Collision Avoidance by a Modified Least-Mean-Square-Error Classification Scheme for Indoor Autonomous Land Vehicle Navigation

Ling-Ling Wang
Institute of Computer Science and Information Engineering
National Chiao Tung University
Hsinchu, Taiwain 30050
Republic of China

Wen-Hsiang Tsai*
Department of Computer and Information Science
National Chiao Tung University
Hsinchu, Taiwan 30050
Republic of China
Received May 17, 1990; accepted October 29, 1990

In this article, a new collision-avoidance scheme is proposed for autonomous land vehicle (ALV) navigation in indoor corridors. The goal is to conduct indoor collision-free navigation of a three-wheel ALV among static obstacles with no *a priori* position information as well as moving obstacles with unknown trajectories. Based on the predicted positions of obstacles, a local collision-free path is computed by the use of a modified version of the least-mean-square-error (LMSE) classifier in pattern recognition. Wall and obstacle boundaries are sampled as a set of 2D coordinates, which are then viewed as feature points. Different weights are assigned to different feature points according to the distances of the feature points to the ALV location to reflect the locality of path planning. The trajectory of each obstacle is predicted by a real-time LMSE estimation method. And the maneuvering board technique used for nautical navigation is employed to determine the speed of the ALV for each navigation cycle. Smooth collision-free paths found in the simulation results are presented to show the feasibility of the proposed approach.

この論文では、室内の通路で運転される自動地上移動車(ALV)のための新しい衝突回避方法を提案している。 室内において、事前に位置情報が与えられていない静止している障害物と軌道が不明な移動する障害物の間での、3輪ALVの無衝突

*To whom all correspondence should be sent.

Journal of Robotic Systems 8(5), 677–698 (1991).
 CCC 0741-2223/91/050677-22$04.00

運転の確立を目標としている。 障害物の予測位置に基づき、改良された最小二乗誤差(LMSE)分類法によるパターン認識を用いて、ローカル無衝突経路を計算している。 壁や障害物の境界は、2次元座標値のセットとしてサンプリングされ、フィーチャー・ポイントとして認識される。 経路計画の局所性を反映するために、異なるフィーチャー・ポイントには、ALVとフィーチャー・ポイントの距離により異なる重みが与えられる。 リアルタイムのLMSE予測法により、各障害物の軌道を予測する。 そして、海上航行で使われるmaneuvering board手法を用いて、ALVの各運行サイクルの速度を決定する。 提案された試みの可能性を示すために、シュミレーションのなかで見つけられたスムースな無衝突経路について解説している。

1. INTRODUCTION

Research and development of autonomous land vehicles (ALVs) have recently attracted widespread interest.[1-4] Planning a collision-free path is one of the fundamental requirements for ALV navigation. Many previous works were concerned with methods for generating a path in a known environment[5-10] or among unknown static obstacles.[11-13] However, obstacles are not always static. Some studies deal with path planning among obstacles that move along known trajectories.[14,15] In a real situation, an ALV may face unexpected moving obstacles such as human beings. Failure to consider possible intrusion of previously unknown moving obstacles prohibits flexible ALV navigation.

There exist several techniques for path planning in the presence of unknown moving obstacles. Tychonievich et al.[16] 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[17] introduced an estimation approach to predict the future positions of 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[18] decomposed the path planning process of a robot 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 robot when unexpected obstacles are encountered.

Static obstacles with no *a priori* position information, as well as moving obstacles with unknown trajectories, are considered in this study for ALV navigation. The goal is to achieve obstacle avoidance in a corridor. The global path for normal ALV navigation in a corridor is assumed to be known in advance. But the corridor environment might change due to obstacle intrusion

in every navigation cycle. Therefore, a local navigation path should be recalculated in every navigation cycle. The obstacles, both static (e.g., walls) and moving (e.g., human beings), and the corridor walls are sensed as sets of sampled 2D points. The trajectories of moving obstacles are first predicted by a real-time LMSE estimation algorithm.[19] A local collision-free navigation path for an ALV is then computed by a modified version of the LMSE classifier in pattern recognition.[20] Different weights are assigned to different 2D points according to the distances of the points to the ALV to emphasize the locality of path planning, and the speed value of the ALV is determined by the maneuvering board technique used for nautical navigation.[16] Integration of the three techniques provides a simple, fast, and intuitive approach to collision avoidance for local path planning.

In the remainder of this article, the LMSE classification method used for classifying patterns in pattern recognition, and the modified version of it for guiding an ALV in an obstacle-free corridor, are introduced in section 2. In section 3, the modified LMSE classification method is applied to ALV navigation in a corridor with unknown moving or static obstacles. The trajectory prediction of the obstacles and the speed control of the ALV for obstacle avoidance are also described. Simulation results are presented in both sections. Conclusions appear in the last section.

2. ALV NAVIGATION IN A CORRIDOR BY A MODIFIED LMSE CLASSIFICATION SCHEME

The LMSE classification method used in pattern recognition[20] to classify two different classes of patterns is modified in this study for conducting ALV navigation in a corridor. Assume that the wall on the left of the ALV has been sampled as a class of point patterns, and that on the right of the ALV as another class of point patterns. Part of the decision boundary between the two classes of patterns obtained from the modified LMSE classifier in the vicinity of the ALV is then used as the local navigation path of the ALV. Collision of the ALV with the walls is avoided automatically in this way. The remainder of this section includes a detailed description of the approach, followed by some simulation results.

2.1. Use of a Modified LMSE Classification Scheme for Finding Local ALV Navigation Paths in a Corridor

Given two classes of discrete point patterns, ω_1 and ω_2, we can use the LMSE classifier to determine a linear decision boundary, $g(\mathbf{x}) = \mathbf{w}^t\mathbf{y}$, between ω_1 and ω_2, where $\mathbf{y} = [\mathbf{x}, 1]^t$ with $\mathbf{x}$ being a feature vector. To get $\mathbf{w}$, it is assumed first that

$$\mathbf{w}^t\mathbf{y} = +1 \quad \text{if } \mathbf{x} \text{ is from class } \omega_1; \text{ and}$$

$$\mathbf{w}^t\mathbf{y} = -1 \quad \text{if } \mathbf{x} \text{ is from class } \omega_2.$$

The problem is to minimize the mean of the squared errors

$$\frac{1}{n}\left[\sum_{\mathbf{x}_j\in\omega_1}(\mathbf{w}^t\mathbf{y}_j - 1)^2 + \sum_{\mathbf{x}_j\in\omega_2}(\mathbf{w}^t\mathbf{y}_j + 1)^2\right]$$

where n is the total number of point patterns. The solution $\mathbf{w}$ can be proven[20] to be

$$\mathbf{w} = K^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

where

$$K = K_1 + K_2,$$

$$K_i = \sum_{\mathbf{x}_j\in\omega_i}\mathbf{y}_j\mathbf{y}_j^t,$$

$$\boldsymbol{\mu}_i = \sum_{\mathbf{x}_j\in\omega_i}\mathbf{y}_j.$$

To fit the need of local path planning for ALV navigation in a corridor, a modification of the LMSE classifier is used in this study, in which different weights are used to place different degrees of emphasis on the squared errors. The problem becomes the minimization of the weighted squared error

$$\frac{1}{n}\left[\sum_{\mathbf{x}_j\in\omega_1}\delta_j^1(\mathbf{w}^t\mathbf{y}_j - 1)^2 + \sum_{\mathbf{x}_j\in\omega_2}\delta_j^2(\mathbf{w}^t\mathbf{y}_j + 1)^2\right]$$

where the weights have the following properties:

$$\sum_{j=1}^{n_1}\delta_j^1 = 1,$$

$$\sum_{j=1}^{n_2}\delta_j^2 = 1$$

with n_1 and n_2 being the numbers of patterns in ω_1 and ω_2, respectively. The solution $\mathbf{w}$ can be derived to be

$$\mathbf{w} = K^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

where

$$K = K_1 + K_2,$$

$$K_i = \sum_{\mathbf{x}_j\in\omega_i}\delta_j^i\mathbf{y}_j\mathbf{y}_j^t,$$

$$\mu_i = \sum_{\mathbf{x}_j \in \omega_i} \delta_j^i \mathbf{y}_j.$$

The equation of the decision boundary is $g(\mathbf{x}) = 0$.

Shown in Figures 1 and 2 are two illustrative examples of using the modified LMSE classifier to separate two classes of patterns, where each class of patterns denotes a set of sampled points of a partial corridor wall. Intuitively, the point patterns that are closer to the ALV location should be given larger weights. Therefore, the weights may be taken to be inversely proportional to the distances between the point patterns and the ALV. Figure 1 shows the result of applying the classifier to part of a straight corridor, and Figure 2 shows that of applying the classifier to a right-angle turning in a corridor. The portion of the decision boundary between the left and right walls in the vicinity of the ALV can be used to determine a collision-free local path for ALV navigation in the corridor. This indicates the feasibility of using the modified LMSE classifier for local path computation and collision avoidance in indoor environments.

2.2. Simulation Results

A three-wheel, rectangular-shaped ALV is considered in the simulation, where the front wheel has driving power and the two rear wheels are free. A criterion for adjusting the front wheel direction is required so as to keep the ALV free from collision with the walls. Once the local navigation path (i.e., the portion of the decision boundary between the left and right walls in the vicinity of the ALV) is determined, the ALV will first turn and then move toward the path if the ALV is not right on the path. A table look-up control strategy

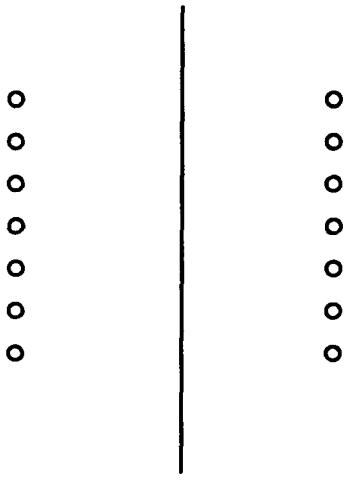


Figure 1. Use of the modified LMSE classifier to find a decision boundary between the left and the right walls in a corridor.

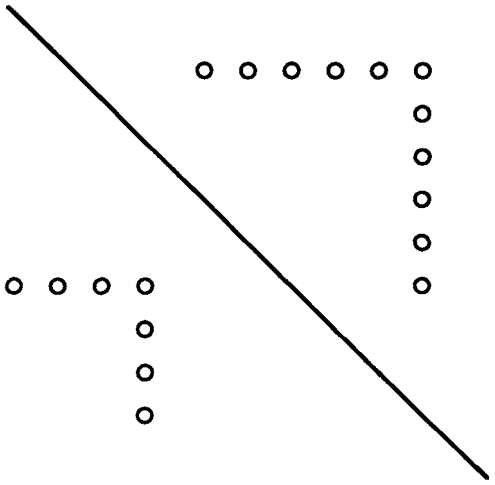


Figure 2. Use of the modified LMSE classifier to find a decision boundary between a pair of right-angle-turn walls in a corridor.

derived in ref. 21 is employed to obtain the front wheel direction of the ALV after the local navigation path is computed in each navigation cycle.

Figure 3 shows the top view of the walls of a corridor, including a right-angle turning. The rectangle in the figure denotes an ALV that travels along the corridor with a constant speed. The corridor wall model is stored in advance in

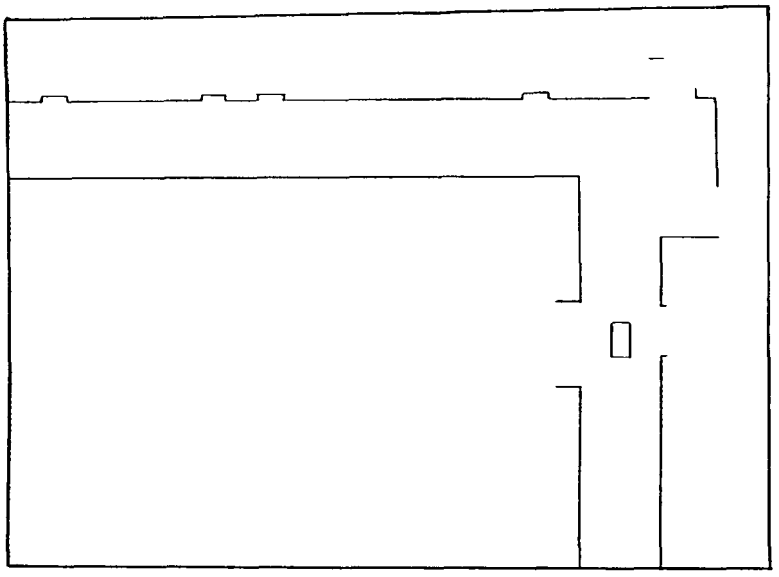


Figure 3. The top view of the corridor in a building.

the ALV. In addition, a global path is needed for the ALV to travel in the corridor. Otherwise, it will not be able to decide whether to turn left or right when a turning corner is faced. Attached to each line segment (corresponding to the top view of a wall) in the model is a flag to indicate whether the ALV should be on the left or right side of the line segment while the ALV is traveling along the part of the corridor represented by the line segment. Together, these flags indicate a global path for the ALV in the corridor.

In each navigation cycle, the left and right walls in the vicinity of the ALV are imaged, extracted, and sampled as two classes of point patterns. The point patterns that are closer to the current ALV location are given larger weights (in proportion to the inverse of the point distance to the ALV location). And the computed decision boundary is used as the local navigation path of the ALV. Figure 4 shows simulated snapshots of the ALV at four different positions in the corridor, where the boldfaced line segments representing the wall portions around the ALV are used to determine the local navigation path. The continuous navigation trajectory of the ALV in the corridor is shown in Figure 5. The trajectory is seen to be well away from the walls. On an average, the computation time for determining the local navigation path in each navigation cycle is about 0.2 s, using a PC/AT computer with an 80386-33 CPU.

3. ALV NAVIGATION AMONG MOVING OR STATIC OBSTACLES

In a real situation, when the ALV travels in the corridor, there may appear some unknown moving obstacles, e.g., human beings, which may collide with the ALV. To handle such a situation, the trajectories of the moving obstacles must be predicted first, and a safe local navigation path of the ALV is then computed according to the predicted trajectories of the obstacles and the environment of the corridor. It is desired that the prediction process be as accurate as possible without expensive computation. Linear prediction models usually given simple but effective solutions, especially for local collision-free path computation. The real-time LMSE estimation algorithm[19] is thus employed in this study to approximate and predict the trajectories of moving obstacles. The modified LMSE classifier is then used again to determine the local navigation path of the ALV.

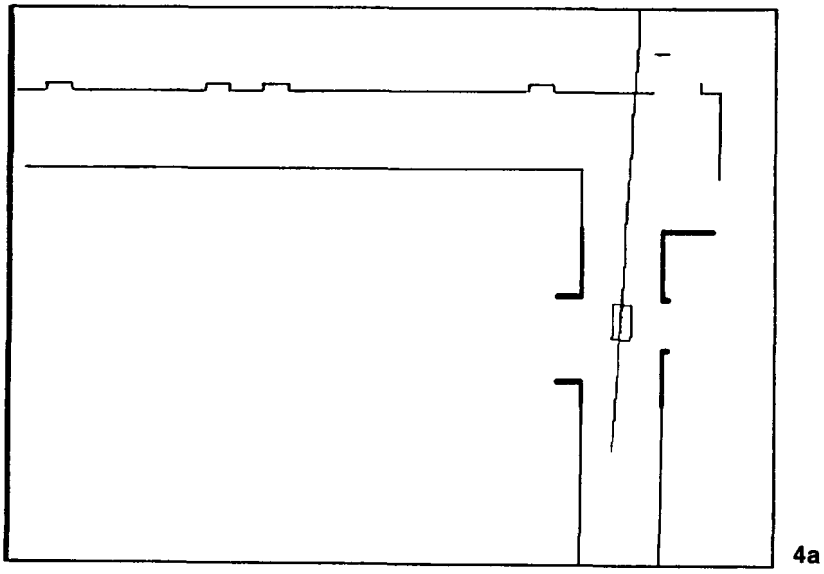
3.1. A Review on the Real-Time LMSE Estimation Algorithm

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:

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



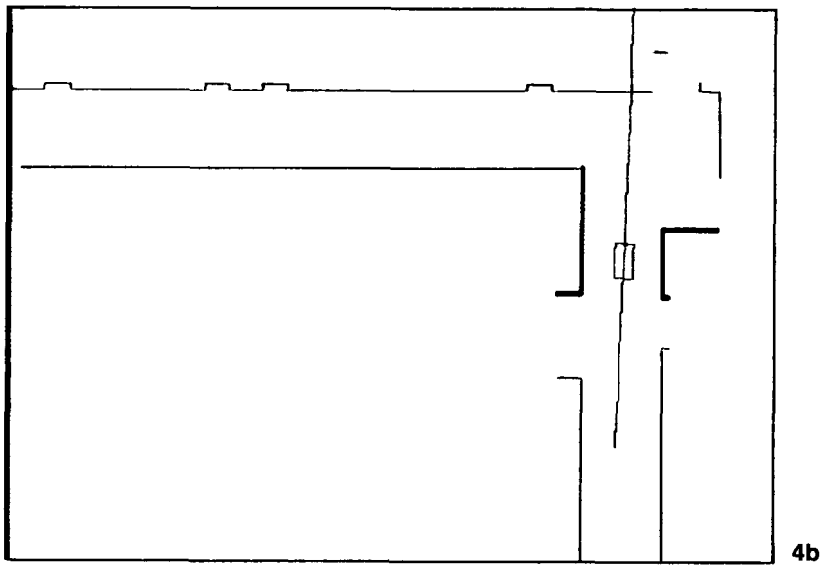
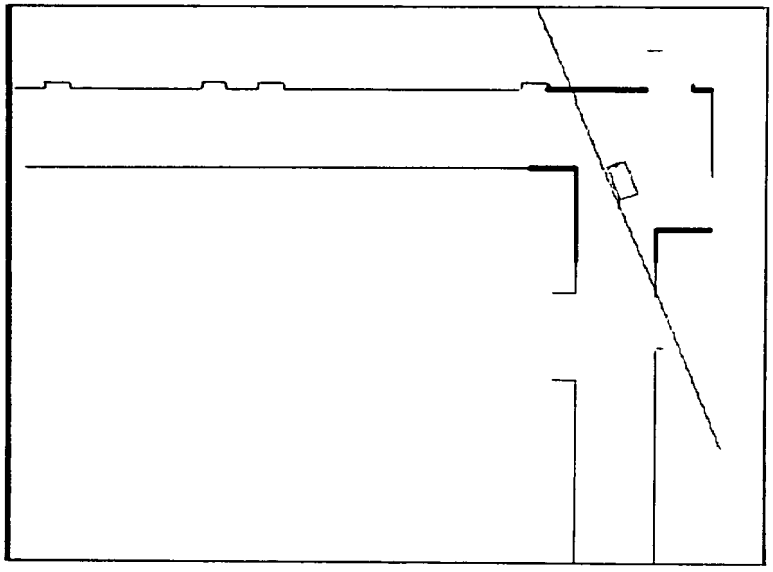


Figure 4. Simulated snapshots of an ALV at four different positions in the corridor of Figure 3. (a) An ALV is running along a straight corridor. (b) The ALV is approaching a right-angle turning in the corridor. (c) The ALV is turning left in the corridor. (d) The ALV has turned left in the corridor.



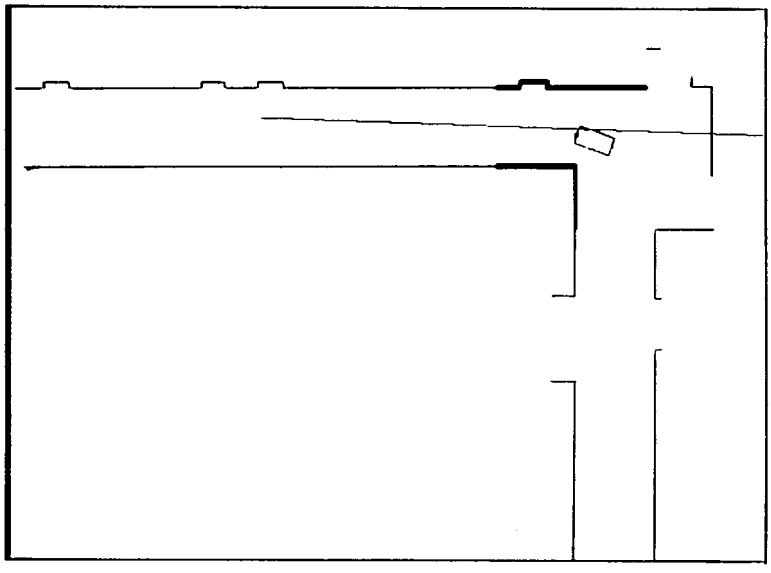


Fig. 4 *Continued*

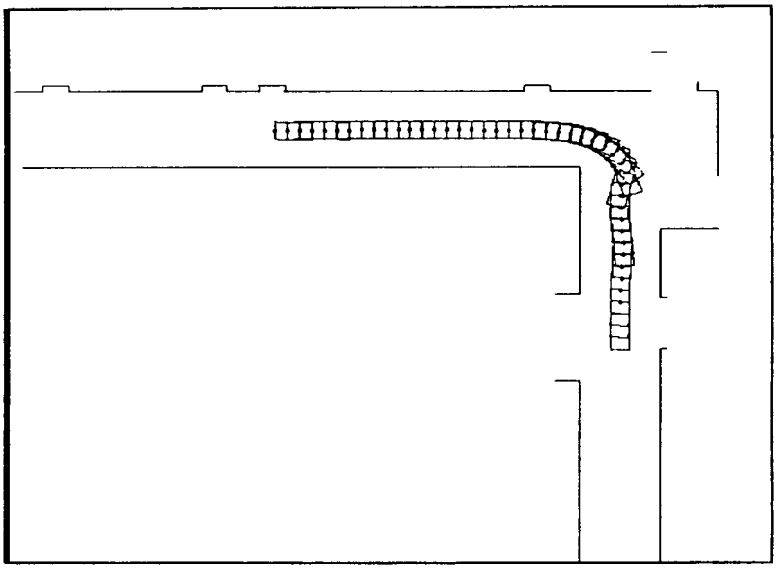


Figure 5. The continuous trajectory of the ALV navigation shown in Figure 4.

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

$$E_m = X_m - T_m A_m.$$

Let the optimal solution, $\hat{A}_m$, be defined as the one such that the criterion function J_m

$$J_m = \sum_{i=1}^{m} e_i^2$$

is minimized. $\hat{A}_m$ can be solved[19] to be

$$\hat{A}_m = (T_m^t T_m)^{-1} T_m^t X_m. \tag{2}$$

When fresh experimental data are continuously supplied, it is desired to improve the parameter estimates by making use of the new information. If a recursive formula can be obtained, the estimates can be updated step by step without repeatedly computing the solution of eq. (2). For this, first define matrix P_m as

$$P_m = (T_m^t T_m)^{-1}.$$

Then it can be proven[19] that the solution $\hat{A}_{m+1}$ for the $(m+1)$th time instant can be computed from that of the mth time instant, $\hat{A}_m$, as follows;

$$\hat{A}_{m+1} = \hat{A}_m + \gamma_{m+1} P_m \mathbf{t}_{m+1} \{x_{m+1} - \mathbf{t}_{m+1}^t \hat{A}_m\},$$

$$P_{m+1} = P_m - \gamma_{m+1} P_m \mathbf{t}_{m+1} \mathbf{t}_{m+1}^t P_m$$

where

$$\mathbf{t}_{m+1} = \begin{bmatrix} t_{m+1} \\ 1 \end{bmatrix},$$

$$\gamma_{m+1} = \frac{1}{1 + \mathbf{t}_{m+1}^t P_m \mathbf{t}_{m+1}}.$$

The real-time LMSE estimation algorithm is a simple modification of the above least-squares estimation algorithm in which an exponential weighting scheme is used to place heavier emphasis on more recent data. Such a type of data emphasis is appropriate for the locality property of collision-free path planning studied in this research. In the algorithm, the error function is defined as

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

in which 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. P_m is defined as

$$P_m = \frac{1}{\lambda}(T_m^t T_m)^{-1}.$$

Then it can be shown[19] that

$$\hat{A}_{m+1} = \hat{A}_m + \gamma_{m+1} P_m \mathbf{t}_{m+1}\{x_{m+1} - \mathbf{t}_{m+1}^t \hat{A}_m\},$$
$$P_{m+1} = \frac{1}{\lambda}\{P_m - \gamma_{m+1} P_m \mathbf{t}_{m+1} \mathbf{t}_{m+1}^t P_m\},$$

where

$$\mathbf{t}_{m+1} = \begin{bmatrix} t_{m+1} \\ 1 \end{bmatrix},$$

$$\gamma_{m+1} = \frac{1}{1 + \mathbf{t}_{m+1}^t P_m \mathbf{t}_{m+1}}.$$

To apply the above real-time LMSE estimation algorithm to local obstacle trajectory 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

$$\begin{aligned} x_i &= at_i + b \qquad i = 1, 2, \ldots, m \\ y_i &= ct_i + d \qquad i = 1, 2, \ldots, m \end{aligned} \tag{4}$$

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. (3) 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. Then we can predict the positions of the obstacle in the next cycles using these estimated parameters and the equations in (4).

3.2. Use of the Modified LMSE Classification Scheme for ALV Navigation Among Moving Obstacles

The modified LMSE classifier can also be used to plan a local collision-free path for an ALV in a corridor or in a planar workspace populated with obstacles. Again, the left and right corridor walls around the ALV are imaged, extracted, and sampled as two classes of patterns. For a moving obstacle, its current position and three-step-ahead predicted positions are regarded as a third class of patterns. The decision boundary between the left corridor wall and the obstacle and that between the right corridor wall and the obstacle are found by the modified LMSE classifier. For each decision boundary, the turn-

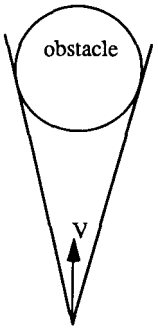


Figure 6. The avoidance cone for a static obstacle.

ing direction of the ALV is computed by the table look-up control strategy found in ref. 21. The decision boundary that results in the smallest turning angle for the ALV to adjust is selected as the local navigation path in the next navigation cycle so that the ALV will deviate as little as possible from its current location, while still being able to avoid collision with the obstacles.

3.3. Determination of ALV Speed

Once the local navigation path is determined, the turning direction of the ALV is computed by the table look-up control strategy found in ref. 21. If we can dynamically vary the speed of the ALV, the probability of collision will decrease. In the following, the maneuvering board technique used in nautical navigation[16] is employed to determine the speed of the ALV.

The basic maneuvering board technique is illustrated in Figures 6 and 7, in which obstacles are represented as enlarged circles such that the ALV can be represented as a point. Figure 6 shows the avoidance cone for a static obstacle, which is defined by two tangent lines from the ALV to the circle representing the obstacle. The movement direction of the ALV from one position to the next and its speed value form a velocity vector. A velocity vector is represented as

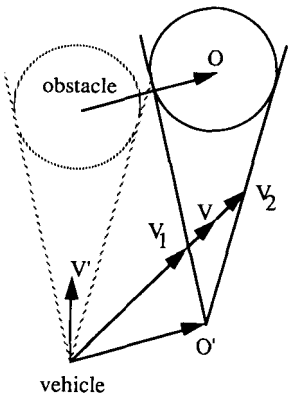


Figure 7. The avoidance cone for a moving obstacle.

an arrow in Figures 6 and 7. A vector is said to fall within an area if the vector head is located in the area. Clearly, any velocity vector V that falls within the avoidance cone will result in a collision of the ALV with the obstacle.

The avoidance cone for a moving obstacle is shown in Figure 7 (consisting of the solid lines). Any velocity vector V of the ALV falling within this cone represents a collision course with the moving obstacle. The reason is that any such velocity vector V is the vector sum of a velocity vector V', which represents a collision vector if the obstacle were static, and another velocity vector O', which is the velocity vector O of the obstacle translated to the ALV location. That is, if the ALV runs with the direction of the vector V and at a speed between $|V_1|$ and $|V_2|$, as shown in Figure 7, the ALV will collide with the obstacle. Here, the $|V_1|$ and $|V_2|$ values are called the boundary speeds.

For the ALV, we set a default speed value. In each navigation cycle, after the turning direction of the ALV is determined and the corresponding boundary speeds are computed, we can check if the default speed lies between the two boundary speeds. If not, the default speed is used as the ALV speed in the next navigation cycle. Otherwise, the boundary speed, which is closer to the current ALV speed, is used.

3.4. Static Obstacle Avoidance

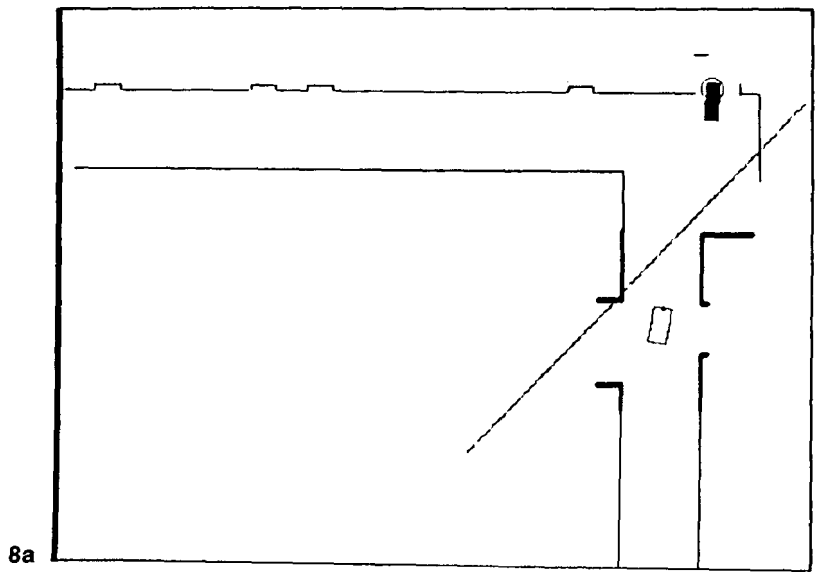
The previous scheme proposed for moving obstacle avoidance can also be applied to static obstacles. In this case, just regard the predicted positions of a static obstacle to be all the same as the current object position, and consider the current object position as a single-pattern class. In fact, it is not necessary to distinguish static obstacles from moving ones in the proposed collision-avoidance scheme.

3.5. Simulation Results of Moving or Static Obstacle Avoidance

An example illustrating how an ALV avoids collision with an unknown moving obstacle in a corridor is first given. Shown in Figure 8 are several simulated snapshots of an ALV and a moving obstacle at different positions, where the unfilled rectangle represents the ALV, the four small black rectangles denote the current position and the predicted obstacle positions of three steps ahead, and the circle denotes the current position of the obstacle. Figure 9 gives the top views of the continuous motions of the ALV and the obstacle at four time instants, illustrating the computed collision-free trajectory of the ALV. Shown in Figures 10 and 11 are two examples illustrating how an ALV avoids a static obstacle in a corridor. On average, the computation time for determining the local navigation path in each cycle is about 0.4 s in the case of avoiding a moving obstacle.

4. CONCLUSION

Several techniques have been integrated in this study to provide a collision avoidance scheme for ALV navigation. The real-time LMSE estimation algo-



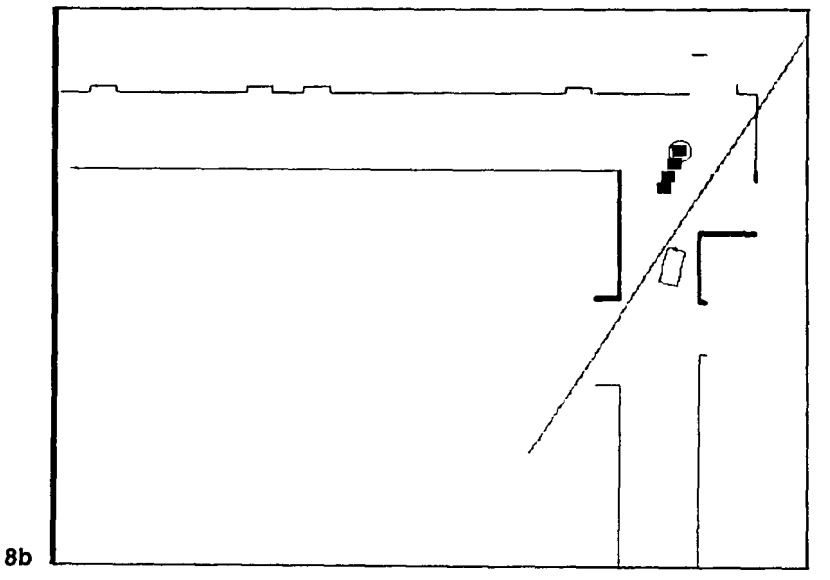
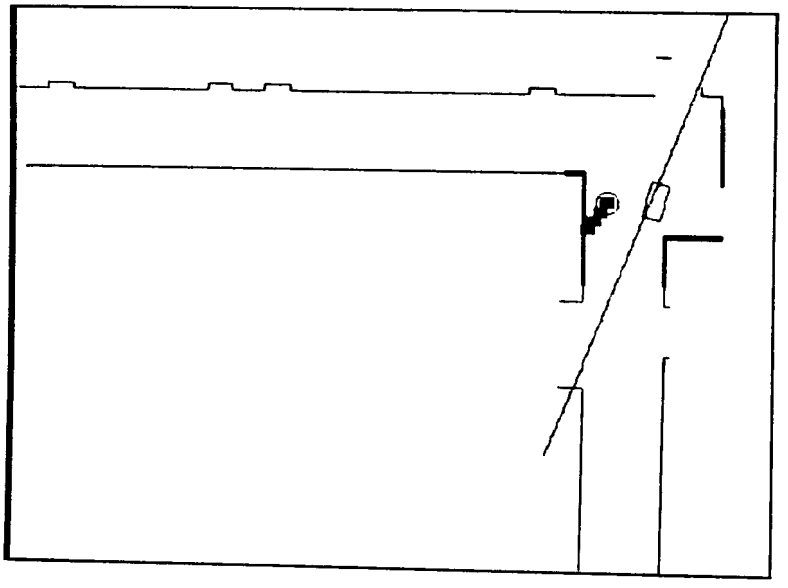
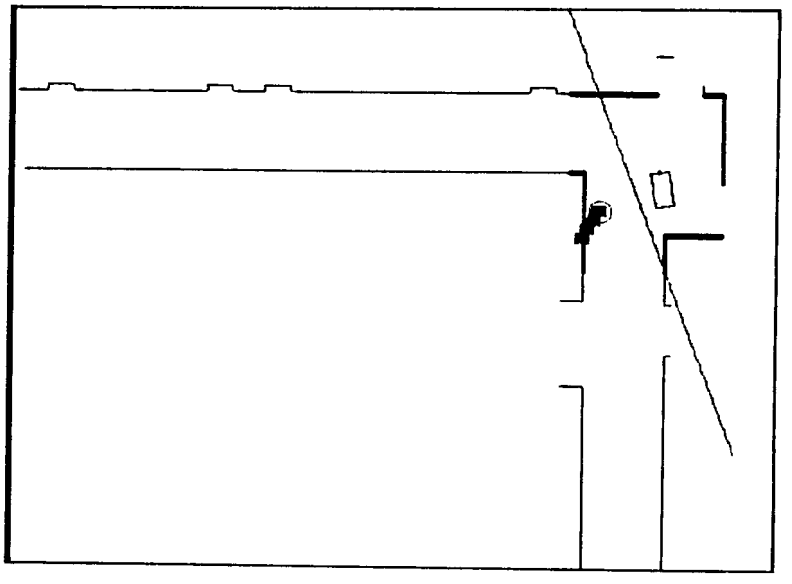


Figure 8. Simulated snapshots of an ALV and a moving obstacle at six different positions. (a) An ALV and an obstacle encounter in the corridor. (b) The ALV is approaching the obstacle. (c) The ALV has avoided the obstacle. (d) The ALV is going to turn left. (e) The ALV is turning left. (f) The ALV has turned left.

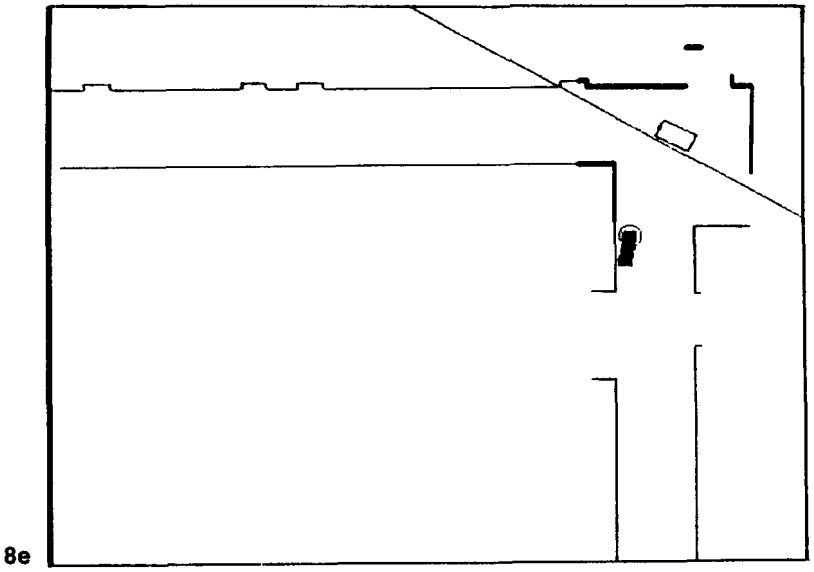


8c

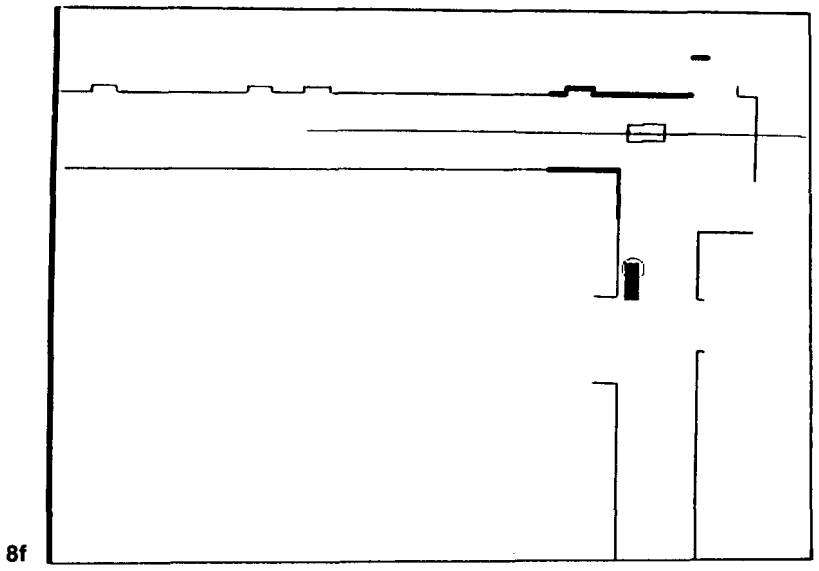


8d

Fig. 8 *Continued*



8e



8f

Fig. 8 *Continued*

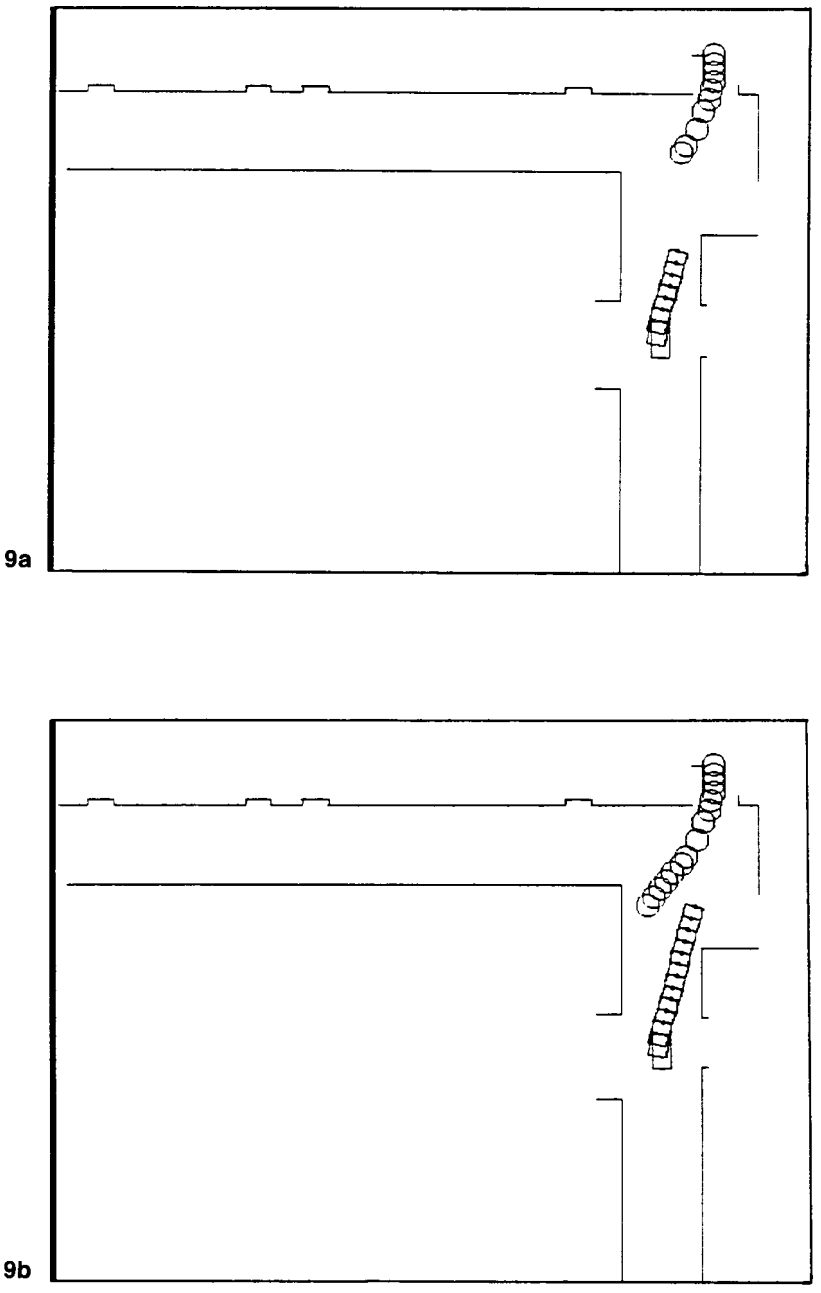
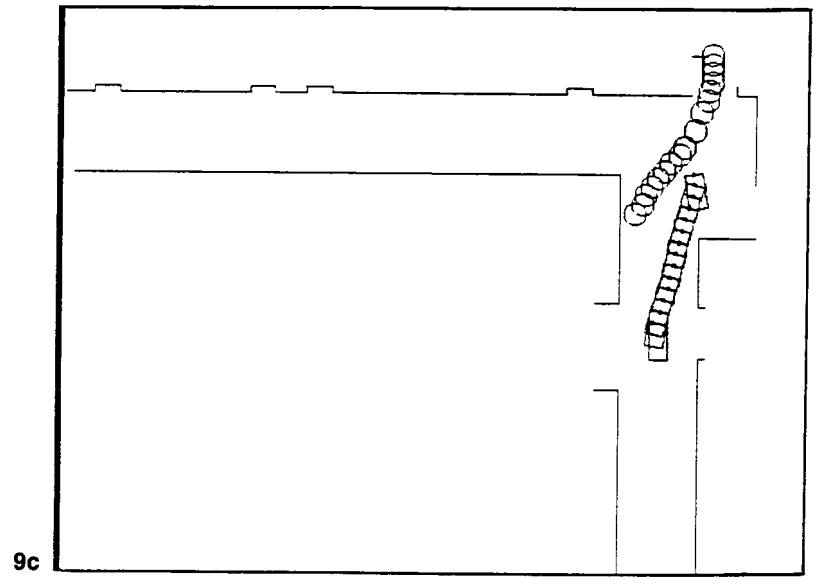
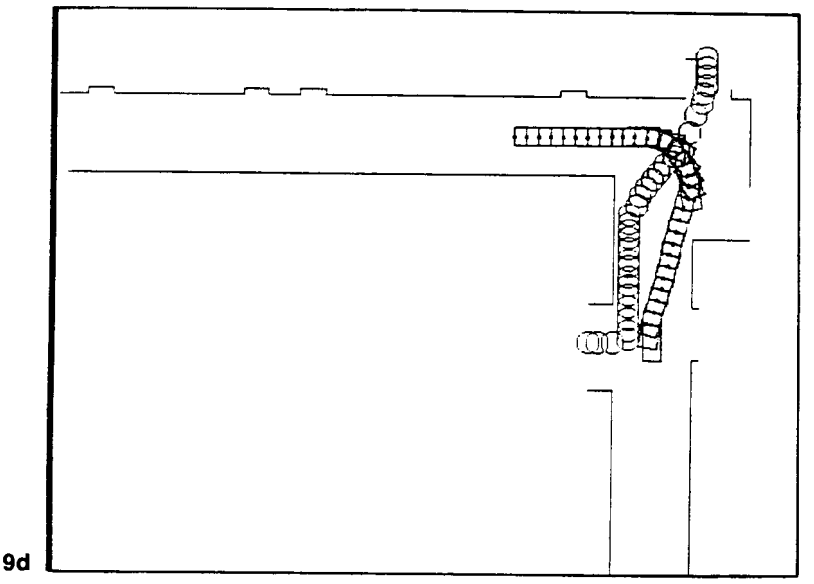


Figure 9. Continuous trajectories of the ALV navigation and the obstacle movement of Figure 8. (a) In the first 6 navigation cycles. (b) In the first 11 navigation cycles. (c) In the first 13 navigation cycles. (d) In the first 29 navigation cycles.



9c



9d

Fig. 9 *Continued*

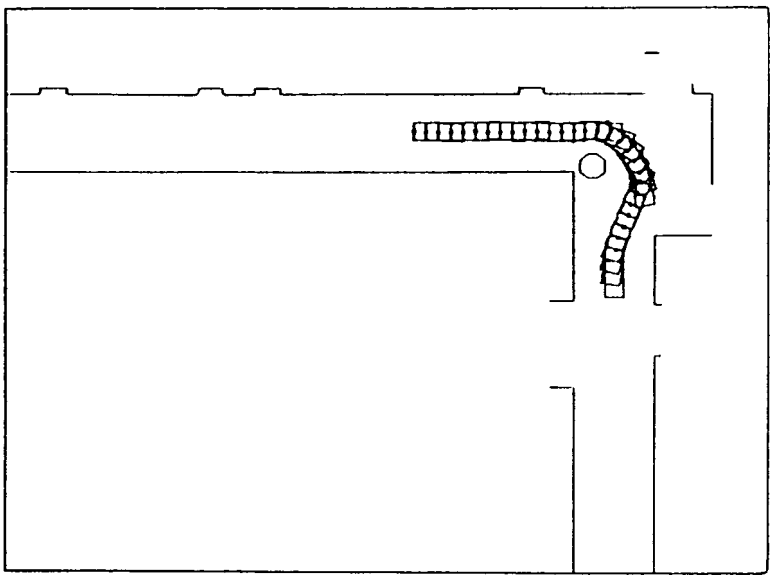


Figure 10. Continuous trajectory of an ALV having avoided a static obstacle (case 1).

rithm has been employed to predict the motions of obstacles. A modified version of the LMSE classifier used in pattern recognition has been employed to plan a collision-free local path among obstacles. The maneuvering board technique, used for nautical navigation, has been used to determine the speed of the ALV. In addition, a realistic control strategy on the translation and rotation of

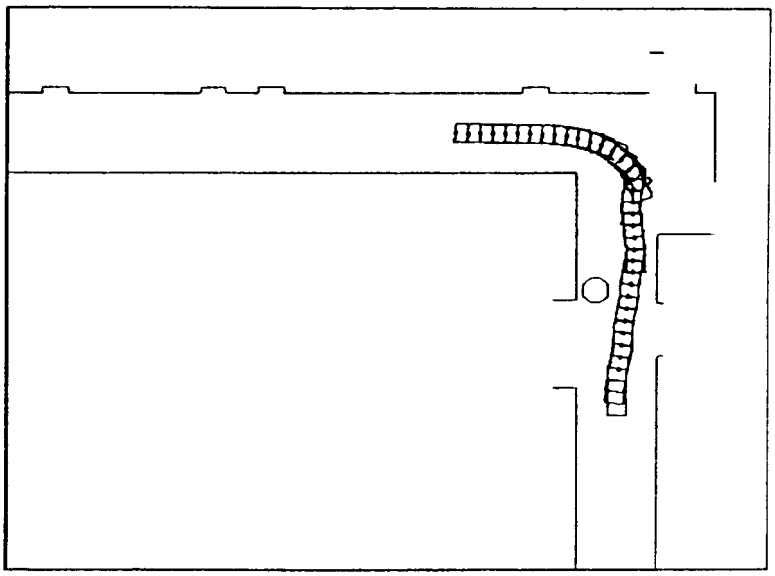


Figure 11. Continuous trajectory of an ALV having avoided a static obstacle (case 2).

a three-wheel ALV has been considered in the simulation. Combination of these techniques provides a feasible collision-avoidance scheme for guiding an ALV in a corridor to avoid unknown moving or static obstacles, as shown by the simulation results.

Further research may be directed to practically implementing the proposed approach on an experimental autonomous vehicle. Some urgent responses of the vehicle must also be considered when the computed collision-free path is too narrow for the vehicle to pass or when the vehicle is too close to the obstacle.

This work was supported by the National Science Council, Republic of China under Grant NSC79-0404-E009-18.

References

1. H. P. Moravec, "The Stanford cart and the CMU rover," *Proc. IEEE,* **71**(7), (1983).
2. R. L. Madarasz, L. C. Heiny, R. F. Cromp, and N. M. Mazur, "The design of an autonomous vehicle for the disabled," *IEEE J. Robotics Automat.,* **2**(3), (1986).
3. A. M. Waxman, "A visual navigation system for autonomous land vehicle," *IEEE J. Robotics Automat.,* **3**(2), (1987).
4. C. Thorpe, M. H. Hebert, T. Kanade, and S. A. Shafer, "Vision and navigation for the Carnegie–Mellon NAVLAB," *IEEE Trans. Pattern Analysis Machine Intell.,* **10**(3) (1988).
5. O. Takahashi and R. J. Schilling, "Motion planning in a plane using generalized Voronoi diagrams," *IEEE Trans. Robotics Automat.* **5**(2), (1989).
6. R. C. Arkin, "Navigational path planning for a vision-based mobile robot," *Robotica,* **7,** (1989).
7. G. T. Wilfong, "Motion planning for an autonomous vehicle," in *Proc. IEEE Int. Conf. Robotics Automat.,* Philadelphia, PA, 1988.
8. S. Kambhampati and L. S. Davis, "Multiresolution path planning for mobile robots," *IEEE J. Robotics Automat.,* **2**(3), (1986).
9. D. T. Kuan, J. C. Zamiska, and R. A. Brooks, "Natural decomposition of free space for path planning," in *Proc. IEEE Int. Conf. Robotics Automat.,* St. Louis, MO, 1985.
10. R. A. Brooks, "Solving the find-path problem by good representation of free space," *IEEE Trans. Syst., Man, Cybernet.,* **13**(3), (1983).
11. R. C. Arkin, "Motor schema-based mobile robot navigation," *Int. J. Robotics Res.,* **8**(4), (1989).
12. J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Trans. Syst., Man, Cybernet.,* **19**(5), (1989).
13. A. C. -C. Meng, "Free space modeling and geometric motion planning under unexpected obstacles," in *Proc. IEEE Int. Conf. Artific. Intell. Applicat.,* San Diego, CA, March 1988.
14. K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *Int. J. Robotics Automat.,* **5**(3), (1986).
15. K. Fujimura and H. Samet, "A hierarchical strategy for path planning among moving obstacles," *IEEE Trans. Robotics Automat.* **5**(1), (1989).
16. L. Tychonievich, et al., "A maneuvering-board approach to path planning with moving obstacles," in *Proc. 11th Int. Joint Conf. Artific. Intell.,* Detroit, MI, 1989.
17. N. Kehtarnavaz and S. Li, "A collision-free navigation scheme in the presence of moving obstacles," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recog.,* Ann Arbor, MI, 1988.

18. J. P. H. Steele and G. P. Starr, "Mobile robot path planning in dynamic environments," in *Proc. IEEE Int. Conf. Systems, Man, Cybernet.*, Beijing and Shenyang, China, 1988.
19. T. C. Hsia, *System Identification: Least-squares Methods*, Lexington Books, Lexington, MA, 1977.
20. P. A. Devijver and J. K. Kittler, *Pattern Recognition: A Statistical Approach*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
21. P. Y. Ku and W. H. Tsai, "Model-based guidance of autonomous land vehicles for indoor navigation," in *Proc. of 1989 Workshop on Computer Vision, Graphics and Image Processing*, Taipei, Taiwan, ROC, 1989.