Collision Avoidance by Fuzzy Logic Control for Automated Guided Vehicle Navigation

Pai-Shih Lee and Ling-Ling Wang*
Institute of Computer Science
National Tsing Hua University
Hsinchu, Taiwan 30050 R.O.C.

Received January 10, 1994; revised June 6, 1994; accepted July 5, 1994

A fuzzy approach to collision avoidance for automated guided vehicle (AGV) navigation is proposed. Static obstacles with no a priori position information as well as moving obstacles with unknown trajectories are considered in this study. Intuitive and subjective human ideas of collision avoidance are modeled into fuzzy rules. Fuzzy logic is applied in the inference procedure for AGV navigation, such that the AGV is guided from the starting point toward the target without colliding with obstacles. Furthermore, the proposed method can also be used for the navigation of multiple AGVs, where each AGV must avoid other AGVs as well as obstacles in the environment. Simulation results are presented to show the feasibility of the proposed fuzzy approach. © *1994 John Wiley & Sons, Inc.*

自動誘導車両（AGV）の運行における衝突回避を実現するための、ファジー理論を提案する。この研究では、事前情報の無い制止物体と共に、未知の軌道を持った移動物体についても考慮している。衝突回避に使われる人間の直感的および主観的思考を、ファジー理論を使ってモデル化している。AGVを出発点から目標に向かって障害物を避けながら誘導する場合などに、AGVの運行における推論手順にこのファジー理論が応用される。さらに、ここで提案する方法は、複数のAGVの運行にも使える。この場合、各AGVは他のAGVまたはその環境中の障害物と衝突してはならない。シミュレーションの結果を使い、提案したファジー理論の可能性を示す。

*To whom all correspondence should be addressed.

1. INTRODUCTION

Current research and development of automated guided vehicles (AGVs) have attracted the attention of researchers in the areas of engineering, computer science, biology, psychology, locomotion, and others. This is due to the great application potential of AGVs. Possible applications include automatic freeway driving,[1] guidance of the blind and disabled,[2] security guarding, tourist guiding for sight seeing, exploration of dangerous regions, document transfer, etc.

For vehicles to navigate automatically in an environment, an important factor is to prevent the vehicle from colliding with obstacles. Planning a collision-free path among obstacles is one of the fundamental requirements for AGV navigation. Many previous works are concerned with methods for generating a path in a known environment.[3-9] Some of these algorithms have the ability to find an optimal path among many feasible paths. However, to provide more flexibility and autonomy to the AGV system, motion planning in the presence of unknown static obstacles is needed. Although in such a case it is difficult to find an optimal path, it is more desirable because of its suitability in a real situation.

One approach to motion planning among unknown static obstacles is the virtual force field method.[10] This method uses a two-dimensional Cartesian histogram grid as a world model, which is updated continuously with range data sampled by on-board range sensors. The virtual repulsion forces generated from the obstacles and the virtual attractive force generated from the target are obtained according to the histogram grid. The AGV is pushed away from the obstacles by the repulsion forces, and pulled toward the target by the attractive force. Another approach to generating motions among unknown static obstacles is based on the edge detection method.[11] First, two vertical visible edges of the obstacles are determined from the sensor data. The horizontal line connecting two vertical edges is considered to be one of the boundaries of the obstacle. An AGV is then steered around either side of the visible edges. A drawback of this method is its sensitivity to sensor accuracy. Frequent misreading of ultrasonic sensors makes the performance unstable. The third approach is the wall-following method.[12] If the AGV encounters an obstacle while navigating, it follows the obstacle's countour until the AGV has passed by the obstacle.

The above approaches only deal with unknown static obstacles. However, obstacles are not always stationary. Some studies deal with motion planning among moving obstacles that move along known trajectories.[13] In a real situation, an AGV may face unexpected moving obstacles such as human beings. Failure to consider possible intrusion of previously unknown moving obstacles prohibits flexible AGV navigation.

There exist few techniques for motion planning in the presence of unknown moving obstacles. Tychonievich et al.[14] extended the maneuvering board method commonly used for nautical navigation to find a collision-free path through a field of moving obstacles, from a starting point to a goal point, where the goal point can itself be in motion. It is assumed that the instantaneous velocities and positions of the obstacles are known in advance but may be uncertain.

Kehtarnavaz and Li[15] introduced an estimation approach to predict the positions of moving obstacles using an autoregressive model. For each obstacle, a collision region around the obstacle path from its current position to the predicted position is defined. By drawing tangent lines between these regions, the shortest collision-free path between the current and a desired location of the vehicle can be obtained.

Steele and Starr[16] decomposed the path planning process of an AGV into two supporting processes: (1) using a graph search method to plan a path for avoiding all known static obstacles, and (2) using a field potential approach to control the motion of the AGV when unexpected obstacles are encountered.

The concept of fuzzy theory was first introduced in 1965 by Zadeh.[17] Henceforth, researchers have found numerous ways to utilize this theory to generalize existing techniques and to develop new algorithms in many research and application fields. Recently, fuzzy theory has been getting more and more important in the field of control, artificial intelligence, pattern recognition, robotics, etc. The success of many applications of the fuzzy set theory have been completely established.

Fuzzy control is one of the most successful applications of fuzzy theory. Several collision-avoidance approaches based on fuzzy logic have been used to control AGVs in stationary environments.[18-21] In these methods, many parameters are required as the input of the fuzzy inference procedures, such as the wall distance, the wall angle, the heading angle of the AGV, the width of the road, the road shape, etc. These methods only deal with static obstacles, and thus are not practical for real navigation environments.

To be suitable for time-varying environments, static obstacles with no a priori position information as well as moving obstacles with unknown trajectories are considered in this study for AGV navigation. Fuzzy logic control is applied for the guidance of an AGV from a starting point toward the target without colliding with any obstacle. Intuitive motions of human beings are modeled into fuzzy rules such that the AGV has the capability, like human beings, of avoiding the obstacles. These fuzzy rules can be dynamically weighted according to the nearness degree of the found obstacles. Furthermore, the proposed approach can also be used for the navigation of multiple AGVs, with few modification to the original algorithm.

In the remainder of this article, brief concepts of the fuzzy set theory and fuzzy inference method are introduced in section 2. The computational framework of the proposed approach and detailed description of each control module are described in sections 3 and 4, respectively. Some simulation results are shown in section 5, and conclusions appear in the last section.

2. FUZZY THEORY

In this section we give a brief note on fuzzy theory and its principal ideas.[22]

2.1. Fuzzy Set

A classical (or crisp) set is a collection of elements. For a universal set X, each single element $x \in X$ can either belong to or not belong to a set A, where $A \subseteq X$. A way of defining set A is to use the characteristic function μ_A, in which 1 indicates membership and 0 nonmembership.

For a fuzzy set, the characteristic function allows various degrees of membership for each element. Let X be a universal set, then a fuzzy set $\tilde{A}$ in X is a set of ordered pairs:

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) \mid x \in X\}$$

where $\mu_{\tilde{A}}(x)$ is called the membership function of x in $\tilde{A}$. The height of $\tilde{A}$ is defined as the supremum of $\mu_{\tilde{A}}(x)$ over X.

2.2. Fuzzy Set Operations

Most crisp set operations, such as *union, intersection,* and *complement,* have analogs in the fuzzy set theory. For any two fuzzy sets $\tilde{A}$ and $\tilde{B}$ defined in X, with membership functions being $\mu_{\tilde{A}}(x)$ and $\mu_{\tilde{B}}(x)$, respectively, the membership function $\mu_{\tilde{C}}(x)$ of their *intersection* $\tilde{C} = \tilde{A} \cap \tilde{B}$ is pointwise defined by

$$\mu_{\tilde{C}}(x) = \min\{\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)\}, \quad x \in X.$$

Similarly, the membership function $\mu_{\tilde{D}}(x)$ of the *union* $\tilde{D} = \tilde{A} \cup \tilde{B}$ is pointwise defined by

$$\mu_{\tilde{D}}(x) = \max\{\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)\}, \quad x \in X.$$

And the membership function $\mu_{\complement\tilde{A}}(x)$ of the *complement* of a fuzzy set $\tilde{A}$ is defined by

$$\mu_{\complement\tilde{A}}(x) = 1 - \mu_{\tilde{A}}(x), \quad x \in X.$$

2.3. Fuzzy Inference

The correlation-minimum inference method proposed by Mamdani[23] is applied in this study for fuzzy inference. Fuzzy inference is a reasoning method using fuzzy theory, whereby human knowledge is expressed using linguistic rules, for example, IF $\tilde{A}$ THEN $\tilde{B}$ (or $\tilde{A} \Rightarrow \tilde{B}$), where $\tilde{A}$ and $\tilde{B}$ are fuzzy variables defined in two universal sets X and Y, respectively. Assume that $\tilde{A}'$ is the input fuzzy variable. Then Mamdani's inference procedure can be illustrated by Figure 1. First, the intersection $\tilde{A}' \cap \tilde{A}$ of the input fuzzy set $\tilde{A}'$ and antecedent fuzzy set $\tilde{A}$ is computed. Then the height h of $\tilde{A}' \cap \tilde{A}$ is used to clip the top of the consequent fuzzy set $\tilde{B}$, and the resultant fuzzy set $\tilde{B}'$ is the inference output.

If the input variable is a nonfuzzy value x', it can be regarded as a fuzzy set $\tilde{A}'$ with the membership function $\mu_{\tilde{A}'}(x) = 1$ if $x = x'$, and 0 otherwise. So the inference process is identical to the above procedure. This is illustrated in Figure 2.

The output $\tilde{B}'$ of the inference procedure is a fuzzy set. In practical control applications, a nonfuzzy output value is desirable, so defuzzification of the fuzzy set $\tilde{B}'$ is necessary. Common defuzzification strategies include the maximum criterion method, the mean of maximum methods, and the center of area method.[24] The center of area method is used in this study because it usually yields superior results.[25] In this method, the centroid b of the fuzzy set $\tilde{B}'$ is chosen as the real-valued output:

$$b = \frac{\int y \mu_{\tilde{B}'}(y)\, dy}{\int \mu_{\tilde{B}'}(y)\, dy}.$$

If a fuzzy rule is multiantecedent, for example,

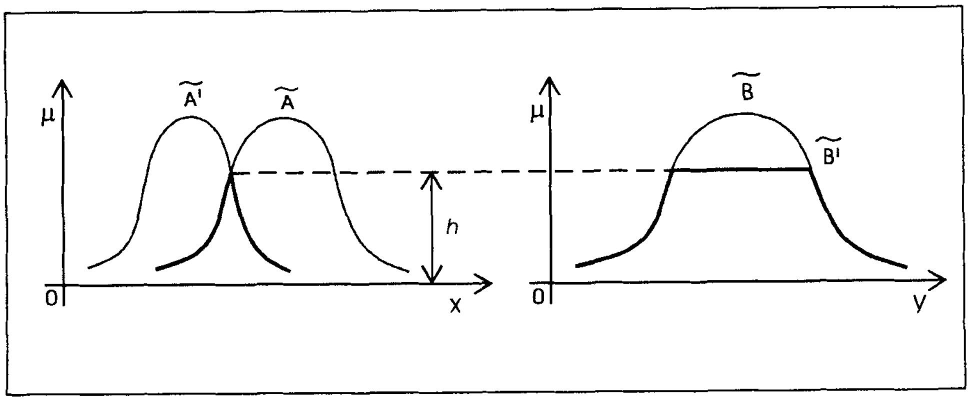


Figure 1. A fuzzy inference example.

IF $\tilde{A}$ AND $\tilde{B}$ THEN $\tilde{C}$

or

IF $\tilde{A}$ OR $\tilde{B}$ THEN C′,

the output fuzzy set $\tilde{C}'$ is obtained by clipping $\tilde{C}$ according to the minimum or maximum value of two height values h_A and h_B, as shown in Figures 3 and 4, respectively.

When there is more than one fuzzy rule, superimposing these fuzzy rules is needed to produce the inference result. Assume that there are m rules $\tilde{A}_1 \Rightarrow \tilde{B}_1$, $\tilde{A}_2 \Rightarrow \tilde{B}_2$, . . . , $\tilde{A}_m \Rightarrow \tilde{B}_m$, and the input $\tilde{A}$ activates all the rules in parallel. When the fuzzy rules have unequal importance, or the reliability of these fuzzy rules is not identical, a reasonable method is to give each fuzzy rule a different weight. The weight $w_i(0 \leq w_i < 1)$, associated with the ith fuzzy rule, is to indicate the importance of the rule. As shown in Figure 5, the output $\tilde{B}$ of these m rules is thus

$$\tilde{B} = \sum_{i=1}^{m} w_i \tilde{B}_i'.$$

The larger the weight value, the more contribution the corresponding fuzzy rule gives to the output.

3. FUZZY APPROACH TO OBSTACLE AVOIDANCE

Fuzzy inference and some intuitive fuzzy rules are applied in this study for the guidance of an AGV. The AGV moves among unknown obstacles and is expected to reach a specified target. The unknown obstacles may be stationary or moving in arbitrary directions and at different speeds. Collision-free motions are generated cycle by cycle according to the inference result.

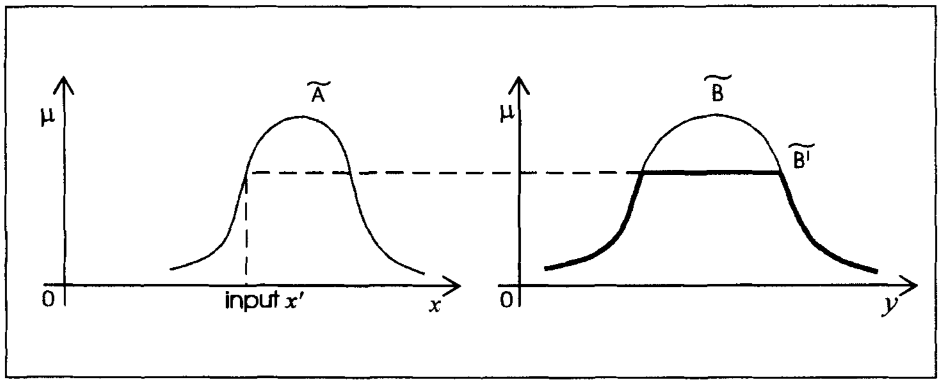


Figure 2. A fuzzy inference example with input being a nonfuzzy value.

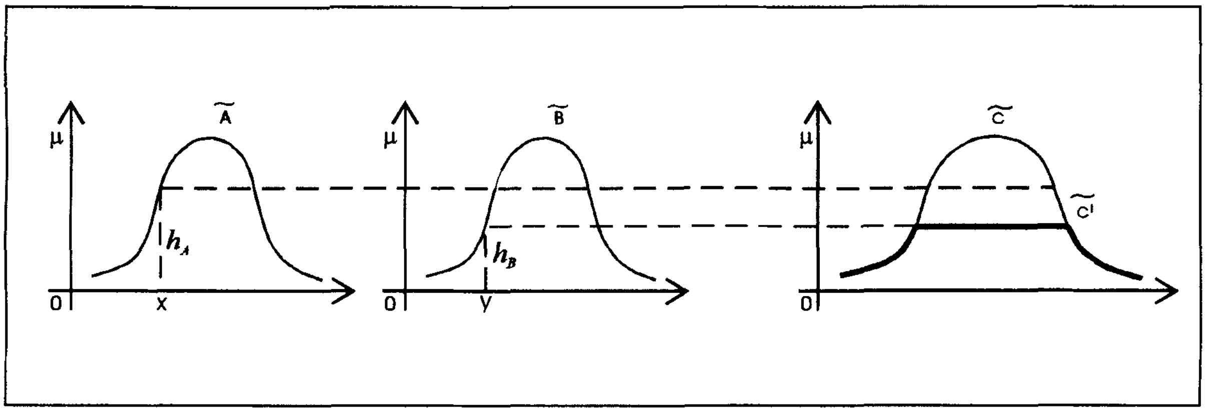


Figure 3. Fuzzy inference for a multiantecedent rule: IF $\tilde{A}$ AND $\tilde{B}$ THEN $\tilde{C}$. Minimum of h_A and h_B is used to clip $\tilde{C}$.

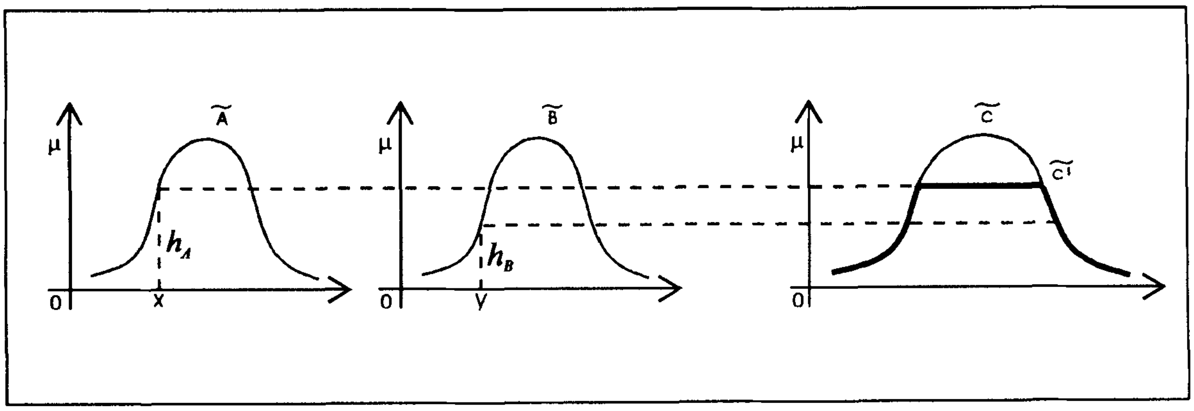


Figure 4. Fuzzy inference for a multiantecedent rules: IF $\tilde{A}$ OR $\tilde{B}$ THEN $\tilde{C}$. Maximum of h_A and h_B is used to clip $\tilde{C}$.

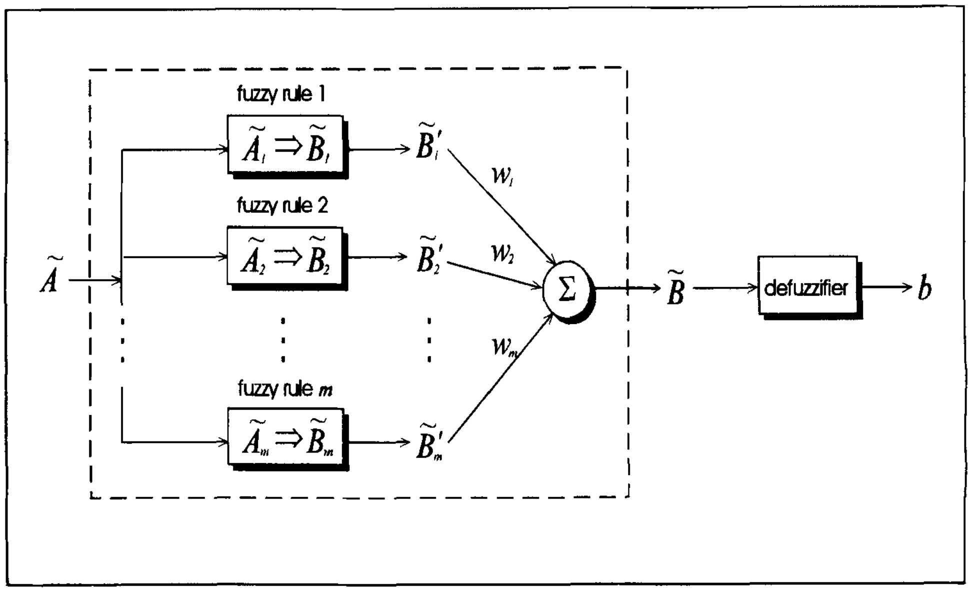


Figure 5. Superimposition of multiple fuzzy rules.

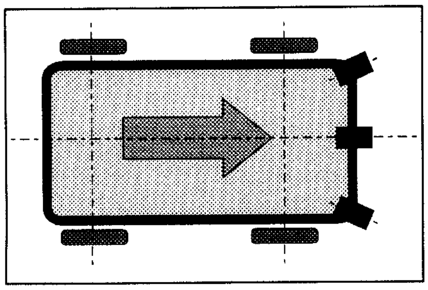


Figure 6. The AGV configuration.

Shown in Figure 6 is the simulated AGV configuration. Three ultrasonic sensors are mounted at the front of the AGV to detect obstacles to the left front, right front, and direct front locations, respectively. The proposed navigation algorithm is composed of six fuzzy control modules: *determining-far-near module, controlling-speed module, directing-toward-target module, avoiding-static-obstacle module, avoiding-moving-obstacle module,* and *handling-trap-state module,* as shown in Figure 7.

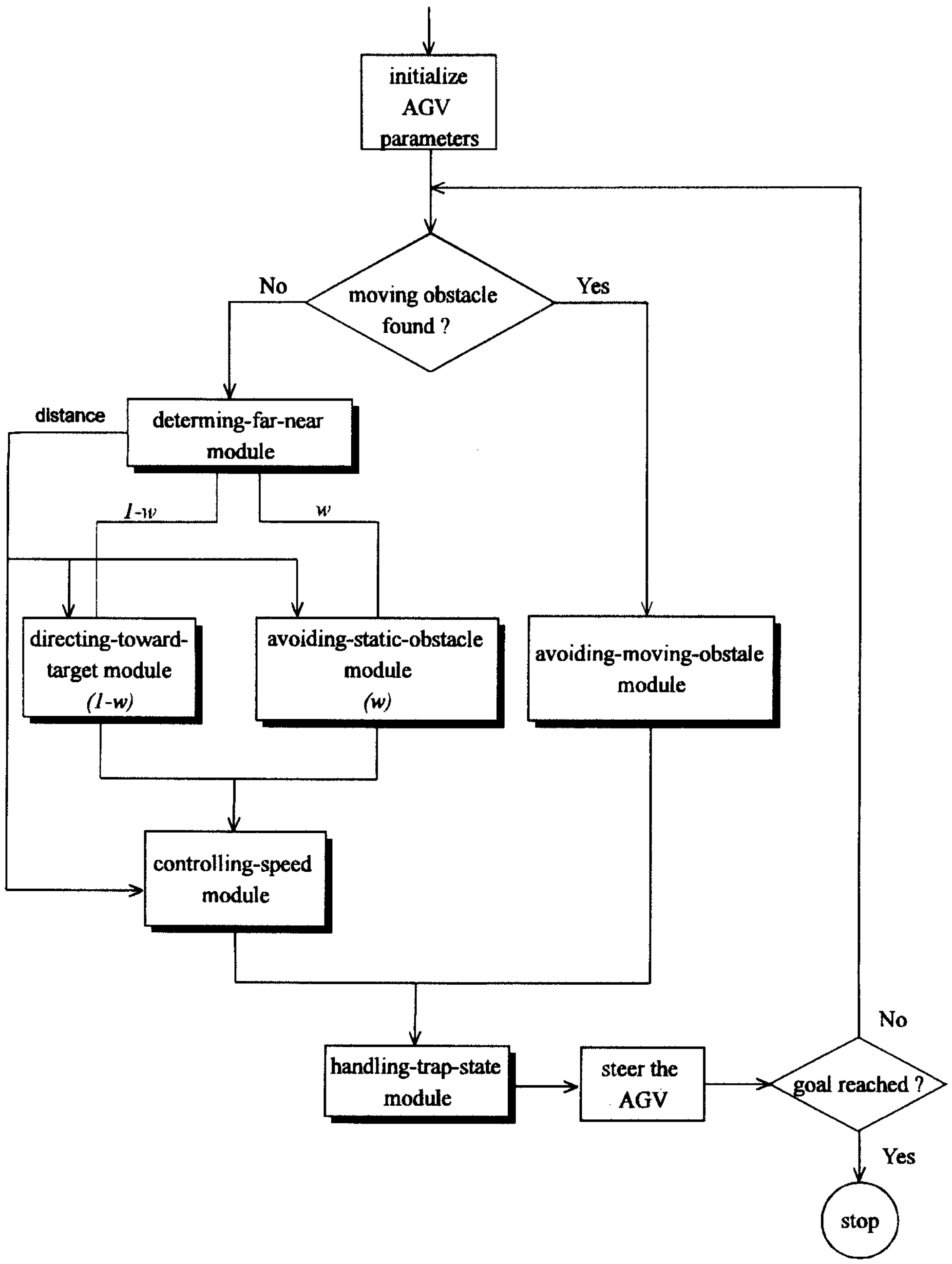

initialize AGV parameters
moving obstacle found ?
No
Yes
distance
determing-far-near module
1-w
w
directing-toward-target module (1-w)
avoiding-static-obstacle module (w)
avoiding-moving-obstale module
controlling-speed module
handling-trap-state module
steer the AGV
goal reached ?
No
Yes
stop

Figure 7. The computational framework of a navigation cycle for collision avoidance.

Before the navigation sessions, some parameters should be initialized in advance—the effective distances of ultrasonic sensors, and the maximum speed and acceleration of the AGV, for example. At the beginning of each navigation cycle, the AGV first scans the environment to see whether there exist obstacles in front. If there exist moving obstacles, the *avoiding-moving-obstacle module* is executed, which is used to avoid the AGV from colliding with moving obstacles by changing the speed of the AGV. The orientation of the AGV is not changed in this module.

On the other hand, if static obstacles are found, the *determining-far-near module* is executed to determine whether they are near to the AGV. The computed nearness-degree value w and the far-degree value (1-w) are then taken as the weights of the *avoiding-static-obstacle module* and the *directing-toward-target module,* respectively. So if a larger nearness-degree value is obtained, a larger weight value is assigned to the *avoiding-static-obstacle module.* Otherwise, a larger weight value is assigned to the *directing-toward-target module.* That is, if the found static obstacles are near to the AGV, it is more urgent to avoid these obstacles than to direct the AGV toward the target. But when the static obstacles are distant or no static obstacle is found, it is more reasonable for the AGV to steer toward the target. In addition, the distances between the static obstacles and the AGV are passed to the *avoiding-static-obstacle module,* the *directing-toward-target module,* and the *controlling-speed module* as the input of their fuzzy control rules.

In both the *avoiding-static-obstacle module* and *directing-toward-target module,* only the orientation of the AGV is changed. In the former module, the orientation of the AGV is changed according to the locations of the static obstacles. But it is changed according to the direction of the target in the latter module. After obtaining the orientation of the AGV, the speed of the AGV is determined in the *controlling-speed module based* on the distance between the AGV and the closest static obstacle.

Under some circumstances, the AGV may get blocked. The *handling-trap-state-module* is used to check whether the AGV is in a trap state and help it to leave this state.

Finally, the orientation and speed computed from the above-mentioned inference modules are used to steer the AGV. Then, in accordance with the location of the AGV, whether the goal is reached is determined. If it is not reached, the next navigation cycle begins.

4. CONTROL MODULES

In this section, detailed specifications of the fuzzy rules in each control module are given. These rules are derived from human beings' intuitive motions of collision avoidance.

4.1. Avoiding-Moving-Obstacle Module

When an AGV is navigating toward the target and a moving obstacle is detected, this module is executed to control the AGV such that it will not collide with the moving obstacle. Instead of turning its orientation, the AGV just changes its speed to prevent collision, according to the distance between the moving obstacle and the AGV. If the distance is short, the AGV reduces speed. The output of this module is the reduced speed value of the AGV. Computation of the distance between a moving obstacle and an AGV is not trivial, which will be discussed later. The fuzzy rules and the corresponding membership functions of this module are shown in Figure 8.

4.2. Determining-Far-Near Module

Three ultrasonic sensors mounted at the front of an AGV are used to detect obstacles to the left front, right front, and direct front locations, respectively. The distance readings of these three ultrasonic sensors are the input of this module. The output of this module is the nearness degree of the found static obstacles, which is used to determine the weights of the *directing-toward-target module* and the *avoiding-static-obstacle module.*

There are two fuzzy rules to decide the nearness degree of the found static obstacles. If one of the obstacles detected by the three sensors is close to the AGV, there is a high nearness-degree value. Otherwise, if all the found obstacles are distant, there is a low nearness-degree value. The larger the value, the higher the degree of nearness. The fuzzy rules and their corresponding membership functions are shown in Figure 9.

4.3. Avoiding-Static-Obstacle Module

When an AGV is navigating toward the target, if one of the three (left, center, and right) ultrasonic sensors detects static obstacles, this module is executed to change the orientation of the AGV to avoid collision. The vehicle is steered toward the direction with more free space according to the locations of

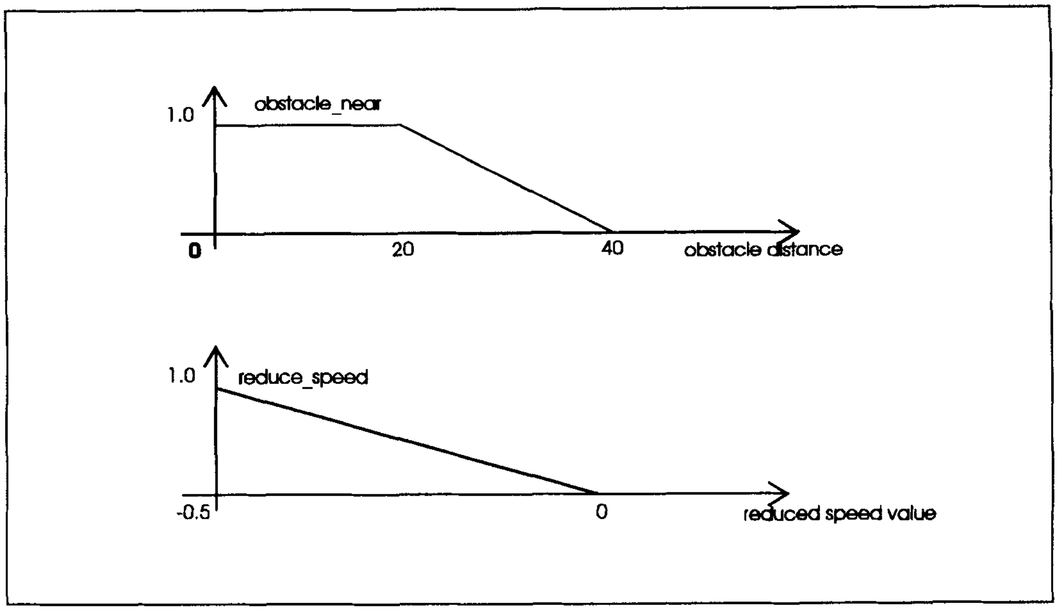


Figure 8. Fuzzy rules and membership functions of *avoiding-moving-obstacle module.* Rule 1: IF obstacle_*near* THEN *reduce_speed.*

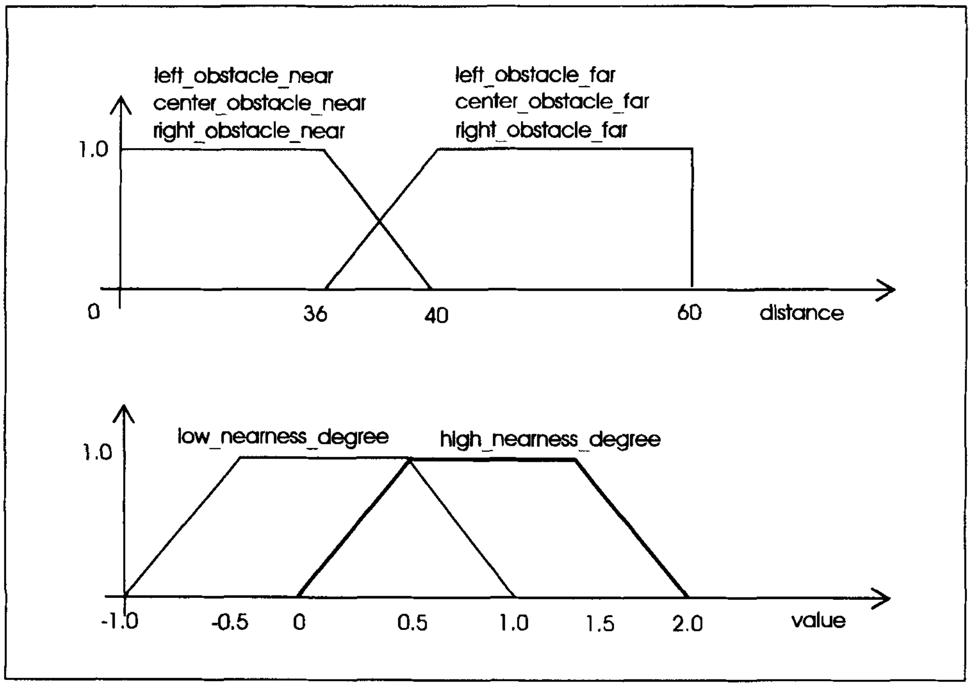


Figure 9. Fuzzy rules and membership functions of *determining-far-near module.*

Rule 1:	IF	*left_obstacle_near*	OR	*center_obstacle_near*	OR
		right_obstacle_near	THEN	*high_nearness_degree.*	
Rule 2:	IF	*left_obstacle_far*	AND	*center_obstacle_far*	AND
		right_obstacle_far	THEN	*low_nearness_degree.*	

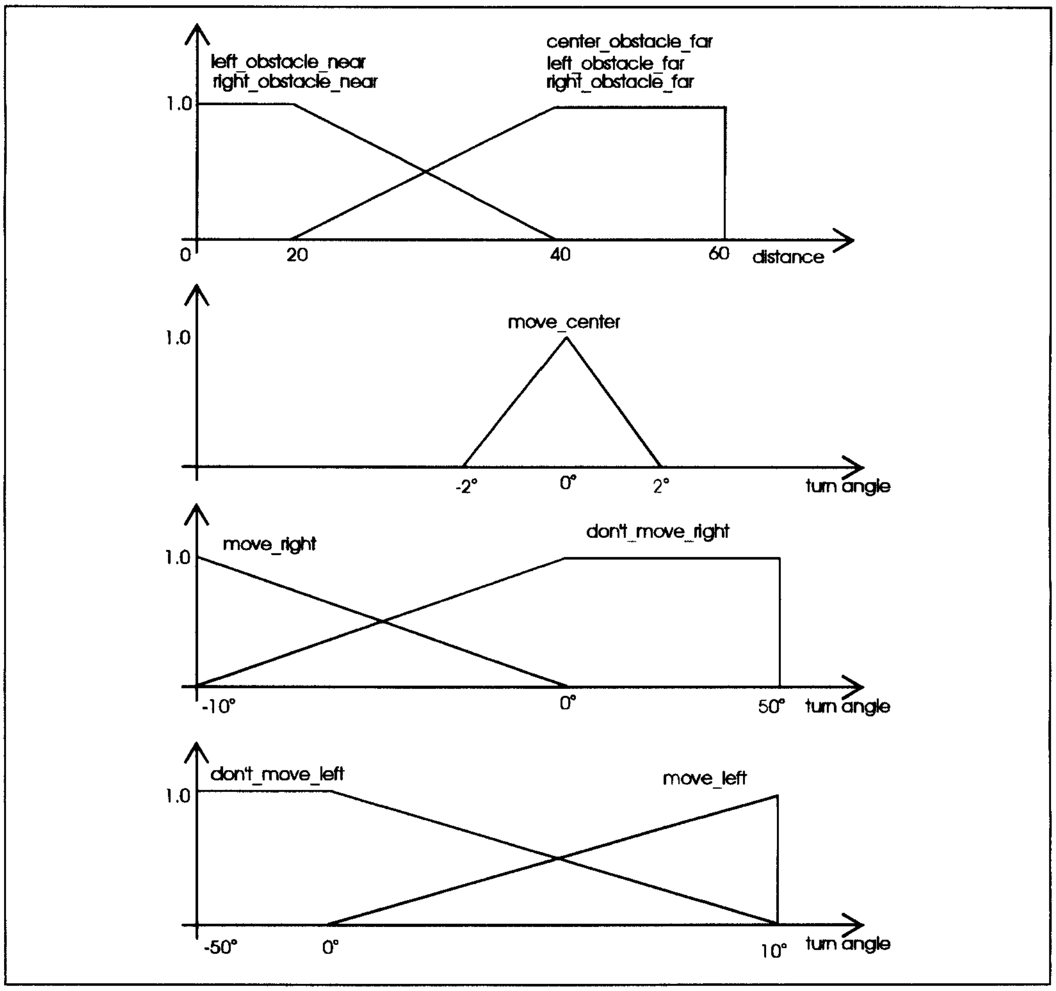


Figure 10. Fuzzy rules and membership functions of *avoiding-static-obstacle module.*

Rule 1:	IF	*right_obstacle_*	THEN	*move_right.*
Rule 2:	IF	*left_obstacle_far*	THEN	*move_left.*
Rule 3:	IF	*center_obstacle_far*	THEN	*move_center.*
Rule 4:	IF	*right_obstacle_near*	THEN	*don't_move_right.*
Rule 5:	IF	*left_obstacle_near*	THEN	*don't_move_left.*

obstacles. For example, if the right obstacle is far, the AGV could move right. The output of this module is the turn angle of the AGV. The fuzzy control rules and their corresponding membership functions are shown in Figure 10.

4.4. Directing-Toward-Target Module

This module is used to guide an AGV from its current location toward the target. The input of this module is the angle between the moving direction of the AGV and the vector from the AGV location to the target location; the output is the turn angle of the AGV. The fuzzy rules of this module and their corresponding membership functions are shown in Figure 11.

4.5. Controlling-Speed Module

In each navigation cycle, if we can dynamically vary the speed of the AGV, its flexibility will be increased. The purpose of this module is to control the speed of the AGV. When the detected static obstacles are distant or no obstacle is found, the AGV accelerates. Otherwise, it decelerates such that it can take a safe turn to avoid the static obstacles.

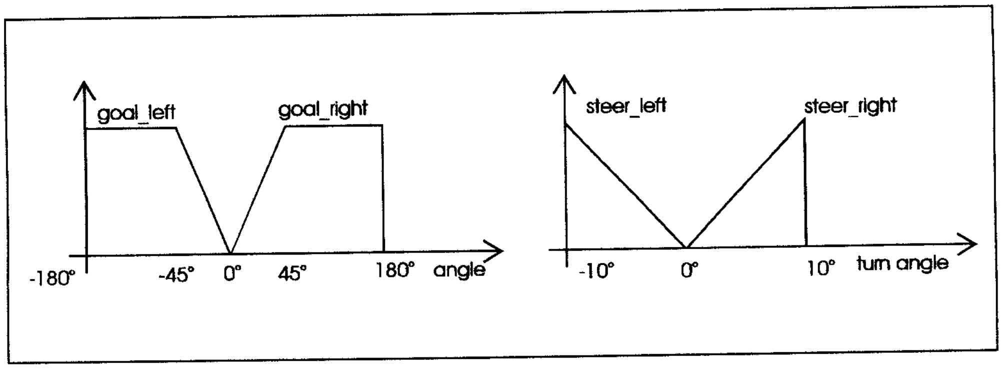


Figure 11. Fuzzy rules and membership functions of *directing-toward-target module.*
Rule 1: IF *goal_left* THEN *steer_left.*
Rule 2: IF *goal_right* THEN *steer_right.*

There are two input values in this module, including the speed of the AGV and the distance between the AGV and the nearest static obstacle. The output of this module is the increased or reduced speed value of the AGV. The minimum speed value is zero; that is, the AGV is not allowed to back up. The fuzzy control rules of this module and their corresponding membership functions are shown in Figure 12.

4.6. Handling-Trap-State Module

The purpose of this module is to help an AGV to recover from a trap state. A trap state occurs when-

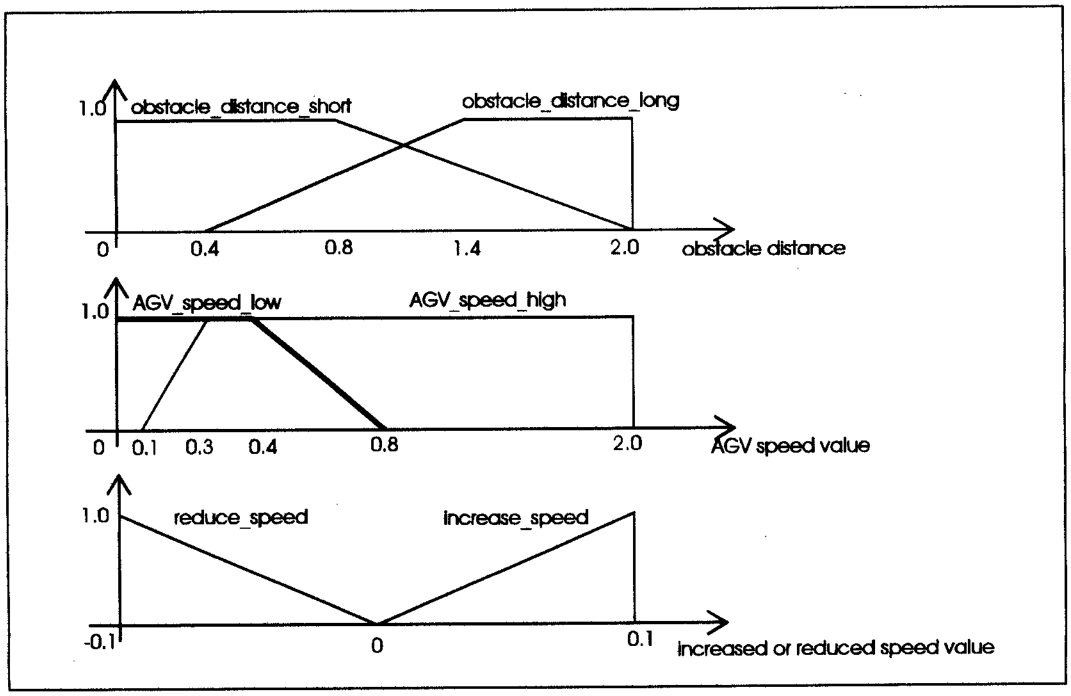


Figure 12. Fuzzy rules and membership functions of *controlling-speed module.*
Rule 1: IF *obstacle_distance_long* AND *AGV_speed_low*
THEN *increase_speed.*
Rule 2: IF *obstacle_distance_short* AND *AGV_speed_high*
THEN *reduce_speed.*

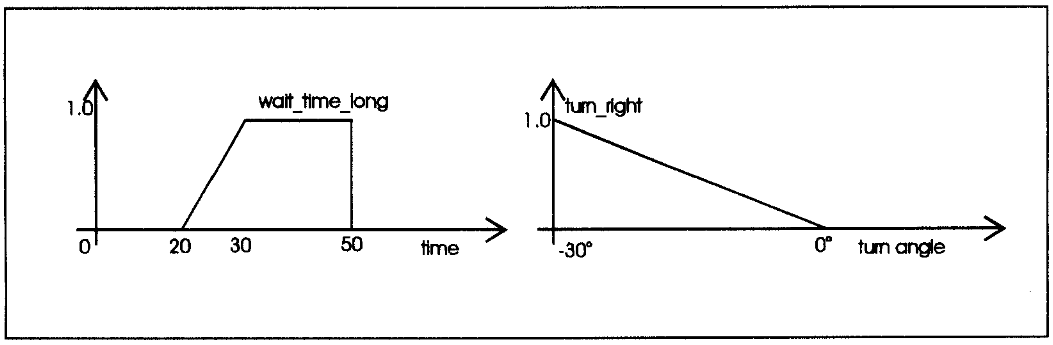


Figure 13. Fuzzy rules and membership functions of *handling-trap-state module.*
Rule 1: IF *wait_time_long* THEN *turn_right.*

ever an AGV is blocked; that is, it neither moves nor turns but just keeps waiting endlessly. A common trap state is when the distance readings of the left-front sensor and the right-front sensor of the AGV are equal; the AGV can neither turn right nor left but keeps going forward. Because the obstacles detected are static, the AGV will eventually reduce its speed to zero. Consequently, the AGV is trapped. Another common trap state occurs when two AGVs move toward each other on a straight line and thus each takes the other one for a moving obstacle. To avoid collision, the two AGVs reduce speed to let the other one to pass by and eventually stop to wait. So these two AGVs wait for each other endlessly.

A trap state may be detected by simply checking if the speed of the AGV is zero for a long period of time. A simple control rule is applied in this module to resolve this problem. When an AGV get trapped, we simply turn it right. The longer the wait time, the larger the turn angle. The fuzzy rules of this module and their corresponding membership functions are shown in Figure 13.

4.7. Computation of the Distance of the Moving Obstacle

It is not mentioned in subsection 4.1 how to measure the distance between a moving obstacle and the AGV. In this study, a collision zone[26] of the moving obstacle is constructed to resolve the problem. It is assumed that the AGV has the ability to obtain the location of a moving obstacle from the input sensors.

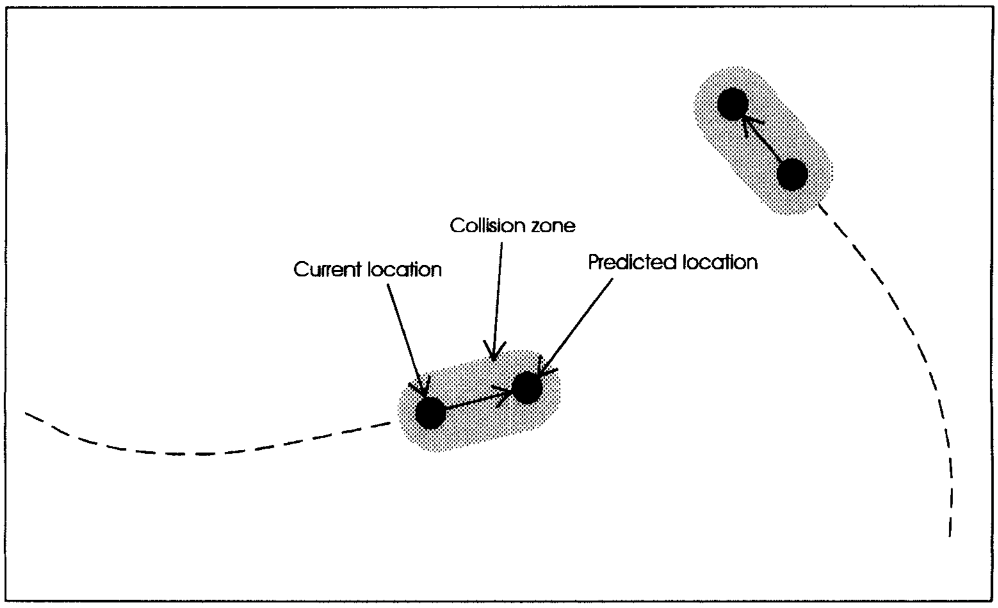


Figure 14. Two moving obstacles and their collision zones.

First, the trajectory of the moving obstacle is predicted, and then a collision zone around the obstacle path from its current position to the predicted position is formed, as shown in Figure 14. The shortest distance between the collision zone and the AGV is used as the distance between the moving obstacle and the AGV.

To establish the collision zone for each moving obstacle, a path prediction algorithm is required. It is desired that the prediction process be as accurate as possible without expensive computation. Linear prediction models usually give simple but effective solutions. In the following, we first briefly introduce the real-time least-mean-square error (LMSE) estimation algorithm,[27] and then adapt this algorithm to predict the paths of moving obstacles.

Let x be a variable related linearly to a time variable t, that is, let

$$x = at + b \tag{1}$$

where a and b are two unknown constant parameters to be estimated by a sequence of m observations x_i on x at m different time instants t_i, $i = 1, 2, \ldots, m$. The m observation data provide the following set of m linear equations:

$$x_i = at_i + b \quad i = 1, 2, \ldots, m,$$

which can be arranged into a simple form

$$X_m = T_m A_m$$

where

$$X_m = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix},$$

$$T_m = \begin{bmatrix} t_1 & 1 \\ t_2 & 1 \\ \vdots & \\ t_m & 1 \end{bmatrix},$$

$$A_m = \begin{bmatrix} a \\ b \end{bmatrix}.$$

Define an error vector $E_m = (e_1, e_2, \ldots, e_m)^t$ for each inexact solution A_m, such that

$$E_m = X_m - T_m A_m.$$

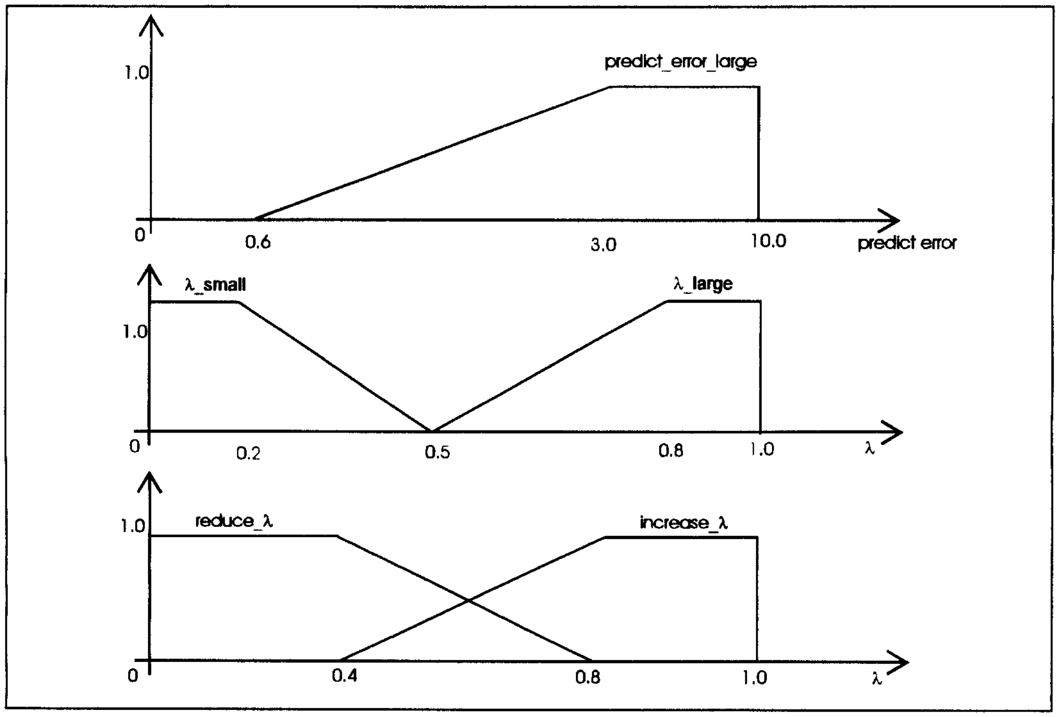


Figure 15. Fuzzy rules and membership functions of λ *control module.*

In the real-time LMSE estimation algorithm, the optimal solution, $\hat{A}_m$, is defined as the one such that the error function

$$J_m = \sum_{i=1}^{m} \lambda^{m-i} e_i^2, \quad 0 < \lambda < 1,$$

is minimized. The squared errors coming from more recent data are given larger weights than those from earlier ones. The smaller the λ value, the heavier the weights assigned to more recent data. Such a type of data emphasis is appropriate for the locality property of collision-free motion planning. Define P_{m-1} as

$$P_{m-1} = \frac{1}{\lambda}(T_{m-1}^t T_{m-1})^{-1}.$$

Then it can be shown[27] that

$$\begin{aligned} \hat{A}_m &= \hat{A}_{m-1} + \gamma_m P_{m-1} \mathbf{t}_m \{x_m - \mathbf{t}_m^t \hat{A}_{m-1}\}, \\ P_m &= \frac{1}{\lambda}\{P_{m-1} - \gamma_m P_{m-1} \mathbf{t}_m \mathbf{t}_m^t P_{m-1}\}, \end{aligned} \tag{2}$$

where

$$\mathbf{t}_m = \begin{bmatrix} t_m \\ 1 \end{bmatrix},$$

$$\gamma_m = \frac{1}{1 + \mathbf{t}_m^t P_{m-1} \mathbf{t}_m}.$$

The "real-time" means that when fresh experimental data x_m are supplied, the estimation $\hat{A}_m$ can be obtained (from Eq. (2)) by simply using this new information and the previous estimation $\hat{A}_{m-1}$, instead of all the data $x_1, x_2, \ldots,$ and x_m.

To apply the above estimation algorithm to local obstacle trajector prediction, assume that a sequence of m observations on the position of an obstacle have been made at m different time instants. By the following two sets of linear equations

$$\begin{cases} x_i = at_i + b & i = 1, 2, \ldots, m \\ y_i = ct_i + d & i = 1, 2, \ldots, m \end{cases} \tag{3}$$

where (x_i, y_i) denote coordinates of the position of the obstacle at time instant t_i, we can use the formulas in Eq. (2) to estimate and update parameters a, b, c, and d using the fresh observation on the position of the obstacle obtained in each navigation cycle.

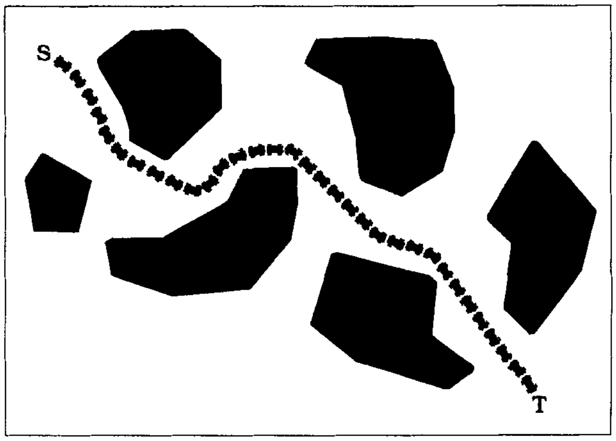


Figure 16. An AGV navigates among static obstacles.

Then we can predict the positions of the obstacle in the next cycles using these estimated parameters and the equations in Eq. (3).

In the real-time LMSE estimation algorithm, the weight λ is fixed. However, it is not reasonable to fix this value in all the navigation sessions. We adjust the weight dynamically according to the prediction error, so that the predicted path is as accurate as possible. If the prediction error is large and λ is large, we reduce the value of λ to place more emphasis on the recent data. But if the prediction error is large and λ is small, we increase the value of λ to place more emphasis on previous data. The prediction error is defined as follows:

$$\text{error} = \sqrt{(x_i - x_p)^2 + (y_i - y_p)^2}$$

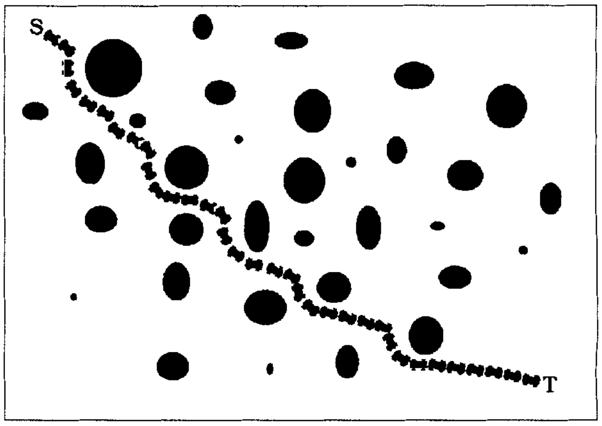
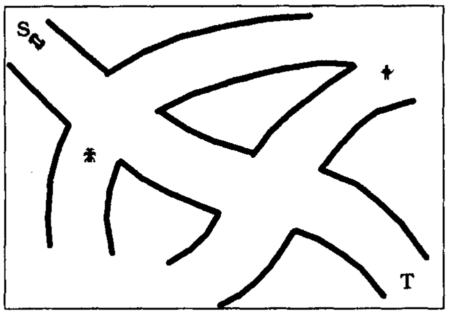
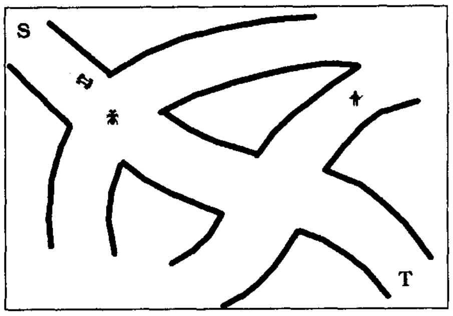


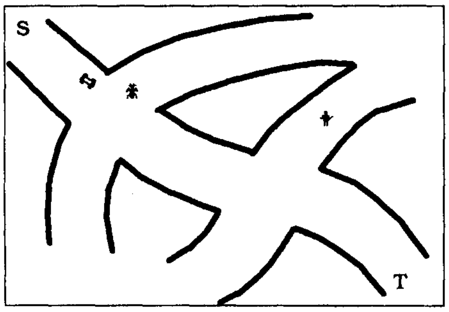
Figure 17. An AGV navigates among cluttered static obstacles.



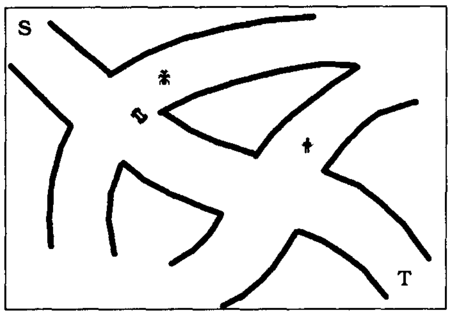
(a) An AGV wishes to go to target T from starting point S.



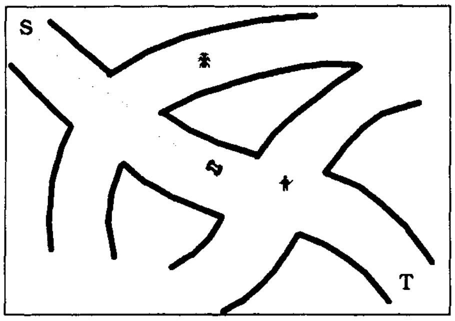
(b) The AGV is approaching a moving beetle.



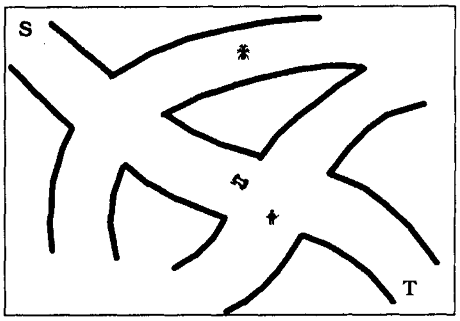
(c) The AGV waits for the beetle to pass by.



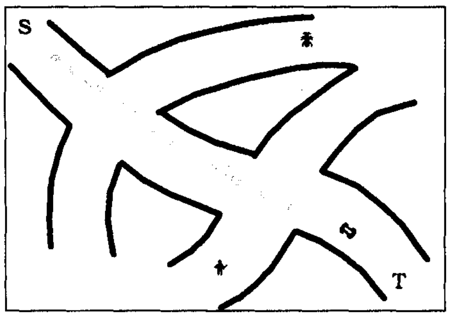
(d) The AGV steers toward the target.



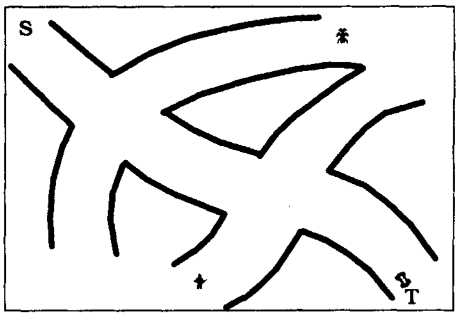
(e) The AGV is approaching a walking man.



(f) The walking man has passed by.

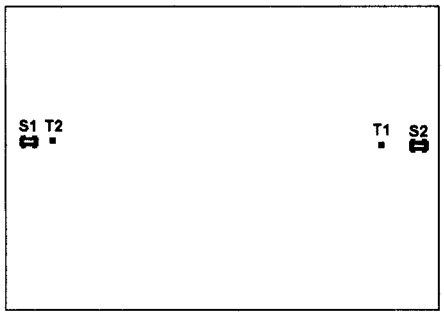


(g) The AGV steers toward the target.

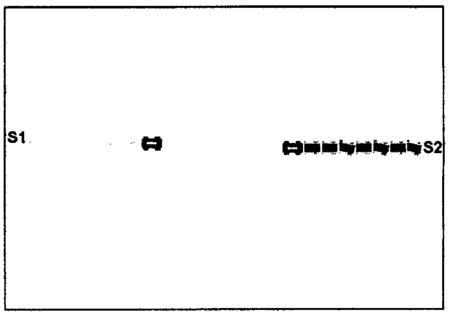


(h) The AGV reaches the target.

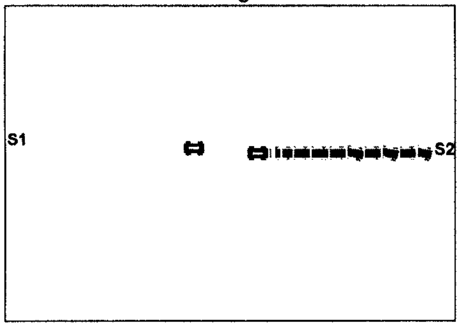
Figure 18. Simulation of an AGV navigating among moving obstacles in a street.



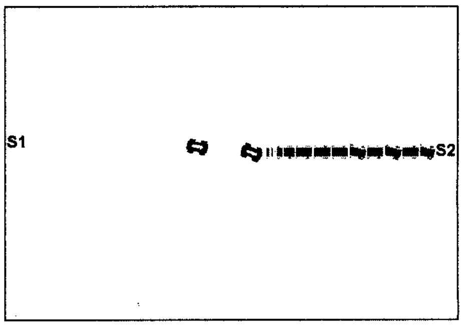
(a) AGV1 wants to go to target T1 from S1 and AGV2 to target T2 from S2.



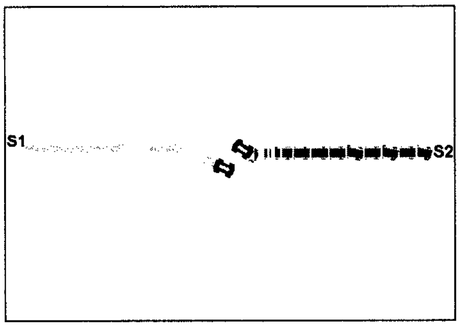
(b) Both AGVs steer toward their targets.



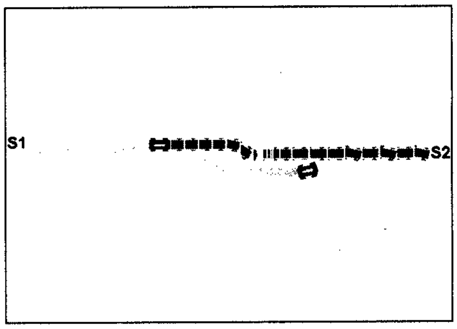
(c) Both AGVs detect each other and reduce speed.



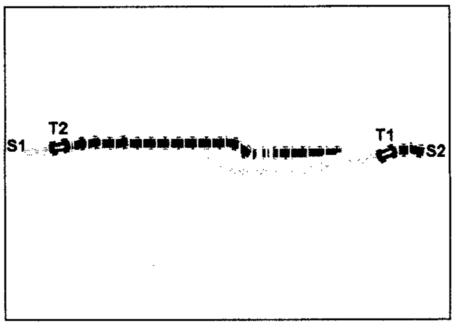
(d) Both AGVs detect a trap state and turn right to leave this state.



(e) Both AGVs have avoided collision with each other.

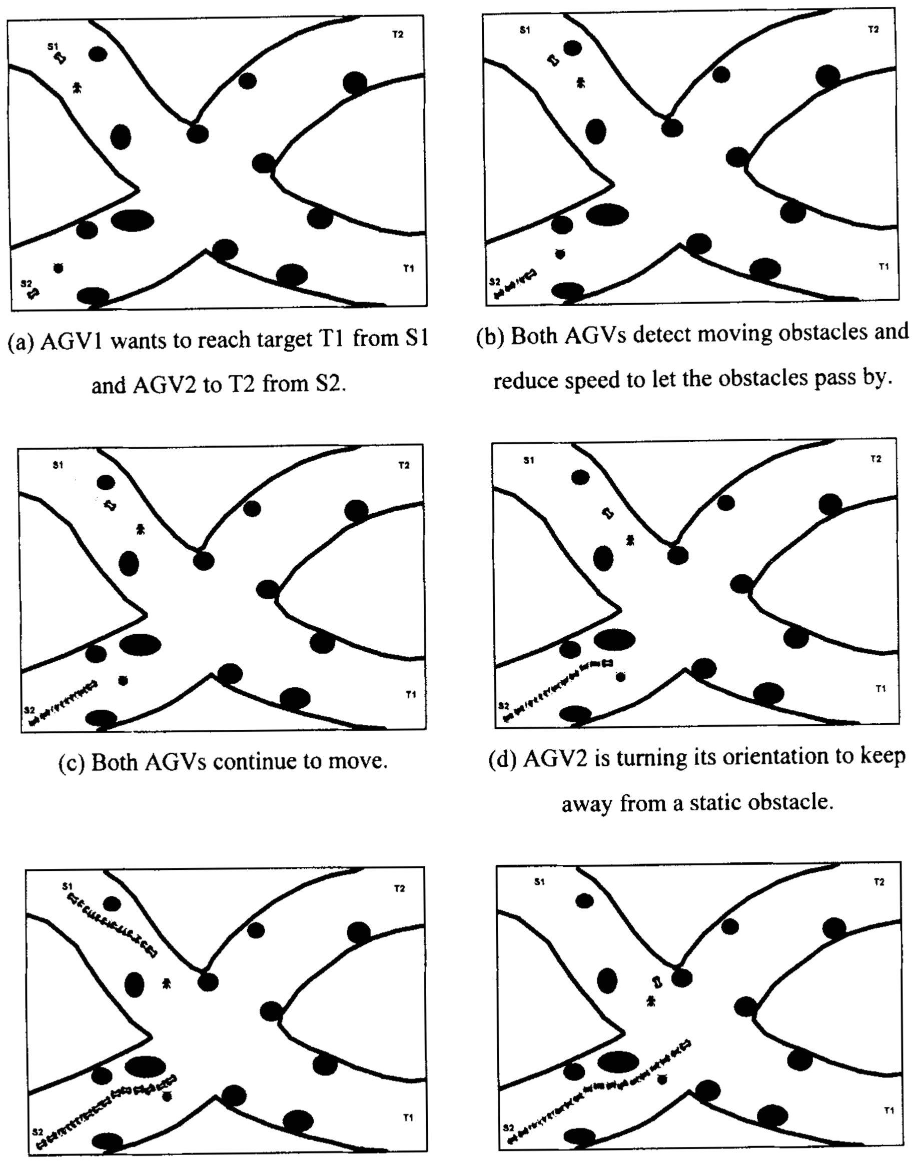


(f) Both AGVs steer toward their targets.



(g) Both AGVs reach their targets.

Figure 19. Simulation of two AGVs steering toward each other.



(a) AGV1 wants to reach target T1 from S1 and AGV2 to T2 from S2.

(b) Both AGVs detect moving obstacles and reduce speed to let the obstacles pass by.

(c) Both AGVs continue to move.

(d) AGV2 is turning its orientation to keep away from a static obstacle.

(e) AGV2 has avoided collision with the static obstacle.

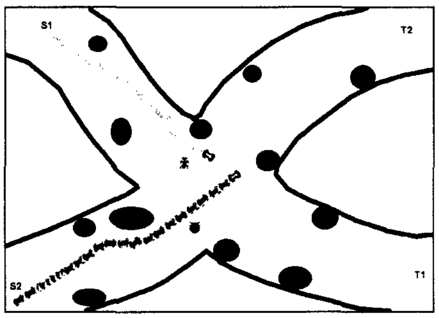
(f) Both AGVs steer toward their targets.

Figure 20. Simulation of two AGVs moving in streets with two moving obstacles and several static obstacles.

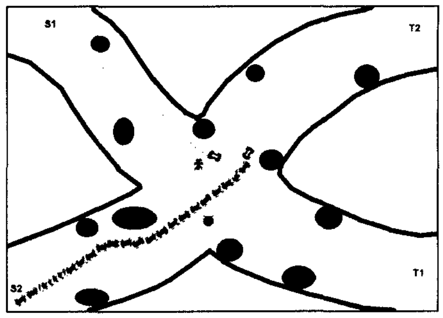
where (x_i, y_i) are the coordinates of the current position of the moving obstacle, and (x_p, y_p) are the coordinates of the predicted position of the moving obstacle computed in the previous cycle. The fuzzy rules used to adjust the weight λ and their corresponding membership functions are shown in Figure 15.

5. EXPERIMENTAL RESULTS

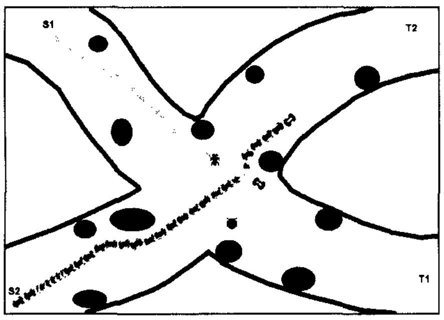
The proposed approach has been implemented in the C language on a 33 Mhz 486-PC with Windows 3.1 environment. The computation is rather fast. To demonstrate the feasibility of the proposed approach, some examples are shown in Figures 16–20. In these examples, the maximum acceleration of AGVs is 0.9 pixel/cycle^2, and the maximum velocity is not limited. Figure 16 shows the collision-free motions among static obstacles for an AGV moving from a starting point **S** to the target **T**. The positions and shapes of these obstacles are not known in advance. Figure 17 shows the trajectory of an AGV navigating in a cluttered environment. The experimental results



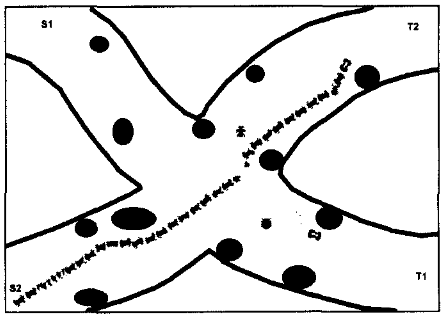
(g) AGV1 detects AGV2.



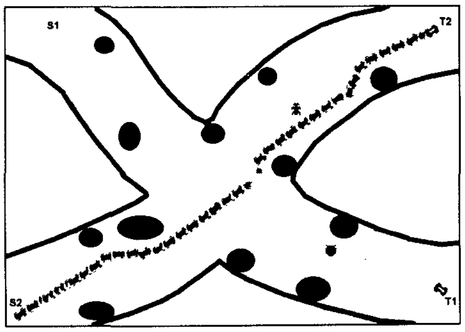
(h) AGV1 reduces speed to let AGV2 pass by.



(i) Both AGVs steer toward their targets.



(j) Both AGVs turn their orientations to keep away from the static obstacles.



(k) Both AGVs reach their targets.

Figure 20. (*continued*) Simulation of two AVGs moving in streets with two moving obstacles and several static obstacles.

indicate that the path planned by the proposed fuzzy approach is feasible and near optimal. The average ratio of the path length obtained from the proposed approach to the distance between the start and target positions is about 1.15. This is acceptable for AGVs navigating among cluttered environments.

Figure 18 shows several snapshots of the vehicle at different positions in the environment with two moving obstacles, denoted as a beetle icon and a man icon, respectively. The closer the positions of the vehicle icons, the lower the speed value of the AGV. The trajectories of the moving obstacles are not known in advance. Figures 19 and 20 show two cases that two AGVs encounter in the environment. Figure 19 shows the simulated snapshots of two AGVs navigating toward each other on a straight line. In Figure 20, two AGVs navigate in the streets with static and moving obstacles. Experiment results show that the proposed approach is also feasible for the navigation of multiple AGVs in an environment.

6. CONCLUSION

In this article, we have proposed a fuzzy control approach for the guidance of an AGV to the target among unknown static and moving obstacles. Weighted fuzzy rules have been designed for inference in the navigation sessions. Simulation results have been performed to show the feasibility of the proposed approach. A small number of parameters are needed in the inference procedure, and the computation is fast. Experimental results have shown that the proposed approach can be applied to the navigation of multiple AGVs in the environment, with few modifications to the original algorithm.

Further research may be directed to practically implementing the proposed approach on an experimental autonomous vehicle. Hence some important issues concerning navigation of the AGV will be studied, such as the sensor modeling. In addition, exploration of complex trap problems, walking in a maze, for example, is also a suggestion for further research.

This work was supported by National Science Council, Republic of China under Grant NSC82-0401-E-007-046.

REFERENCES

1. L. L. Wang and W. H. Tsai, "Car safety driving aided by 3-D image analysis techniques," *Proc. Microelectronics and Information Science and Technology Workshop*, Hsinchu, Taiwan, R.O.C., 1986, pp. 687–701.
2. R. L. Madarasz, L. C. Heing, R. F. Crimp, and N. M. Mazur, "The design of an autonomous vehicle for the disabled," *IEEE J. Rob. Autom.*, **RA-2**, 117–126, 1986.
3. O. Takahashi and R. J. Schilling, "Motion planning in a plane using generalized Voronoi diagrams," *IEEE Trans. Rob. Autom.*, **5**(2), 143–150, 1989.
4. R. C. Arkin, "Navigational path planning for a vision-based mobile robot," *Robotica*, **7**, 49–63, 1989.
5. G. T. Wilfong, "Motion planning for an autonomous vehicle," *Proc. IEEE Int. Conf. Rob. Autom.*, Philadelphia, PA, 1988.
6. S. Kambhampati and L. S. Davis, "Multiresolution path planning for mobile robots," *IEEE J. Rob. Autom.*, **2**(3), 135–145, 1986.
7. D. T. Kuan, J. C. Zamiska, and R. A. Brooks, "Natural decomposition of free space for path planning," *Proc. IEEE Int. Conf. Rob. Autom.*, St. Louis, MO, 1985, pp. 168–173.
8. R. A. Brooks, "Solving the find-path problem by good representation of free space," *IEEE Trans. Syst. Man Cybern.*, **13**(3), 190–197, 1983.
9. Y. K. Hwang, N. Ahuja, "Gross motion planning—A survey," *ACM Comput. Surv.*, **24**(3), 219–290, 1992.
10. J. Borenstein and Y. Koren, "Real time obstacle avoidance for fast mobile robots," *IEEE Trans. Syst. Man Cybern.*, **19**(5), 1179–1187, 1989.
11. I. Borenstein and Y. Koren, "Obstacle avoidance with ultrasonic sensors," *IEEE J. Rob. Autom.*, **RA-4**(2), 213–218, 1988.
12. G. Bauzil, M. Briot, and P. Ribes, "A navigation subsystem using ultrasonic sensors for the mobile robot Hilare," *Proc. 1st Int. Conf. Robot Vision and Sensory Controls*, Standford-upon-Avon, UK, 1981, pp. 47–58.
13. K. Fujimura, and H. Samet, "A herarchical strategy for path planning among moving obstacles," *IEEE Trans. Rob. Autom.*, **5**(1), 61–69, 1989.
14. L. Tychonievich, et al., "A maneuvering-board approach to path planning with moving obstacles," *Porc. 11th Int. Joint Conf. Artifical Intelligence*, Detroit, MI, 1989.
15. N. Kehtarnavaz and S. Li, "A collision-free navigation scheme in the presence of moving obstacles," *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, Ann Arbor, MI, 1988, pp. 1017–1021.
16. J. P. H. Steele and G. P. Starr, "Mobile robot path planning in dynamic environments," *Proc. IEEE Int. Conf. Syst. Man Cybern.*, Beijing and Shenyang, China, 1988.
17. L. A. Zadeh, "Fuzzy sets," *Information Control*, **8**, 338–353, 1965.
18. M. Maeda, Y. Maeda, and S. Murakami, "Fuzzy drive control of an autonomous mobile robot," *Fuzzy Sets and Systems*, **39**, 195–204, 1991.
19. Y. Nagai and N. Enomoto, "Fuzzy control of a mobile robot for obstacle avoidance," *J. Inf. Sci.*, **45**(2), 231–248, 1988.
20. M. Sugeo and M. Nishida, "Fuzzy control of model car," *Fuzzy Sets Syst.*, **16**, 103–113, 1985.
21. J. Gasos, M. C. Carcia-Alegre, and R. Garcia, "Fuzzy strategies for the navigation of automous mobile robots," *Proc. I.F.E.S.*, 1991, pp. 1024–1034.
22. H. J. Zimmermann, *Fuzzy Set Theory and its Applications*, Kluwer Academic Publishers, Boston, 1991.
23. E. H. Mamdani, "Application of fuzzy logic to approximate reasoning using linguistic synthesis," *IEEE Trans. Comput.*, **C-26**(12), 1182–1191, 1977.
24. C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller," *IEEE Trans. Syst. Man Cybern.*, **20**(2), 404–435, 1990.
25. M. Braae and D. A. Rutherford, "Fuzzy relations in a control setting," *Kybernets*, **7**(3), 185–188, 1978.
26. N. Kehtarnavaz and N. Griswold, "Establishing collision zones for obstacles moving with uncertainty," *Computer Vision, Graphics, and Image Processing*, **49**, 95–103, 1990.
27. T. C. Shia, *System Identification: Least-Squares Methods*, Lexington Books, Lexington, MA, 1977.