Neural Comput & Applic (2008) 17:161–173
DOI 10.1007/s00521-006-0079-1

ORIGINAL ARTICLE

An optimized modular neural network controller based on environment classification and selective sensor usage for mobile robot reactive navigation

Seong-Joo Han · Se-Young Oh

Received: 30 August 2006 / Accepted: 13 December 2006 / Published online: 10 January 2007

Abstract A new approach to the design of a neural network (NN) based navigator is proposed in which the mobile robot travels to a pre-defined goal position safely and efficiently without any prior map of the environment. This navigator can be optimized for any user-defined objective function through the use of an evolutionary algorithm. The motivation of this research is to develop an efficient methodology for general goal-directed navigation in generic indoor environments as opposed to learning specialized primitive behaviors in a limited environment. To this end, a modular NN has been employed to achieve the necessary generalization capability across a variety of indoor environments. Herein, each NN module takes charge of navigating in a specialized local environment, which is the result of decomposing the whole path into a sequence of local paths through clustering of all the possible environments. We verify the efficacy of the proposed algorithm over a variety of both simulated and real unstructured indoor environments using our autonomous mobile robot platform.

Keywords Reactive navigation · Evolutionary robotics · Neurocontroller · Environment classification · Cooperative coordination

S.-J. Han (✉) · S.-Y. Oh
Department of Electrical Engineering,
Pohang University of Science and Technology (POSTECH),
Pohang 790-784, South Korea
e-mail: hanisl@postech.ac.kr

1 Introduction

There has been an increasing interest in developing service robots that learn to navigate in complex, partially known, and unpredictable environments. Because these environmental characteristics differ from the highly structured and constrained environments of the industrial robots, the mobile robot cannot be fully pre-programmed to carry out a pre-defined set of actions. Furthermore, it is often very difficult to find a single control algorithm that applies to all working environments. Therefore, these robots need a reactive navigational capability where they appropriately react to any change of the environment with onboard sensors.

To cope with reactive navigation, various approaches have been proposed. Lumelsky and Stepanov [1] proposed the Bug approach, which consisted of two actions ''moving directly toward the target'' and ''following the obstacle boundary.'' The Bug approach then evolved to the DistBug algorithm [2] that employed range sensors to define a set of new heuristic leaving conditions, that is, the condition for switching actions from ''following the obstacle boundary'' to ''moving directly toward the target,'' according to varying complex obstacle configurations. Khatib [3] proposed an artificial potential field (APF) approach, which consists of two force components ''attractive force for reaching the goal'' and ''repulsive force for avoiding obstacles,'' derived from the range sensors. This approach was further developed into the virtual force field [4] and the vector field histogram [5] by Borenstein and Koren. Although these APF approaches can produce fast and smooth control commands, the robot stalled in some obstacle configurations called a ''local minimum.''

Various techniques have been developed to escape such local minima—use of additional force components [6, 7], additional heuristics [1, 8, 9], and the computational intelligence technique [6]. Along with these potential field approaches, the behavior-based approach has been introduced where different behaviors take care of the navigation task as a function of the sensor data. In order for these (either learned or pre-designed) behaviors to cope with new situations, cooperative behavior coordination was used that concurrently uses some of the behaviors. In, AFREB (Adaptive Fusion of REactive Behaviors) [10], a neural network (NN) supervisor learned the necessary strengths of behaviors by trial and error. For example, when ''the goal is in sight,'' a combination of ''goal attraction'' and ''keep away'' primitive behaviors were used.

In another line of research, soft computing which attempts to model the biological organism's learning and adaptation has also been employed to solve the navigation problem. This technique has relieved a human some of the laborious design task. For instance, an NN with backpropagation learning has been applied to human teaching [10–12] and reinforcement learning [13, 14] of the task. On the other hand, evolutionary optimization of robot design, or evolutionary robotics (ER), was used to optimize the parameters of a human designed controller [6] or those of a known control structure [15–18], or even the control structure itself [19, 20]. Herein, the evolutionary algorithm (EA) is invaluable for optimizing multi-objective problems in unsupervised learning context. Furthermore, the EA has another advantage that it can accommodate any desired cost function without any change in the algorithm itself. Floreano and Mondada [16] and Miglino et al. [18] proposed an ER approach to evolve an NN controller for developing ''battery recharging,'' ''looping maze,'' and ''obstacle avoidance'' behaviors in simple environments. Later, Hoffmann [17] proposed a soft computing approach, which hybridized all of the fuzzy logic, NN, and EA, to learn the wall following behavior in simple environments. However, most of these ER approaches were confined to learning or evolving a limited set of specialized behaviors in simple and limited environments in a reactive navigation mode.

This paper presents a new ER based on NNs approach to designing a full-blown navigator for mobile robots to travel to a goal position safely and efficiently without any prior map of the environment. The only assumption is that the robot knows where it is and in which relative direction the goal is located. The objective function was designed from the basic philosophy that the robot should move to the goal safely with minimum time and energy using a simplest possible NN architecture at any given time. To this end, the NN has a dynamically reconfigurable structure that not only optimizes its weights but also minimizes the sensor connectivity (hence uses a minimum number of sensors to carry out the local navigation task) to the output layer as a function of the local environment. It is predicated on that usage of more sensors than is necessary may even harm the resulting control performance.

However, even with EA optimization, the system performance is largely a function of the robot environment which may vary a great deal along the robot path. Hence, the size of the single NN will grow with complexity of the environment which will inevitably lead to a long search time for the EA. The solution therefore seems to be the use of a modular neural network (MNN) architecture where each NN module is optimized to a prototype local environment obtained by clustering the wide spectrum of possible navigation environments. Eventually, we hope to develop a reactive navigation controller, which works well in an actual residential environment, in contrast to many of the previous ER approaches evolving several behaviors in a few rather simple environments. Furthermore, in order to limit the number of controller modules used to generate more efficient commands under greatly varying environments, cooperative coordination was adopted in our research which provides different behavior coefficients according to the similarity of the current environment to the prototype environment under which each NN module has been optimally trained. This type of coordination also lends itself to generating emergent behaviors as well.

This paper is organized as follows. Section 2 explains the NN controller architecture and the environment classification methodology used in our research. Section 3 briefly explains EA and introduces the basic ingredients of our ER approach in detail. Section 4 demonstrates its navigating performance under both simulated and real environments. Section 5 finally discusses some important issues emanating from the proposed ER research.

2 NN controller for robot navigation

In this section, the structure of the feedforward NN, which is the building block of the overall control architecture, as well as the construction of the MNN architecture will be described in detail.

2.1 The robot and its NN controller module

The robot, adapted from the Active-Media Pioneer 2-DX, is equipped with eight ultrasonic sensors as shown in Fig. 1. The eight ultrasonic sensors are embedded in a Polaroid 6500 Series Sonar Ranging Module which has a typical absolute accuracy of ±1% over the range from 15 cm to 10 m according to its technical specification. However, we used them up to about 4 m as its maximal range considering the sensor interference due to beam spreading and the scale of the indoor environment. The variable θ_g denotes the relative orientation of the target. The single layer perceptron (SLP) NN [21] shown in Fig. 2 generates a steering command from the inputs consisting of the eight sensor readings and the goal orientation θ_g. The simultaneous consideration of both the local information coming from the sensor readings and the global (goal) information can generate the sequence of steering commands that perform obstacle avoidance and target seeking concurrently. The output of the NN takes a continuous real value ranging from –1 to +1 through a sigmoid function. This real-valued output is then linearly mapped to the steering angle ranging from –90° to +90° with 1° precision.

2.2 The MNN architecture

Since the navigation environment is usually complex, partially unknown, and unpredictable, a single NN controller can hardly take charge of the whole task. If a single NN is used, it must have a complex structure with many internal parameters to solve the problem of navigation which is highly nonlinear. Therefore, a MNN, based on the principle of divide and conquer, has been employed to solve many nonlinear problems with good generalization and fault-tolerance capabilities. Because the MNN uses many simple NN modules each covering a specific local environment, it can quickly and easily find good local solutions [22]. Figure 3 shows the MNN architecture used in this research.

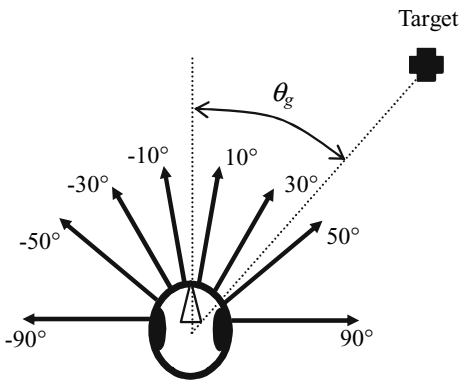


Fig. 1 The autonomous mobile robot testbed

2.3 Environment classification for gating of the NN controller modules

Environment classification has been employed for the purpose of decomposing a complex navigational task into simpler subtasks in this research. About 3,000 ultrasonic patterns have been obtained from a great variety of both real and simulated environments where the raw sensor measurements went through normalization followed by an eight level quantization. All the pre-processed measurement data have been clustered using the follow-the-leader (or leader clustering) algorithm [23]. The clustering algorithm creates new clusters whenever the distance from the new pattern to existing cluster centers (or prototypes) exceeds a preselected threshold δ. It is one of the simplest and fastest clustering method which is also suitable for on-line applications. This on-line feature is essential to add new environments that cannot be accounted for by existing cluster prototypes. Unlike the K-means algorithm, it does not specify the number of clusters at the outset but the rough size of the clusters with a threshold parameter δ. It is also in line with the threshold distance parameter used to choose N similar environments in cooperative coordination. Incidentally, this result was also compatible to the result of applying the K-means to the same data with nine clusters.

The number and the locations of the cluster centers depend on δ and sometimes, the incoming sequence of the patterns. To reduce these effects, the leader clustering has been processed many times (1,000 times in our research) with a fixed threshold value but with different sequences of input presentation. Then, the solution with a minimum number of cluster centers is selected as optimal with the given threshold. This threshold has been varied with 1 cm increments in our research. The set of nine cluster centers for environment classification was chosen after carefully looking at the curve in Fig. 4 showing the number of clusters vs.

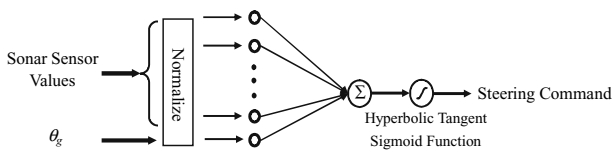


Fig. 2 The NN controller for autonomous mobile robot navigation

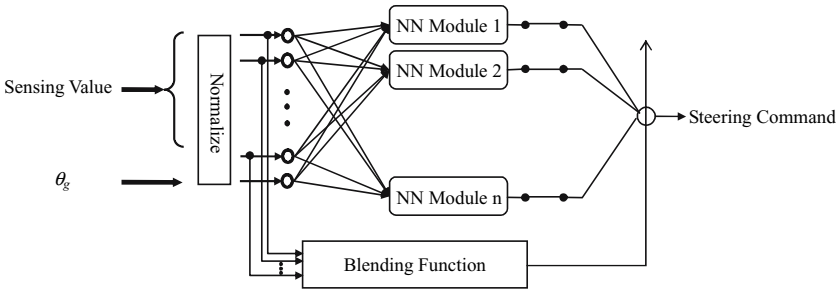


Fig. 3 Modular NN architecture with a blending function

the threshold. There is a justification behind the choice of using nine cluster prototypes. In Fig. 4, we observe two places—9 → 12 and 15 → 19—of abrupt change in number of resulting clusters for a small change of threshold. But, 9 clusters seem to be more stable in that it remained constant over a wider range of threshold than 15 clusters. Furthermore, the prototype environments represented by the clusters diagrammed in Fig. 5 mostly resemble a classification of the typical local indoor environments in Fig. 6.

The prototype local environments in Fig. 5 were obtained after clustering a wide spectrum of indoor environments such that each cluster was distant enough (that is, more than a certain Euclidean distance away) from the rest to be meaningful. However, sometimes due to some cylindrical objects, trash cans, and other kinds of obstacles, some incident environment turned out to be close to more than one prototype. In order to deal with some local environmental situations resembling more than one prototype, cooperative coordination in Sect. 2.4 was developed for navigation control to handle the ''in-between'' situations. These prototype environments will be utilized as the basis of finding the most suitable NN controller module. The MNN architecture in Fig. 3 utilizes the result of local environment classification as the blending function which simply becomes a gating function.

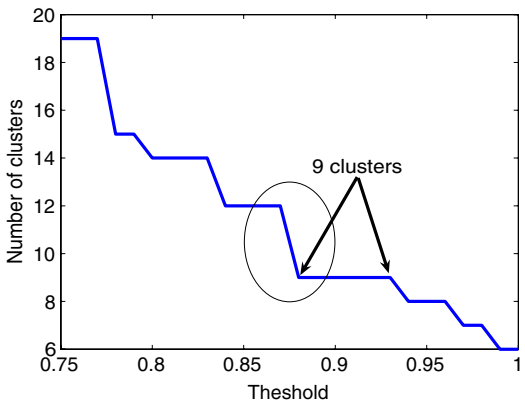


Fig. 4 The curve for the number of clusters vs. the threshold for new cluster creation

2.4 Cooperative coordination of the NN controller modules

Each control module in charge of local path planning will be optimized with respect to a particular sensory environment called a cluster prototype. Therefore, as the current environment moves away from this prototype environment, the performance gradually degrades. The farthest point would be the midpoint between two adjacent clusters. In this case, a proper blend of the two or more control behaviors will enhance the performance compared to the case of adopting only one behavior. At the other extreme, use of too many modules will also worsen the performance due to interference between the modules that have been pre-trained in widely varying environments. Therefore, it would be a good strategy to choose a threshold leading to selecting a proper number of relevant modules by looking at the average distance between clusters. It would then provide a good balance between the number of relevant modules used and the range of interference.

In addition, the size variation of the indoor environment may in general affect the performance of navigation. However, quantization of the sensor range as well as the module cooperation technique can cope well with these environmental variations. For example, the left corner in a small size environment is resolved by blending narrow corridor and left corner modules. On the other hand, the museum, a special case of a huge environment, is a very easy environment for navigation because most of the local environments are perceived as the free space. In another special case, however, of a relatively small indoor environment, the maze in a micro mouse contest is a very difficult environment for reactive navigation. In this case, the wall following technique may be more preferable.

In order to blend the outputs of the modules, the Euclidean distance information between the current

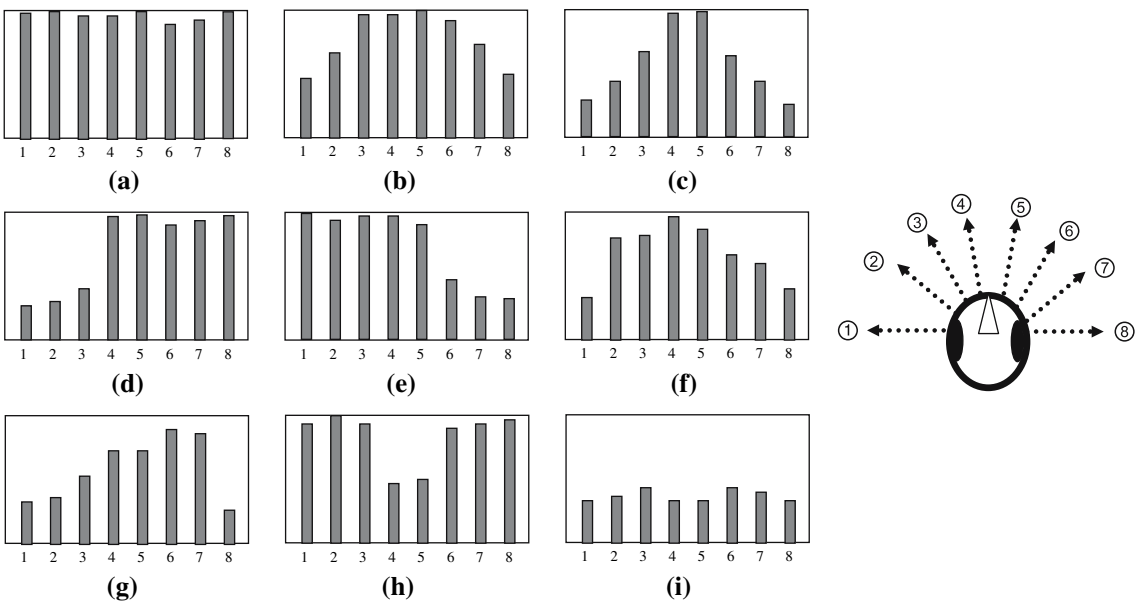


Fig. 5 Sensory patterns for nine clustered prototype environments defined in Fig. 6

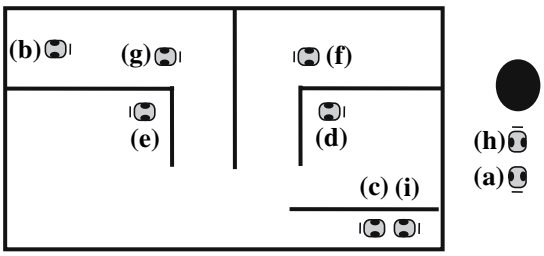


Fig. 6 Illustration of the prototype environments: **a** open space, **b** wide corridor, **c** narrow corridor, **d** left wall, **e** right wall, **f** left corner, **g** right corner, **h** isolated obstacle, **i** U-shape wall

sensory pattern and the prototype patterns has been used to compute the blending function in Fig. 3. The coordination mechanism was changed from competitive gating to a cooperative blending type. However, a key issue for cooperative coordination is how to select the relevant modules as well as how to combine the outputs of the selected modules. In this paper, a simple coordination method based on Euclidean distance is employed as follows. The average inter-cluster distance, d_{avg}, is used for module selection, i.e., determining whether the module should enter cooperation or not. Choosing this average distance of 0.94, the threshold for switching to employ cooperation is chosen to be one half of this average, i.e., 0.47. Therefore, if the distance between the input sensory pattern and the closest stored environment prototype exceeds the threshold of 0.47, a cooperative coordination mechanism is activated that employs distance weighted combination:

$$\Delta\theta = \sum_{\text{Selected modules}} w_i \cdot f^i(\mathbf{s}, \theta_{\text{g}}) \tag{1}$$

$$w_i = \frac{(d_{\text{avg}} - d_i)^2}{\sum_i (d_{\text{avg}} - d_i)^2}. \tag{2}$$

Here, $\Delta\theta$ is the final steering control command for the robot, $\mathbf{s}$ is the ultrasonic range pattern, f^i is the output of the ith module, and d_i is the Euclidean distance between the current range pattern and the ith prototype.

3 Evolution of the NN controllers

Each NN controller module, in charge of navigation around a particular local environment, is optimally trained using an EA. The different training environments have been created by a human who drives the robot with a remote controller.

3.1 Evolutionary algorithm

Evolutionary algorithms [24] are stochastic search methods inspired from the process of biological evolution. EAs operate on a population of potential solutions applying the principle of survival of the fittest to produce better and better solutions. Specifically, EAs repeatedly apply the reproduction process of potential solutions, the variation process of survived

solutions, and the selection process of better solutions to gradually find the fittest solution.

There are three main avenues of research in EA: genetic algorithm (GA), evolution strategy (ES), and evolutionary programming (EP). GA emphasizes chromosomal operators. ES and EP emphasize behavioral change at the level of individual and species, respectively [25].

3.2 Evolution of the controller modules

Evolving the NN controller modules follows the procedure in Fig. 7. However, our variation process, i.e., searching for improved controllers, employs a hybrid evolving technique that searches for both optimal sensor connectivity (not all the ultrasonic sensors are used to generate the steering commands at any given time) and optimal weights of NN using GA and EP, respectively. GA is used for evolving a sensor usage that is directly coupled with the NN structure, while the real-valued weights of the NN controller are evolved using EP, which seems to be more efficient for optimizing the behavioral trait [24]. The hybrid evolving technique shown in Fig. 8 has a hierarchical loop structure. GA searches for the best solutions of the NN structure in the outer loop and then the inner loop of EP is activated to search weights of the controller with the selected NN structure.

The chromosome for evolving the NN controllers consists of the sensor usage bits and the real-valued NN weights as shown in Fig. 9. The Boolean part signifies that the NN controller uses sensors associated with only ''1'' valued linkage to generate the steering command. Therefore, this chromosome is decoded by multiplying Link_i and Weight_j. In GA, we employed one-point crossover and bit mutation as the variation operator. In EP, we employ self-adaptive mutation [24] as the variation operator:

$$\sigma_i'^j = \sigma_i^j \exp\left(\tau N_i(0,1) + \tau' N_i^j(0,1)\right) \tag{3}$$

$$x_i'^j = x_i^j + \sigma_i'^j N_i(0,1) \tag{4}$$

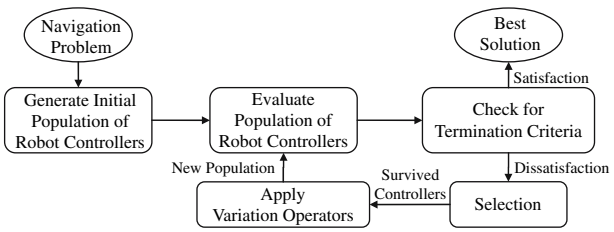


Fig. 7 Process of ER

where x_i^j is jth component of ith individual, σ_i^j is jth standard deviation of ith individual, τ and τ' are operator-set parameters, and N represents the normal distribution. The parameters used for carrying out the proposed EA learning process are shown in Table 1.

3.2.1 *Cost function*

A good selection of the cost function is very important to the control performance. The basic philosophy in designing our cost function is that the robot should move to the goal safely and smoothly in minimum time with the simplest possible NN architecture. The cost function is defined as:

$$\text{Cost} = \alpha\,\text{Cost}_\text{S} + \beta\,\text{Cost}_\text{P} \tag{5}$$

$$\text{Cost}_\text{S} = \sum_{k=1}^{M} \text{Link}_k \tag{6}$$

$$\text{Cost}_\text{P} = \sum_t (Q_1\text{Col} + Q_2\text{Osc} + Q_5\text{Clr}) + Q_3\text{Lng} + Q_4\text{Arr} \tag{7}$$

$$\text{Osc} = \begin{cases} 0.1 & \text{if the robot oscillates} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$\text{Col} = \begin{cases} 10,000 & \text{if the robot collides} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

$$\text{Lng} = \text{total length of the trajectory} \tag{10}$$

$$\text{Arr} = \begin{cases} 100 & \text{if the robot cannot reach the goal before time - out} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

$$\text{Clr} = \begin{cases} 1.0 - \dfrac{\min_i s_i}{\text{Avg}_i(s_i)} & \text{if } \min_i s_i \le \Delta \\ 0 & \text{if } \min_i s_i > \Delta \end{cases}. \tag{12}$$

The gross cost ''Cost'' is defined as a weighted sum of the structural cost and the performance cost. The structural cost ''Cost_S'' is proportional to the number of sensors that have been actually used for generating the steering command at any time while the performance cost ''Cost_P'' is a weighted sum of five factors as follows. First, the cost factor ''Osc'' is the amount of oscillation in the actual followed path. The three consecutive steering commands like right-left-right or left-right-left are considered to be an oscillation. The cost factor ''Osc'' guarantees a relatively straight and

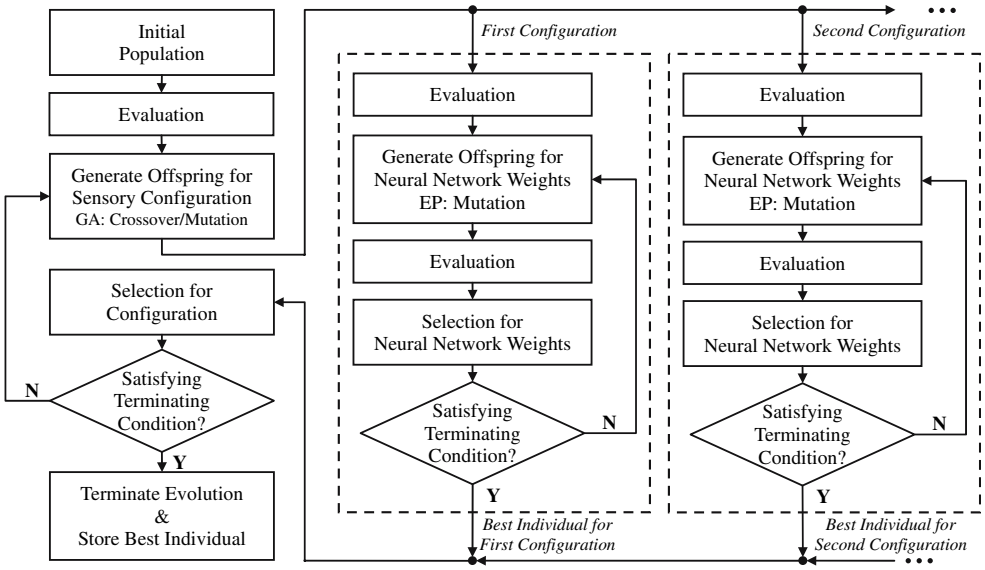


Fig. 8 Hierarchical evolving procedure with structure learning at the outer loop

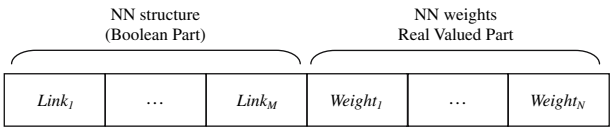


Fig. 9 Chromosome encoding of the NN controller

Table 1 The parameters used for EA

Number of generations	200
Population size	30
Individual size	18
Elite size	3
Tournament size	15
Crossover rate for GA	0.3
Mutation rate for GA	0.05

smooth trajectory. Second, the cost factor ''Col'' measures the occurrence of collision. If the robot controlled by the candidate solution collides with an obstacle, this solution incurs a very high cost which makes it difficult to survive to the next generation. Third, the cost factor ''Lng'' measures the total length of the trajectory so that it targets on a relatively efficient path generation, i.e., shorter path. Fourth, the cost factor ''Arr'' aims at convergence to the goal. The case of ''if the robot cannot reach the goal'' indicates that the length of the trajectory exceeds a pre-defined threshold caused either by getting stuck on a local minimum or by generating an excessive detour. Lastly, the cost factor ''Clr'' for obstacle clearance ensures a safe trajectory. In (12), s_i is the ith sensor reading. The reason for dividing by $\mathrm{Avg}(s_i)$ is to resolve the difference between an open space (e.g., wide corridor) and a narrow environment (e.g., narrow corridor). Because the robot in a narrow environment has no chance of keeping the obstacles beyond a certain distance, the trajectory passing the obstacles with equal distances becomes an optimal trajectory in our simulation run. This guarantees that the controller generate a compromised trajectory between safety and efficiency. The relative weights α, β for ''Cost'' and $Q_1 \sim Q_5$ for ''$\mathrm{Cost_P}$'' are the empirical constants all set to ones in our experiments.

3.2.2 Selection mechanism

Chromosomes are selected based on the result of evaluation. The mixed selection method which combines rank-based and tournament selection methods has been used for both GA and EP. In rank-based selection, the rank ordering of the cost value of the chromosomes within the current population determines the probability of selection [2]. In tournament selection, chromosomes with more wins against a subset of randomly selected opponents from the current population are selected. The rank-based selection guarantees convergence due to preserving the best solution while the tournament selection guarantees search diversity due to having a finite survival probability of the relatively inferior chromosomes.

4 Experimental results

The proposed ER algorithm has first been implemented on a simulator in order to speed up the evolution process and further to guarantee the safety of the robot. Then it was applied to the Pioneer robot in

our laboratory environment. In order to find an optimum controller, the robot first moves, for a given pair of starting and goal positions, using an NN controller candidate among the population.

4.1 Results for a single NN controller

Each NN controller module is specialized for each local environment prototype shown in Fig. 5. The process for evolving each NN controller starts from the initial population of identical solutions consisting of the pre-trained NN weights to alleviate a long search time. The pre-trained initial population is acquired through supervised learning using human teaching data followed by further evolving of the NN weights using all the sensory links in a variety of obstacle scattered environments. The training patterns were acquired by a human through a remote controller. Figure 10 shows the evolution of the NN controller with all sensory links for generating an initial population in an obstacle scattered environment. It contains mostly the open space of the previous environment prototypes in Fig. 5 but also several other prototypes. Since evolution is sought to minimize the total cost, the individual costs may sometimes increase. For example, ''Cost of oscillation'' in Fig. 10b went higher toward the end because its importance in the total cost is relatively low compared to the ''Cost of obstacle clearance.'' Figure 11 shows the actual controlled path at several stages of evolution.

Figure 12 shows the evolution of the NN controller module for open space navigation—most distance readings are large enough—using aforementioned pre-trained initial population. Evolution of the other modules has similar tendencies. It is surprising that use of fewer sensors than the full set still shows a good performance. The results for all the controller modules are summarized in Table 2. Since we placed more priority on the control performance than on the minimal structure in this paper, the performance cost dominated the overall cost. In the future however, robots may be equipped with a great variety of sensors and the relative importance of using a simper control architecture may go up. In this case, we may increase β in (6) to reflect this change. It is up to the designer to change the relative importance to meet his objective. Figure 13 shows the actual controlled trajectories for each of the best evolved NN controllers in their corresponding environments.

4.2 Results for a MNN controller

The evolved NN controller modules and the environment classification module are combined into the MNN controller in Fig. 3. For our experiment, only linear NNs (SLP) were used with no hidden layers.

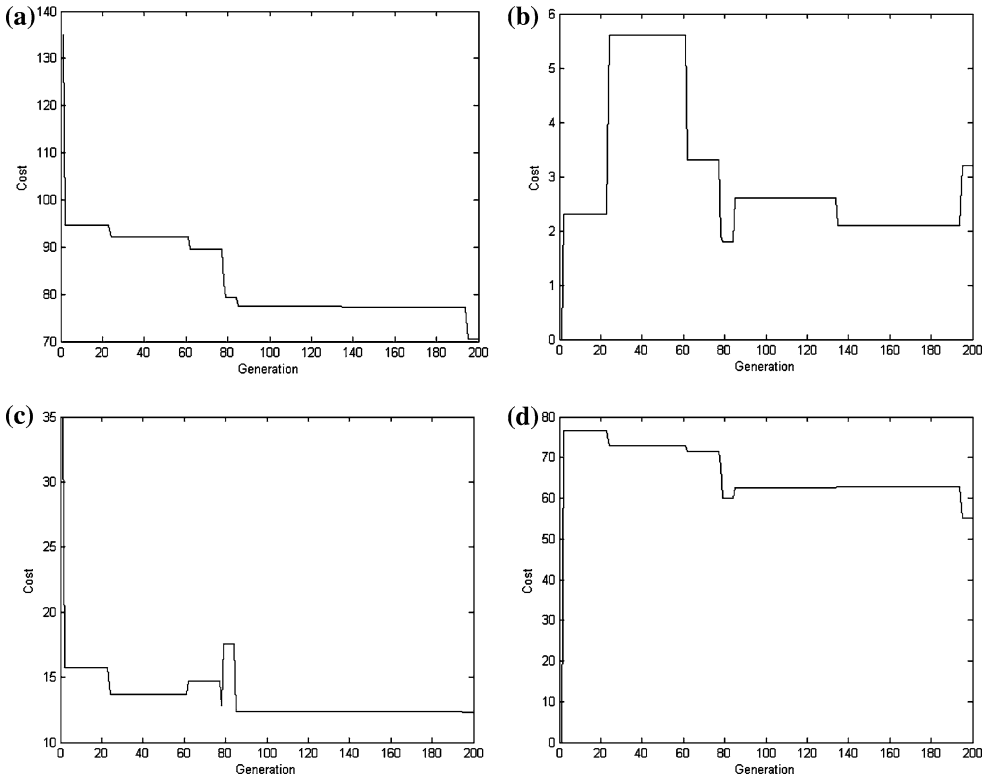


Fig. 10 Evolution of the cost function with all sensors used: **a** total performance cost, **b** cost of oscillation, **c** cost of path length, **d** cost of obstacle clearance

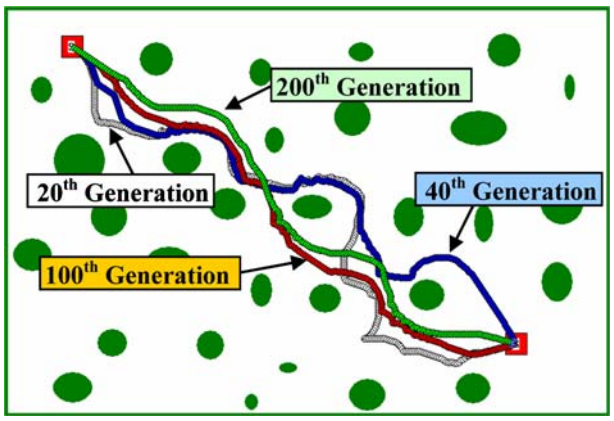


Fig. 11 Control trajectory during several stages of evolution

Hence, the size of the NN depends on how many sensory inputs are used to generate the control signal. While the MNN architecture as a whole, like a single NN case, utilizes all of the eight ultrasonic sensors, each module will select the most effective subset of them as found by the EA to derive the steering command. As shown in Table 2, different sizes of NN module such as ''4-by-1,'' ''5-by-1,'' ''6-by-1'' are used depending on the specific local environment the robot is passing through.

For the same navigational task, Fig. 14 compares the performance of the pre-trained single controller and the modular controller with the resulting cost values listed in Table 3. Figure 14a shows that a single NN optimized to a particular environment outperforms the MNN in that environment as shown in Table 3. However, the MNN in Fig. 14b, c performs much better than a single NN in general environments. Because the single NN controller has been optimized to the particular condition of the obstacles and the start–goal configurations in environment (Fig. 14a), it shows a better performance only in this particular environment. The higher cost associated with the smoother trajectory in Fig. 14b, c arises from the fact that the clearance cost takes a larger portion of the total performance cost. That is, as shown in the cost function plot in Fig. 10, the clearance cost undergoes a relatively larger change than the other cost components (note that the cost scale of the obstacle clearance is larger than that of the others). The overall cost is improved by enhancing clearance while sacrificing oscillation with the current weighting scheme. However, a smoother trajectory could have been obtained by decreasing the weight of clearance and increasing the weight of oscillation.

The weights of the individual cost functions must be carefully chosen by observing their dynamic ranges. The cost function ''Col'' and ''Arr'' are handled separately from the rest. At the first occurrence of any collision or any failure to arrive at the goal within a

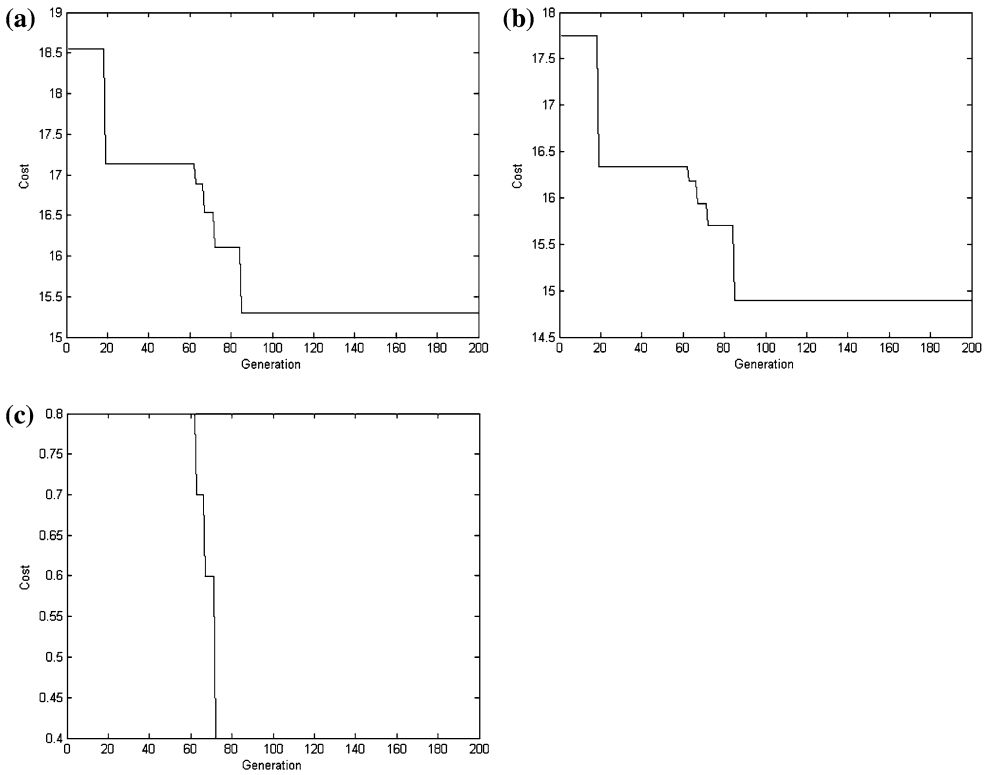


Fig. 12 Evolution of the cost function for open space navigation: **a** total cost, **b** performance cost, **c** structural cost

Table 2 Evolution of a single NN controller for each of the nine prototype object environments

Nine object environments	Cost improvement (init. pop. → fin. result)	Unused sensors for the controller
Open space	18.55 → 15.30	–50°, –10°, 10°, 30°
Wide corridor	90.90 → 83.06	–50°, –10°, 10°, 30°
Narrow corridor	141.80 → 124.26	–50°, 50°
Left wall	26.17 → 13.67	–50°, –10°, 50°
Right wall	74.39 → 15.19	–50°, 30°
Left corner	33.50 → 25.78	–50°, 10°, 90°
Right corner	36.45 → 28.56	–50°, 30°
U-shape wall	101.19 → 59.89	–90°, –50°, 10°
Isolated obstacle	67.87 → 21.60	–50°, –10°, 10°, 50°, 90°

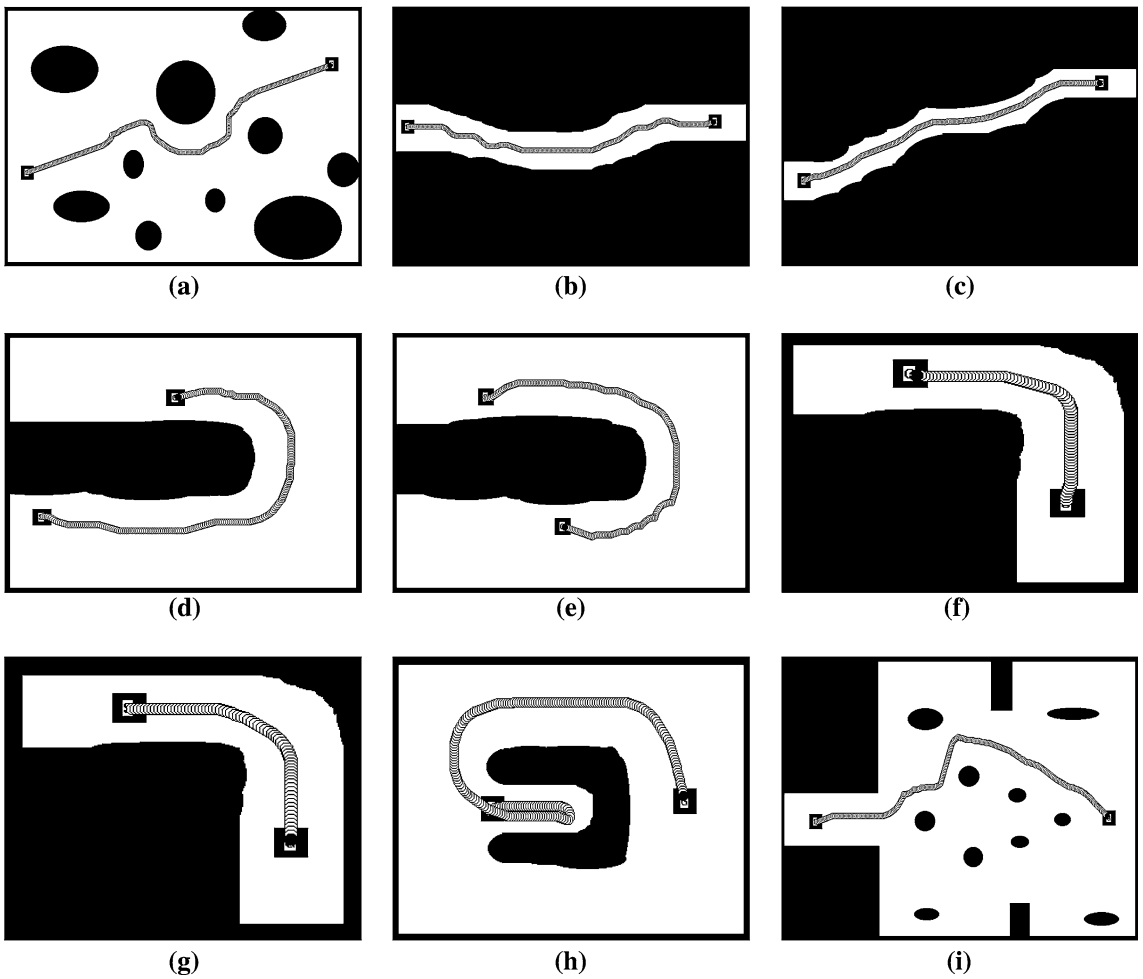


Fig. 13 Actual controlled trajectories of the controller modules for nine prototype environments: **a** open space, **b** wide corridor, **c** narrow corridor, **d** left wall, **e** right wall, **f** left corner, **g** right corner, **h** U-shape wall, **i** isolated obstacle

specified time limit (called time-out) in a controlled trajectory, the corresponding controller solution will be discarded. They have an overriding effect on the rest of the performance consideration. If neither becomes a problem, the weights on the rest of the performance (total trajectory length, oscillation, and obstacle clearance) will indicate the subjective preference on each performance. As shown in Fig. 10, the dynamic range of the clearance and then the amount of oscillation dominate the path length, a proper weighting scheme must be chosen by the designer to suppress the influence of any of them.

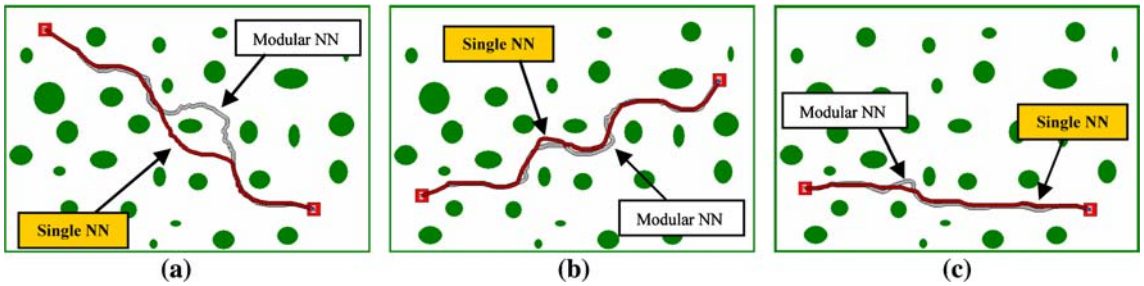


Fig. 14 Comparison of the controlled trajectories between the single and the modular NN controllers under three different environments

Table 3 Comparison of the cost values between the single and the modular NN controllers

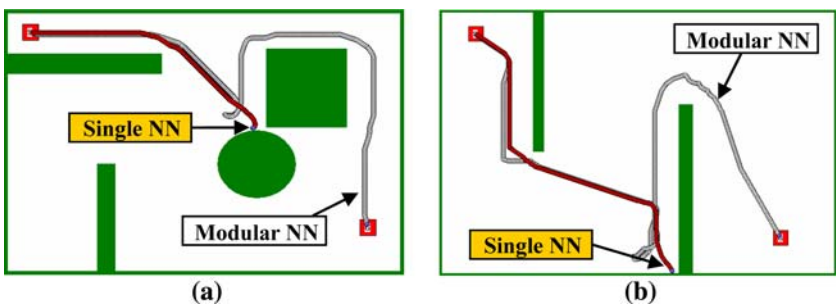
		Cost of oscillation	Cost of clearance	Cost of path length	Performance cost
Fig. 14a	Single NN	3.2	55.00	12.30	70.50
	MNN	1.80	67.82	13.65	83.27
Fig. 14b	Single NN	3.1	65.70	12.90	81.70
	MNN	4.10	58.42	13.70	76.22
Fig. 14c	Single NN	3.9	46.51	10.70	61.11
	MNN	2.30	44.61	11.05	57.96

Under the same obstacle configuration as in Fig. 14, we exchanged the start and goal positions and ran the experiment for ten times. The cost statistics were such that the mean and standard deviation of the cost function were 195 and 309, respectively, for a single NN while the same were 107 and 141, respectively, for the MNN. The MNN definitely has a more stable performance. Furthermore, when the robot was told to navigate a different environment containing a long obstacle type shown in Fig. 15, the MNN successfully reached the goal while the single NN failed due to the trap into a local minimum. The results of using competitive coordination and cooperative coordination in the same environment are compared in Fig. 16. While in competitive coordination, the robot failed to enter the door at the first approach, cooperative coordination succeeded by properly blending the outputs from the corridor module and the left turn module.

4.3 Real robot experiments

The proposed algorithm was finally applied to our Pioneer robot in Fig. 17a for verification. The testbed has been built on a commercial Active-Media Pioneer 2-DX platform with the differential drive type of wheels. The ultrasonic sensors read the range information at every 40 ms and send it to the navigation algorithm in a PC. It then computes the steering commands to the microcontroller via a serial port which then drives the motors. Thus, 40 ms is the control cycle time spent on sensing, environment classification, and command generation. The experiment utilized the result in Table 2 so that each control module utilizes only those sensor values that are most effective to handle the current environmental situation. This allows fast evolution and simple controller usage by removing those sensor values that do

Fig. 15 Comparison of the controlled trajectories between the single and the modular NN controllers



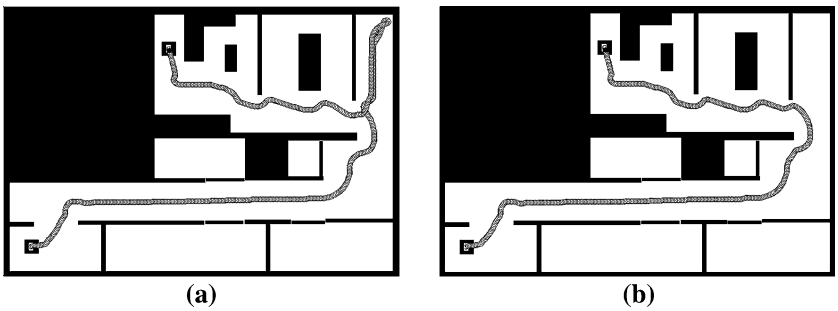


Fig. 16 Simulation result using the proposed MNN controller in our office environment using **a** competitive coordination, **b** cooperative coordination

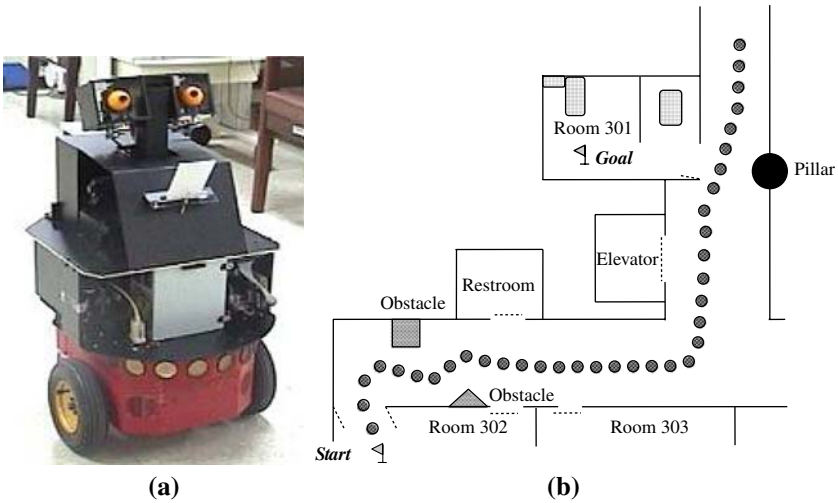


Fig. 17 Real world navigation result using the proposed MNN controller: **a** the robot hardware, **b** navigation path

not affect the current control effort. The navigation environment was chosen to be a path between Room 301 and Room 302 within our building as shown in Fig. 17b. Unlike the simulation result in Fig. 16, the robot could not enter Room 301 due to the limitation of our sensor model used. The door opening was not detected due to the narrow width of the door as well as the robot's oblique angle approach to the door. Other than missing the detection of the open door, the abilities of obstacle avoidance and goal seeking are well demonstrated.

5 Conclusion and future work

A MNN approach to the navigation control of mobile robots in rather complex daily environments has been presented. The NN was designed to optimize an arbitrary user-selected objective as a function of both the sensory connectivity and the control performance using both GA and EP. The selective usage of input sensors amounts to selective attention in human brain and may lead to more effective sensor utilization. Furthermore, in order to enhance the generalization capacity of a single NN, the MNN approach has been used. Finally, a cooperative module coordination algorithm blending several module outputs can indeed further enhance the navigation performance compared to competitive coordination where only one module is selected.

Although the cooperative method adopts a heuristic approach by taking a linear combination of the module outputs based on distances to the prototype environments, use of an EA to further optimize these weights may produce even better results in the future. In regard to a real robot experiment, the narrow door missing problem due to the limitation of the sonar sensors may be solved in the future by using a more accurate sensor model.

Although we think that our algorithm can also cover dynamic obstacles as it is, more experiments are needed to verify that. Finally, since this paper assumed that the goal position is precisely known a priori as in most research for reactive navigation, we could also create intermediate subgoals along the way in order to adapt the controller to many possible variants of the environment. By stopping at each subgoal and then reorienting toward the next subgoal, we could pass through the door that we failed to enter in our real

experiment. For practical application, however, some kind of goal shape based homing (like visual servoing) would be a viable alternative, since the dead reckoning is usually inaccurate.

Acknowledgments This research was supported by the Brain Neuroinformatics Research Program sponsored by Korean Ministry of Commerce, Industry and Energy and also in part by the Ministry of Education of Korea toward the Electrical and Computer Engineering Division at POSTECH through its BK21 program.

References

1. Lumelsky VJ, Stepanov AA (1987) Path-planning strategies for a point mobile automaton moving amidst obstacles of arbitrary shape. Algorithmica 2:403–430
2. Kamon I, Rivlin E (1997) Sensory-based motion panning with global proofs. IEEE Trans Robot Autom 13:814–822
3. Khatib O (1986) Real-time obstacle avoidance for manipulators and mobile robots. Int J Robot Res 5:90–98
4. Borenstein J, Koren Y (1989) Real-time obstacle avoidance for fast mobile robots. IEEE Trans Syst Man Cybern 19:1179–1187
5. Borenstein J, Koren Y (1991) The vector field histogram—fast obstacle avoidance for mobile robots. IEEE J Robot Autom 7:278–288
6. Im KY, Oh SY, Han SJ (2002) Evolving a modular neural network-based behavioral fusion using extended VFF and environment classification for mobile robot navigation. IEEE Trans Evol Comput 6:413–419
7. Na YK, Oh SY (2003) Hybrid control for autonomous mobile robot navigation using neural network based behavior modules and environment classification. Auton Robots 15:193–206
8. Chattergy R (1985) Some heuristics for the navigation of a robot. Int J Robot Res 4:59–66
9. Lumelsky VJ (1991) A comparative study on the path length performance of maze-searching and robot motion planning algorithms. IEEE Trans Robot Autom 7:57–66
10. Zalzala MS, Morris AS (1996) Neural networks for robotic control: theory and applications. Ellis Horwood, London New York
11. Omidvar O, Smagt P (1997) Neural systems for robotics. Academic, San Diego, London
12. Pomerleau DA (1991) Efficient training of artificial neural networks for autonomous navigation. Neural Comput 3:88–97
13. Tani J, Fukumura N (1994) Learning goal-directed sensory-based navigation of a mobile robot. Neural Netw 7:553–563
14. Zalama E, Gómez J, Paul M, Perán JR (2002) Adaptive behavior navigation of a mobile robot. IEEE Trans Syst Man Cybern 32:160–169
15. Berlanga AS, Isasi P, Molina JM (2002) Neural network controller against environment: a coevolutive approach to generalize robot navigation behavior. J Intell Robot Syst 33:139–166
16. Floreano D, Mondada F (1998) Evolutionary neurocontrollers for autonomous mobile robots. Neural Netw 11:1461–1478
17. Hoffmann F (2000) Soft computing techniques for the design of mobile robot behaviors. Int J Inf Sci 122:241–258
18. Miglino O, Lund HH, Nolfi S (1995) Evolving mobile robots in simulated and real environments. Artif Life 2:417–434
19. Nelson L, Grant E, Galeotti JM, Rhody S (2004) Maze exploration behaviors using an integrated evolutionary robotics environment. Robot Auton Syst 46:159–173
20. Odagiri R, Yu W, Asai T, Yamakawa O, Murase K (1998) Measuring the complexity of the real environment with evolutionary robot: evolution of a real mobile robot Khepera to have a minimal structure. In: IEEE World Congress on Computational Intelligence
21. Haykin S (1999) Neural networks: a comprehensive foundation. Prentice-Hall, New Jersey
22. Auda G, Kamel M (1999) Modular neural networks: a survey. Int J Neural Syst 9:129–151
23. Tou JT, Gonzalez RC (1974) Pattern recognition principles. Addison-Wesley, Reading
24. Bäck T, Fogel D, Michalewicz Z (1997) Handbook of evolutionary computation. Oxford University Press, London
25. Watanabe K, Hashem MMA (2004) Evolutionary computations: new algorithms and their applications to evolutionary robotics. Springer, Berlin New York