An Incremental-Learning-by-Navigation Approach to Vision-Based Autonomous Land Vehicle Guidance in Indoor Environments Using Vertical Line Information and Multiweighted Generalized Hough Transform Technique

Guan-Yu Chen and Wen-Hsiang Tsai

***Abstract*—An incremental-learning-by-navigation approach to vision-based autonomous land vehicle (ALV) guidance in indoor environments is proposed. The approach consists of three stages: initial learning, navigation, and model updating. In the initial learning stage, the ALV is driven manually, and environment images and other status data are recorded automatically. Then, an off-line procedure is performed to build an initial environment model. In the navigation stage, the ALV moves along the learned environment automatically, locates itself by model matching, and records necessary information for model updating. In the model updating stage, an off-line procedure is performed to refine the learned model. A more precise model is obtained after each navigation-and-update iteration. Used environment features are vertical straight lines in camera views. A multiweighted generalized Hough transform is proposed for model matching. A real ALV was used as the testbed, and successful navigation experiments show the feasibility of the proposed approach.**

***Index Terms*—Autonomous land vehicle, guidance, incremental learning, model matching, navigation, weighted generalized Hough transform.**

I. Introduction

A. Motivation

Because of the fast development of computer vision techniques, vision-based guidance of autonomous land vehicles (ALV's) has been intensively studied in the recent years, and model-based methods are often used in practical experiments. However, the establishment of environment models is really a time-consuming work. It is thus desired to design a process for automatic modeling of navigation environments. With this process, it is not necessary to measure the environment manually. Instead, just drive the ALV manually once along the desired path, and all jobs about initial model learning will be automatically accomplished without human involvement.

However, certain problems arise when a fully automatic model establishing process is performed. The noise of image processing and the shake of the ALV will reduce the accuracy of the obtained model. Since the noise coming from image processing will not appear at the same place in each navigation cycle and the error caused by ALV shaking may be eliminated by averaging several observations, it is possible that more training using multiple navigation data will reduce the inaccuracy. This means that after initial model learning, we could utilize the information recorded in each navigation cycle to update the original coarse model, and hopefully a more reliable refined model could be obtained after several navigation trainings. This leads to our study of "incremental-learning-by-navigation" for ALV guidance in indoor environments.

B. Survey of Related Studies

A lot of successful ALV systems have been established for various purposes. For environment learning, the autonomous mobile robot HERMIES developed by Saussure *et al.* [1] includes a successful learning system in which the robot, placed in an arbitrary initial location without any prior specification of its environment, successively discovers and navigates around some obstacles, searches for a desired target, and performs a learned sequence of manipulations on the control panel device. Another system developed by Onoguchi *et al.* [2] is a visual navigation robot working in a nuclear power plant. The operation of this system is divided into two stages. In the first stage, a multi-information local map describing information necessary for self-location measurement is created interactively from stereo images collected during remote-controlled navigations. The second stage is autonomous navigation. Lebégue and Aggarwal [3], [4] developed an integrated system to generate architectural CAD models using a mobile robot. The system consists of a segment detector, a tracker and a CAD modeler optimized for environments with prominent three-dimensional (3-D) orientations. Nashashibi *et al.* [5] proposed an approach to build a rough geometric model for a 3-D terrain using a laser range finder. They also gave algorithms to build snapshot models with planar faces from range data, and to perform 3-D data fusion between these snapshot models, in order to build incrementally a reliable 3-D model [6]. Ishiguro *et al.* [7] also presented a strategy for establishing models of an unknown environment by a mobile robot. In their implementation, panoramic sensing was used to perceive the structure of the environment.

In Ku and Tsai [8], a model-based navigation was proposed. The corridor contour is used to match the model and the input pattern are extracted from video camera images. So, the global location of the ALV can be known. In Chang and Tsai [9], a vision-based collision avoidance method is proposed for ALV navigation. The corridor contour of a building is used as the model, and laser markers and cameras are used to detect corridor contours and obstacles. A new guidance approach by model matching was proposed in Cheng and Tsai [10]. Two laser light sources were employed to reduce the time for computing vertical positions by triangulation. A matching scheme using distance weight correlation is also proposed. In Su and Tsai [11], model-based navigation and collision avoidance in building corridors and elevators was proposed. The multiple corner position information was matched with the model to locate the ALV accurately. Pan and Tsai [12] proposed an integrated approach to automatic model learning and path generation for vision-based ALV guidance in building corridors.

C. Overview of Proposed Approach

The goal of ALV learning and guidance of this study is to equip the ALV with the capabilities to explore the environment with its sensors, construct an appropriate model of the environment, and navigate smoothly and safely in the learned environment.

The proposed approach, incremental-learning-by-navigation for ALV guidance in indoor environments, consists roughly of three stages. The first stage is initial learning, in which the ALV is driven manually along a path decided by the driver and the environment images captured by the camera and the control status data are recorded. Then, a certain off-line procedure is performed to construct the initial model. This is accomplished by calculating the relation between the ALV and the environment features observed in each learning cycle, and matching the features with the partially learned model. The second stage is to allow the ALV to navigate automatically alone the desired path. And the third stage is to update the learned model with the information collected in the previous navigation. The second

Manuscript received April 14, 1996; revised July 5, 1997. This work was supported by the National Science Council under Project NSC-83-E-04-333.

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

Publisher Item Identifier S 1083-4419(98)07302-6.

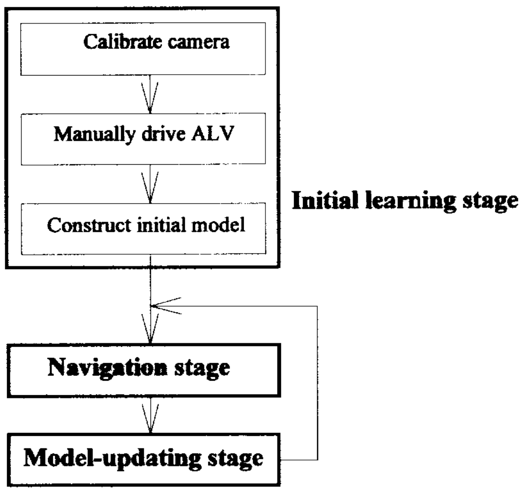


Fig. 1. Illustration of the proposed "incremental-learning-by-navigation" approach.

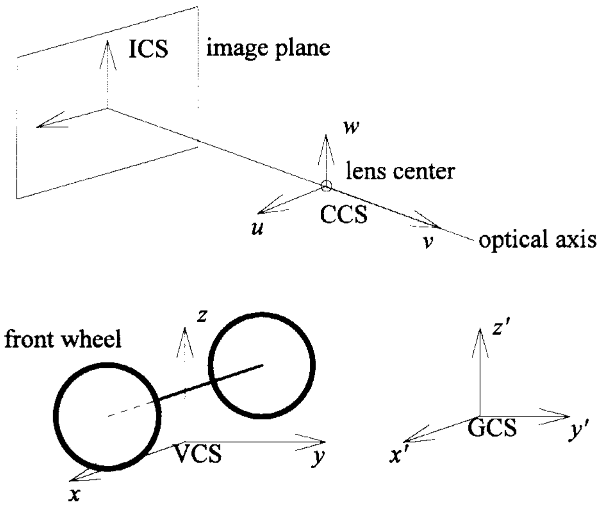


Fig. 2. The four coordinate systems ICS, CCS, VCS, and GCS.

and the third stages, namely, ALV navigation and model updating, may be repeated several times in order to obtain a more reliable model. The relation of these three stages is illustrated in Fig. 1.

The second stage, the navigation task, can also be roughly divided into three phases. The first phase is to calculate the relation between the ALV and the environment features observed in the image taken in the current cycle. The second phase is to perform a model matching scheme so that the ALV can locate itself accurately. The third phase is to drive the vehicle toward a favorable direction by a control strategy. The detailed procedure of navigation will be described later.

Selecting stable environment features and developing effective methods to extract these features are the most important keys to success of model-based ALV guidance, especially to the model learning and update works. In this study, we extract all vertical line features in each environment image with no care about what the lines really represent. One reason for using vertical lines as features is that they can be treated as points in the top view; this facilitates the use of many well-developed point matching techniques to locate the ALV. Another reason is that vertical lines can be extracted more easily and stably by many image processing techniques; this leads to the use of less image processing time which is necessary for a real-time navigation system. Compared with other approaches, our method has the advantages of improving the initial learning result constantly in every navigation session. Most existing approaches have no such incremental learning capability.

The remainder of this paper is organized as follows. In Section II, the proposed method to extract environment features for model learning and ALV guidance is described. In Section III, the proposed "incremental-learning-by-navigation" approach is described in detail. In Section IV, employed image processing techniques and some experimental results are presented. Finally, the conclusion of this paper is given in Section V.

II. Extraction Environment Features for Model Learning and ALV Guidance

A. Coordinate Systems and Transformations

In the proposed ALV guidance process, the following four coordinate systems are used to describe the vehicle location and the navigation environment.

1) The vehicle coordinate system (VCS): denoted as x–y–z. The origin V of the VCS is chosen to be at the middle point of the line segment which connects the two contact points of the two front wheels with the ground. The x-axis and y-axis are on the ground and parallel to the short and the long sides of the vehicle body, respectively. The z-axis is vertical to the ground.
2) The camera coordinate system (CCS): denoted as u–v–w. The camera is associated with the camera coordinate system whose origin C is attached to the camera lens center. The v-axis is coincident with the optical axis and the u–w plane is parallel to the image plane.
3) The image coordinate system (ICS): denoted as u–w. The image plane of the image coordinate system is coincident with the u–w plane of the CCS and its origin I is the image plane center.
4) The global coordinate system (GCS): denoted as x'–y'–z'. The origin G of the global coordinate system is located at a certain fixed position. The x'-axis and y'-axis are defined to lie on the ground.

Fig. 2 shows these coordinate systems. Since the origins of the ICS, CCS, and VCS are attached to some points on the ALV, the ICS, CCS, and VCS are moving with the vehicle during navigation. On the contrary, the GCS is fixed and is defined to be coincident with the VCS when the ALV is at the starting position in the initial model learning stage.

The transformations between these four coordinate systems can be found in [13] and [16]. Note that since the ALV always navigates on the ground, the relation between the two two-dimensional (2-D) coordinate systems x–y and x'–y' is sufficient to determine the position and orientation of the vehicle. In other words, the translation vector (x'_p, y'_p) and the rotation angle ω of the ALV in the x'–y' coordinate system as shown in Fig. 3 determine the position and the direction of the vehicle in the GCS, respectively. The transformation between the GCS and the VCS can be written as

$$(x' \quad y' \quad 1) = (x \quad y \quad 1) \cdot \begin{bmatrix} \cos\omega & \sin\omega & 0 \\ -\sin\omega & \cos\omega & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x'_p & y'_p & 1 \end{bmatrix}. \quad (1)$$

In the following sections, the combination of the vehicle position and direction is referred to the *vehicle location* and is denoted by a triple (x_p, y_p, ω).

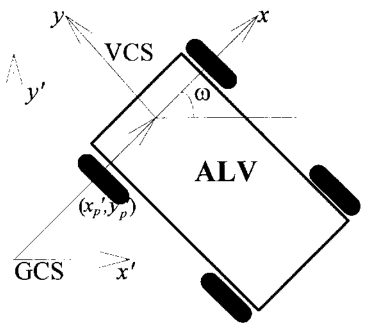


Fig. 3. The relation between VCS and GCS.

B. Ideas of Locating and Matching Vertical Line Features for ALV Guidance

In this study, the term "*vertical lines*" are defined to be the ones which are parallel to the z'-axis of the GCS. Vertical lines may be the edges of walls, windows, doors, bulletin boards, paintings on the walls, and so on. Vertical line features provide abundant 3-D information and may be collected to serve as the basic elements of the environment model. For simplicity, we only utilize those vertical lines which appear nearly parallel to the w-axis of the ICS in the captured images. This simplifies the work of searching vertical lines in the image and also speeds up the image processing procedure. However, this requires that the swing angle be nearly zero, i.e., the w-axis must be nearly parallel to the z'-axis, and that the tilt angle be small (approximately within the limit of $\pm 3^\circ$). This requirement must be satisfied when the camera is installed.

With no depth information, a single image is insufficient for locating a vertical line in the GCS without any heuristic. However, if a point on a certain vertical line is known to be on a known plane, the location of the line may be uniquely decided. In our study, the intersection point of a vertical line with the base line of the corridor is used to locate the vertical line, since this intersection point is known to be on the ground. The job of finding the intersection point can be easily done by simple image processing techniques.

The equations for calculating the VCS coordinates of a point located on a known plane, e.g., on the ground, are derived in [14] and [16]. By the way, if the vehicle location, (x_p', y_p', ω), is known, the GCS coordinates of the vertical lines can also be obtained by (1). In our approach, a rough estimation of the vehicle location is first obtained by the use of the information of the odometer, which gives the navigation distance during a cycle, as well as the photo-encoder, which feeds back the turn angle of the front wheels. Then by matching the collected vertical line features with those in the learned model, the error in the rough estimation of the vehicle location can be corrected, and so safe ALV guidance is feasible.

C. Multiweighted Generalized Hough Transform

As mentioned in the previous section, a vertical line can be viewed as a point from a top-view. As a result, the learned model, which is a collection of some vertical line features, can be treated as a set of points, or a point pattern. Thus, the ALV location problem may be solved by a point matching scheme. Our approach to point matching is based on a modification of the distance-weighted generalized Hough transform (DWGHT) proposed by Jeng and Tsai [15]. The DWGHT is useful for inexact matching of point patterns and may be employed to detect or locate object shapes with noisy or distorted boundaries caused by image sensing or preprocessing, so it is suitable for our application since our feature patterns, the vertical lines, are often distorted by erroneous image processing or by the shake of the ALV.

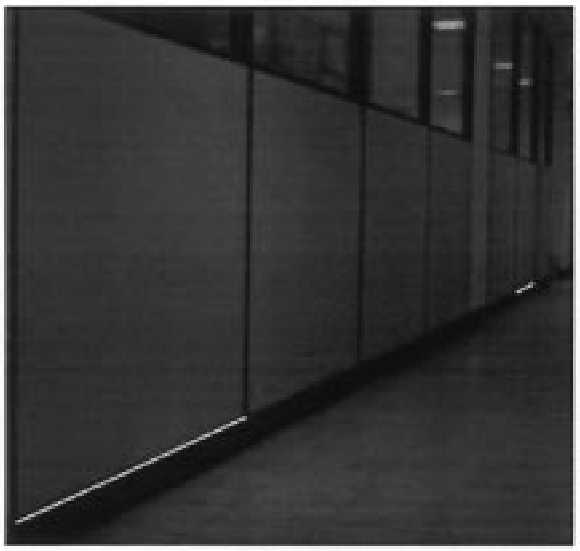


Fig. 4. A view of the navigation environment. The further line segment looks shorter in the captured images, while the nearer one looks longer. Both line segments are of the same length actually.

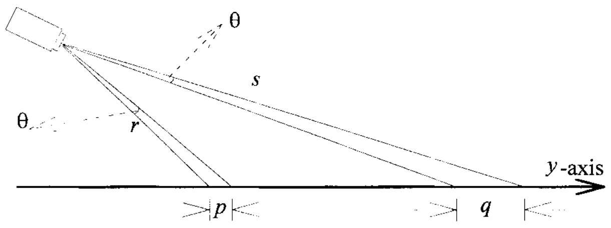


Fig. 5. Illustration of position estimation error due to the little variation of the tilt angle.

The basic idea of the DWGHT is to replace the unity increment value used in the cell value incrementation stage of the conventional generalized Hough transform proposed by Ballard [17] with distance-weighted increment values. The distance-weighted (DW) cell value incrementation strategy for the DWGHT for a certain cell C in the Hough counting space can be describe as follows:

```
for each point $P$ at $(x_e, y_e)$ in the input point pattern,
for each scale $\tilde{S}$,
for each orientation $\tilde{\Theta}$, and
for each displacement vector $(r, \theta)$
    compute $(x, y) = (x_e, y_e) + (r \cos\theta, r \sin\theta)$
for each cell $NC$ at location $(x', y')$
    in the neighborhood $NH$ of cell $C$ at $(x, y)$,
    set $H\left(x', y', \tilde{S}, \tilde{\Theta}\right) = H\left(x', y', \tilde{S}, \tilde{\Theta}\right) + W(d)$     (2)
```

where d is the Euclidean distance between C and NC, i.e., $d = \sqrt{(x'-x)^2 + (y'-y)^2}$, and the distance-weighted function $W(d)$ is

$$W(d) = \frac{1}{1+d^2}. \tag{3}$$

However, in the DWGHT, each point in the point pattern has the same importance, and the weight only depends on the distance of the

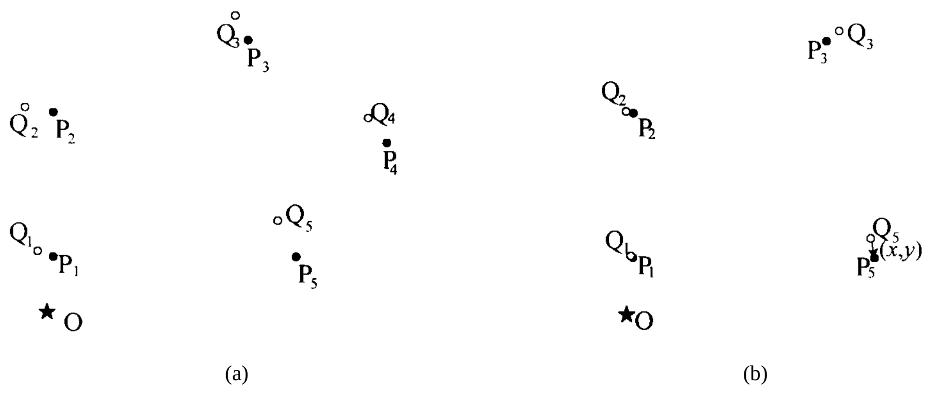


Fig. 6. Illustration of the multiweighted generalized Hough transform: (a) original positions of template pattern L and input pattern N and (b) positions of template pattern L and input pattern N after rotating the input pattern with a certain rotation angle γ_0 and with respect to the origin O.

matched point pairs, but not on the importance of the point itself. In our approach, we propose the use of a multiweighted generalized Hough transform (MWGHT), in which each point is attached with an additional weight, called the *confidence weight*.

There are two reasons to add this new weight. First, during the learning-by-navigation process, the confidence of a feature point should be increased if the corresponding vertical line appears at an identical position again in the next image because multiple occurrences of a vertical line indicate that the vertical line feature is reliable. Second, it is found that the errors of the feature points coming from locations far from the vehicle are relatively larger than those of the feature points near the vehicle. Two examples can illustrate this fact. First, as shown in Fig. 4, the two segments are of the same length actually, but in the perspective image, the further line segment looks shorter while the nearer one looks longer. As a result, image processing errors in pixel length may cause relatively larger errors in the estimations of the positions of further points. Second, Fig. 5 shows the errors caused by the variations of the tilt angle. Two sources of variations of the tilt angle values are shakes of the ALV and imprecise camera calibration results. Let the errors caused by the variations of a tilt angle at a short distance and at a long distance be p and q, respectively. Also, let the distance between a further point P_f and the camera be s, and that between a nearer point P_n and the camera be r. If the variation of the tilt angle is θ, then since θ is small, the errors caused by the variation of the tilt angle forP_f and P_n, respectively, can be written as

$$p = r\theta, \quad q = s\theta. \tag{4}$$

Since $r < s$, we have $p < q$. The conclusion is that the error due to the variation of the tilt angle is directly proportional to the distance between the point and the camera (or the ALV).

The above discussions show the need of a confidence weight for each feature point. In the rest of this section, an algorithm for the proposed MWGHT will be presented. The algorithms for assigning and updating the confidence weights will be described in Section III-C.

Algorithm 1: Multiweighted Generalized Hough Transform

Input: An input point pattern **N** and a template point pattern **L**.

Output: The displacement vector (x_t, y_t, γ) which transforms **N** to **L** through a translation (x_t, y_t) and a rotation γ.

Steps

Step 1. Set up a 3-D Hough counting space $\mathbf{H}(x_t, y_t, \gamma)$ including the maximum reasonable displacement, and set all values of the cells in **H** to zero.

Step 2. Increase the values of the cells in **H** according to the following cell value incrementation strategy:

```
for each rotation angle $\gamma$,
for each point $P_i$ with location $(x_p, y_p)$ and
    confidence weight $W_p$ in template pattern $\mathbf{N}$,
for each point $Q_j$ with location $(x_q, y_q)$and
    confidence weight $W_q$ in input pattern $\mathbf{L}$,
    compute$(x, y) = (x_p, y_p)$
        $- (x_q \cos\gamma - y_q \sin\gamma, x_q \sin\gamma + y_q \cos\gamma)$      (5)
for each cell $NC$ with location $(x', y', \gamma)$ in the
    neighborhood $NH$ of the cell $C$ at $(x, y, \gamma)$,
    set $\mathbf{H}(x', y', \gamma) = \mathbf{H}(x', y', \gamma)$
        $+ W_p \cdot W_q \cdot \frac{1}{1 + d_1/D_0}$      (6)
```

where D_0 is a pre-selected constant, and d_1 is the Euclidean distance between C and NC, i.e.,

$$d_1 = \sqrt{(x' - x)^2 + (y' - y)^2}. \tag{7}$$

Step 3. Find out the location of the cell with the maximum value in **H**.

Step 4. Exit with the corresponding displacement vector (x_t, y_t, γ) as the output.

An example may be used to illustrate how Algorithm 1 works. A template pattern **L** and an input pattern **N** are given, and what desired is the displacement vector (x_t, y_t, γ) which transforms **N** to **L** through a translation (x_t, y_t) and a rotation γ. The original positions of the template pattern, represented in black dots, and the input pattern, represented in white dots, are shown in Fig. 6(a). The positions of **N** and **L** after rotating **N** with a certain rotation angle γ_0 and with respect to the origin O are shown in Fig. 6(b). For a point P_i in **L** and a point Q_j in **N**, e.g., P_5 and Q_5, the values of x and y in (5) compose the translation vector from Q_j to P_i, as shown in Fig. 6(b). The value of the cell C (x, y, γ_0) is increased by the increment $W_p \cdot W_q$. And the value of the neighborhood cell NC $(x+1, y, \gamma_0)$ is increased by the increment $W_p \cdot W_q \cdot 1/(1+1/D_0)$ because the distance from C to NC is 1. For other neighborhood cells, the increment value can be calculated from (6). For other rotation angles, and for each point pair, similar operations are applied. Finally, a maximum value search is performed for the whole Hough cell space, and the displacement vector (x_t, y_t, γ) corresponding to the cell with maximum cell value is the desired output.

The time complexity of Step 2 is $O(n_r n_N n_L m^2)$, where n_r is the number of candidate rotation angles, n_N is the number of points in the input pattern $\mathbf{N}$, n_L is the number of points in the template pattern $\mathbf{L}$, and m is the size of the neighborhood NH. The time complexity of Step 3 is $O(n_r n_x n_y)$, where $n_r \times n_x \times n_y$ is the dimension of the Hough cell space. The space complexity of the entire algorithm is $O(n_r n_x n_y)$.

Due to the limitation of computer memory space and speed, a hierarchical version of the MWGHT is used in our implementation. In the MWGHT, the constant D_0 scales the distance weight function and should be chosen carefully. If D_0 is too small, the weight function will drop sharply while the distance gets larger. As a result, the effect of the distance weight is eliminated. If D_0 is too large, the value of the distance weight function always approaches to one, and consequently the value in each cell is nearly the same and the real maximum cell value is no longer distinguishable. In our experiments, the value of D_0 was set to be 15 cm.

III. Strategies for Model Learning and ALV Guidance

As mentioned in Section I-C, the proposed incremental-learning-by-navigation approach consists roughly of three stages. In Section III-A, an algorithm for establishing the initial learned model will be described first. In Section III-B, the proposed approach to guiding the ALV will be described next. And in Section III-C, the algorithm for updating the learned model will be illustrated finally.

A. Construction of Initial Model

The goal of the first stage of the proposed approach is to construct the initial model. The works for establishing the initial model are accomplished by the following algorithm.

Algorithm 2: Construction of Initial Model

Step 1: Perform camera calibration.

Step 2: Drive the ALV manually to the starting location, and set up the GCS of the current model by the position and orientation of the ALV.

Step 3: Start the ALV.

Step 4: Take an image of the environment using the camera.

Step 5: Record the counter of the odometer and the turn angles of the front wheels.

Step 6: Manually drive the ALV with a certain distance and an appropriate turn angle.

Step 7: If the ALV reaches the goal, go to Step 8 to perform off-line processing; else, go to Step 4 for the next cycle.

Step 8: Perform image processing to find the local vertical line features in the image taken in each cycle.

Step 9: Compute the VCS and the GCS coordinates of the local features to form a local model.

Step 10: If the first cycle is processed, add the local model to an empty model to form a global model, and go to Step 16; otherwise, go to Step 11.

Step 11: Extract desired feature points from the learned global model with a certain window in the VCS to form an extracted model.

Step 12: Perform the MWGHT to match the local model with the extracted model.

Step 13: Compute the actual slant angle and position of the ALV using the result of the previous matching.

Step 14: Re-compute the more precise GCS coordinates of the local features according to the actual slant angle and position of the ALV computed in the last step.

Step 15: For each local feature point p in the local model, check if there exists in the learned global model any feature point in the neighborhood of point p. If there exists none, add the point p to the learned global model; otherwise, compute the weighted centroid of point p and those points in the learned global model within the neighborhood of p, and add the resulting centroid point to the global model (the detailed computation process is described in Section III-C).

Step 16: Go to Step 9 if there exists a subsequent cycle; otherwise, stop.

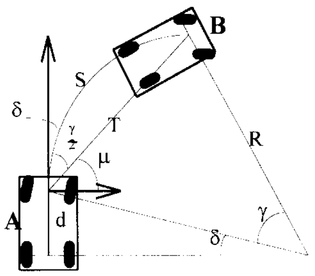


Fig. 7. The vehicle location before and after the ALV moves a distance S forward.

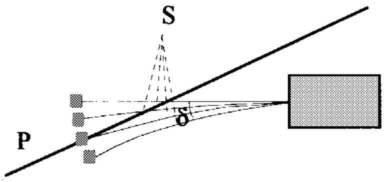


Fig. 8. Illustration of adjustment of the front wheels in a path.

In Step 9, the estimated position and orientation of the ALV is used to calculate the rough GCS position of the feature points. In our system, the estimated position and orientation of the ALV can be obtained by using the information of the feedback sensors, namely, the odometer and the photo-encoder on the front wheels. In general, when the ALV move from a known position, the new position of the ALV can be estimated by using the moving distance S and the turn angle of the front wheels. The equations to calculate the estimated ALV location are derived as follows. They can be also found in [12]. As shown in Fig. 7, the vehicle is located at A. After moving a distance S forward, the vehicle will be new location B, which is the desired estimated ALV location. The relative location of B with respect to A is denoted by a vector $\mathbf{T}$. The rotation radius R can be written as

$$R = \frac{d}{\sin \delta} \tag{8}$$

where d is the distance between the front wheels and the rear wheels, and δ is the turn angle of the front wheels. And the angle γ can be determined as

$$\gamma = \frac{S}{R}. \tag{9}$$

So, the length of vector $\mathbf{T}$ can be solved to be

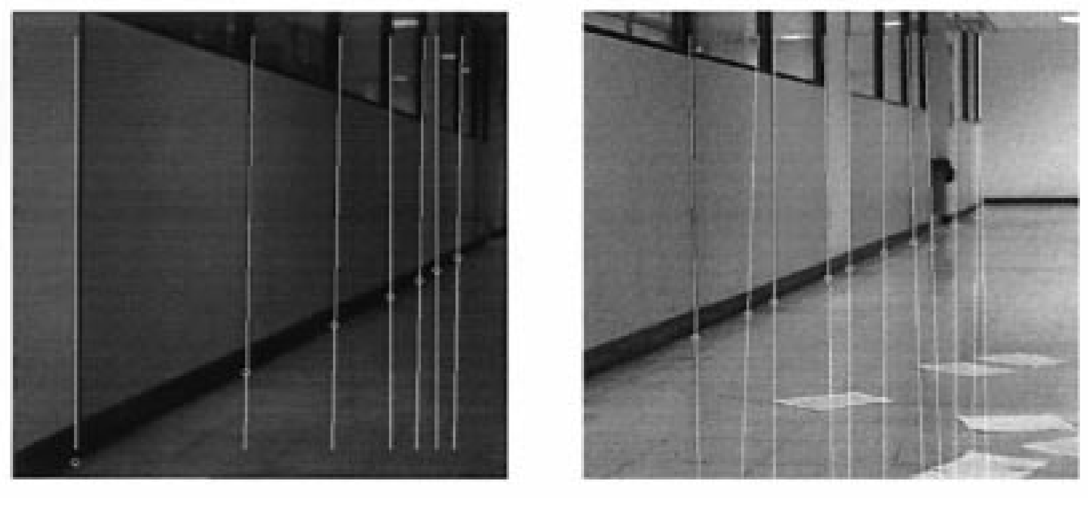
$$||\mathbf{T}|| = R\sqrt{2(1 - \cos \gamma)} \tag{10}$$

and the direction of vector $\mathbf{T}$ is

$$u = \frac{\pi}{2} - \delta - \frac{\gamma}{2}. \tag{11}$$

The VCS coordinates of location B with respect to location A can thus be computed by

$$\begin{aligned} x_B &= ||\mathbf{T}|| \cos u \\ y_B &= ||\mathbf{T}|| \sin u. \end{aligned} \tag{12}$$



(a) (b)

Fig. 9. Two examples of image processing results: (a) image with plain floor and (b) image with pieces of paper on the floor.

After the front wheel location of the ALV is determined, the rear wheel location $(\tilde{x}_b, \tilde{y}_b)$ of the ALV can also be determined to be

$$\begin{aligned}\tilde{x}_b &= x_B + d \sin \gamma \\ \tilde{y}_b &= y_B - d \cos \gamma.\end{aligned} \tag{13}$$

Since the GCS coordinates of location A are known, and the VCS coordinates of location B with respect to location A can be obtained from (12). The GCS coordinates of location B can be calculated by coordinate system transformations. Thus the desired estimated ALV location is obtained.

As mentioned previously, the time complexity of the MWGHT is proportional to the size of the data set, i.e., the number of points in the input pattern and the template pattern. As a result, the number of vertical lines in a feature pattern must be controlled within a reasonable range. Fortunately, in the indoor navigation environment, this number is usually not large. In order to achieve the goal of real-time navigation, the processing time of the MWGHT must be reduced. This work can be done by extracting just the feature points near the current ALV position from the learned global model, instead of using the full set of the model. Feature points which are impossible to appear in the camera view, e.g., the points which are far away from or behind the ALV, are of no help in the matching process. Such feature points should be discarded to speed up the matching. In our approach, a certain window in the VCS are used to extract the desired points from the learned global model (see Step 11).

The positions of the feature points in the local model are derived from the estimated ALV location. After performing the MWGHT, the displacement from the local model to the learned global model is obtained. Note this displacement is also the displacement from the estimated vehicle location to the actual one. As a result, the actual vehicle location (x_p, y_p, ω) can be obtained by

$$x_p = \hat{x}_p + x_t, \quad y_p = \hat{y}_p + y_t, \quad \omega = \hat{\omega} - \gamma \tag{14}$$

where $(\hat{x}_p, \hat{y}_p, \hat{\omega})$ is the estimated vehicle location, and (x_t, y_t, γ) is the displacement vector.

B. Steps of Navigation Cycle

Basically, an ALV navigation process includes the tasks of grabbing and processing images, locating the ALV, making guidance decisions, and executing steering control procedures. The proposed navigation process is described by the following algorithm.

Algorithm 3: ALV Navigation Process

Step 1: Read the learned global model and the planned path.

Step 2: Take an image of the environment using the camera and compute the initial position and orientation of the static ALV.

Step 3: Start the ALV.

Step 4: Take an image of the environment using the camera.

Step 5: Perform image processing to find the local vertical line features.

Step 6: Compute the VCS and the GCS coordinates of the local features to form a local model.

Step 7: Extract desired feature points from the learned global model with a certain window in the VCS to form an extracted model.

Step 8: Perform the MWGHT to match the local model with the extracted model.

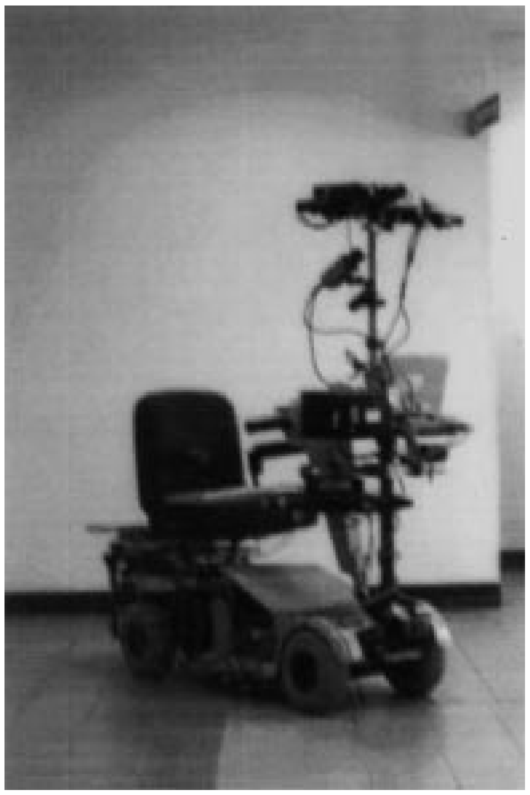
Step 9: Compute the actual slant angle and position of the ALV using the result of the previous matching.

Step 10: Re-compute the more precise GCS coordinates of the local features to refine the local model according to the actual slant angle and position of the ALV computed in the last step. Store the coordinates and the confidence weights of the local features for updating the model (see Algorithm 4 described later).

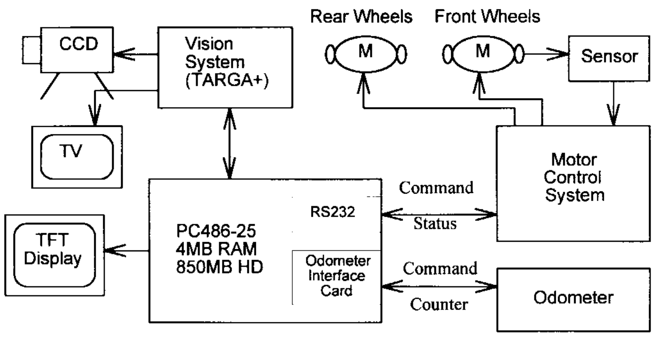
Step 11: Determine the turn angle of the front wheels to guide the ALV close to the extracted path portion and turn the front wheels of the ALV (the details are illustrated later).

Step 12: If the ALV reaches the goal of the desired path, then stop; else, go to Step 4.

The scheme for adjusting the driving wheel direction δ in this study is based on the wheel adjustment strategy described in [12]. The basic idea is to search a turn angle of the front wheels to drive the ALV as close to the desired path as possible. As shown in Fig. 8, given a reasonable moving distance S and a fixed turn angle of the front wheels, the location of the ALV can be estimated, as discussed in Section III-A. Given a path P, either a straight line or a circular segment, define $D_P^F(\delta)$ as the distance from the midpoint of the ALV front wheels to the given path P after the ALV traverses a certain distance S with the turn angle δ, where S may be assigned to be the average navigation distance during a cycle. Also, define $D_P^B(\delta)$ as the distance from the midpoint of the ALV back wheels to the given



(a)



(b)

Fig. 10. The prototype ALV used in the experiments: (a) external view and (b) system structure.

path P. Finally, define measure L_P to be

$$L_P = D_P^F(\delta) + D_P^B(\delta). \tag{15}$$

To find the turn angle of the front wheel to drive the ALV as close to the path as possible, an exhaustive search is performed to find the angle that produces the minimal value of L_P, and the obtained angle is used as the turn angle for safe navigation.

C. Strategies for Updating the Learned Model

The proposed algorithm for updating the learned model after a navigation training is described as follows.

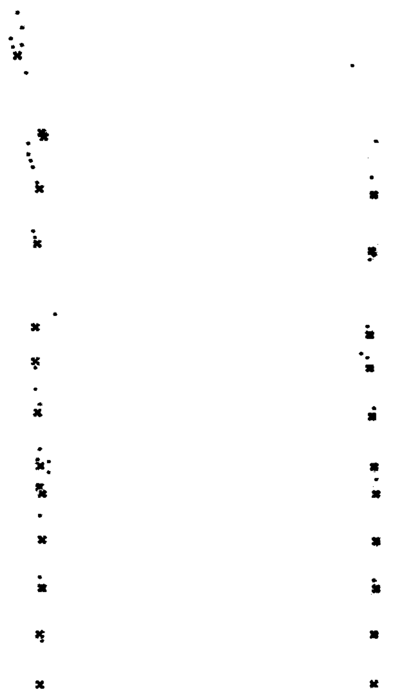
Algorithm 4: Strategies for Updating the Learned Global Model:

Step 1: For each local feature point p recorded in the navigation session, check if there exists in the learned global model any feature point in the neighborhood of point p. If there exists none, add the point p to the learned global model; otherwise, compute the weighted centroid of point p and those points in the learned global model within the neighborhood of p, and add the resulting centroid point to the global model.

Step 2: Normalize the confidence weights of the feature points in the learned model.

Step 3: Discard those feature points whose normalized confidence weights are smaller than a certain threshold value T_s.

Step 4: Adjust the confidence weight of the primary point, defined to be the feature point nearest to the origin of the GCS, to be the maximum of all the confidence weights.



(a) (b)

Fig. 11. An example of learned global models: (a) initial learned model and (b) refined model after five iterations of incremental-learning-by-navigation. The crosses represent real vertical lines and the spots are noise. Note the removal of some noise points.

In Step 1, the size of the neighborhood can be arbitrarily chosen. Choosing a large size of the neighborhood would cause the combination of two distinct feature points. Choosing a small size of the neighborhood would leave unmerged a group of feature points which come from inexact computation results of a single feature point (i.e., a single vertical line in the environment). However, the choice of the neighborhood size does not affect the result of the MWGHT too much. A reason is that, in the MWGHT, several close feature points with small confidence weights are equivalent to a feature point at the position of their centroid and with a large confidence weight.

When a new feature point is added into the model, the initial confidence weight attached to this new feature point P_k is assigned by the following equation:

$$W_{P_k} = \frac{1}{1 + (x_{VCS}^2 + y_{VCS}^2)/C_d} \tag{16}$$

where x_{VCS} and y_{VCS} are the x and y coordinates of P_k in the VCS, respectively, and C_d is a predefined constant. The coordinates (x_C, y_C) and the confidence weight W_C of the weighted centroid of

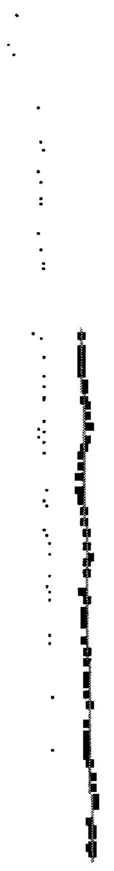


Fig. 12. Illustration of the top view of a navigation session. The black squares mark the trace of the ALV, the gray line within the black-square trace is the planned path, and the little black spots are extracted feature points. Note the closeness of the ALV trace to the planned path.

a group of points is calculated as follows:

$$x_C = \frac{\sum W_{P_i} \cdot x_{P_i}}{\sum W_{P_i}}, \; y_C = \frac{\sum W_{P_i} \cdot y_{P_i}}{\sum W_{P_i}}, \; W_C = \sum W_{P_i} \quad (17)$$

where (x_{P_i}, y_{P_i}) and W_{P_i} are the coordinates and the confidence weight of a point P_i in the group, respectively. In Step 2, the normalized confidence weight $\hat{W}_{P_i}$ of a feature point P_i is calculated by the following equation:

$$\hat{W}_{P_i} = m \cdot \frac{W_{P_i}}{\sum W_{P_i}} \quad (18)$$

where W_{P_i} is the original confidence weight of P_i, and m is the number of feature points in the learned global model.

Note that the average value of the normalized confidence weights is always one. The normalized confidence weight of a stable feature point will get larger and larger during the learning-by-navigation iterations, while the weight of a noise feature point will get smaller and smaller. When the normalized confidence weight of a feature point gets lower than a certain threshold, the feature point is regarded as noise. Such a kind of feature point contributes nothing in the matching scheme, and thus may be removed from the learned model (see Step 3). After a sufficient number of learning-by-navigation iterations, only stable feature points are left in the learned global model and all noise points will be removed. Consequently, the goal to establish a stable and practical model for the guidance of the ALV may be achieved.

The primary point mentioned in Step 4 is chosen to be the one closest to the origin. It is also the nearest feature point to the starting location of the ALV in each navigation session. It plays an important role in determining the initial location of the ALV. In real navigation trainings, the primary point can usually be found in just one or two cycles in the whole navigation session. It contrasts with other feature points which might appear in four or more captured images in the navigation cycles. As discussed in the previous section, fewer occurrences result in smaller confidence weights. Thus, in order to emphasize the importance of the primary point, it is assigned with the maximum confidence weight in Step 4.

IV. Experimental Results

The image processing work of our system can be divided into three steps. The first step is to find vertical edges. In the second step, the Hough transform is performed to detect vertical lines using the edge points. To speed up the system, only nearly vertical lines are searched. The third step is to find the cross points of the detected vertical lines and the base lines of corridors. Then the cross points are used to locate our vertical line features, as discussed in Section II. Two examples of image processing results is shown in Fig. 9.

The external view of the prototype of the ALV is shown in Fig. 10(a). The ALV is computer-controlled with a modular architecture, as shown in Fig. 10(b), including four major components, namely, a vision system, a central processor PC, a motor control system, and a dc power system. The vision system consists of one camera, a TV monitor, and a TARGA+ image frame grabber. The central processor PC is an IBM PC/AT compatible personal computer with an Intel 80 486 CPU, 4 MB of main memory, one floppy disk driver, a 850 MB hard disk, and a TFT display. The motor control system consists of a main control board with an Intel 8085 controller, a motor driver, and two motors.

The ALV learning and navigation experiments were performed in a building corridor in National Chiao Tung University. By using the proposed approach, many successful navigation sessions have been conducted. The navigation speed of the vehicle is about 30 cm/s. The computation time of a navigation cycle ranges approximately from 1.5 s to 3.5 s for different images. Fig. 11 shows an example of learned global models. Fig. 11(a) is the initial learned model, and Fig. 11(b) is the refined model after five iterations of learning-by-navigation. There are totally 39 and 22 feature points in Fig. 11(a) and (b), respectively. The crosses and the spots in Fig. 11 represent the feature points whose normalized confidence weights are larger and less than 1.0, respectively. Thus, the crosses are real vertical lines, and the spots might be noise. By observing Fig. 11, we can find that some noise points were eliminated during the incremental learning process. This shows the effectiveness of our approach. Fig. 12 shows the trace of the ALV in one navigation session. In the figure, the black squares represent the trace of the ALV, the little black spots represent the vertical line features, and the gray straight line represents the planned path. Note the closeness of the ALV trace to the planned path.

V. Conclusion

An incremental-learning-by-navigation approach has been proposed for ALV learning and navigation in indoor corridors. Computer vision techniques have been proposed to locate an ALV by the use of the vertical line features in a corridor. The approach is reliable because of the robustness of the proposed MWGHT matching scheme. It is also flexible because the learned environment model can be updated after each navigation session. Each navigation session becomes a training to the ALV; even a coarse initial learned model can be refined to be a more precise one after several passes of navigation. The proposed approach has been implemented on a prototype ALV and successful navigation sessions in real time confirm the effectiveness of the approach.

REFERENCES

[1] G. Saussure, C. R. Weisbin, and P. F. Spelt, "Navigation and learning experiments by an autonomous robot," *Robot. Comput.-Intergr. Manuf.,* vol. 6, no. 4, pp. 295–301, 1989.

[2] K. Onoguchi, M. Watanabe, Y. Okamoto, Y. Kuno, and H. Asada, "A visual navigation system using a multi-information local map," in *Proc. 1990 IEEE Int. Conf. Robotics Automation,* Cincinnati, OH, May 1990, vol. 2, pp. 99–105.

[3] X. Lebègue and J. K. Aggarwal, "Extraction and interpretation of semantically significant line segments for a mobile robot," in *Proc. 1992 IEEE Int. Conf. Robotics Automation,* Nice, France, May 1992, pp. 1778–1785.

[4] ——, "Generation of architectural CAD models using a mobile robot," in *Proc. 1994 IEEE Int. Conf. Robotics Automation,* San Diego, CA, May 1994, vol. 1, pp. 711–717.

[5] F. Nashashibi, M. Devy, and P. Fillatreau, "Indoor scene terrain modeling using multiple range images for autonomous mobile robots," in *Proc. 1992 IEEE Int. Conf. Robotics Automation,* Nice, France, May 1992, vol. 1, pp. 40–46.

[6] F. Nashashibi and M. Devy, "3-D incremental modeling and robot localization in a structured environment using a laser range finder," in *Proc. 1993 IEEE Int. Conf. Robotics Automation,* May 1993, vol. 1, pp. 20–27.

[7] H. Ishiguro, T. Maeda, T. Miyashita, and S. Tsuji, "A strategy for acquiring an environmental model with panoramic sensing by a mobile robot," in *Proc. 1994 IEEE Int. Conf. Robotics Automation,* San Diego, CA, May 1994, vol. 1, pp. 724–729.

[8] P. Y. Ku and W. H. Tsai, "Model-based guidance of autonomous land vehicle for indoor navigation," in *Proc. 1989 Workshop Computer Vision, Graphics, Image Processing,* Taipei, Taiwan, R.O.C., Aug. 1989, pp. 165–174.

[9] K. T. Chang and W. H. Tsai, "Collision avoidance for autonomous land vehicle navigation in indoor environments by laser light projection," in *Proc. 1991 Workshop Computer Vision, Graphics, Image Processing,* Tainan, Taiwan, R.O.C., Aug. 1991, pp. 334–339.

[10] S. D. Cheng and W. H. Tsai, "Model-based guidance of autonomous land vehicle in indoor environments by structured light using vertical line information," in *Proc. 1991 Workshop Computer Vision, Graphics, Image Processing,* Tainan, Taiwan, R.O.C., Aug. 1991, pp. 340–345.

[11] Y. M. Su and W. H. Tsai, "Autonomous land vehicle guidance for navigation in building corridors and elevators by computer vision, radio, and photoelectric sensing techniques," in *Proc. 1992 Conf. Computer Vision, Graphics, Image Processing,* Nantou, Taiwan, R.O.C., Aug. 1992, pp. 237–244.

[12] F. M. Pan and W. H. Tsai, "Automatic environment learning and path generation for indoor autonomous land vehicle guidance using computer vision techniques," in *Proc. 1993 Nat. Computer Symp.,* Chia-Yi, Taiwan, R.O.C., 1993, pp. 311–321.

[13] J. D. Foley and A. V. Dam, *Fundamentals of Interactive Computer Graphics.* Reading, MA: Addison-Wesley, 1982.

[14] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision.* Reading, MA: Addison-Wesley, 1993, vol. 2.

[15] S. C. Jeng and W. H. Tsai, "A study of generalized Hough transform: Fast, scale- and orientation-invariant, and distance-weighted algorithms," Tech. Rep., Inst. Comput. Sci. Inf. Eng., National Chiao Tung Univ., Hsinchu, Taiwan, R.O.C., 1991.

[16] C. C. Lai and W. H. Tsai, "Outdoor autonomous land vehicle guidance by road information using computer vision and fuzzy wheel adjustment techniques," Tech. Rep., Inst. Comput. Sci. Inf. Eng., National Chiao Tung Univ., Hsinchu, Taiwan, R.O.C., 1993.

[17] D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recognit.,* vol. 13, no. 2, pp. 111–122, 1981.

Requirements Specification and Analysis of Digital Systems Using Fuzzy and Marked Petri Nets

Victor R. L. Shen and Feipei Lai

***Abstract*—Fuzzy information often appears in the system requirements. Fuzzy Petri nets (FPN) are Petri nets in which certain fuzzy truth-values are assigned to its transitions. In this paper, we show how the FPN model can be used for formal specification and verification of digital systems. The consistent FPN model is actually a state machine, from which we can obtain a consistent marked Petri net (MPN) model. Based on the consistent MPN model, the hardware prototype at register transfer level can be easily induced by using the optimization rules. Finally, main results are presented in the form of three theorems and are supported by some experiments.**

***Index Terms*— Formal analysis, knowledge base, Petri nets, requirements specification.**

I. INTRODUCTION

In developing a top-down design, we usually start from the system level and keep on breaking the design process down until we reach a level where the design can be constructed with standard off-the-shelf parts or can be synthesized with synthesis tools. Consequently, the requirements specification plays a central role among the hardware development stages. However, the real world is not linearly quadratic and many situations can not be modeled accurately by simple mathematical equations. Fuzzy information often appears in the system requirements. Especially, how to make a decision among numerous requirement clauses thus becomes a hard part as reasoning a large scale knowledge base. Meanwhile, the problem of how to deal with the uncertain, inexact, and vague nature of information has received much attention during the last decade [1]. Furthermore, many researchers are interested in the *executable specification* or *high-level synthesis* nowadays [14], [19].

The Algorithmic State Machine (ASM) chart was developed by Osborne in 1973 [15]. Since then it has been widely applied to express the abstract algorithm and to support the conversion of the algorithm into hardware. But, it does not provide a straightforward way to deal with the fuzzy information and to solve some hardware problems like *deadlocks* and *hazards.* This motivates us to propose an attractive alternative to solve the problems.

A knowledge-based approach to implementing a digital system contains the following advantages [7].

1) The programming paradigm easily supports data-driven computation. The subproblem ordering task can be conveniently cast in a data-driven approach.
2) Production rules naturally represent the design knowledge adopted.
3) A knowledge base is easily extended as compared to manipulating procedural code.

Manuscript received August 12, 1995; revised July 26, 1996 and May 1, 1997. This work was supported in part by the National Science Council of Taiwan under Grant NSC 85-2215-E002- 017.

V. R. L. Shen is with the Department of Electrical Engineering, Mingchi Institute of Technology, Taipei 243, Taiwan, R.O.C. (e-mail: rlshen@ccsun.mit.edu.tw).

F. Lai is with the Department of Electrical Engineering and the Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: flai@cc.ee.ntu.edu.tw).

Publisher Item Identifier S 1083-4419(98)07305-1.