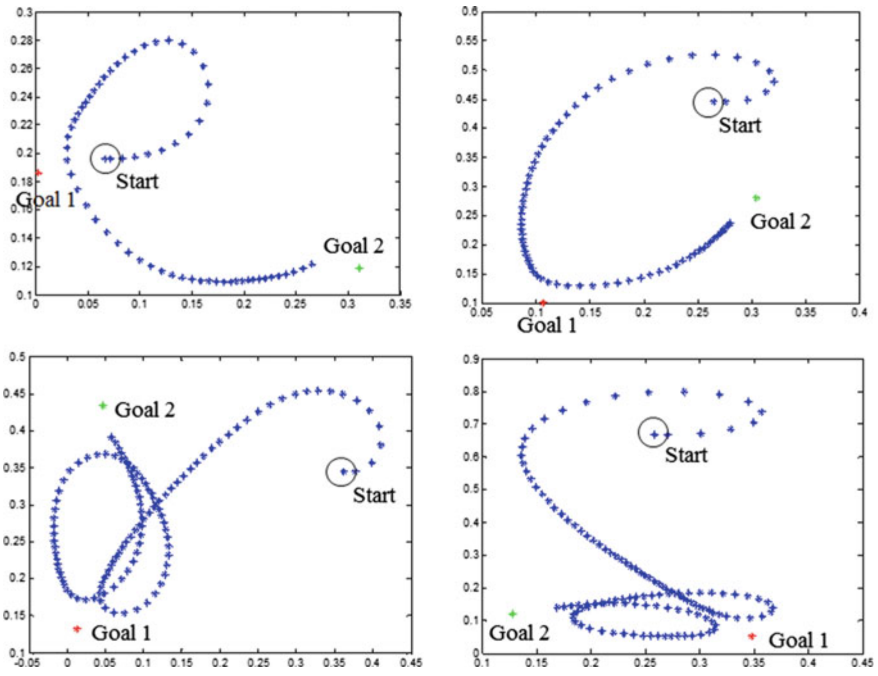

Fig. 6 Mobile robot trajectory when there are two goal targets

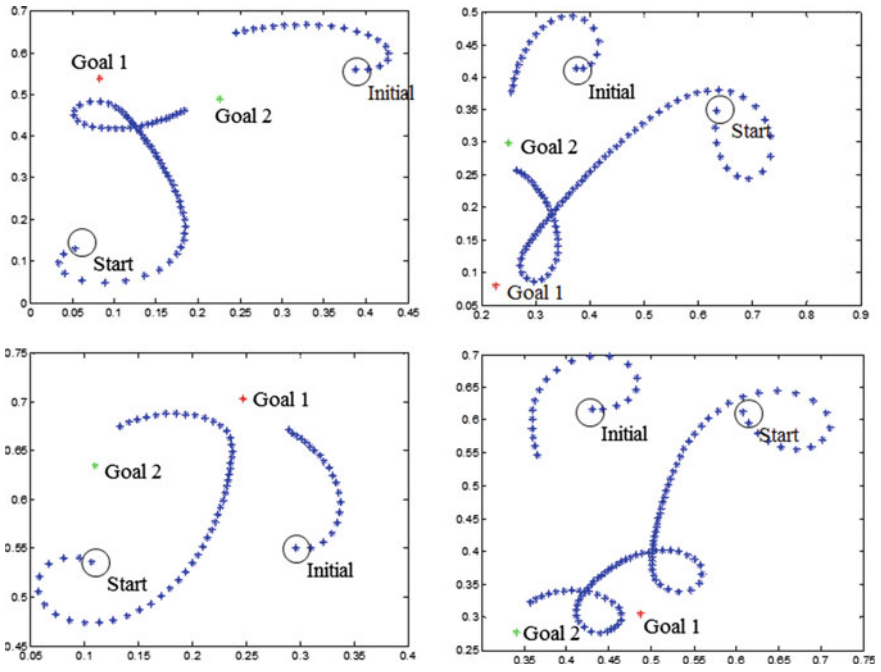

Fig. 7 Mobile robot trajectory when external disturbance exist

In the experimental results, the mobile robot always moved to goal points with proper sequences. In the figure, 'Start' is the starting position of the mobile robot and 'Goal 1' and 'Goal 2' are positions of goals 1 and 2, respectively. In Fig. 6, some trajectories showing strange morphology like circling and that is because in this simulation, mobile robot was permitted to rotate only one direction. Since the robot and goal positions are generated randomly in the training and test phases, we can conclude that the RNN properly generates not only given training trajectories but also unlearned trajectories, therefore guarantees generalization capability of the RNN. To prove robustness of the RNN, in an additional experiment, external disturbance was given to kidnap the robot to other places. After 20 steps moving from the initial stating position, the mobile robot's position was moved abruptly to randomly selected position on the x-y plane. The results is plotted in Fig. 7.

In the figure, 'Initial' is the initial position of the mobile robot and 'Start' is the starting position of the mobile robot after kidnapping. As shown in the figure, even though the mobile robot was kidnapped, it could generate a proper trajectory. Finally, the change of the context node activation value is plotted in Fig. 8 according to the mobile robot's trajectory. We found that only one context node out of 10 context nodes was changed through time.

In this figure, we can find some regularity. If the robot gets closer to the goal 1, the context node value gets increased. And after turning to goal 2, the context node

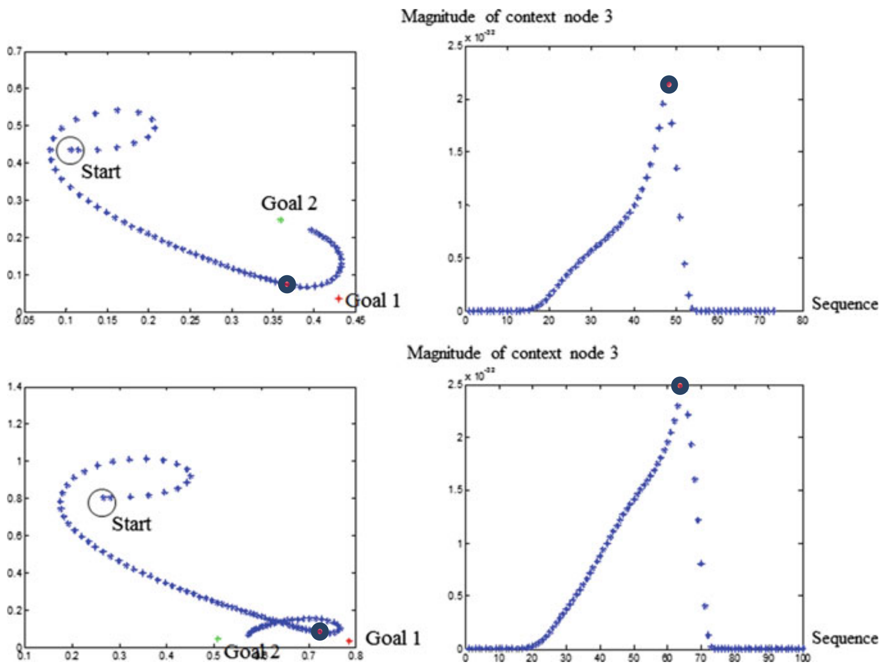

Fig. 8 Change of context node 3

value gets decreased. The turning point is indicated on Fig. 8 using big circle containing red point on the center of it. It is not enough to conclude from Fig. 8 that the context layer has a memory in which a clear switching point exists. However, it looks like that the mobile robot has strong tendency to reach the red point first.

5 Conclusion

In this paper, we tested various capability of the RNN through mobile robot simulation. From the fact that the mobile robot could generate proper sequences even with the external disturbance, we confirmed that the RNN was robust and reliable behavioral computational model. Such kind of characteristics of the RNN may be extended to more complex task such as object manipulation by using humanoid robot arms and walking pattern generator for humanoid robots.

Acknowledgment This work was supported by a grant from the National Research Foundation of Korea (NRF) funded by the Korea government (MSIP) (No. NRF-2014R1A2A1A10051551).

References

1. Hopfield, J.: Neural networks and physical systems with emergent collective computational abilities. In: Proceedings of the National Academy of Sciences, vol. 79, USA (1982)
2. Kohonen, T.: The self-organizing map. Proc. IEEE **78**(9) (1990)
3. Rojas, R.: Unsupervised learning and clustering algorithms. In: Neural Networks. Springer, Berlin, pp. 99–121 (1996)
4. Bengio, Y.: Learning deep architectures for AI. Found. Trends® Mach. Learn. **2**(1) 1–127 (2009)
5. Hinton, G.E.: Deep belief networks. Scholarpedia **4**(5) 5947 (2009)
6. Yokoya, R., et al.: Experience-based imitation using rnnpb. Adv. Robot. **21**(12) 1351–1367 (2007)
7. Sugita, Y., Tani, J.: Learning semantic combinatoriality from the interaction between linguistic and behavioral processes. Adapt. Behav. **13**(1) 33–52 (2005)
8. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Cogn. Model. (1988)
9. Hinton, G.: A practical guide to training restricted Boltzmann machines. Momentum **9**(1) 926 (2010)