2003 International Symposium on Advanced Intelligent Systems
September 25-28, 2003, Jeju, Korea

A Fuzzy-Neural Network Based Human-Machine Interface for Voice Controlled Robots Trained by a Particle Swarm Optimization

Keigo Watanabe*, Amitava Chatterjee†, Koliya Pulasinghe*, Kiyotaka Izumi* and Kazuo Kiguchi*
*Department of Advanced Systems Control Engineering, Graduate School of Science and Engineering,
Saga Univeristy, 1 Honjomachi, Saga 840-8502, Japan
E-mail: {watanabe, izumi, kiguchi}@saga-u.ac.jp
†Electrical Engineering Department, Jadavpur University, Kolkata - 700 032, India
E-mail: cha_ami@yahoo.co.in

Abstract— **Particle swarm optimization (PSO) is employed to train fuzzy-neural networks (FNN), which can be employed as an important building block in real life robot systems, controlled by voice-based commands. The FNN is also trained to capture the user spoken directive in the context of the present performance of the robot system. The system has been successfully employed in a real life situation for navigation of a mobile robot.**

I. Introduction

In the last ten years or so, many research efforts have been directed towards development and exploration of particle swarm optimization (PSO) technique which is based on the social metaphor of bird flocking or fish schooling [1]–[3]. One of the main advantages of PSO, which has endeared itself to the research community is that it is comparatively simple in operation and easier to understand compared to other evolutionary computations presently available, e.g., genetic algorithm (GA), evolutionary programming, genetic programming, etc.

The early works on PSO have shown the employment of the algorithm for a number of benchmark problems with a variety of dimensions. Most of these experimentations indicated that PSO can be very useful in certain problem domains to arrive at a fast solution [4]. To overcome getting stuck in local minima, different improved variations of PSO have been also reported which additionally employ static and varying inertia weights and constriction factor [2]. PSO has also recently evolved as a viable alternative to tune fuzzy and neuro-fuzzy systems [5]. However one of the main question that still remains unanswered is that can PSO be effectively applied in developing a real world system? This question inspired us to undertake the present work.

This paper describes the effective utilization of PSO to train a Takagi-Sugeno (TS)-type fuzzy-neural network (FNN) as an important building block of voice-controlled robot systems. Development of socially responsible, mature robots guided by spoken language commands [7] can be very useful for nursing and aiding the elderly people, for physically handicapped people, for people struck with paralyses and even as companion for children. The effectiveness of the proposed PSO-trained FNN for voice-controlled robot systems is aptly demonstrated by applying it for navigation of a wheeled mobile robot.

II. The Particle Swarm Optimization

PSO always initializes a pool of particles with random positions and velocities in a multidimensional space. In each iteration k, PSO calculates the fitness function f for each potential solution, by utilizing the current positional coordinates of the jth particle $\boldsymbol{x}_j$ in N-dimension. If the value of the fitness function, f, is not found satisfactory, then the position $\boldsymbol{x}_j$ and velocity $\boldsymbol{v}_j$ of the jth particle is updated according to position and velocity update relations. The new velocity of the jth particle in nth dimension, for the $(k+1)$th iteration, is calculated as an additive influence of three major components, i) component *I*: the current velocity (at kth iteration) of the jth particle in nth dimension (denoted by v_{jn}), ii) component *II*: the difference between the nth dimension component of the best position obtained by the jth particle, until now (denoted by p_{jn}) and the current position (at the kth iteration) of the jth particle in nth dimension and iii) component *III*: the difference between the nth dimension component of the best position obtained by any particle in the topological neighborhood of the jth particle, until now (denoted by p_{gn}) and the current position (at the kth iteration) of the jth particle in nth dimension. The influence of each of the components II and III is stochastically weighted and added to component I to obtain the updated velocity. This velocity update relation can be given as:

$$\begin{aligned} v_{jn}(k+1) &= v_{jn}(k) + \varphi_1(p_{jn}(k) - x_{jn}(k)) \\ &\quad + \varphi_2(p_{gn}(k) - x_{jn}(k)) \end{aligned} \tag{1}$$

φ_1 and φ_2 are two uniformly distributed random positive numbers, used to provide the stochastic weighting, and they are restricted by the maximum value of $\varphi_{\max}$. Usually $\varphi_{\max}$ is chosen between 1.6 and 2.0. To prevent any unwanted exploration of a particle along a given dimension, its velocity in that dimension is restricted by its maximum permissible value ($v_{n\max}$). This option is exercised to keep the random search of the potential solutions, in quest of a better fitness,

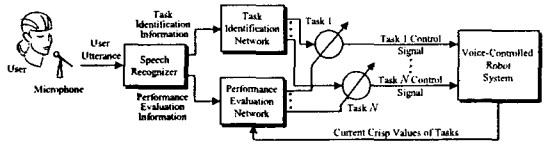

Fig. 1. The architecture of the proposed voice-controlled robot systems.

within control. Then the new position of the jth particle in nth dimension is calculated as:

$$x_{jn}(k+1) = x_{jn}(k) + v_{jn}(k+1) \tag{2}$$

To overcome too aggressive or negative search situation, various researchers have recently proposed improved velocity update rules with dynamic inertia weights or employing constriction coefficients. Velocity update relations with dynamic inertia weights W can be given as:

$$\begin{aligned} v_{jn}(k+1) &= W(k+1)v_{jn}(k) + \varphi_1(p_{jn}(k) - x_{jn}(k)) \\ &\quad +\varphi_2(p_{gn}(k) - x_{jn}(k)). \end{aligned} \tag{3}$$

As opposed to the proposal of incorporating inertia weights which only exercise its influence over component I of the velocity update relation in (1), constriction coefficients are employed to exercise wider control over each of the three major components of the velocity update relation in (1), in an effort to prevent explosion of the system. The proposed velocity update relation can be given as:

$$\begin{aligned} v_{jn}(k+1) &= \chi(v_{jn}(k) + \varphi_1(p_{jn}(k) - x_{jn}(k)) \\ &\quad +\varphi_2(p_{gn}(k) - x_{jn}(k))). \end{aligned} \tag{4}$$

It has been argued that there is no requirement of restricting velocity in any dimension for PSO employing constriction coefficient. The constriction coefficient χ makes this requirement redundant. However it has also been pointed out that employing constriction coefficient with a liberal $v_{n\max}$ equal to the dynamic range of the variable may be quite useful [4].

III. Fuzzy Neural Network in Voice-Controlled Robot Systems

The proposed voice-controlled robot system is shown in form of a schematic diagram in Fig. 1. The system is composed of four major building blocks: a speech recognizer (SR), a task identification network (TIN), a performance evaluation network (PEN) and the robot system under consideration. The input to the system appears in form of spoken commands from a human. The entire system is designed so that the linguistic nature of spoken directive of the human user is translated in form of a quantified and crisp desired action for the robot system. To start with, each and every running utterance from a user is stripped off by a SR module to create a pseudo sentence. Creating a pseudo sentence from a user utterance implies that the in-vocabulary (IV) words are segregated from out-of-vocabulary (OOV) words. For example, if a user utters *"Robot, can you move backward very slow"*, then the OOV words *Robot, can* and *you* are stripped off to create the pseudo sentence *"move backward very slow"* comprising of IV words. This pseudo sentence actually consists of the meaningful semantic action along with its linguistic adjective for the robot to perform its action. Spotting of proper IV words in a running utterance is achieved by training a left-to-right category of Hidden Markov Models (HMMs) [6]. The construction of this pseudo sentence is achieved by using the HMM Toolkit (HTK) distributed as a freeware in the web by Speech Vision and Robotics Group of the Engineering Department, Cambridge University. This speech recognizer employs phoneme-based recognition of IV words where the phonemes of each IV word are designed as tri-state left-to-right HMM models. The strength of a keyword recognizing module depends on how intelligently the IV words can be recognized and it requires efficient training of the HMM based speech recognizer. Special attention has been provided to identify similar actions with synonyms. Initially HMMs are trained so that they can identify isolated words and later they are added together to build the database for the SR.

Once the pseudo sentence is constructed, it comprises of two parts: each definite task/action along with the evaluation of performing that task. For example, in the pseudo sentence *"move backward very slow"*, the task/action part is *"move backward"* and the evaluation of performance is *"very slow"*. The selection of the task part comprises of the qualitative nature of the job and performance evaluation part is executed to determine the quantitative, crisp output signal to be generated by the network for the robot system, for that given task as commanded by the user. For selection of the specific action, the pseudo sentence is fed to a single layer perceptron based artificial neural network (ANN). Here each output node characterizes the presence of a specific IV word in the pseudo sentence by generating a high output. Each output node employs a hard limiting characteristic function and the ANN is overtrained to specifically identify the set of IV words. For a set of M IV words with P distinct tasks the ANN employs M input nodes and P output nodes. Normally $P \leq M$ to take care of different words with similar tasks i.e. synonyms in user utterance.

Once the task identification ANN identifies the exact linguistic nature of the task assigned, it becomes very important to determine the quantitative degree of severity with which the task has to be performed. This very important part of the total system is implemented with the help of the PEN. This PEN is implemented with the TS-type FNN which is trained with the help of PSO. This FNN-based PEN is employed to acquire fuzzy linguistic information from the domain expert to train the architecture so that it can produce precise, quantitative control signal for the robot system as an output, based on the user command. The FNN-based PEN utilizes both the linguistic task identification and performance evaluation clauses in the pseudo sentence as its one input. For example in the pseudo sentence *"move backward very slow"*, the linguistic task identification clause *"move backward"* is associated with a fuzzy predicate (FP) symbolizing the performance evaluation clause *"very slow"*. Each linguistic task identification input is associated with an FP which is

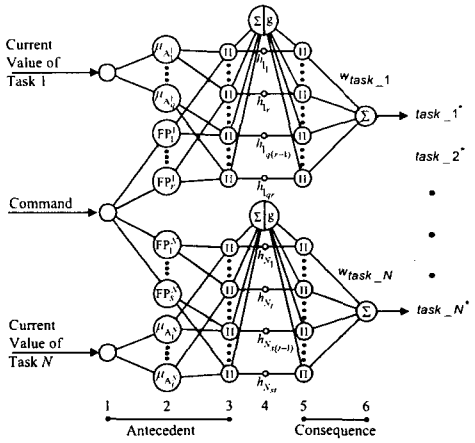

Fig. 2. The FNN employed as the performance evaluation network.

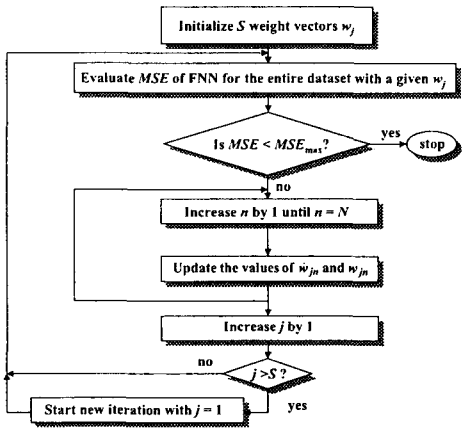

Fig. 3. The training of the FNN employing particle swarm optimization.

characterized as a singleton. Each FP belongs to a global FP database for all actions. One sample FP database can be $\{very\ slow,\ slow,\ carry\ on,\ fast,\ very\ fast\}$. The other input of the FNN corresponds to the current quantitative value of the tth task selected from TIN. The architecture of the FNN is shown in Fig. 2.

IV. Training of the TS-type FNN Employing PSO

The PSO training algorithm of the TS-type FNN based PEN is shown in Fig. 3, where the algorithm has been employed to train the weights w_{ir} in the defuzzification layer of the FNN architecture. The PSO problem has been defined as an N-dimensional problem where N is the total number of output weights in the FNN-based PEN. Our objective is to train the output weights of the FNN so that the PEN can produce desired crisp output control signal according to the pseudo sentence derived from the spoken directive from the user and the current state of the robot task. The FNN is trained in batch mode with a training dataset of I data pairs. The problem is formulated as a minimization problem where the fitness function is based on the mean-squared-error (MSE):

$$MSE = \frac{1}{I}\sum_{i=1}^{I}(y_{di} - y_{ai})^2. \tag{5}$$

Here y_{di} is the desired output and y_{ai} is the actual output from the FNN for the task i in the training phase. y_{di} is obtained from a knowledge base, created using the knowledge of the domain expert for that specific robot system problem domain.

All the training data pairs are created by employing completely randomly chosen numbers for each input. Then, we choose a possible size of population S which gives us a possible set of weight vectors $\{\boldsymbol{w}_1\ \boldsymbol{w}_2\ \cdots\ \boldsymbol{w}_S\}$. Each weight vector $\boldsymbol{w}_j$ is a potential particle for the PSO algorithm and is an $R \times 1$ vector where R is the total number of output weights in the FNN. Hence $\boldsymbol{w}_j = [w_{j1}\ w_{j1}\ \cdots\ w_{jR}]$ where $R = P \times Q$. To start with, all the weights in each weight vector $\boldsymbol{w}_j$ are randomly initialized in a discourse of $[w_{left}, w_{right}]$.

In the iteration, numerical values of each component weight in each possible weight vector are updated according to the PSO algorithm. At the end of the kth iteration, the nth component weight w_{jn} in the jth particle is updated by

$$\begin{aligned}\dot{w}_{jn}(k+1) &= W(k+1)\dot{w}_{jn}(k) + \varphi_1(p_{jn}(k) - w_{jn}(k)) \\ &\quad +\varphi_2(p_{gn}(k) - w_{jn}(k)) \qquad (6)\end{aligned}$$

$$w_{jn}(k+1) = w_{jn}(k) + \dot{w}_{jn}(k+1). \tag{7}$$

In our version of the PSO we have employed a linearly adaptable inertia weight W which starts with a high value W_{high} and linearly decreases to W_{low} at the end of the maximum number of iterations, i.e. $iter_{\max}$, if the algorithm continues till the end without meeting the termination criterion. Hence inertia weight for the $(k+1)$th iteration is given by

$$W(k+1) = W_{low} + \left(\frac{W_{high} - W_{low}}{iter_{\max}}\right)(iter_{\max} - (k+1)). \tag{8}$$

For each dimension we can choose different discourse of W, i.e. $[W_{high},\ W_{low}]$, if we wish so. Similarly the maximum permissible velocity of each particle, i.e. the maximum permissible rate of change of each weight, $\dot{w}_{\max}$, can also be accordingly chosen independent of each other.

V. Navigation of a Mobile Robot

The proposed system is implemented for the navigation of Khepera, which can be considered as a miniature version of a robot driven wheelchair. The robot has 2 degrees-of-freedom with turning and moving facility. Figure 4 shows the experimental setup. Here the output control signal from the FNN-based PEN is extracted in form of velocity. Figure 5 shows the MFs chosen for the input variable, current velocity for the FNN network. The knowledge base obtained from the domain expert in form of fuzzy MFs, to derive desired velocity along with the contextual meaning of the spoken directives, is given in Fig. 6. The desired velocity at each time step is obtained from this figure and output (current velocity v) sensed from the Khepera, according to the following relation:

$$Desired\_Velocity = Velocity\_Factor \times v. \tag{9}$$

The ANN employed for the TIN was trained with randomly initialized weights in the discourse $[-0.5, 0.5]$ and a small

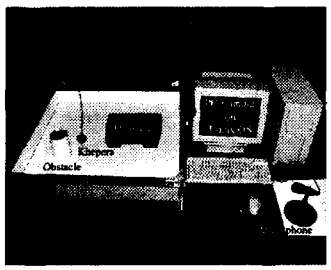

Fig. 4. The experimental setup for Khepera.

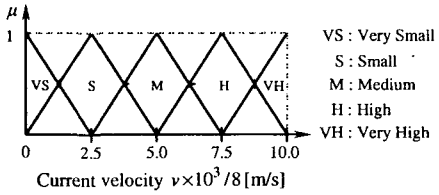

Fig. 5. Membership functions of current velocity of Khepera.

learning rate of value 0.1. The termination criterion is set at an error value of 0.01. For the training process of FNN-based PEN, we have chosen three possible population sizes, 20, 30 and 40. Our objective was to achieve the desired performance with as small a population size as possible, because in that case the training process of FNN will undertake less computation burden. Figure 7 shows the variation of MSE with iterations for training FNN-based PEN for this case study.

In each case, the potential weights for each particle in the PSO was initialized in the discourse $[0, 1]$. The maximum permissible velocity, i.e. the incremental change in position per iteration, $\dot{w}_{\max}$, is kept fixed at 0.1 in each dimension. Each of the inputs of the FNN employs 5 MFs, i.e. 5 MFs for the

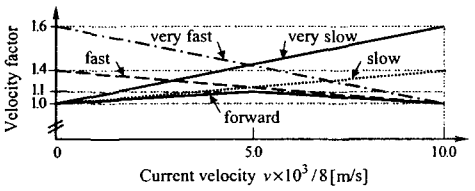

Fig. 6. The knowledge base of the desired velocity of Khepera.

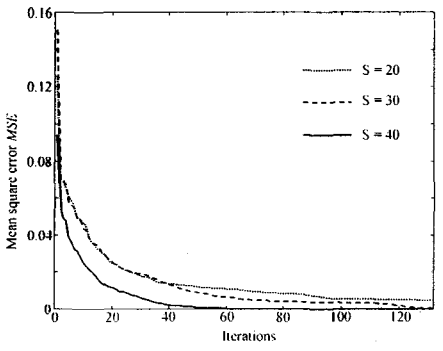

Fig. 7. The Training performance of PSO for Khepera.

TABLE I

USER SPOKEN DIRECTIVES AND THE OUTPUT CONTROL SIGNALS OF FNN.

User directives	Control outputs of the FNN
robot, move forward	2.075
can you go very fast	3.065 ‡
please turn left	No output
I want you move fast	3.634
robot, please turn right	No output
move very slow	2.662
robot, turn right	No output
please go fast	3.404
robot, go very fast	4.298 ‡
stop	0

input velocity and 5 singletons for the fuzzy predicates. Hence there are total 25 possible rules and 25 output weights to be adapted in the defuzzification layer. The size of the training dataset was chosen as 10,000 and the termination criterion was set at MSE = 0.0001. The parameters for the inertia weight W, i.e. W_{high} and W_{low} are chosen as 0.2 and -0.3 respectively for each dimension, in each of the three possible cases employing different population sizes. It can be seen that with 40 particles, the algorithm converged in 132 iterations but the results are not satisfactory with S = 20 or 30. Hence the PSO configuration, employed with a population size of 40, was accepted for the FNN-based PEN and this trained FNN network was implemented in real life for the navigation. The user-spoken utterances for which the robot was tested in real life are given in Table I, which also shows the velocity outputs produced by the FNN as control signals. The importance of the contextual meaning in the spoken directive is amply shown by entries 2 and 9 in Table I (shown by ‡ marks), in which two different increments in output velocity are, from the previous velocity, exhibited for the same user utterance.

VI. CONCLUSIONS

The present work has demonstrated the feasibility of employing particle swarm optimization (PSO) techniques for efficient training of a fuzzy-neural network (FNN). The PSO trained FNN has been successfully employed as an important building block in real life voice-controlled robot systems.

REFERENCES

[1] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proc. 1999 Congr. Evolutionary Computation, IEEE Service Center*, Piscataway, NJ, 1999, pp. 1945–1950.

[2] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proc. 2000 Congr. Evolutionary Computation*, San Diego, CA, July 2000, pp. 84–88.

[3] J. Kennedy, "The particle swarm: Social adaptation of knowledge," in *Proc. 1997 Int. Conf. Evolutionary Computation*, Indianapolis, IN, April 1997, pp. 303–308.

[4] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, Feb. 2002.

[5] A. Conradie, R. Miikkulainen, and C. Aldrich, "Adaptive control utilising neural swarming," in *Proc. Genetic and Evolutionary Computation Conference*, New York, USA, 2002.

[6] R. C. Rose and D. B. Paul, "A hidden Markov model based keyword recognition system," in *Proc. IEEE ICASSP '90*, 1990, pp. 129–132.

[7] K. Pulasinghe, K. Watanabe, K. Kiguchi, and K. Izumi, "Modular fuzzy neuro controller driven by voice commands," in *Proc. ICCAS 2001*, 2001, pp. 194–197.