Correspondence

Obstacle Avoidance for Autonomous Land Vehicle Navigation in Indoor Environments by Quadratic Classifier

Ching-Heng Ku and Wen-Hsiang Tsai

***Abstract*— A vision-based approach to obstacle avoidance for autonomous land vehicle (ALV) navigation in indoor environments is proposed. The approach is based on the use of a pattern recognition scheme, the quadratic classifier, to find collision-free paths in unknown indoor corridor environments. Obstacles treated in this study include the walls of the corridor and the objects that appear in the way of ALV navigation in the corridor. Detected obstacles as well as the two sides of the ALV body are considered as patterns. A systematic method for separating these patterns into two classes is proposed. The two pattern classes are used as the input data to design a quadratic classifier. Finally, the two-dimensional decision boundary of the classifier, which goes through the middle point between the two front vehicle wheels, is taken as a local collision-free path. This approach is implemented on a real ALV and successful navigations confirm the feasibility of the approach.**

***Index Terms*— ALV navigation, collision-free path, computer vision, obstacle avoidance, obstacle detection, pattern recognition, quadratic classifier.**

I. Introduction

A. Survey of Related Studies

In recent years, autonomous land vehicles (ALV's) have been studied intensively. How to guide the ALV to navigate in a certain environment and avoid obstacles in the mean time is the major goal. In the study of obstacle avoidance, two cases can be identified, namely, navigation in a known environment or in an unknown one. In a known environment, the vehicle usually generates a collision-free path in an off-line phase using the map of an environment knowledge base and a certain scheme of path planning, such as the A^* and the breadth-first search algorithms in Hyland and Fox [1], the dynamic programming algorithm in Cesarone and Eman [2], the use of visibility graphs in Acosta and Moras [3], and the potential field method in Kim and Khosla [4]. Alternatively, some on-line methods have also been proposed using the information of the global environment to generate a collision-free path in real time, such as the uses of the cubic function in Onoguchi *et al.* [5], the B-spline curves in Yang [6], and the least-mean-square-error classification in Wang and Tsai [7].

In an unknown environment, the vehicle can only use locally observed features to generate a collision-free path. Some approaches to achieving the goal of safe navigation in an unknown environment have been proposed, such as the method of combining certainty grids and potential fields in Borenstein and Koren [8], and the use of the Bug2 and Tarry algorithms in Skewis and Lumelsky [9]. Besides, Bauer *et al.* [10] used the properties of geometry and kinematics for wheel control in an unknown environment.

Manuscript received August 27, 1996; revised September 18, 1998. This work was supported by the National Science Council under Grant NSC84-2213-E009-122.

The authors are with the Department of Computer and Information Science, National Chiao Tung University, Hsinchu 300, Taiwan R.O.C.

Publisher Item Identifier S 1083-4419(99)03543-8.

Some vision-based navigation methods [14], [15] for mobile robots with obstacle avoidance capability have also been proposed. Ohya [14] used a model edge map for vehicle navigation on a planned path. Obstacles are detected by computing the difference between the edges estimated from the three-dimensional (3-D) environment model and the edges detected from the actual camera image. The system [15] consists of three independent vision modules, the edge module, the RGB module, and the HSV module, for obstacle detection. The obstacle boundaries from the individual modules are combined into a single obstacle boundary which is converted to motor commands. Yang [16] used an adaptive-network-based fuzzy classifier to define the three-dimensional obstacle regions that must be avoided. Biewald [17] used a human-like conception and a more qualitative world model to plan routes.

In this study, an algorithm using the quadratic classifier in pattern recognition [11] is proposed for collision avoidance for ALV navigation in an unknown indoor environment. This algorithm employs real-time operations to find safe collision-free paths. A local navigation path is calculated for each navigation cycle. A difference from the fuzzy classifier method [16] is that the path is a two-dimensional quadratic decision boundary of the classifier by which the vehicle can be guided smoothly. The boundary is generated by treating all obstacles and the two sides of the vehicle body as patterns. A systematic method is proposed for separating the patterns into two classes for use as the design sample input for the classifier.

B. Overview of Proposed Approach

The first step of the proposed obstacle avoidance approach for ALV navigation in a corridor environment is to analyze the image captured by a monocular camera to find out obstacles. Second, a collision-free path is generated by the distribution of obstacles. Finally, a precise turning angle is obtained for ALV wheel control at the current position using the collision-free path. These steps are executed cyclically. The system flowchart is shown in Fig. 1. The details are described as follows.

1) Step 1: Image acquirement: The front view of the vehicle is captured by a wide-angle (8 mm) camera mounted on the vehicle. The image is used to extract relevant information of the unknown environment.
2) Step 2: Obstacle detection: Image points that compose the baselines of the obstacles, including the walls of the corridor and the objects that appear in the way of ALV navigation in the corridor, are detected using an obstacle detection algorithm introduced in Section IV.
3) Step 3: Coordinate transformation: The coordinates of the image points obtained in Step 2 in the image coordinate system (ICS) are transformed into the space coordinates in the vehicle coordinate system (VCS) using some formulas introduced in Section II-D. These space coordinates are all on the x-y plane in the VCS.
4) Step 4: Pattern generation: The patterns representing the obstacles and the two sides of the vehicle body are clustered into two classes using a pattern generation algorithm introduced in Section II-B. In additional to the two classes of patterns, some series of additional points are included as patterns using a pattern addition algorithm described in Section II-C. The

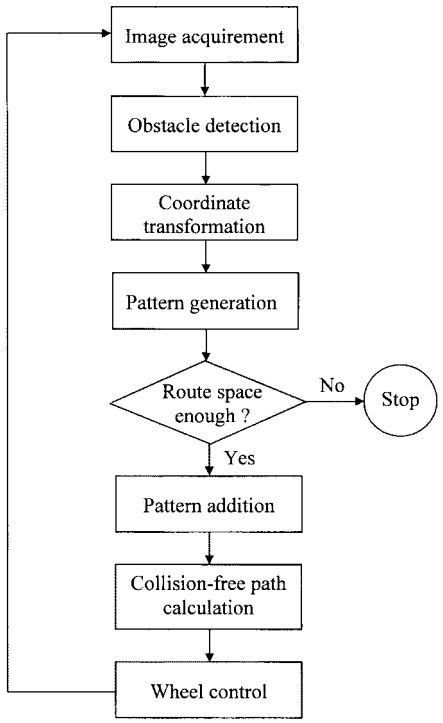


Fig. 1. System flowchart.

additional patterns are generated according to the locations of obstacle patterns and the vehicle width in this study. Besides, whether there exists enough route space to allow the vehicle to go through or not is also checked. If yes, continue; otherwise, stop the vehicle.

5) Step 5: Collision-free path calculation: A quadratic collision-free path is generated using some formulas derived in Section II-A. The path is the decision boundary of the quadratic classifier designed with input patterns coming from Step 4.
6) Step 6: Wheel control: Proper control of the ALV front wheels is executed according to the turning angle computed with some formulas derived in Section III based on the collision-free path generated in Step 5. In this way, the vehicle keeps its trajectory on a collision-free path continually.
7) Step 7: Repeat Steps 1–6 until there is no path way to go through.

In the processes described above, the proposed system does not use any environment knowledge set up in advance and the guidance of the ALV is based on local visual information. At least three advantages are found in this approach. First, the curve properties of the quadratic path are used to match the kinematic trajectory of the vehicle. Second, by following the quadratic path it is smoother to go through unknown environments with obstacles than other approaches using linear paths. Third, the current location of the vehicle is taken into consideration while generating the quadratic collision-free path.

In the remainder of this study, the proposed ALV navigation method is described in Section II, including a review of the quadratic classifier in pattern recognition, the use of the proposed approach to find collision-free paths, the generation and addition of patterns, and the coordinate transformations of patterns. The vehicle wheel control according to the turning angle for path following is described in Section III. The image processing techniques employed for obstacle detection are described in Section IV. Experimental results are found in Section V. And conclusions are given in Section VI.

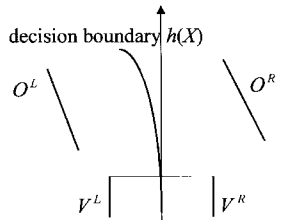


Fig. 2. Two-dimensional decision boundary, $h(X)$, passing through the origin of the vehicle coordiante system.

II. Proposed ALV Navigation Method

A. Principle of Using Quadratic Classifier for Finding ALV Navigation Paths for Obstacle Avoidance

From the result of the pattern generation processes described in Sections II-B and II-C, we obtain two pattern groups, L and R, each of which includes patterns representing obstacles, corridor walls on one side of the vehicle, and the two sides of the vehicle body. We use the quadratic classifier to determine a quadratic decision boundary, $h(X)$, between L and R. The decision boundary is then taken as a collision-free path which the ALV follows to achieve safe navigation. The path does not go through the area that consists of the patterns of L or R, and is constrained to go through the middle point between the two front wheels of the vehicle in this study. See Fig. 2 for an illustration, where O^L and O^R are patterns representing obstacles, and V^L and V^R are patterns representing the vehicle body sides. Each pattern of the two pattern groups consist of x and y values in the vehicle coordinate system (VCS). We denote the coordinates of the ith pattern in L as $[x_i^L \;\; y_i^L]^T$ and those of the jth pattern in R as $[x_j^R \;\; y_j^R]^T$.

According to the theory of pattern recognition, we can find a quadratic decision boundary between the patterns of two classes to form a quadratic classifier. A general representation of the quadratic classifier is as follows:

$$\begin{aligned} h(X) &= X^T Q X + V^T X + v_0 \\ Q &= \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix} \\ V &= \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \end{aligned} \tag{1}$$

where v_0 is a constant, and the vector $X = [x_1 \;\; x_2]^T$ specifies a pattern of L or R. If $h(X) < 0$, it means that X belongs to L; if $h(X) > 0$, it means that X belongs to R; and if $h(X) = 0$, it means that X falls on the decision boundary.

In this study, we project patterns from the 3-D space onto a plane represented by the corridor floor, so the dimension of patterns are reduced from three to two. Besides, since it is constrained that the collision-free path goes through the middle point between the two front wheels of the vehicle, i.e., the origin of the VCS, the value of v_0 in (1) can be set to zero. Now (1) can be represented as follows:

$$\begin{aligned} h(X) &= X^T Q X + V^T X \\ &= \sum_{i=1}^{2} \sum_{i=1}^{2} q_{ij} x_i x_j + \sum_{i=1}^{2} v_i x_i \\ &= \sum_{i=1}^{3} \alpha_i y_i + \sum_{i=1}^{2} v_i x_i \\ &= A^T Y + V^T X \\ &= [\alpha_1 \;\; \alpha_2 \;\; \alpha_3 \;\; v_1 \;\; v_2] \cdot [y_1 \;\; y_2 \;\; y_3 \;\; x_1 \;\; x_2]^T \end{aligned} \tag{2}$$

where

$$A = [\alpha_1 \quad \alpha_2 \quad \alpha_3]^T = [q_{11} \quad q_{12} \quad + q_{21} \quad q_{22}]$$
$$X = [x_1 \quad x_2]^T \quad \text{and}$$
$$Y = [y_1 \quad y_2 \quad y_3]^T = [x_1^2 \quad x_1 x_2 \quad x_2^2]^T.$$

Note that in (2), the originally quadratic equation is simplified to a linear one, so we can use the design technique for linear classifiers to find the coefficients $(\alpha_1, \alpha_2, \alpha_3, v_1, v_2)$.

Let $M = [Y^T \; X^T] = [y_1 \; y_2 \; y_3 \; x_1 \; x_2]^T$. The values of matrix M come from those of the patterns of L and R. Matrix M for L and R will be denoted as M^L and M^R, respectively. If the values of x_1 and x_2 in matrix M are substituted by those of the ith point of $L, x_1 = x_i^L$ and $x_2 = y_i^L$, we denote the resulting matrix M by M_i^L. Similarly, if the values of x_1 and x_2 in matrix M are substituted by those of the ith point of $R, x_1 = x_i^R$ and $x_2 = y_i^R$, we denote the resulting matrix M by M_i^R.

Accordingly, the nonlinear solutions of the coefficients for the quadratic classifier whose input design patterns come from the points of L and R are equal to the linear solutions of the coefficients for the linear classifier whose input design patterns come from the points of M^L and M^R. So, the five coefficients of (2) can be solved and the result is as follows [11]:

$$[\alpha_1 \quad \alpha_2 \quad \alpha_3 \quad v_1 \quad v_2]^T = [\tfrac{1}{2}K_L + \tfrac{1}{2}K_R]^{-1}(D_R - D_L) \quad (3)$$

where D_L and D_R, and K_L and K_R are the means and variances of M^L and M^R, respectively, computed as follows:

$$D_L = \frac{1}{m}\sum_{i=1}^{m} M_i^L = \frac{1}{m}\sum_{i=1}^{m}[(x_i^L)^2 x_i^L y_i^L (y_i^L)^2 x_i^L y_i^L]^T$$
$$D_R = \frac{1}{n}\sum_{i=1}^{n} M_i^R = \frac{1}{n}\sum_{i=1}^{n}[(x_i^R)^2 x_i^R y_i^R (y_i^R)^2 x_i^R y_i^R]^T$$
$$K_L = \frac{1}{m}\sum_{i=1}^{m}(M_i^L - D_L)\cdot(M_i^L - D_L)^T$$
$$K_R = \frac{1}{n}\sum_{i=1}^{n}(M_i^R - D_R)\cdot(M_i^R - D_R)^T \quad (4)$$

and m and n are the numbers of points of L and R, respectively.

By (3) and (4), the five coefficients of (2) can be obtained and a quadratic decision boundary, $h(X)$, in (2) is generated. This decision boundary is a collision-free path which the ALV can follow, as shown in Fig. 2. In Section II-B, pattern generation and pattern clustering for generating collision-free paths are described.

B. Pattern Generation for Quadratic Classifier Design

After performing the pattern coordinate transformations described in Section II-D, we can get the space coordinates, (x, y, z), of each pattern. Because the z coordinates are all zero, we can project the patterns onto the x-y plane without changing the values of the x and y coordinates. Hence in the following analysis, only the x and y values instead of all x, y, and z values are used, i.e., the three-dimensional coordinates in the space are treated as two-dimensional coordinates on a plane.

Before analyzing the patterns, we first introduce the definition of the minimum distance of the two clusters of points. If there are two clusters, U and V, of variant points, U_i and V_j, where $1 \le i \le m; 1 \le j \le n; m$ is the number of points in cluster U; and n is the number of points in cluster V, let $D(U_i, V_j)$ denote the distance between any point U_i in U and any point V_j in V. The minimum distance between U and V, denoted as D_{UV}, is defined to be the minimum of all $D(U_i, V_j)$ values, i.e., $D_{UV} = \min_{i,j} D(U_i, V_j)$. The value D_{UV} will be used to determine the width of the corridor space through which the vehicle passes between the two clusters U and V of obstacle points.

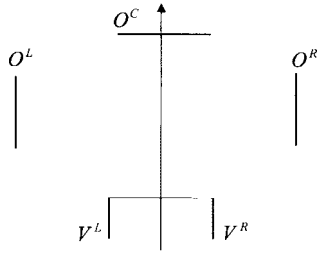


Fig. 3. Three kinds of obstacles and two kinds of vehicle patterns in the vehicle coordinate system.

In this study, obstacles are all assumed to lie on the ground, so there are intersections between the surfaces of obstacles and the ground, which are called baselines in this study. All patterns detected by the techniques described in Section IV are considered to be the baselines of certain obstacles, so we will now analyze the types of the distributions of the obstacle baselines instead of the patterns. The purpose is to group all baselines into three sets, O^L, O^C, and O^R for use in classifier design where the components of O^L and O^R represent the baselines which are located on the left and the right sides of the y axis, respectively, and the components of O^C represent the baselines which lie across the y axis in the VCS (see Fig. 3). And this is accomplished in this study by the following rule:

assign B_i to

$$\cdot \begin{cases} O^L, & \text{if } STARTX(B_i) < 0 \text{ and } ENDX(B_i) < 0 \\ O^c, & \text{if } STARTX(B_i) < 0 \text{ and } ENDX(B_i) > 0 \\ O^R, & \text{if } STARTX(B_I) > 0 \text{ and } ENDX(B_i) > 0 \end{cases} \quad (5)$$

where B_i specifies a baseline, and $STARTX(B_i)$ and $ENDX(B_i)$ specify the values of the x coordinates of the starting and ending points, respectively, of B_i.

On the other hand, as shown in Fig. 3, we also regard the vehicle as two sets of patterns, V^L and V^R, representing the points of the projections of the left and right sides of the vehicle, respectively, in the x-y plane of the VCS. And we want to associate separately each of these two sets of patterns, V^L and V^R, with one of the sets O^L, O^C, and O^R for finding the collision-free path. A scheme for this purpose proposed in the following considers the location of the vehicle and enforces the collision-free path to go through the origin of the VCS, i.e., to go through the middle point between the two front wheels of the vehicle. The scheme is to separate the five kinds of patterns, O^L, O^C, O^R, V^L, and V^R, into two groups, L and R, by the following rule:

set $L = V^L \cup O^L \cup L^C, R = V^R \cup O^R \cup R^C$

with

$$L^C = \begin{cases} O^C, & \text{if } (O^L \cup O^R = \phi \text{ and } SLOPE(O^C) > 0) \\ & \text{or } (O^R = \phi \text{ and } O^L \neq \phi \text{ and} \\ & \quad D_{O^L,O^C} >|, W_V) \\ & \text{or } (O^R \neq \phi \text{ and } O^L \neq \phi \text{ and} \\ & \quad D_{O^L,O^C} < D_{O^R,O^C}) \\ \phi, & \text{otherwise,} \end{cases}$$

$$R^C = \begin{cases} O^C, & \text{if } (L^L \cup O^R = \phi \text{ and} \\ & \quad SLOPE(O^C) < 0) \\ & \text{or } (O^L = \phi \text{ and } O^R \neq \phi \text{ and} \\ & \quad D_{O^R,O^C} > W_V) \\ & \text{or } (O^R \neq \phi \text{ and } O^L \neq \phi \text{ and} \\ & \quad D_{O^L,O^C} > D_{O^R,O^C}) \\ \phi, & \text{otherwise} \end{cases} \quad (6)$$

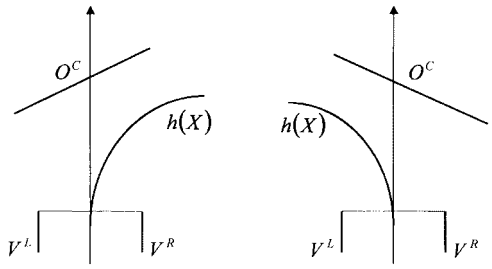


Fig. 4. Only O^C exists.

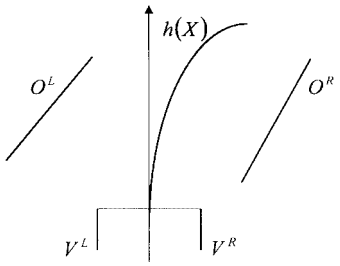


Fig. 5. Only O^C is nonexistent.

where the $SLOPE(O^C)$ means the slope of the baseline O^C, the value W_V means the width of the vehicle, the value D_{O^L,O^C} means the minimum distance between O^L and O^C, and the value D_{O^R,O^C} means the minimum distance between O^R and O^C. The set L is the union of V^L, O^L, and L^C, and the set R is the union of V^R, O^R, and R^C. The idea behind the above rule is explained in the following.

According to the existences of the three kinds of baselines, O^L, O^C, and O^R, denoted as N^L, N^C, and N^R, we get eight different kinds of their combinations, each denoted by (N^L, N^C, N^R), where the definitions of N^L, N^C, and N^R are as follows:

$$N^L = \begin{cases} 0, & \text{if } O^L = \phi; \\ 1, & \text{if } O^L \neq \phi, \end{cases} \quad N_C = \begin{cases} 0, & \text{if } O^C = \phi; \\ 1, & \text{if } O^C \neq \phi, \end{cases}$$

$$N^R = \begin{cases} 0, & \text{if } O^R = \phi; \\ 1, & \text{if } O^R \neq \phi. \end{cases}$$

In grouping the three kinds of baselines and the two kinds of vehicle patterns into two sets L and R, all possible merges of eight kinds of combinations of the baselines with the two kinds of vehicle patterns need be considered. The following are the descriptions of all of the different situations.

1) (0, 0, 0): O^L, O^C, and O^R are all nonexistent, so set $L = V^L$ and $R = V^R$.
2) (0, 0, 1): Only O^R exists, so set $L = V^L$ and $R = V^R \cup O^R$.
3) (1, 0, 0): Only O^L exists, so set $L = V^L \cup O^L$ and $R = V^R$.
4) (0, 1, 0): Only O^C exists, so we check the slope of O^C. If it is larger than zero, then set $L = V^L \cup O^C$ and $R = V^R$; otherwise, set $L = V^L$ and $R = V^R \cup O^C$. This enforces the vehicle to pass the obstacle specified by O^C from the right-hand side or the left-hand side of O^C, considering the slope of O^C (see Fig. 4).
5) (1, 0, 1): Only O^C is nonexistent, so we calculate the space width between O^L and O^R. If $D_{O^L,O^R} \leq W_V$, with W_V being the width of the vehicle, then stop the vehicle; otherwise, set $L = V^L \cup O^L$ and $R = V^R \cup O^R$. This enforces the vehicle to go through the space between O^L and O^R (see Fig. 5).

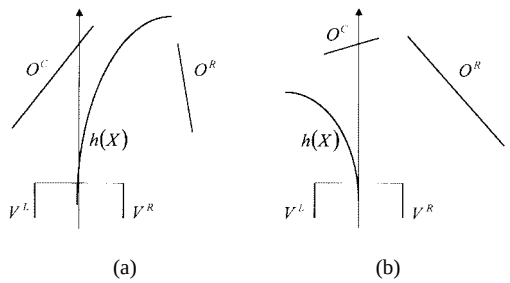


Fig. 6. Only O^L is nonexistent.

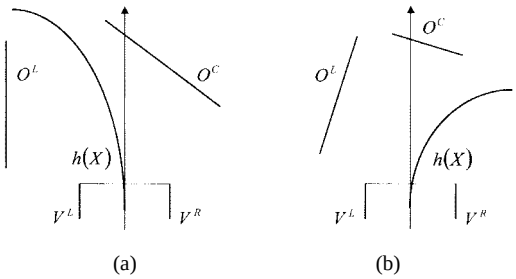


Fig. 7. Only O^R is nonexistent.

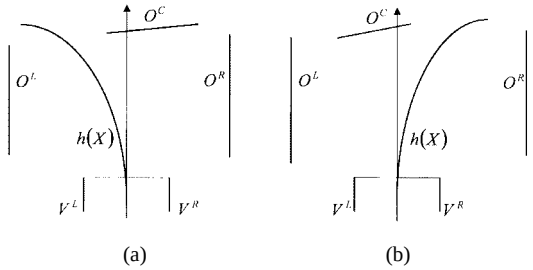


Fig. 8. All O^L, O^C, and O^R exist.

6) (0, 1, 1): Only O^L is nonexistent, so we check the space width between O^C and O^R to see whether it is larger than the width of the vehicle or not. If $D_{O^C,O^R} > W_V$, then set $L = V^L \cup O^C$ and $R = V^R \cup O^R$; otherwise, set $L = V^L$ and $R = V^R \cup O^C \cup O^R$. This enforces the vehicle to pass the wider corridor space, considering the space width between O^C and O^R (see Fig. 6).
7) (1, 1, 0): Only O^R is nonexistent, so we check the space width between O^L and O^C to see whether it is larger than the width of the vehicle or not. If $D_{O^L,O^C} > W_V$, then set $L = V^L \cup O^L$ and $R = V^R \cup O^C$; otherwise, set $L = V^L \cup O^L \cup O^C$ and $R = V^R$. This enforces the vehicle to pass the wider corridor space, considering the space width between O^L and O^C (see Fig. 7).
8) (1, 1, 1): All of O^L, O^C, and O^R exist, so we calculate the space width between O^L and O^C, and that between O^C and O^R in order to find the wider corridor space to go through. If $\max(D_{O^L,O^C}, D_{O^C,O^R}) < W_V$, then stop the vehicle; otherwise, if $D_{O^L,O^C} > D_{O^C,O^R}$, then set $L = V^L \cup O^L$ and $R = V^R \cup O^C \cup O^R$; if $D_{O^L,O^C} \leq D_{O^C,O^R}$, then set $L = V^L \cup O^L \cup O^C$ and $R = V^R \cup O^R$. This enforces the vehicle to go through the space between O^L and O^C, or between O^C and O^R (see Fig. 8).

The above analysis is concluded by Rule (6). The space between the two groups, L and R, is guaranteed to be wide enough to allow

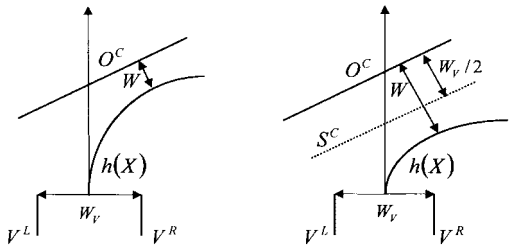


Fig. 9. Two diffrent situations without and with a series of additional points.

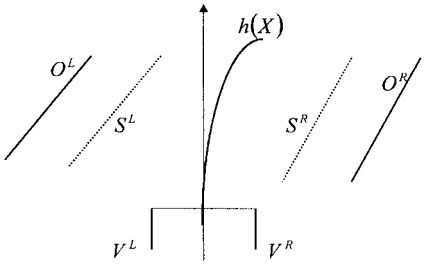


Fig. 10. Two series of additional pints, S^L and S^R, are included as patterns before generating decision boundary, $h(X)$.

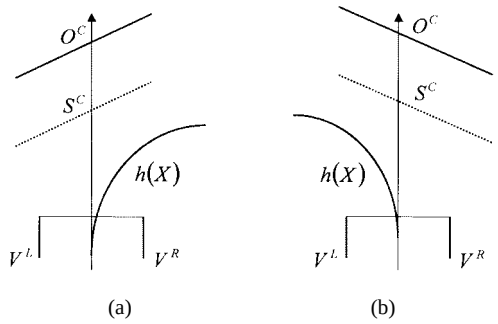


Fig. 11. The additional points S^C are generated.

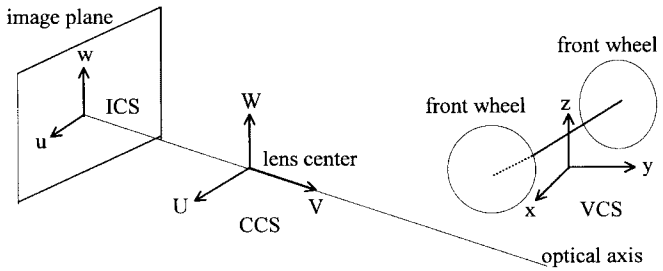


Fig. 12. The three coordinate systems.

the vehicle goes through. L and R are used as part of the input for designing the quadratic classifier described in Section II-A. In addition to L and R, some additional patterns generated using a pattern addition algorithm described in Section II-C next constitute the remaining part of the input for designing the quadratic classifier.

C. Addition of Patterns for Creating Safe Distance Between Obstacles and ALV Navigation Paths

Because the vehicle has width in real implementation and the collision-free path generated is enforced in this study to go through the middle point of the two font wheels of the vehicle, it is found necessary to create some artificial points as additional patterns for designing the classifier in generating collision-free paths. To illustrate this, check the two different situations, one without and the other with a series of additional points, S^C, as shown in Fig. 9. In Fig. 9(a), although a correct path is generated, the space specified by the width W between the path and the obstacle O^C is smaller than half of the vehicle body width, $W_V/2$, so that the vehicle cannot pass the object without collision. On the contrary, in Fig. 9(b) due to the addition of the series of dotted points, S^C, as extra patterns, which is parallel to the obstacle O^C with a distance of $W_V/2$, the generated path will allow sufficient space (with a distance large than $W_V/2$ to O^C) for the ALV to pass the obstacle O^C without collision. In short, the additional point series works to keep a safe distance between obstacles and the collision-free path so that the generated path guarantees that no collision will occur.

Hence in this study, in addition to the image points of the detected obstacles and the two sides of the vehicle body, a series of additional points at a distance of $W_V/2$ to the obstacles are also included as patterns. According to the three kinds of baselines, O^L, O^C, and O^R, three kinds of series of additional points, S^L, S^C, and S^R, are generated, respectively. As shown in Fig. 10, the additional points S^L are generated in such a way that for each point$P$ in O^L, there is a corresponding point Q in S^L with Q being on the right-hand side of P at a distance of $W_V/2$. S^R is generated similarly except each point in S^R is put to the left of its corresponding one in O^R. As shown in Fig. 11, S^C is also generated similarly except that each point in S^C is put at the lower side of its corresponding one in O^C. Additionally, S^L, S^C, and S^R are assigned to be in the same classes (L or R) as O^L, O^C, and O^R, respectively, so Rule (6) for generating L and R should now be modified as follows:

$$\text{set } L = L' \cup S^L \cup L^S, R = R' \cup S^R \cup R^S,$$

with

$$L^S = S^C \text{ and } R^S = \phi \quad \text{if } O^C \in L;$$
$$\text{or} \quad L^S = \phi \text{ and } R^S = S^C \quad \text{if } O^C \in R.$$

where $L'$and R' are the L and R in (6), respectively.

D. Pattern Coordinate Transformations

In this study, the ALV navigation environment is described by the following three coordinate systems, as shown in Fig. 12.

1) The image coordinate system (ICS): denoted as u-w. The origin I is the image plane center and the u-w plane coincides with the image plane. Any point in the image is specified by the coordinates (u, w).
2) The camera coordinate system (CCS): denoted as U-V-W. Every camera has a camera coordinate system and its origin C is attached to its lens center. The V-axis is the optical axis and the U-W plane is the same as that of the ICS. Any point related to the origin C in the space is specified by the coordinates (U, V, W).
3) The vehicle coordinate system (VCS): denoted as x-y-z. The origin V is at the middle point of the line segment which connects the two contact points of the two front wheels of the vehicle with the ground. The x-axis and y-axis are on the ground and parallel to the short side and the long side of the vehicle body, respectively. The z-axis is vertical to the ground. Any point related to the origin V in the space is specified by the coordinates (x, y, z).

The transformations between the ICS, CCS, and VCS are described as follows (see Fig. 13). Assume that any point in the image plane

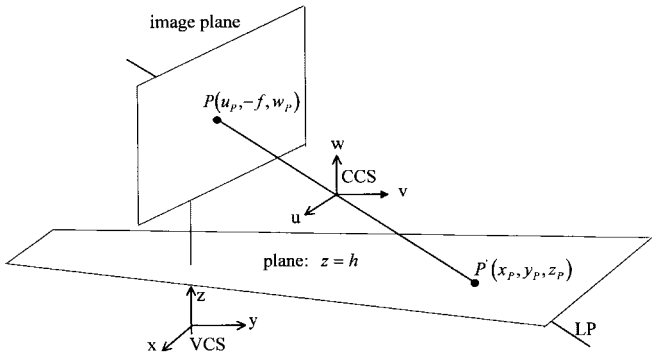


Fig. 13. Illustration of the coordinate transformations between the camera coordinate system (CCS) and the vehicle coordinate system (VCS).

has the CCS coordinates $(u_P, -f, w_P)$ where $(u_P, w_P,)$ specify the coordinates in the ICS and f is the focus length. We get the VCS coordinates (x_P, y_P, z_P) of point P in the image [12] as

$$\begin{aligned} x_P &= u_P(\cos\theta\cos\psi + \sin\theta\sin\phi\sin\psi) + f(\sin\theta\cos\phi) \\ &\quad + w_P(\sin\theta\sin\phi\cos\psi + \cos\theta\cos\psi) + x_d \\ y_P &= u_P(\sin\theta\cos\psi + \cos\theta\sin\phi\sin\psi) - f(\cos\theta\cos\phi) \\ &\quad - w_P(\cos\theta\sin\phi\cos\psi + \sin\theta\sin\psi) + y_d \\ z_P &= u_P(\cos\theta\sin\psi) - f\sin\phi + w_P(\cos\theta\cos\psi) + z_d \end{aligned}$$

where (x_d, y_d, z_d) are the VCS coordinates specifying the translation vector from the origin of the VCS to the origin of the CCS, θ is the pan angle, ϕ the tilt angle, and ψ the swing angle, of the camera with respect to the VCS.

As shown in Fig. 13, after backprojecting a point P in the image into the VCS, we can get a line LP which passes the lens center and P. The equation of line LP is

$$\frac{x - x_d}{x_P - x_d} = \frac{y - y_d}{y_P - y_d} = \frac{z - z_d}{z_P - z_d} = k. \tag{7}$$

Let point P' be the intersection point of the plane $z = h$ and the line LP. By substituting $z = h$ into (7), the desired VCS coordinates $(x_{P'}, y_{P'}, z_{P'})$ of point P' can be solved to be

$$\begin{aligned} x_{P'} &= x_d + \frac{h - z_d}{z_P - z_d}(x_P - x_d) \\ y_{P'} &= y_d + \frac{h - z_d}{z_P - z_d}(y_P - y_d) \\ z_{P'} &= h. \end{aligned} \tag{8}$$

In this study, the origin V of the VCS is on the ground, so the plane $z = 0$ is on the ground. All obstacle points on the ground possess the property $h = 0$, so the coordinates of such points in the VCS detected in Section IV can be obtained by Formula (8) with $h = 0$.

III. Path Following

In Section II, collision-free paths, $h(X)$, for obstacle avoidance are generated to be the decision boundary of the quadratic classifier. When the vehicle navigates by following the quadratic path, $h(X)$, it can pass the obstacles safely. In this section, we describe a method proposed in this study for generating an optimal turning angle by which the vehicle can control the wheels to follow on the quadratic path, $h(X)$, in every navigation cycle.

Before investigating the computation of the turning angle for the wheels, the kinematic trajectory of the vehicle is introduced first. If the behavior model of the vehicle is understood, the location at which the vehicle arrives can be estimated. As shown in Fig. 14, the vehicle

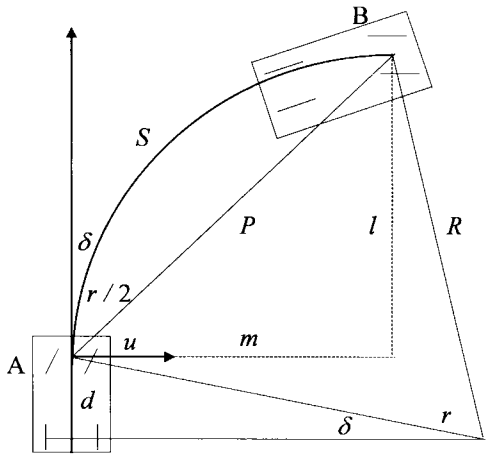


Fig. 14. Analysis of the path, S, which the vehicle navigates from location A to location B by turning an angle of δ.

moves a distance S from location A to location B by turning an angle of δ. We assume that the speed, V, and the navigation time, T, of the vehicle are both known. So, the navigation distance S is a constant computed by

$$S = VT.$$

The translation specified by the values m and l can be acquired by the following equations [13]:

$$\begin{aligned} m &= P\cos u \\ l &= P\sin u \\ P &= R\sqrt{2(1-\cos r)} \\ R &= \frac{d}{\sin\delta} \\ u &= \frac{\pi}{2} - \delta - \frac{r}{2} \\ r &= \frac{S}{R} \end{aligned} \tag{9}$$

where R is the rotation radius, d is the distance between the front wheels and the rear wheels, r and P are the corresponding angle and secant line of S, respectively, and δ is the turning angle of the front wheels.

According to Formula (9), the translation (m, l) of the next location with respective to the current one is computed as follows:

$$\begin{aligned} m &= \frac{d}{\sin\delta}\sqrt{2\left(1-\cos\frac{VT\sin\delta}{d}\right)}\cos \\ &\quad \cdot\left(\frac{\pi}{2} - \delta - \frac{VT\sin\delta}{2d}\right) \\ l &= \frac{d}{\sin\delta}\sqrt{2\left(1-\cos\frac{VT\sin\delta}{d}\right)}\sin \\ &\quad \cdot\left(\frac{\pi}{2} - \delta - \frac{VT\sin\delta}{2d}\right). \end{aligned}$$

So, m and l are determined by the vehicle length d, the vehicle speed V, the navigation time T, and the turning angle δ of the front wheels. In this study, the length and speed of the vehicle are constants, and the navigation time set as the interval between the issues of two system commands is also a constant. So, the turning angle δ determines the values of m and l alone.

When the coordinates of location B are generated, the system can estimate the possible locations at which the vehicle arrives. We hope

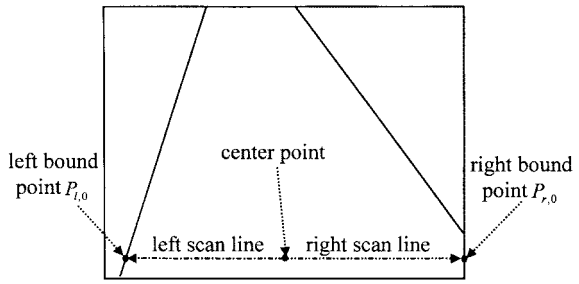


Fig. 15. Determining the start and the end columns, using the left scan line and the right scan one, respectively, for vertical scans of the image.

Fig. 16. An example of obstacle detection results.

that the coordinates, m_i and l_i, of the next location of the vehicle are as close as to the collision-free path $h(X)$.0 Let the coordinates, m_0 and l_0, specify the location closest to the path $h(X)$. The previous analysis tells that the best coordinates, m_0 and l_0, which keeps the vehicle away from collision by following the collision-free path, can be obtained by selecting a proper turning angle. The turning angle of the vehicle is limited in this study to be within thirty degrees to the left or right. It is computed as follows:

$$\begin{aligned}\min_{\delta_i\in\{-30,\mathrm{K},30\}}|h(X_i)| &= \min_{\delta_i\in\{-30,\ldots,30\}}|X_i^T Q X_i + V^T X_i| \\ &= \min_{\delta_i\in\{-30,\ldots,30\}}|\alpha_1 m_i^2 + \alpha_2 m_i l_i \\ &\quad + \alpha_3 l_i^2 + v_1 m_i + v_2 l_i| \qquad (10)\end{aligned}$$

in which $X_i = [m_i\ l_i]^T$ and $|h(X_i)|$ specifies the absolute value of the sum of $X_i^T Q X_i$ and $V^T X_i$.

According to the property of the quadratic classifier, if $h(X) < 0$, then X belongs to the left-hand side of $h(X)$; if $h(X) > 0$, then X belongs to the right-hand side of $h(X)$; and if $h(X) = 0$, then X falls on $h(X)$. The closer the location specified by the coordinates (m_i, l_i) to the path $h(X)$, the smaller the absolute value of $h(X)$. Hence, m_0 and l_0 make the absolute value of $h(X)$ minimum.

After getting the turning angle δ by (10), the vehicle immediately controls the front wheels to turn δ degrees and keeps moving forward until the next control command is received. It is in this way that the vehicle keeps on the collision-free path to achieve collision avoidance and navigation in real time.

IV. Obstacle Detection

In this study, obstacles are assumed to lie on the ground and represented by baselines. Obstacles that have to be detected include the walls of the corridor and the objects that appear in the way of

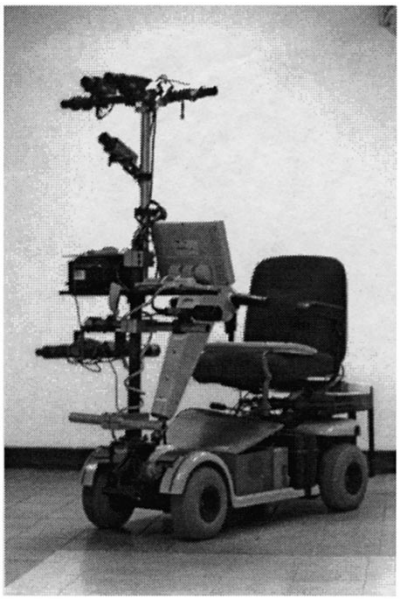


Fig. 17. Prototype ALV used in this study.

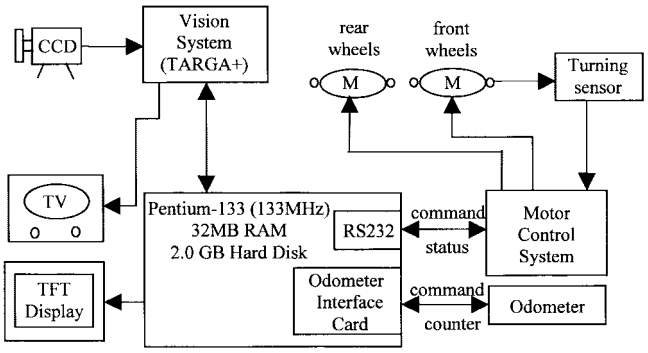
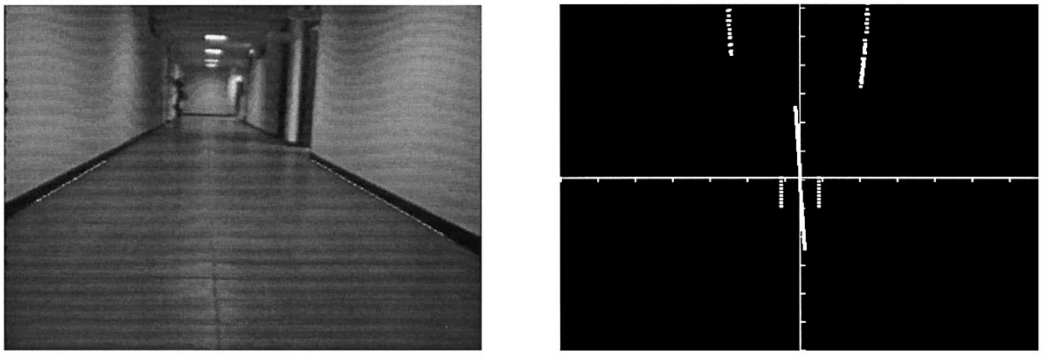


Fig. 18. System structure of the prototype ALV.

ALV navigation in the corridor. By the result of obstacle detection, the space coordinates of obstacles are computed and the vehicle uses this information to find a safe collision-free path. We use only a monocular camera to detect obstacles in the corridor for real-time computation. Additionally, the properties of baselines, which are the intersections of obstacles and the ground, are used to compute the three-dimensional coordinates of the obstacles because all the points on the baselines have the property that the heights are zero.

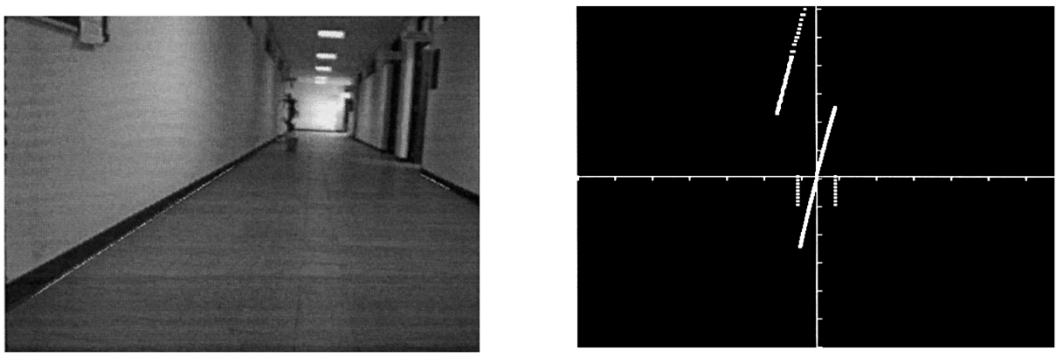
In this section, an obstacle detection method is proposed. The points that compose the baselines of obstacles are detected by local thresholding since the gray levels of the obstacles and those of the ground are generally different. Each baseline consists of a cluster of points. After baseline points are detected, region growing is used to collect the baselines. The details of the obstacle detection method are described as follows.

1) Step 1: Determining a threshold value and performing local thresholding. First, compute the average gray value, denoted as $GAVG$, of the image. The purpose of computing the $GAVG$ first is to understand the lighting of the environment and give a reference in determining the threshold value, TV. This makes the detection of obstacles more stable. When the $GAVG$ value is high, it means that the environment is bright and the contrast is strong, so a lower TV value is set up, and vice versa. More specifically, we set TV as $TV = TH - GAVG$, where TH



(a) (b)

Fig. 19. Experimental result of generated collision-free path for a case with left and right obstacles O^L and O^R.



(a) (b)

Fig. 20. Experimental result of generated collision-free path for a case with left obstacles O^L.

is a preselected constant value obtained from experimental experience.

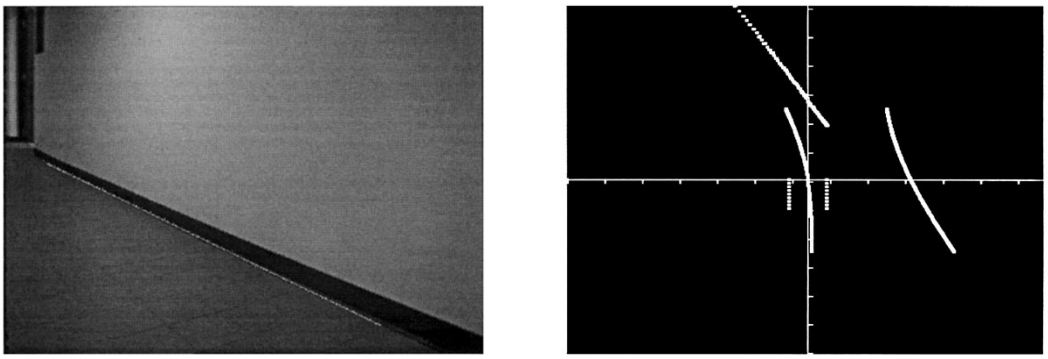
2) Step 2: Determining the start and end columns for vertical scans of the image. The purpose is to detect the left and right corridor boundary in the bottom line of the image. The values are assigned to the initial and end values of Step 3. We scan the bottom line in the image from the center to the left and to the right bounds, called the left scan line and the right one, respectively. See Fig. 15 for an illustration. Let $P_{l,0}$ be the first point in the left scan line such that $|gl(l,0)-gl(l-1,0)|>TV$, where $gl(i,j)$ means the gray level of the image point at (i,j). Let $P_{r,0}$ be the first point in the right scan line such that $|gl(r,0)-gl(r+1,0)>TV$. Obstacle detection is performed on the image area between the lth and the rth columns.
3) Step 3: Detecting the baselines of the obstacles. The image is vertically scanned in a way described by the following steps from the lth column to the rth column.

   a) Step 3.1: Scan the ith vertical line of the image from bottom to top. Let $P_{i,j}$ be the first point in the scan line such that $|gl(i,j)-gl(i,j+1)>TV$. If this point is found, continue; otherwise, go to Step 3.3.
   b) Step 3.2: Region growing is performed to combine the clusters of points that compose the baselines. The points detected in Step 3.1 are merged with other points detected in the previous cycles to form the desired baselines. The criterion for merging a newly detected point with previously detected points is to check if the new point is within the 15×15 neighborhood of any previously detected point. If the point is independent of others or if a group of points is small in size to compose a baseline, the points are treated as noise and discarded.
   c) Step 3.3: Move the scan line to the $(i+1)$th column and go to Step 3.1.

The above algorithm detects the intersection points of the obstacles and the ground to form the baselines of the obstacles. An example of results is shown in Fig. 16, in which the white points are the detected points and the white lines consisting of the white points are the resulting baselines of the obstacles. Some other detected points on the ground which cannot compose a baseline are treated as noise and are ignored.

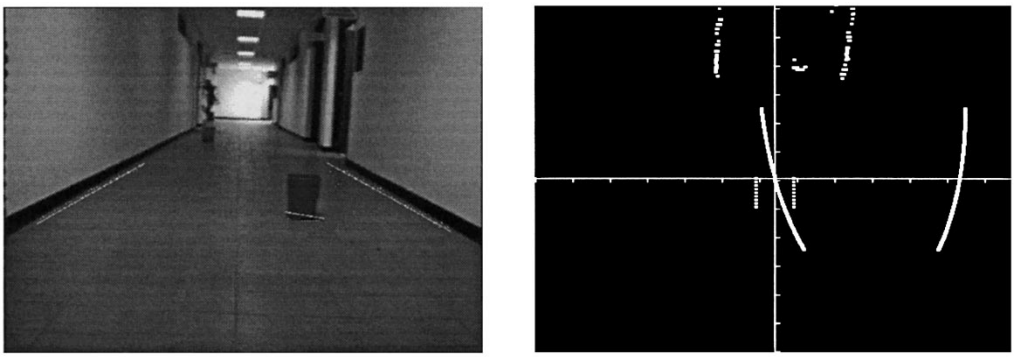
V. Experimental Results

The proposed method has been implemented and used to autonomously steer a real ALV, as shown in Fig. 17. This prototype ALV is modified from a commercial vehicle by adding power conversion systems, some control circuits, cameras, and an on-board computer. The vehicle is four-wheeled with two motors controlling the front and rear wheels, respectively. The width between the front wheels is 48 cm, and that between the rear wheels is 55 cm. The vehicle body is 135 cm in length. The front wheels can be controlled to turn leftward or rightward, while the rear wheels can be driven to go forward or backward. Above the front wheels is a vertical pole with two cross-shaped racks and one load-bearing platform. The lower crossbar is mounted with one CCD camera used in this study and a laser tube used in other studies. The X-Y-Z coordinates of the camera used in this study in the VCS are $(-19.87, 13.05, 72.11)$. The



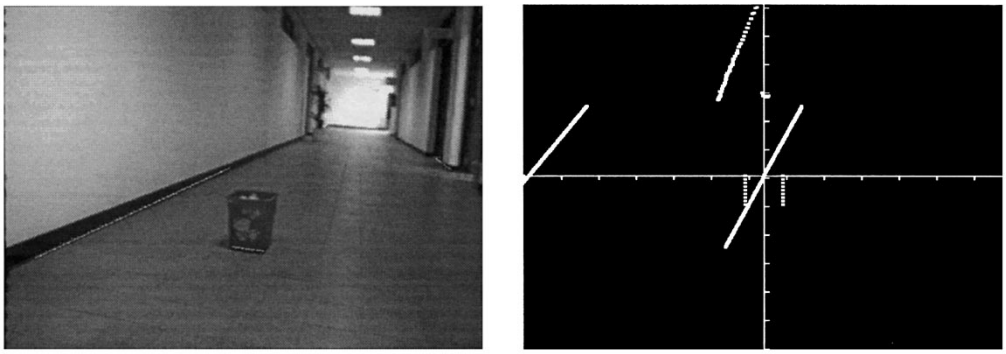
(a) (b)

Fig. 21. Experimental result of generated collision-free path for a case with obstacles O^C.



(a) (b)

Fig. 22. Experimental result of generated collision-free path for a case with obstables, O^L and O^R.

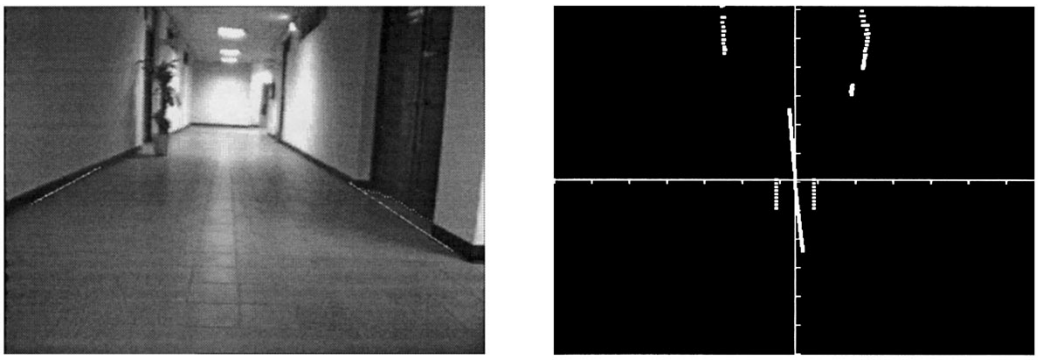


(a) (b)

Fig. 23. Experimental result of generated collision-free path for a case with obstables, O^L and O^C.

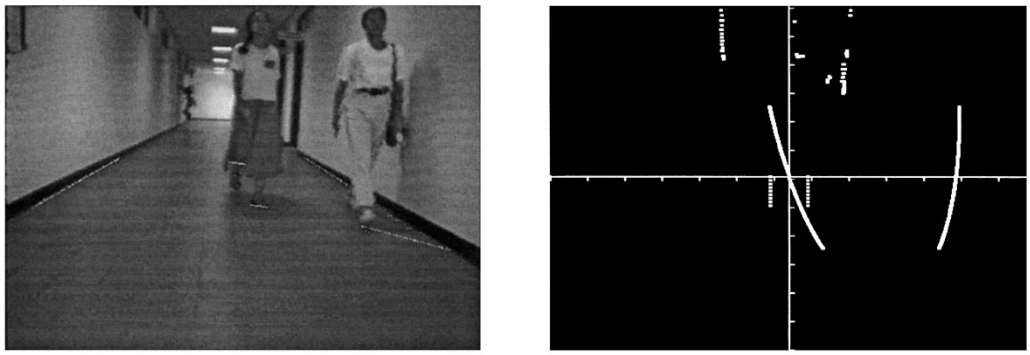
pan, tilt, and swing angles of the camera are $-0.09, -0.15$, and 0.04 degrees, respectively. In addition to the camera used in this study, the three cameras mounted on the upper crossbar and the CCD camera attached to the middle of the pole are all used in other studies. On the platform, there are two monitors. One is used to display images and the other is used as a computer monitor. Below the person-carrying seat, there is a Pentium-133 personal computer, a pair of 12-volt batteries, a set of power converters, and a motor control system.

The architecture of the ALV system is illustration in Fig. 18. The ALV system is composed of three main modules: a visual system, a power system, and a motor control system. The set of the power converters together with the batteries supplies the power of the entire ALV system. The motor control system consists of a main control board with an Intel 8085 controller, a motor driver, and two motors. Turning and speed adjustments are accomplished by sending commands to the motor control system through an RS-232 interface. The visual system is made up of two monitors, a PC, an image frame grabber, and a CCD camera. In our experiments, images grabbed by the image frame grabber are 512 $\times$ 486 pixels in resolution. Throughout the entire navigation process, all image processing and decision making tasks are performed in the Pentium-133 MHz microprocessor equipped with 32-mega byte RAM.



(a) (b)

Fig. 24. Experimental result of generated collision-free path when the vehicle goes through the door of a classroom.



(a) (b)

Fig. 25. Experimental result of generated collision-free path when two persons go through the corridor.

Figs. 19–25 show some experimental results in a real indoor corridor environment. Every figure includes two parts (a) and (b). Part (a) is a real image captured by a wide-angle camera. The white points in (a) are the patterns of the detected obstacles and the white lines consist of the patterns of the obstacles. Part (b) shows the x-y plane of the VCS. All the coordinates of the patterns of the obstacles and the vehicle in the VCS are shown. Besides, the generated collision-free path that goes through the origin of the VCS is also shown in (b). Note that some quadratic paths have two parts (such as the two curves of a hyperbola); only the ones going through the vehicle are the desired ones. According to the generated path, a modification of the turning angle is also computed, which is described in the caption of (b).

Fig. 19 shows that the vehicle keeps moving forward with a small left turn when left and right obstacles, O^L and O^R, exist. Fig. 20 shows the case with O^L only, in which the vehicle turns the wheels to the right-hand side of O^L. Fig. 21 shows that the case with O^C only, in which the vehicle turns the wheels to the left-hand side of O^C. The hyperbola curve on the right part of (b) is the unused part of the generated quadratic decision boundary. Fig. 22 shows that the vehicle passes the left-hand side of the obstacle which is on the ground of the corridor. Fig. 23 shows that the vehicle passes the right-hand side instead of the left-hand one of O^C since the space width between O^L and O^C is too narrow to allow the vehicle to go through. Figs. 24 and 25 show that the vehicle passes the door of a classroom and two persons going through the corridor, respectively.

The above cases and a lot of other experiments show that this approach indeed can find out safe collision-free paths. Additionally, in continuous navigation, this approach also proved to be able to modify the turning angles of the vehicle wheels in real time to achieve the purpose of obstacle avoidance. An example of continuous navigation is shown in Fig. 26, which is a top view of the experimental environment and the ALV location in all the navigation cycles. The velocity of the vehicle is 31.75 cm per second that is equivalent to 1.14 km per hour. This velocity is acceptable in many application indoor environments. In addition, the vehicle modifies the turning

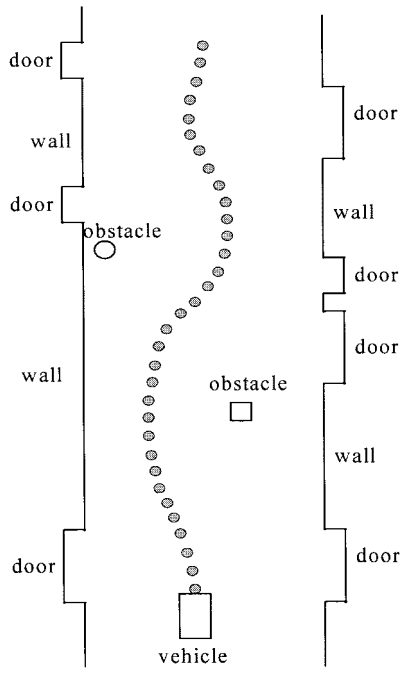


Fig. 26. Top view of the experimental indoor environment and an ALV continuous navigation.

angle every 1.5 s such that this approach can deal in reasonable response time with common obstacles that might cause collisions in indoor environments.

VI. Conclusions

In this study, a vision-based obstacle avoidance approach for ALV navigation has been proposed. The vehicle can detect obstacles, including walls and objects in the way, in an unknown indoor environment and safe collision-free paths can be generated from quadratic classifier design in real time. According to the collision-free path, the vehicle can modify the turning angle of the wheels to achieve the purpose of collision avoidance. Besides, a systematic method has been proposed for generating input patterns for classifier design to compute safe quadratic paths.

The use of quadratic paths instead of linear ones produces smoother paths and prevents dead-reckoning navigation to increase the flexibility of ALV applications in unknown complex environments with obstacles. Additionally, quadratic paths also match the ALV trajectory better than linear ones. A method for computing the optimal turning angle to avoid collisions in real time has also been proposed. The proposed approach has been implemented on a real ALV and a lot of successful navigations confirm the feasibility of the approach.

References

[1] J. C. Hyland and S. R. Fox, "A comparison of two obstacle avoidance path plannings for autonomous underwater vehicles," in *Proc. Symp. Autonomous Underwater Vehicle Technology*, Washington, DC, June 1990, pp. 216–222.

[2] J. Cesarone and K. F. Eman, "Mobile robot routing with dynamic programming," *J. Manufact. Syst.*, vol. 8, no. 4, pp. 257–266, 1989.

[3] C. Acosta and R. G. Moras, "Path planning simulator for a mobile robot," *Comput. Indust. Eng.*, vol. 19, no. 1–4, pp. 346–350, 1990.

[4] J. O. Kim and K. Khosla, "Real-time obstacle avoidance using hormontic potential functions," *IEEE Trans. Robot. Automat.*, vol. 8, pp. 338–349, June 1992.

[5] K. Onoguchi, M. Watanabe, Y. Okamoto, Y. Kuno, and H. Asada, "A visual navigation system using a multi-infomation local map," in *Proc. 1990 IEEE Int. Conf. Robotics Automation*, Cincinnati, OH, vol. 2, pp. 767–774, May 1990.

[6] D. C. H. Yang, "Collision-free path planning by using nonperiodic B-spline curves," *J. Mech. Design*, vol. 115, pp. 679–684, Sept. 1993.

[7] L. L. Wang and W. H. Tsai, "Collision avoidance by a modified least-mean-square-error classification scheme for indoor autonomous land vehicle navigation," *J. Robot. Syst.*, vol. 8, no. 5, pp. 677–798, Oct. 1991.

[8] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, pp. 1179–1187, Sep./Oct. 1989.

[9] T. Skewis and V. Lumelsky, "Experiments with a mobile robot operating in a cluttered unknown environment," in *Proc. 1992 IEEE Int. Conf. Robotics Automation*, Nice, France, vol. 2, pp. 1482–1487, May 12–14, 1992.

[10] R. Bauer, W. Feitern, and G. Lawitzky, "Steer angle fields: An approach to robust manoeuvring in cluttered, unknown environments," *Robot. Auton. Systems*, vol. 12, pp. 209–212, 1994.

[11] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd ed. San Diego, CA: Academic, 1990.

[12] C. C. Lai, "Outdoor autonomous land vehicle guidance by road information using computer vision and fuzzy wheel adjustment techniques," M.S. thesis, Inst. Comput. Inf. Sci., National Chiao Tung Univ., Hsinchu, Taiwan, R.O.C., June 1993.

[13] S. D. Cheng and W. H. Tsai, "Model-based guidance of autonomous land vehicles in indoor environments by structured light using vertical line information," *J. Elect. Eng.*, vol. 34, pp. 441–452, Dec. 1991.

[14] A. Ohya, A. Kosaka, and A. Kak, "Vision-based navigation of mobile robot with obstacle avoidance by single camera vision and ultrasonic sensing," in *Proc. 1997 IEEE/RSJ Int. Conf. Intelligent Robot Systems*, Grenoble, France, Sept. 1997, vol. 2, pp. 704–711.

[15] L. M. Lorigo, R. A. Brooks, and W. E. L. Grimsou, "Visually-guided obstacle avoidance in unstructured environments," in *Proc. 1997 IEEE/RSJ Int. Conf. Intelligent Robot .Systems*, Grenoble, France, vol. 1, pp. 373–379, Sept. 1997.

[16] Y. G. Yang and G. K. Lee, "Path planning using an adaptive-network-based fuzzy classifier algorithm," *13th Int. Conf. Computers Applications*, Honolulu, HI, Mar. 1998, pp. 326–329.

[17] R. Biewald, "Real-time navigation and obstacle avoidance for nonholonomic mobile robots using a human-like conception and neural parallel computing," in *Int. Workshop Parallel Processing Cellular Automata and Array*, Berlin, Germany, Sept. 1996, pp. 232–240.

Dynamic Fuzzy Control of Genetic Algorithm Parameter Coding

Robert J. Streifel, Robert J. Marks, II, Russell Reed, Jai J. Choi, and Michael Healy

***Abstract*— An algorithm for adaptively controlling genetic algorithm parameter (GAP) coding using fuzzy rules is presented. The fuzzy GAP coding algorithm is compared to the dynamic parameter encoding scheme proposed by Schraudolph and Belew. The performance of the algorithm on a hydraulic brake emulator parameter identification problem is investigated. Fuzzy GAP coding control is shown to dramatically increase the rate of convergence and accuracy of genetic algorithms.**

I. Introduction

Genetic algorithms are powerful search techniques which have been applied to many practical problems. However, the accuracy of the final solution found by binary coded genetic algorithms is limited by the number of bits used to code search parameters into strings. The low resolution of binary coding does not seriously affect the solution for many problems (e.g., integer and combinatorial searches). Accuracy becomes a more important consideration when

1) the search space consists of floating point parameters;
2) the parameters have a large dynamic range;
3) a relatively small number of bits are used to code the parameters.

The standard genetic algorithm uses no problem specific information except the relative fitness of the coded binary strings. Lack of gradient information can cause slow progress in search regions where the objective function has nearly zero gradient. The combination of low slope areas and low resolution binary coding can cause slow convergence on many practical problems.

The fuzzy genetic algorithm parameter (GAP) coding methodology presented in this paper is specifically designed to improve the search performance on a parameter identification problem. Conventional genetic algorithm parameter coding is static, the coding is constant for the entire search. This results in slow convergence. Greater accuracy

Manuscript received April 15, 1996; revised July 5, 1997. This paper was recommended by Associate Editor L. O. Hall.

R. J. Streifel, R. J. Marks, II, and R. Reed are with the University of Washington, Seattle, WA 98195 USA (e-mail: robert.j.streifel@boeing.com).

J. J. Choi and M. Healy are with the Boeing Information and Support Services, Seattle, WA 98195 USA.

Publisher Item Identifier S 1083-4419(99)00903-6.