[17] W. J. Chung, Y. Nakamura, and O. J. Sørdalen, “Prototyping a nonholonomic manipulator,” in *Proc. IEEE Int. Conf. Robot. Automat.*, Nagoya, Japan, 1995, pp. 2029–2036.
[18] Y. Nakamura, “Integrability of dynamic constraints of underactuated mechanisms,” in *Proc. 6th Int. Symp. Robot. Res.*, Pittsburgh, PA, 1993, pp. 99–110.
[19] G. Oriolo and Y. Nakamura, “Free-joint manipulators: motion control under second-order nonholonomic constraints,” in *Proc. IEEE/RSJ Int. Wkshp. Intell. Robots and Syst.*, Osaka, Japan, 1991, pp. 1248–1253.
[20] ——, “Control of mechanical systems with second-order nonholonomic constraints: underactuated manipulators,” in *Proc. IEEE Conf. Decision Contr.*, Brighton, U.K., 1991.
[21] H. Arai and S. Tachi, “Position control of a manipulator with passive joints using dynamic coupling,” *IEEE Trans. Robot. Automat.*, vol. 7, pp. 528–534, 1991.
[22] Y. Nakamura, Y. Iwamoto, and R. Iwamoto, “Space multibody structure connected with free joints and its shape control,” in *Proc. IEEE Conf. Decision Contr.*, San Antonio, TX, 1993, pp. 3126–3131.
[23] D. Seto and J. Baillieul, “Control problems in super-articulated mechanical systems,” *IEEE Trans. Automat. Contr.*, vol. 39, pp. 2442–2453, 1994.
[24] K. Y. Wichlund, O. J. Sørdalen, and O. Egeland, “Control properties of underactuated vehicles,” in *Proc. IEEE Int. Conf. Robot. Automat.*, Nagoya, Japan, 1995, pp. 2009–2014.
[25] Y. Nakamura, R. Iwamoto, and K. Yoshimoto, “Control of nonholonomic mechanisms with drift,” *J. Robot. Soc. Japan*, vol. 13, no. 6, pp. 830–837, 1995.
[26] Y. Nakamura, T. Suzuki, and M. Koinuma, “Nonlinear behavior and control of underactuated robotic mechanisms,” in *7th Int. Symp. Robot. Res.* Oct. 1995, Herrsching am Ammersee, Germany, pp. 67–77.
[27] T. Suzuki, M. Koinuma, and Y. Nakamura, “Chaos and nonlinear control of a nonholonomic free-joint manipulator,” in *Proc. IEEE Int. Conf. Robot. Automat.*, Minneapolis, MN, 1996, pp. 2668–2693.
[28] H. Arai, “Controllability of a 3-DOF manipulator with a passive joint under a nonholonomic constraint,” in *Proc. IEEE Int. Conf. Robot. Automat.*, Minneapolis, MN, 1996, pp. 3707–3713.
[29] A. De Luca, R. Mattone, and G. Oriolo, “Dynamic mobility of redundant robots using end-effector commands,” in *Proc. IEEE Int. Conf. Robot. Automat.*, Minneapolis, MN, 1996, pp. 1760–1767.
[30] S. Wiggins, *Introduction to Applied Nonlinear Dynamical Systems and Chaos*. Splinger-Verlag, 1990.
[31] M. Tabor, *Chaos and Integrability in Nonlinear Dynamics—An Introduction*. New York: Wiley, 1989.

Environment Prediction for a Mobile Robot in a Dynamic Environment

Charles C. Chang and Kai-Tai Song

***Abstract*—The problem of navigating a mobile robot among moving obstacles is usually solved on the condition of knowing the velocity of obstacles. However, it is difficult to provide such information to a robot in real time. In this paper, we present an environment predictor that provides an estimate of future environment configuration by fusing multisensor data in real time. The predictor is implemented by an artificial neural network (ANN) trained using a relative-error-backpropagation (REBP) algorithm. The REBP algorithm enables the ANN to provide output data with a minimum relative error, which is better than conventional backpropagation (BP) algorithms in this prediction application. The mobile robot can, therefore, respond to anticipated changes in the environment. The performance is verified by prediction simulation and navigation experiments.**

***Index Terms*—Artificial neural networks, environment prediction, mobile robots, moving obstacles, training algorithm.**

I. Introduction

To be useful in the real world, a mobile robot should be able to navigate safely in an unstructured environment and accomplish given tasks despite unexpected changes in its surroundings. Many studies have been conducted on motion planning in uncertain but static environments [13]. In the real world, however, obstacles are not always stationary. When obstacles move, navigation methods designed for static environments do not work unless they adopt larger safety margins or shorter sampling times. These remedies may not be easy to realize and they are not efficient solutions. Therefore, new approaches are needed to resolve navigation problems that involve moving obstacles.

Some researchers proposed to plan a global trajectory before the robot moves. Not only a path to the goal is planned, but also the time that the robot would arrive at every intermediate position. One approach is to plan a possible path at first, then plan a trajectory or velocity by considering the trajectories of obstacles [10]. Another approach is to consider time an additional dimension and make a plan on space–time [6], [8], [17]. All these methods require that the robot knows obstacle motion trajectories before it makes a plan. Their applications will be limited, because the obstacle information cannot always be obtained from on-board sensor system and the obstacle motion may change after the robot moves.

Several methods have been proposed to use on-board sensors for handling immediate moving obstacles. In these methods, a mobile robot pre-plans a path to the goal; while traveling along this path, it refines its motion to deal with moving and unexpected obstacles according to sensing information. Different types of sensing information has been used in these methods. For instance, the navigation method

Manuscript received March 4, 1997. This work was supported by the National Science Council, Republic of China, under contract NSC84-2212-E-009-029. This paper was recommended for publication by Associate Editor M. Hebert and Editor V. Lumelsky.

C. C. Chang was with the Department of Control Engineering, National Chiao Tung University, Taiwan, R.O.C. He is now with the Industrial Technology Research Institute, Taiwan, R.O.C.

K.-T. Song is with the Department of Electrical and Control Engineering, National Chiao Tung University, Taiwan, R.O.C.

Publisher Item Identifier S 1042-296X(97)07797-5.

proposed by Sharma [16] simply needs the obstacle positions. The method developed by Kyriakopoulos and Saridis [11] must work with a precise prediction of collision time and position. The approaches used by Tsubouchi *et al.* [18] and Zhu [19] need precise position estimation of obstacles. In Fiorini and Shiller's method [7], the relative velocity information must be available.

All the methods mentioned above offer no experiments to verify their feasibility. Some of them need only current position information about obstacles. These methods can possibly be realized, but since the motion of obstacles is ignored, the performance may not be acceptable. Other methods may have difficulties in experiment because they do not have adequate sensor system to provide detail obstacle information. One exception involving experiments is the work done by De Lamadrid and Gini [12]. Their method, however, can only work in a slowly changing environment. The robot's speed and that of the obstacles in their experiment were slower than 2 cm/s. If an obstacle moves faster, the motion estimation error may be too large, forcing the robot to replan its trajectory frequently, therefore making real-time operation impossible.

We conclude from the methods surveyed above that currently available sensor systems can hardly provide reliable explicit obstacle motion estimates (i.e., their speeds and directions) in real time. An alternative may be to find implicit information, like next-time positions of obstacles. In this paper, we present a practical approach that allows mobile robots to predict implicit motion information of moving obstacles. We propose to use an artificial neural network (ANN) environment predictor to process ultrasonic sensor data for this purpose. The ANN predictor outputs what the future sensor readings will be, i.e., where obstacles will be relative to the robot if the robot and obstacle motion patterns stay the same. Using this information, the robot can respond to anticipated environmental changes in advance. This will result in better navigation performance than a system without motion information.

The prediction concept has been used in robot perception. The authors' previous work [3] proposed to use an ANN structure to process ultrasonic sensor data for environment prediction, but it was just a realization of a simple idea and the accuracy of prediction needed further improvement. Dickmanns *et al.* [5] proposed to use the Kalman filter approach for prediction in an active vision system. Image data are first processed to initialize a four-dimensional [4-D; three-dimensional (3-D) space plus time] object model. Then, exploiting dynamic models for those objects allows the prediction of object states. Baluja and Pomerleau [1] proposed an ANN structure for predicting the next-time input image to help the mobile robot focus its attention on the important features.

The ANN approach has also been used to steer a mobile robot according to sensory data. Biewald [2] trained several ANN's to steer an indoor mobile robot in different static-environment conditions. The simulation with 25 ultrasonic sensor inputs shows the generalization capability of the ANN approach. Pomerleau [14] developed a vision-based guidance system using a neural network to steer a road vehicle. Both the above methods are not aimed to deal with moving obstacles. In dynamic environments, the robot speed in addition to the steering should be taken into consideration. In this case, it would be very difficult to train an ANN for generating direct motion command based on sensor data from different time instants. This is because there are many adequate motion commands for a single condition.

Although the backpropagation (BP) [15] is the most used algorithm for training an ANN, new training methods can be beneficial for a particular application. For their classification problem, Hampshire and Waibel [9] proposed an objective function called classification figure of merit (CFM), which seeks to maximize the difference between the output activation of the node representing the correct classification and all other nodes (representing incorrect classifications). In this paper, we develop a relative-error-backpropagation (REBP) algorithm for training the ANN environment predictor. The REBP algorithm seeks to minimize the relative output error and is more appropriate for this application.

The rest of this paper is organized as follows. We present the structure of the ANN predictor and the REBP training algorithm in Section II. Section III shows the performance of the ANN predictor by simulation. Section IV shows the experimental results of using the ANN predictor for mobile robot navigation in a dynamic environment. Section V offers a discussion, and Section VI concludes the paper.

II. ANN Environment Predictor

Robots need to know obstacle motion information to effectively deal with moving obstacles. Most existing robot navigation methods [7], [8], [11] assume the trajectories (or the speeds and directions) of moving obstacles are known and apply algorithms to determine the action of the mobile robot. However, it is difficult to get this kind of information precisely in real time using on-board sensor systems and data-fusion techniques. Some navigation methods simply use the current sensor information to deal with moving obstacles. However, this does not consider the motion of obstacles, so navigation will not be very efficient. To provide the information about obstacle motion and meet the real-time requirement, we propose a simple but very practical strategy that provides motion information between the two extremes above—very explicit and none.

Many existing mobile robots are equipped with rangefinders for obstacle detection. Rangefinders such as ultrasonic sensors have difficulty identifying obstacles. Only the nearest distance to an object within the ultrasound beam is obtained, so the object's shape, size, and detected point are difficult to determine. As a result, the velocity calculation of obstacles will have large degrees of uncertainty. In our method given below, rangefinders are not used to identify the precise location of each obstacle. We only use the sensor system to tell the robot what area will be occupied by obstacles. This is an implicit form of considering the motion of obstacles. Thus, it is termed an environment predictor rather than a motion estimator. The environment predictor is possible if we succeed in fusing multiple sensor data to obtain the correlation among them. One possibility is to predict future sensor readings. Note that the measurement from any given sensor must have a relationship to historical measurements from sensors nearby if the motions of the robot and obstacles follow certain fixed patterns when sensory data are taken. If we can find this relationship, the sensor's future measurement can be predicted. Knowing the future measurements of rangefinders is equivalent to knowing what area will be occupied by obstacles. It is helpful for navigation among moving obstacles because the robot can respond to anticipated changes in the environment.

A. Structure of the ANN Predictor

In our design, a ring of equally-spaced ultrasonic transducers is used to detect obstacles. Although other types of sensors may also be suitable, we chose ultrasonic sensor mainly because of its simplicity and relative low cost. The environment predictor is actually constructed of multiple sensor predictors. Each of them predicts next-time measurement of a sensor according to historical sensory data. We can obtain such a sensor predictor if the relationship among measurements made at different instants can be formulated. However, it would be difficult to find a mathematical model for this relationship because of the following reasons. First, fusing multisensor data containing temporal relationship is usually very complex. Particularly

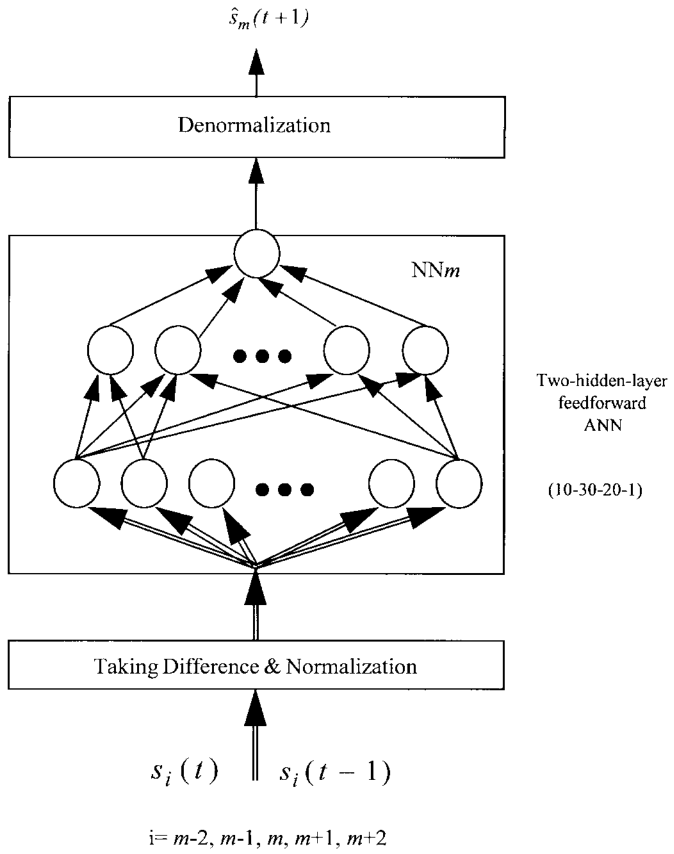

Fig. 1. Structure of the ANN sensor predictor.

in our application, the ultrasonic transducer has a cone shape of an emissive beam and an object may travel through the beams from one sensor to the other. Second, an obstacle may be detected by more than one sensor at the same time (especially when the beams of neighboring sensors overlap). This cannot be distinguished from the multiple-obstacle case. Moreover, obstacle size is not uniform. It is difficult to know how many sensors will detect the obstacle(s) at next instant. Therefore, the relationship among historical sensor measurements is uncertain. Third, it is impossible to locate an obstacle precisely because of the large beam opening angle and other sources of uncertainty associated with ultrasonic sensors. Fourth, multiple sensors are installed at different positions in different directions. The relation between data from different observations is nonlinear. Also, the sensor range limit will cause a prediction saturated, which is another source of nonlinearity. These conditions suggest that formulating predictions can take advantage of the ANN structure, which is model-free, suitable for nonlinear mapping problems, and able to deal with noisy data. Furthermore, the ANN can be realized with hardware and, therefore, the on-board computer can save data processing time.

In determining the structure of the ANN sensor predictor, we first have to clarify how it will be used and what the motion patterns are for the robot and obstacles. The predictor is used to predict future sensor readings by assuming the robot will not change its motion before the next-time instant. The assumption does not mean the robot motion has to stay the same to have a good prediction. In fact, after obtaining predictions, the robot determines its new motion immediately according to the predictions. The maximum direction-change rate of the robot is assumed to be 12.5°/s. Our consideration is as follows. First, we want the robot to behave like a human being who normally moves smoothly among obstacles. According to our observations, a person rarely takes a turn sharper than this limit. Second, this limit makes the robot motion tend rectilinearly, thus simplifying the prediction problem. Because no knowledge about the obstacles is available, we assume their motion is rectilinear and of constant speed, which is reasonable for most short-term observations. On these conditions, the last two sensor readings are sufficient to predict the next one. Fig. 1 shows the structure of the ANN sensor predictor. In the figure, $s_i(t)$ represents the data from sensor i at sample instant t; and the symbol m indicates this predictor is to predict sensor m. The inputs to the predictor are the historical measurements of sensors near the sensor whose reading in the next-time instant is being predicted. Before the measurements enter the ANN module, their difference must be taken [i.e., $s_i(t) - s_i(t-1)$], and both the current measurements and the differences are normalized to a value between zero and one. The larger the measured distance, the closer the value to one. The output of the ANN is the normalized sensor data prediction, which is also a value between zero and one. To convert it into the predicted sensor data, the denormalization module multiplies it by the maximum sensor range. The number of neighboring sensors considered in the predictor depends on the velocities of the robot and obstacles. It also depends on the sensor arrangement of the robot. For the case of a ring of 24 sensors, it is adequate to use seven sensors to predict one sensor reading. The environment predictor is therefore constructed of 24 ANN's (see Fig. 2). Because of the symmetrical sensor arrangement, only one ANN is trained for all 24 predictors.

B. Training Algorithm Minimizing Relative Output Error

In our previous work [3], the ANN was originally trained by the BP algorithm [15]. The BP algorithm propagates output errors back to previous layers and adjusts weights in each layer accordingly. Results with minimal output errors can be achieved in this way. We found the average prediction error for short-distance (within one meter) obstacle is around 48 cm, which was not acceptable. Considering this environment prediction problem, we thought a greater prediction error for distant obstacles could be tolerated because it would not be dangerous in this situation. Any navigation deviation resulting from this error can be compensated for later. On the other hand, the allowable prediction error for nearby obstacles is more important and should be kept smaller because inappropriate navigation can cause a collision in this situation. For this reason, we figured out the conventional BP algorithm, which minimizes the prediction errors for both distant and nearby obstacles may not be the best training scheme. To meet the problem characteristics as described above, minimizing the relative output error [*(output error)/(desired output)*] is more suitable. Therefore, we developed a training algorithm that uses relative output errors as feedback to update the weights. The following is the derivation of the training algorithm that minimizes relative output errors. This algorithm is termed REBP, which is an abbreviation of "relative error backpropagation."

Suppose the ANN has N-layers (including input and output layers). Denote $A_{n,j}$ as the output of the jth neuron of the nth layer, then

$$A_{n,j} = f(net_{n,j}) \tag{1}$$

where $f(\ )$ is the activation function and $net_{n,j}$ is a summation function. We adopt the activation function as

$$f(x) = \frac{1}{1+e^{-x}} \tag{2}$$

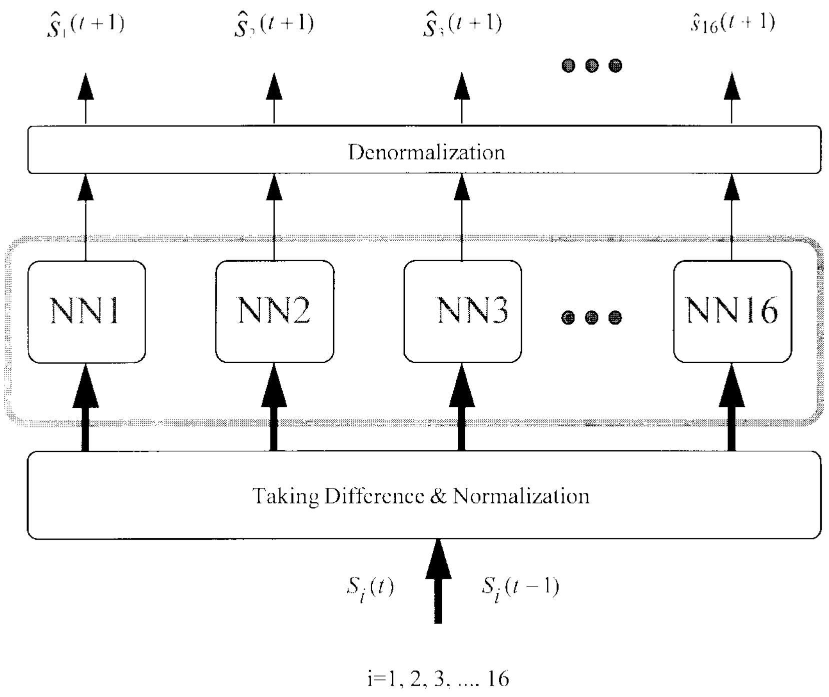

Fig. 2. Environment predictor for the mobile robot.

and the summation function is

$$net_{n,j} = \sum_i W_{n,ij} A_{n-1,i} - \theta_{n,j} \tag{3}$$

where $W_{n,ij}$ is the weight between the ith neuron of the $(n-1)$th layer and the jth neuron of the nth layer; $\theta_{n,j}$ is the bias for the jth neuron of the nth layer.

Define error function E as

$$E \triangleq \frac{1}{2} \sum_j [(T_{N,j} - A_{N,j})/T_{N,j}]^2 \tag{4}$$

where $A_{N,j}$ and $T_{N,j}$ are the actual and desired outputs of the jth neuron of the output layer. We can achieve the minimal relative error by minimizing E. To minimize E we adopt the gradient-descent method to adjust weights. The weights are updated according to the following formulas:

$$\triangle W_{n,ij} = \eta \delta_{n,j} A_{n-1,i} \tag{5}$$
$$\triangle \theta_{n,j} = -\eta \delta_{n,j} \tag{6}$$

where η is the learning rate and $\delta_{n,j}$ is for hidden layers

$$\delta_{n,j} = \left[\sum_k \delta_{n+1,k} W_{n,jk}\right] H_{n,j}(1 - H_{n,j}) \tag{7}$$

($H_{n,j}$ is the jth neuron output of the nth layer); for the output (Nth) layer

$$\delta_{N,j} = \frac{(T_{N,j} - Y_j)Y_j(1 - Y_j)}{T_{N,j}^2} \tag{8}$$

(Y_j is the jth neuron output of the output layer).

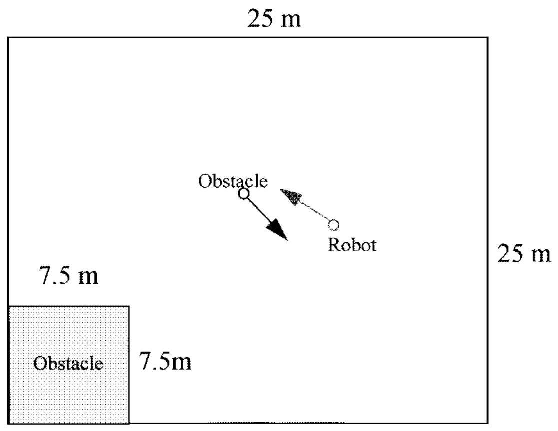

Fig. 3. Environment for generating the training data.

C. Training the ANN Predictor

The ANN predictor is trained off line. The sensor data for training are generated by an ultrasonic sensor model. The sensor model and related environmental considerations for generating training data are summarized as follows.

1) The ultrasonic sensor has a cone-shaped beam angle of 22.5° and an obstacle can be detected within this angle at any distance between 40–500 cm. The sensing range is determined according to the navigation requirement.
2) Movable obstacles are assumed to have rough surfaces so that specular reflection will not occur, i.e., moving obstacles can always be detected.
3) Static obstacles have two types. One is like walls, docking stations, etc. They are assumed to be of smooth plane surface. When the ultrasonic wave's incident angle is beyond 23°, specular reflection will occur and the sensor cannot detect this type of obstacle. The other type of static obstacle is zero-velocity movable obstacles.

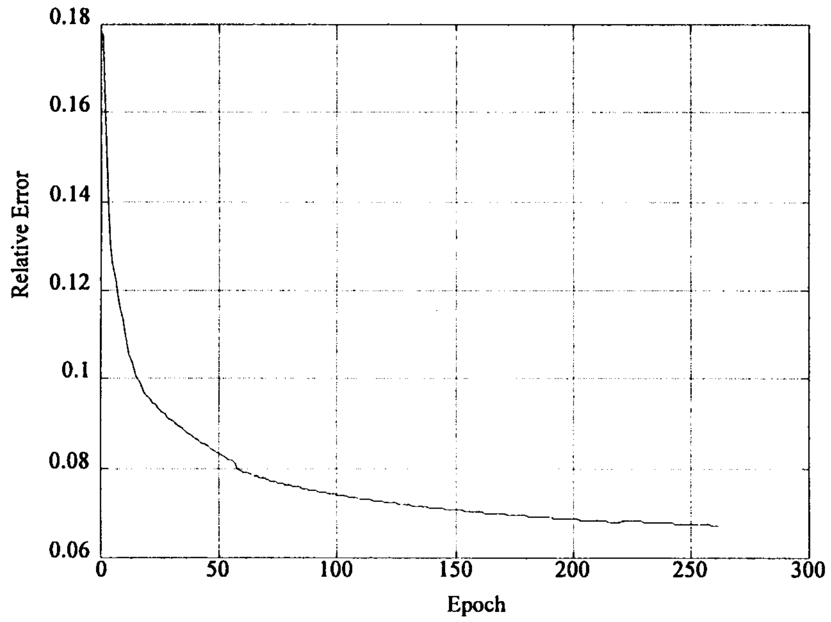

Fig. 4. Error decay curve during training process.

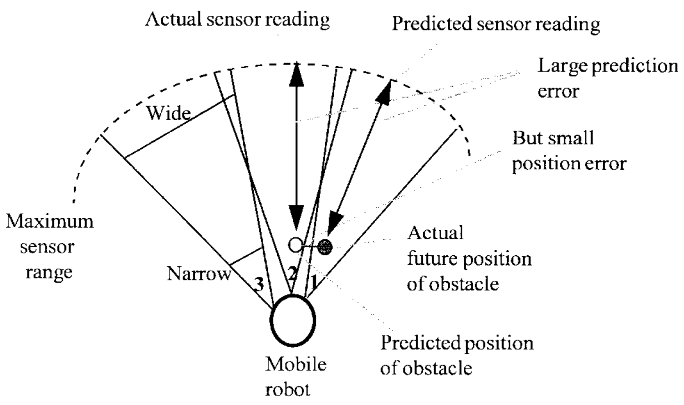

Fig. 5. Explanation of excessive prediction error.

4) The robot is cylindrical and has a diameter of 60 cm, while the moving obstacles are also cylindrical[1] but have a diameter of 40 cm.
5) The maximum speeds of robot and obstacles are 200 cm/s.
6) The robot is assumed to take sensor data every 500 ms. The measurement time of sensors is assumed so short that can be neglected.

With the above considerations, the training data are generated randomly according to the following procedure.

Step 1.) The robot is placed at a random location in a 25 m $\times$ 25 m-square room; a 7.5 m $\times$ 7.5 m object occupies one corner (Fig. 3). This environment contains convex and concave corners—typical environmental features for indoor robots.

Step 2.) In this step, two alternatives are used by turns. For odd turns, a moving obstacle randomly positioned (more obstacles are not needed, as explained later) within the range of the robot's sensors. This is for the robot to learn the motions of movable obstacles. For even turns, there is no moving obstacles, so the sensor detects a plane wall. This is for learning the relative motion of a static object.

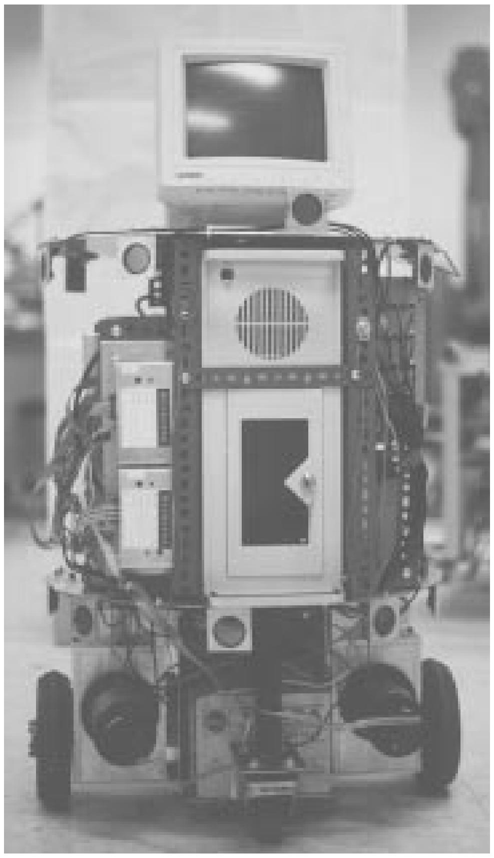

Fig. 6. Picture of the experimental mobile robot.

[1]The obstacle shape is not important, and the ultrasonic sensor cannot distinguish the shapes. What we are concerned with in the prediction is the nearest point of an obstacle. However, we need an obstacle model to generate the training data. Cylindrical shape is simple and good for this purpose.

TABLE I
COMPARISON OF REBP AND CONVENTIONAL BP METHOD

Environment conditions	Error(cm) using current data	Error(cm) using linear predictor	Error(cm) using conventional BP	Error(cm) using REBP	ER using conventional BP	ER using REBP
Vr: 20-200 (cm/sec) Vo: 20-200	37.9	33.5	27.5	21.4	37.8%	77.1%
Vr: 20-80	14.8	9.6	8.2	6.6	80.5%	124.2%
Vr: 20-80 Vo: 20-80	24.5	20.3	24.4	17.8	0.5%	37.6%
Vr: 80-140	16.1	8.7	8.3	7.1	94.0%	126.8%
Vr: 80-140 Vo: 80-140	33.9	27.5	25.7	19.6	32.0%	73.0%
Vr: 140-200	18.4	10.7	9.7	8.1	89.7%	127.2%
Vr: 140-200 Vo: 140-200	45.8	41.6	31.8	26.0	44.0%	76.2%

Step 3.) The velocities of the robot and obstacle are set at random within their speed limits.

Step 4.) Calculate sensor data for five sample instants. Taking the same number of samples for each situation allows the trained ANN to make roughly equally good predictions in all situations.

Step 5.) Repeat Steps 1) through 4) two hundred times to cover the problem space.

After several trials, we determined to adopt a two-hidden-layer network as the sensor predictor. The first hidden layer had 30 processing elements; the second hidden layer had 20. The whole training data for the ANN consisted of 22 000 sets of sensor readings. Initial weights were randomly set between 1 and -1. During the training, every set of training data was put into the ANN and then the weights were adjusted accordingly. All the training data were used cyclically until the output error of the ANN converged. Since the initial weights affect the results because of the local-error-minimum problems, several sets of initial weights had to be tried. It took about 14 h CPU time on a SUN SPARC 2 workstation to train the ANN to convergence. Fig. 4 shows the error decay curve of the training process. In the figure, the relative error (RE) is defined as

$$\mathrm{RE} = \sum_{M} \frac{|d_a - d_p|}{d_a} / M \tag{9}$$

where d_a is the actual distance, d_p is the predicted distance, and M is the number of sample data in the training set.

III. SIMULATION OF THE ANN PREDICTOR

After the training phase, the ANN predictor was tested under various environment conditions. The test data were generated by using the same procedure as that of generating the training data. Table I summarized the simulation results of environment prediction. We compared the prediction error computed by using the REBP ANN predictor, using the conventional BP ANN predictor, using a simple linear predictor, and using the current sensor data to predict the next one. The linear predictor is formulated by the following equation:

$$\hat{s}_i(t+1) = s_i(t) + [s_i(t) - s_i(t-1)] \tag{10}$$

where $\hat{s}_i(t+1)$ is the prediction of sensor i at time $t+1$ and the value is limited to the effective sensor range. Using the current sensor data means using the present data as the prediction for the next-time instant measurements. In Table I, the environment conditions are distinguished by the robot speed (V_r) and the obstacle speed (V_o). If there is no moving obstacle then no obstacle speed is listed. Using the linear predictor can obtain better predictions than using the current sensor data. However, the linear predictor does not consider the correlation between sensors, so the prediction performance is limited. Let us focus on the conventional BP ANN and the REBP ANN predictors. Both improved the predictions much more than the linear predictor did. Define error reduction (ER) as

$$\mathrm{ER} = \frac{|E_p - E_c|}{E_p} \tag{11}$$

TABLE II
COMPARISON OF REBP AND CONVENTIONAL BP METHOD AT DIFFERENT DISTANCES

Next-time distance range (cm)	Error(cm) using current data	Error(cm) using general BP	Error(cm) using REBP
40 - 100	70.3	48.3	12.1
100 - 200	46.1	32.1	14.7
200 - 300	35.7	26.5	17.8
300 - 400	35.5	24.9	24.3
400 - 500	38.0	27.5	28.4

TABLE III
COMPARISON OF CASES WITH DIFFERENT NUMBER OF OBSTACLES

Learning condition	Error(cm) without prediction	Error(cm) by ANN learned from cond. (1)	Error(cm) by ANN learned from cond. (2)	Error(cm) by ANN learned from cond. (3)
(1) 1 obstacle	37.9	22.1	25.1	25.2
(2) 2 obstacles	43.1	31.5	31.9	32.1
(3) 3 obstacles	45.8	33.1	33.7	34.5

where E_p is the prediction error using an ANN predictor and E_c is the error using the current data. In the general environment (V_r: 20–200 cm/s; V_o: 20–200 cm/s), the error reduction is 77.1% on using the REBP ANN and 37.8% on using conventional BP ANN. Also the error reductions of both ANN predictors are better in environments that change faster (see the last two columns of the fourth, sixth, and eighth rows in Table I). Finally, it can be seen that the REBP ANN predictor performs better than the conventional BP ANN predictor in every environment condition in which we tested.

Table II shows comparison of errors at different prediction distances. It was obtained by classifying the test data according to the distance range to be predicted. It can be seen that the short-distance predictions made by the REBP ANN are much better than those made by the conventional BP ANN. Although the REBP algorithm does not improve much in the long-distance predictions, the relative error is already small and acceptable. According to the error functions used, the prediction error using BP ANN and the relative error using the REBP ANN should be similar at each distance. However, from the test results shown in Table II, one can find that the errors are larger than expected when the predicted distance is smaller than 200 cm. The reason for this phenomenon is that the ultrasonic wave beam is narrow over short distances. This can easily result in erroneous sensor predictions: obstacles are mispredicted by neighboring sensors. As shown in Fig. 5, sensor 1 should detect a short-distance obstacle at next sample instant, but it predicts no obstacle and has a maximum-sensor-range prediction. Therefore, the sensor prediction error is large. In this particular situation, the obstacle is predicted instead by the neighboring predictor, i.e., sensor predictor 2. Sensor predictor 2 also has a large prediction error because it should predict no obstacle. These errors, which are apt to occur over short distances, result in larger (relative) prediction errors over near-distance ranges. Although the (relative) prediction errors for both the sensors are large, the error of obstacle prediction is not that large because the actual obstacle position is just beside the predicted one. This type of erroneous prediction does not affect the navigation much.

From the above simulation results, we conclude that both ANN predictors improve the predictions. However, the ANN predictor trained by the REBP algorithm is better than the conventional BP ANN in this particular application. Therefore, minimizing the relative output error of the ANN is more suitable.

During the training of the ANN predictor, we considered cases with only one moving obstacle. Therefore, the relating sensory data at different sample instants are relatively simple. If there were more obstacles within the sensing range of an ANN sensor predictor, the relating sensory data would be more complex and tend toward ambiguity. We now consider the prediction results with the data collected in the environment where more than one moving obstacle exists. In addition to the ANN predictor trained with one moving obstacle, we trained ANN predictors with two and three obstacles, respectively. We proceeded to generate three sets of test data, with one, two, or three obstacles in the environment. Table III is a comparison of test results obtained by using each of these three ANN predictors on each of the data sets in turn. It can be seen that the ANN trained with one moving obstacle performs better than the other two

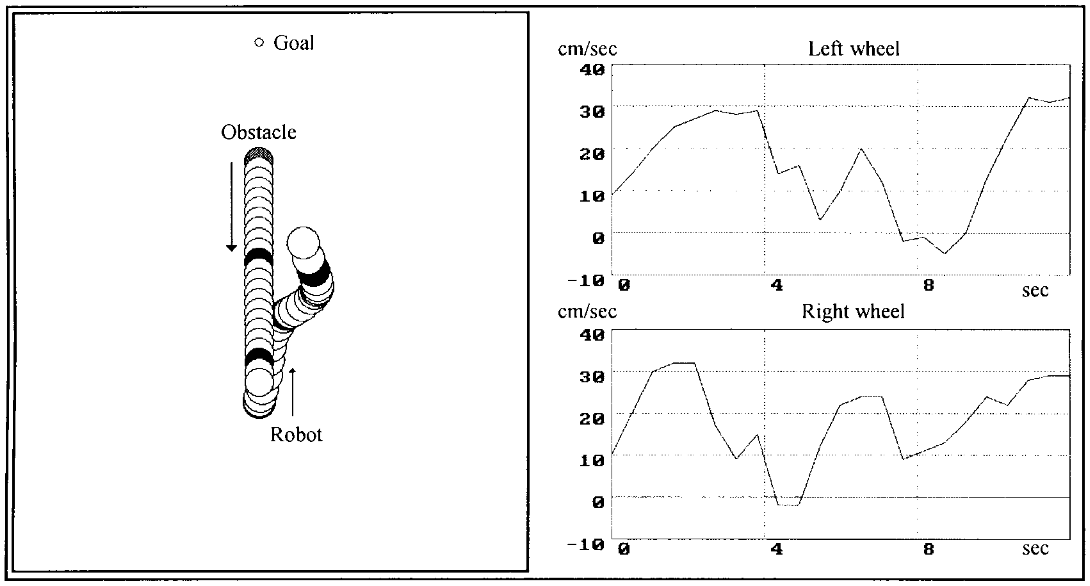

Fig. 7. An experimental result of dynamic navigation with one moving obstacle approaching the robot head on. (The obstacle started to move from sample instant zero at the speed of 27 cm/s.)

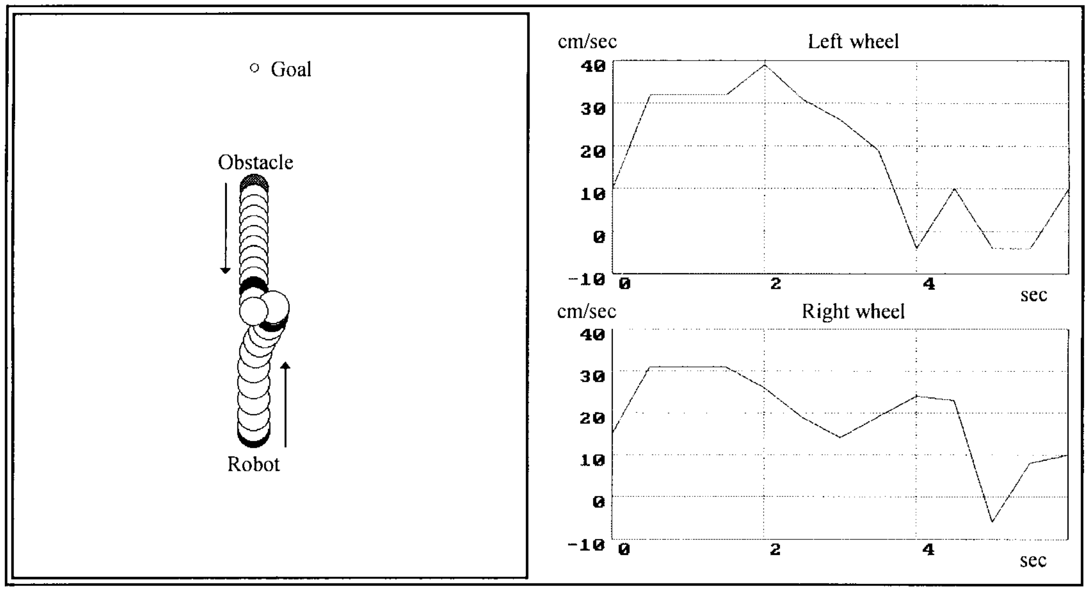

Fig. 8. An experimental result showing a collision occurs if without the predictor. (The obstacle started to move from sample instant zero at the speed of 27 cm/s.)

on all sets of test data. The reason for this phenomenon can be that too many moving obstacles create excessive ambiguity for the ANN to learn effectively. Ambiguity here means the difficulty in identifying which two sensory data at different sample instants are caused by the same obstacle. Thus, training data from cases in which there are more moving obstacles will only deteriorate learning. Therefore, we concentrated on only one moving obstacle when training the ANN predictor. Table III also indicates that no matter with which training situation, prediction will be worse when more obstacles are cluttered together (i.e., multiple obstacles are within the prediction range of a single sensor). This is unavoidable when using relatively inaccurate sensors. However, this does not necessarily mean the prediction will be worse when there are more obstacles around the robot. If obstacles are not in the same predictor range, the prediction will not be affected. If several obstacles move together as a group, they can be regarded as one obstacle and do not affect the prediction either. Problems occur when two or more obstacles are in the beam opening angle of the same sensor. In this case, inaccurate predictions may occur because only the nearest obstacle can be detected by the ultrasonic sensor.

IV. Experimental Results

Navigation experiments have been carried out to show that a mobile robot using the predictor indeed navigates better than that without the predictor. They also demonstrate mobile robot navigation using the predictor in a dynamic environment. Fig. 6 is a picture of the experimental mobile robot developed in our laboratory. It has two

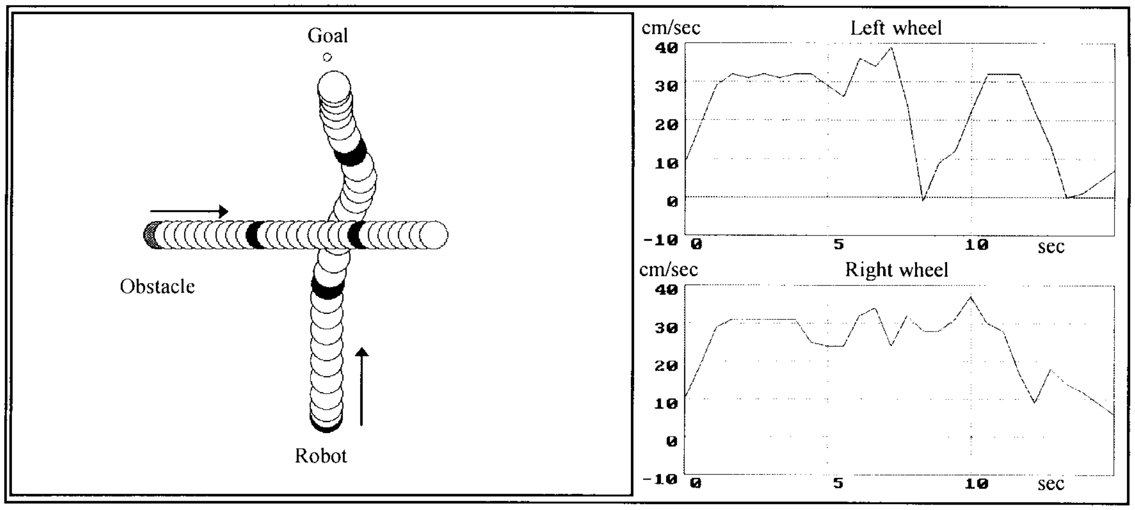

Fig. 9. Navigation experiment of avoiding a slow obstacle. (The obstacle started to move from sample instant zero at the speed of 27 cm/s.)

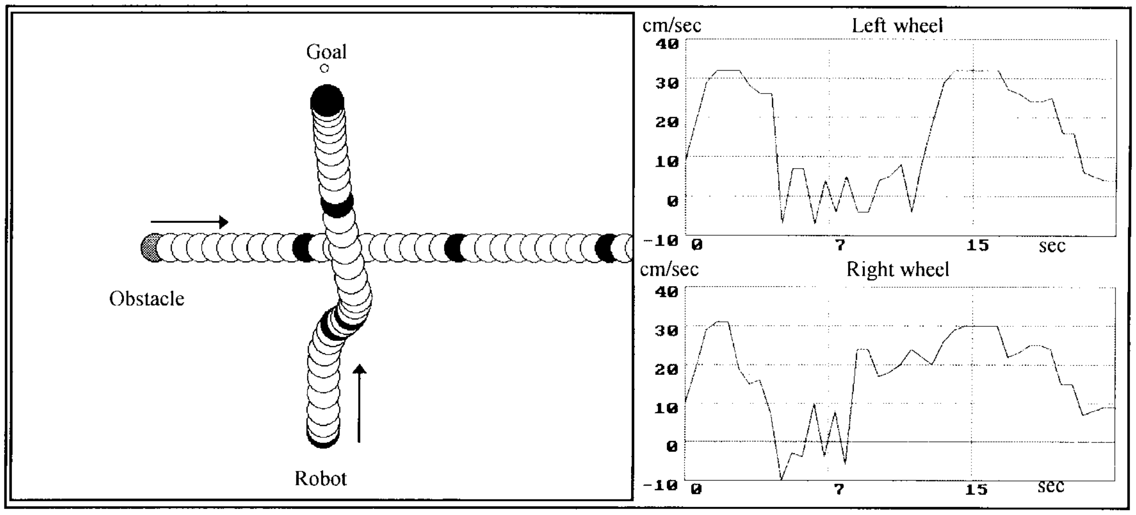

Fig. 10. Navigation experiment of avoiding a faster obstacle. (The obstacle started to move from sample instant zero at the speed of 40 cm/s.)

driving wheels and two casters for balance. Its maximum speed is 40 cm/s. A ring of 16 ultrasonic sensors is installed alternating at heights of 30–60 cm with equal angle spans. The effective range of each sensor is 43–300 cm. A cycle of all 16 ultrasonic sensor measurements takes 150 ms. The sample time for navigation command is 560 ms. Because the angle span is larger than that used in the previous simulation, the ANN predictor for a single sensor was modified to take the input data from five neighboring sensors. It should be noted that navigation algorithms utilizing range data can benefit from the predictor if a proper design is applied. Here we employed the algorithm proposed in [4], which directly uses the predicted information and goal position to generate motion commands. In the experiment, we tested with people as the moving obstacles.

Figs. 7 and 8 are the experimental results with and without the environment predictor, respectively. In each figure, there are two parts shown: the trajectory window (left), and the wheel-velocity window (right). In the trajectory windows, we recorded the trajectories by putting a circle at each sample command time. Every tenth instant, the circle is depicted in black. The small circles represent the trajectories of moving obstacles; the big circles represent the robot. In Fig. 7 the obstacle was predicted to endanger the robot at the third sample instant, so it turned right to avoid the obstacle and then safely moved toward the goal. On the contrary, in Fig. 8 the robot began to turn right at the fourth sample instant. Without the predictor, it not only turned right too late but also turned too little. Therefore, it could not avoid the approaching moving obstacle. To guarantee the robot's safety without the predictor, it has to consider all possible movement that obstacles may have because the robot does not know the motions of obstacles. Therefore, the robot must use a navigation algorithm with a large safe margin, which usually results in unsuitable avoidance.

Figs. 9–11 show the predictor provides useful information to the robot for avoiding obstacles with different speeds. In these three experiments, an obstacle coming from left would cross the robot path. Although the robot at first all predicted an obstacle at the front left and turned right, it generated a suitable motion according to each situation. When the obstacle was slow, the robot passed it in front of it (Fig. 9). When the obstacle was a little bit faster, the robot predicted the obstacle earlier and slowed down its speed. Therefore, the robot

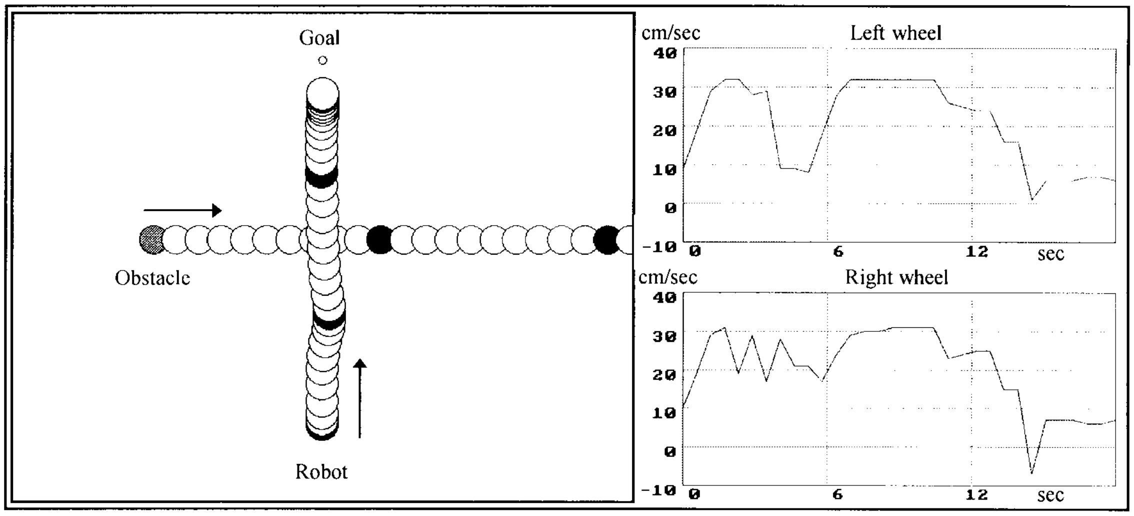

Fig. 11. Navigation experiment of avoiding a very fast obstacle. (The obstacle started to move from sample instant zero at the speed of 60 cm/s.)

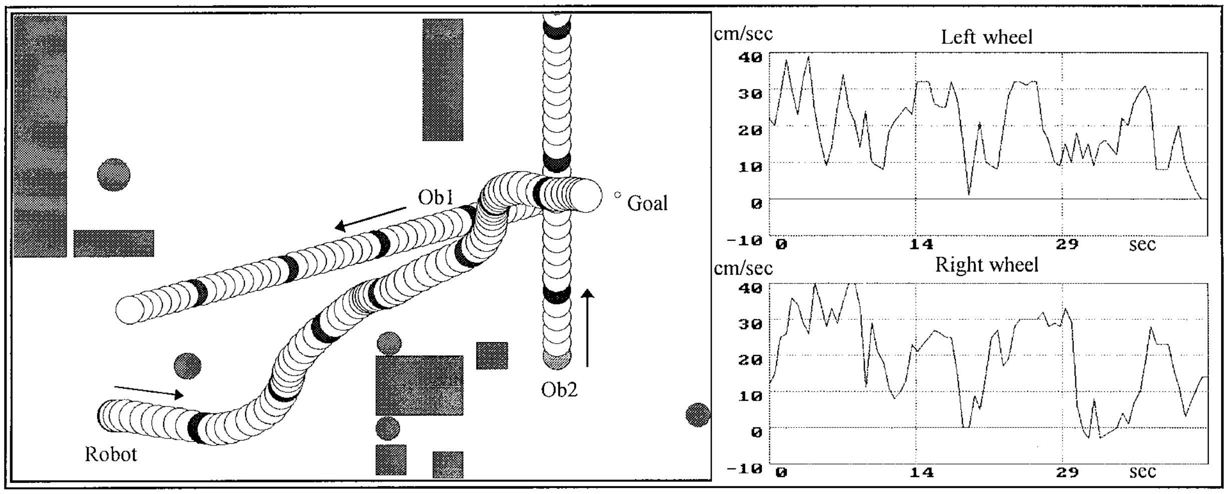

Fig. 12. Navigation experiment with static and moving obstacles. (Obstacle ob1 started to move from sample instant 30 at the speed of 25 cm/s; Obstacle ob2 started to move from sample instant 45 at the speed of 35 cm/s.)

was not able to pass the obstacle in front of it. As shown in Fig. 10, at the 14th sample instant, the robot predicted the obstacle would be on its front right, so it turned left and passed over behind the obstacle. When the obstacle moved even faster, as shown in Fig. 11, the robot only first slowed down a little and went to the goal along an almost straight path. These experimental results show that the robot can be more adaptable to obstacle speed when using the predictor.

Fig. 12 shows the robot can deal with static obstacles as well as moving obstacles using the ANN predictor. In the experiment, the robot at first moved forward right because of the static obstacle (a person) in front of it. After the robot passed the person, it turned left to avoid other obstacles (they were a rectangle carton, two round chairs, and a stopped rectangle mobile robot). At the 35th sample instant, the robot predicted the moving obstacle *ob1* (a person). However, because there was another static obstacle on the right, the robot could only decrease its speed and slightly turn right. Note that if the robot did not use the predictor and thus applied a large safe margin, it could not move toward the goal. On the contrary, it would move backward to avoid the obstacle, which will not be acceptable. At the 50th sample instant, the robot predicted another moving obstacle *ob2* (another person). In this case, the robot could not pass before *ob2*. Therefore, it slowed down and turned left. Until at the 56th sample instant, it predicted that *ob2* would not block the path to the goal, so it began to turn right. Because it was already near the goal, it moved slowly toward the goal. If without the predictor, the robot will employ a large safe margin, then it would turn away from the goal when met *ob2*. The mobile robot is very difficult to accomplish its task in this case.

V. Discussions

To mimic humans' behavior and simplify the predictor design, we assumed that the robot has a limit of the maximum direction-change rate of 12.5°/s. From the general navigation tests, as shown in Fig. 12, it can be seen that this assumption holds for almost any three consecutive positions, even though the complete recorded path is very curvy. To consider the mobile robot may have sharper maneuvering, we may use the wheel velocities as extra inputs to the ANN predictor. In this case, the nonlinear part of the robot motion can also be learned by the ANN, and the sensor data from only two time instants are still sufficient to predict next-time sensor data. According to our test results, using this kind of ANN structure has advantage over using no-wheel-velocity one when the robot takes sharp turns often. However, if the robot does not take sharp turns, the additional learning space will not benefit the predictions, but on the contrary, it deteriorates them because the ANN has to adapt its weights to many different situations. We have used the ANN with the wheel-velocity input to learn the case as shown in Section II, where the robot motion was not very curvy. It made the prediction error decrease by only 0.87 cm. Therefore, we adopted the present ANN structure to simplify the calculation and reduce the burden of the on-board computer.

There is one more point that can be realized to improve the prediction. When training the ANN predictor, because of the symmetrical sensor arrangement we used data from all sensors to train one ANN and this ANN was used as the predictor for each sensor. However, our experimental mobile robot is equipped with two independent drive wheels, and its motion is not omnidirectional. Therefore, the sensors are not completely symmetrical in terms of motion and the workspace for each sensor is different. In this situation, training only one ANN with data from all sensors has a larger learning space than training an ANN for each sensor with separately local data. In practice, a larger learning space will slightly affect the mapping accuracy. Therefore, separately training each ANN predictor will be better. We have made a simulation to verify this design. The 24 sensors were divided into seven groups according to the motion symmetry. There was an ANN trained for each group of sensors. The prediction error indeed decreased, but by only 0.53 cm on average.

VI. Concluding Remarks

The obstacle motion information is of essential importance for mobile robots in a dynamic environment. However, most of the existing research on the dynamic navigation did not consider the capacity of the present sensor system. We develop a real-time environment predictor, which provides dynamic environment information in acceptable detail. The prediction information consists of next-time measurements of ultrasonic rangefinders, which is equivalent to knowing the relative future positions of obstacles. The predictor takes advantage of an ANN structure to fuse multiple ultrasonic sensor data. For navigation application, a greater prediction error for distant obstacles could be tolerated; the allowable prediction error for nearby obstacles is more important and should be kept smaller. Therefore, minimizing relative prediction error is more suitable than minimizing prediction error itself. We developed a learning algorithm termed relative-error-backpropagation (REBP) algorithm, which is useful for problems in which minimizing relative errors is more meaningful. Simulation results show that, for short-distance prediction, the prediction error in using the REBP algorithm is only about 25% of that in using the conventional BP algorithm. Navigation experiments demonstrate that the predictor is indeed useful for dynamic navigation. However, because the predictor only provides motion information implicitly, the robot cannot have a long-term navigation strategy. The future work is to expand this method to involve more delicate sensors, such as CCD cameras so that it can be applied in multirobot navigation in complex environments.

References

[1] S. Baluja and D. A. Pomerleau, "Using the representation in a neural network's hidden layer for task-specific focus of attention," in *Proc. Int. Joint Conf. Artif. Intell.'95*, Montreal, Canada, Aug. 1995, pp. 133–139.

[2] R. Biewald, "A neural network controller for the navigation and obstacle avoidance of a mobile robot," in *Neural Network for Robotic Control*, A. Zalzala and A. Morris, Eds. Singapore: Ellis Horwood, 1996, pp. 162–191.

[3] C. C. Chang and K. T. Song, "A neural network predictor for mobile robot navigation among moving obstacles," in *Proc. Int. Symp. Artif. Neural Networks'94*, Tainan, Taiwan, Dec. 1994, pp. 722–727.

[4] C. C. Chang and K. T. Song, "Dynamic motion planning based on real-time obstacle prediction," in *Proc. IEEE Int. Conf. Robot. Automat.*, Minneapolis, MN, Apr. 1996, pp. 2402–2407.

[5] E. D. Dickmanns, B. Mysliwetz, and T. Christians, "An integrated spatio-temporal approach to automatic visual guidance of autonomous vehicles," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, no. 6, pp. 1273–1284, Nov./Dec. 1990.

[6] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," *Algorithmica*, vol. 2, no. 4, pp. 477–522, 1987.

[7] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using the relative velocity paradigm," in *Proc. IEEE Int. Conf. Robot. Automat.*, Atlanta, GA, May 1993, pp. 560–565.

[8] K. Fujimura and H. Samet, "A hierarchical strategy for path planning among moving obstacles," *IEEE Trans. Robot. Automat.*, vol. 5, no. 1, pp. 61–69, Feb. 1989.

[9] J. B. Hamshire and A. H. Waibel, "A novel objective function for improved phoneme recognition using time-delay neural networks," CMU-CS-89-118, 1989.

[10] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *Int. J. Robot. Res.*, vol. 5, no. 3, pp. 72–89, 1986.

[11] K. J. Kyriakopoulos and G. N. Saridis, "An integrated collision prediction and avoidance scheme for mobile robots in nonstationary environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, Nice, France, 1992, pp. 194–199.

[12] J. F. G. de Lamadrid and N. L. Gini, "Path tracking through uncharted moving obstacles," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, no. 6, pp. 1408–1422, Nov./Dec. 1990.

[13] J. Latombe, *Robot Motion Planning.* Boston, MA: Kluwer, 1991.

[14] D. A. Pomerleau, *Neural Network Perception for Mobile Robot Guidance.* Boston, MA: Kluwer, 1993.

[15] D. E. Rumelhart and J. L. McClelland, Eds., *Parallel Distributed Processing, Vol. 1.* Cambridge, MA: MIT Press, 1986, pp. 318–362.

[16] R. Sharma, "Safe motion planning for a robot in a dynamic, uncertain environment," in *Proc. IEEE Int. Conf. Robot. Automat.*, Sacramento, CA, 1991, pp. 438–443.

[17] C. L. Shih, T. Lee, and W. A. Gruver, "A unified approach for robot motion planning with moving polyhedral obstacles," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, no. 4, pp. 903–915, July/Aug. 1990.

[18] T. Tsubouchi, K. Hiraoka, and T. Naniwa, "A mobile robot navigation scheme for an environment with multiple moving obstacles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot and Systems'92*, Raleigh, NC, 1992, pp. 1791–1798.

[19] Q. Zhu, "Hidden Markov model for dynamic obstacle avoidance of mobile robot navigation," *IEEE Trans. Robot. Automat.*, vol. 7, no. 3, pp. 390–397, June 1991.