Unsupervised Learning of Unexplored Environment by Pushdown Transducer for Autonomous Land Vehicle Navigation

Guan-Yu Chen (陳冠宇) and Wen-Hsiang Tsai (蔡文祥)

Department of Computer and Information Science
National Chiao Tung University
Hsinchu, Taiwan 300, Republic of China
Telephone : 886-3-5712121ext56650 Fax : 886-3-5721490
Email : gis81510@cis.nctu.edu.tw

ABSTRACT

An approach to unsupervised learning of the environment for autonomous land vehicle (ALV) navigation by the use of a pushdown transducer is proposed. The ALV may, without human's involvement, navigate in an unknown closed environment, collect the information of the environment features, and then build a top-view map of the environment for ALV navigation or other applications. In the learning system, the pushdown transducer is employed to guide the ALV to explore the entire navigation environment. The sensed local environment features are encoded into symbols for use as input to the pushdown transducer by a preprocessing unit. Output symbols representing special ALV actions are generated by the transducer to guide the ALV. Some theoretical proofs showing that the proposed transducer is able to explore the entire region of any closed navigation environment in finite cycles are included. A learning system for simulated grid model environments was implemented and several simulation results show the feasibility of the proposed approach.

Keyword : unsupervised learning, autonomous land vehicle navigation, pushdown transducer, formal language theory

1. Introduction

Many vision-based autonomous land vehicles (ALV's) have been developed in recent years, and model-based guidance approaches are often employed in ALV navigation. However, the traditional method of establishing environment models by manual measurement of the navigation environment is a time-consuming work. It is thus desired to design a system for automatic modeling of navigation environments. As a result, several environment learning systems were developed in the recent years [1-9]. Many other mobile robot learning systems were also constructed [10-13] for use in applications other than navigation environment modeling, such as navigation, goal reaching, obstacle avoidance, etc.

For environment learning, Lebègue and Aggarwal [1,2] 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. The basic assumption of their study is that the navigation environment is with prominent 3-D orientations. Such assumption stands in most building corridors. Nashashibi et al. [3] proposed an approach to building a rough geometric model for a 3D terrain using a laser range finder. They also gave algorithms to build snapshot models with planar faces from range data. By performing 3D data fusion between the snapshot models, the proposed approach can build a reliable 3D model incrementally [4]. Ishiguro et al. [5] presented a strategy for establishing the model of an unknown environment by a mobile robot. Panoramic sensing was used to perceive the structure of the environment in their implementation. Kurz [6] introduced an approach to generating environmental maps based on ultrasonic range data. Free-space can be partitioned into situation areas by means of a learning classifier. Then the situation areas can be attached to graph nodes by dead-reckoning and finally a map of the free-space in the form of a graph representation is generated. Dean et al. [7] formulated map learning as the problem of inferring the structure of a reduced deterministic finite automaton from noisy observations and also provided an exploration algorithm to learn the correct structure of the automaton. Pan and Tsai [8] proposed an integrated approach to automatic model learning and path generation for vision-based ALV guidance in building corridors.

In most learning systems, certain involvement from human operators is required to complete the learning process. For example, in our previous work, the ALV should be driven manually by a human operator along the environment for initial learning. However, for some applications, it is impractical to get human's involvement. A typical example of such applications is the use of autonomous mobile robots in nuclear plants or other dangerous regions. For this case, all operations, including the learning process, should be full automatic. As a result, the capability to explore an unknown navigation environment automatically is required, demanding an unsupervised learning scheme. Furthermore, with the capability of automatic exploration, the ALV can serve as a safety guard, being able to patrol around certain environments. The ALV may also reach any goal automatically if it is designed with the capability to explore the entire navigation automatically. The proposed learning scheme is developed to meet these requirements.

As discussed previously, the first step of unsupervised learning is automatic exploration. In the proposed approach, a pushdown transducer, called *navigation transducer*, is designed to guide the ALV to explore the unknown environment. The sensed local environment features are encoded into symbols for use as input to the pushdown transducer by a preprocessing unit, and output symbols representing special ALV actions are generated by the navigation transducer to guide the ALV. The sensed local environment features are then merged into the learned global model with a model matching scheme. In this way, a global map may be generated after the ALV have explored the entire navigation region.

To concentrate on developing a system with the capability of automatic exploration, a simulated grid environment model is introduced to simplify the environment learning works, including map establishment, environment feature detection, ALV self-location, etc. It is also more convenient to develop the kernel of automatic exploration, namely, the navigation transducer, in such an environment model. An unsupervised learning system for grid environment models has been built. Several simulation results and theoretical proofs show the correctness of the proposed approach. Real-world environments may be reduced to grid environment models by appropriate transformations, and the proposed learning scheme may be extended for real-world environments with certain modifications. These topics are beyond the scope of this study and are the directions of future works.

The remainder of this paper is organized as follows. In Section 2, the proposed method for unsupervised learning of ALV navigation environments is described. In Section 3, the proposed navigation transducer is described in detail. In Section 4, the correctness of the proposed navigation transducer is proved. In Section 5, several simulation results are presented. Finally, the conclusion and further works of this paper is given in Section 5.

2. Proposed Environment Learning Systems

2.1 System overview

The proposed ALV learning system consists of three subsystems, a feature location subsystem, a map building subsystem, and an automatic exploration subsystem. The feature location subsystem is designed to extract and locate the environment features. The automatic exploration subsystem consists of a navigation transducer and a preprocessing unit. The preprocessing unit encodes the extracted local environment features into input symbols for the navigation transducer. The navigation transducer is a pushdown transducer which takes the encoded local environment features as input and yields ALV reactions as output corresponding to the current system status. It serves as the guidance kernel, which leads the ALV to explore the entire navigation environment automatically. The map building subsystem builds and keeps track of the learned global model. For each learning cycle, it merges the local environment features to the global model, or adjusts the global model when multi-occurrence, of environment features is encountered. The interaction of these subsystems are shown in Figure 1.

2.2 Steps of learning algorithm

As mentioned previously, the unsupervised learning work is accomplished by the cooperation of the three subsystems. Generally speaking, the environment learning work is incrementally accomplished by several successive cycle runs. In a learning cycle, the feature location subsystem first extracts the local environment features from sensor inputs, matches the features to the global model, and then compute the location of the features. The map building subsystem then merges the extracted local environment features in the global model. The extracted local environment features are also passed to the automatic exploration subsystem. The navigation transducer generates the output according to the encoded input. The output is then passed to the ALV control unit, and the control unit performs a certain operation corresponding to the transducer output, like moving the ALV to a new position. This completes a learning cycle. The same procedures are performed repeatedly in each learning cycle. A flowchart of the proposed learning procedures is shown in Figure 2. A detailed description of the learning algorithm is shown as follows.

Algorithm 1 **Unsupervised learning of unexplored environment for ALV navigation.**

Step 1. Perform sensor calibration.
Step 2. Drive the ALV manually to the starting location and start the ALV.
Step 3. Set the initial global model to empty.
Step 4. Get inputs from the sensors.
Step 5. Extract environment features from the sensor inputs.
Step 6. Calculate the location of the extracted environment features and set up a local model by collecting the extracted local features.
Step 7. If the global model is non-empty, then match the local model with the global model and recalculate the accurate position of the local features by the matching result.
Step 8. Attach the local model to the global model.
Step 9. Encode the extracted local features and send the encoded symbol to the navigation transducer as input.
Step 10. If the output of the navigation transducer is 'stop', then stop the ALV; otherwise, perform the ALV operation corresponding to the transducer output, and then go to Step 4 to start another cycle.

2.3 Grid environment model

The grid environment model is a simulation of an indoor navigation environment which meets the following conditions:

1. The navigation environment is composed of numerous square grids.

2. The obstacles or the walls are located on the edges of the grids and the length of the wall is equivalent to the side-length of the grid. That is, each side of a grid is either fully open or fully occupied by a wall.
3. For each navigation cycle, the ALV may move to the left-, right-, upper-, or lower- neighbor of the grid where the ALV is currently located, depending on the output of the navigation transducer.
4. The fields of view of the sensors of the ALV are within the grid where the ALV is currently located.

As mentioned previously, with the use of the grid environment model, the development of the exploration algorithm and the environment learning works are facilitated. The behavior of the exploration kernel, the navigation transducer, can be defined clearly in such an environment model. Although the grid model environment loses some degree of reality; however, with certain modification the techniques developed from learning of the grid environment model can be applied to learning of the real-world environment.

3. Navigation Transducer

The proposed navigation transducer serves as an automatic exploration kernel, which enables the ALV to explore the entire navigation environment without human involvement. In this section, the definition of a typical pushdown transducer is first introduced. Then, the principle and the structure of the proposed navigation transducer are described. In the remainder of this section, a brief proof showing that the proposed transducer is able to explore the entire environment region and then stops in finite steps will be described.

3.1 Definition of pushdown transducer

A pushdown transducer (PDT) is an eight-tuple, $M = (Q, \Sigma, \Gamma, \Delta, \delta, \lambda, q_0, Z_0)$, where

1. Q is a finite nonempty set of states;
2. Σ is a finite nonempty set of input alphabet;
3. Γ is a finite nonempty set of stack alphabet;
4. Δ is a finite nonempty set of output alphabet;
5. δ is a transition function,
   $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to Q \times \Gamma^*$, where ε denotes empty input;
6. λ is an output function, $\lambda : Q \times \Sigma \times \Gamma \to \Delta$;
7. q_0 is the initial state; and
8. Z_0 is a particular stack symbol, called the start symbol.

The above notations of the PDT are based on those used in Hopcroft and Ullman [14]. An interpretation of the transition function

$$\delta(q,a,Z) = \{(p_1,\gamma_1),(p_2,\gamma_2),\ldots,(p_m,\gamma_m)\}$$

where $q, p_i \in Q, 1 \le i \le m$; $a \in \Sigma$; $Z \in \Gamma$; and $\gamma_i \in \Gamma^*$, $1 \le i \le m$, is that the PDT in state q, with the input symbol a and the top symbol Z on the stack can, for any i, enter state p_i, replace symbol Z by string γ_i, and advance the input head one symbol. The interpretation of the transition function

$$\delta(q,\varepsilon,Z) = \{(p_1,\gamma_1),(p_2,\gamma_2),\ldots,(p_m,\gamma_m)\}$$

is that the PDT in state q with the top symbol Z on the stack, independent of the input symbol being received can, for any i, enter state p_i and replace symbol Z by string γ_i. The interpretation of the output function $\lambda(q,a,Z) = b$, where $q \in Q$, $a \in \Sigma$, $Z \in \Gamma$, $b \in \Delta$, is that the PDT in state q, with input symbol a and the top symbol Z on the stack can output symbol b.

3.2 Principle of navigation transducer

When a man attempts to walk through a building, the most common rules for him to follow is as follows. First, walk along a corridor. Secondly, when a crossing is encountered, i.e., when more than two candidate paths can be chosen, select one of them and put a mark on the chosen path to distinguish it from the unexplored ones, and then go along the chosen path. Thirdly, when an end of the current path is encountered, go backward along the current path to the previous crossing, pick another unexplored path, put a mark on the selected path, and go. Finally, if all available paths are explored, go backward furthermore to another crossing until an unexplored path is found. In this way, one may either find the exit of the building or explore the entire building and find there is no exit.

The proposed navigation transducer is designed based on the above rules. The rules seem straightforward, but the design of the transducer is complicated. Sensed environment features are transformed by the preprocessing unit into symbols first as input to the transducer, and the transducer emits the navigation direction as output to guide the ALV to explore the unknown environment.

The stack of the navigation transducer is used to keep track of the marks, each of which indicates whether a certain path is explored or not. When one crossing is encountered, the symbols representing all unexplored paths from this crossing are pushed down to the stack of the transducer except the selected one, and the navigation transducer guides the ALV to move along the selected path. When the ALV encounters an end of the corridor, according to the third rule, the navigation transducer guides the ALV to retreat backward to the previous crossing; when arriving the previous crossing, a new path is popped out of the stack, and the navigation transducer guides the ALV to follow this new path. Besides, when all paths from the crossing have been explored, a similar retreating procedure is performed again to guides the ALV to the previous crossing. Unfortunately, if no further information is kept, the navigation transducer will not be able to know from which path the ALV comes to this crossing, so the retreating work is unfeasible obviously. To solve this problem, the retreating path should be pushed before other unexplored paths are pushed. By the nature of stack operations, namely, first in last out, the retreating path would be popped out

only after all unexplored paths are popped out. That is, the ALV retreats from one crossing only if all paths from the crossing have been explored. This ensures the ALV to explore the entire region systematically.

The stack operations of the navigation transducer are summarized as follows.

- In the *exploring mode* indicating that the ALV is navigating in an unexplored region, if a crossing is encountered, choose a path to go, push the retreating path into the stack first and then those unexplored paths except the chosen one, and finally guide the ALV to move along the chosen path.
- In the *retreating mode* indicating that the ALV has visited a path end and is retreating to the previous crossing, if the ALV reaches the crossing, pop one path from the stack, and then guide the ALV to go along the popped path.

The state of the navigation transducer is used to check whether the ALV is navigating in the exploring mode or in the retreating mode. The state transition function yields the corresponding new state in accordance with the current state, the input, and the content of the stack. The detailed implementation of the navigation transducer is illustrated next.

3.3 Implementation of navigation transducer

The proposed navigation transducer is constructed according to the principle described previously. The preprocessing unit transforms the surrounding environment features into input symbols. Fifty-two input symbols are identified. They are categorized into seven sets by their properties in order to simplify the illustration of the transition functions, as shown in Figure 3. The first set of input symbols includes "*one-way grids*". They correspond to the grids with two walls on their four borders. The second set of input symbols includes "*ending grids*". They are the ones with three walls on their borders. The third set of input symbols includes "*crossing grids*". The symbols in this set correspond to grids with three or more open sides on their borders. The fourth set includes "*explored grids*". The preprocessing unit transforms a grid into a symbol of an "explored grid" if the grid is a crossing and is explored once before the current transition. The squares are colored in gray to represent the attribute of "being explored". The fifth set includes "*one-way starting grids*". Each symbol in this set corresponds to a starting grid with only one open side. The sixth set includes "*multi-way starting grids*". Each symbol in this set corresponds to a starting grid with two or more open sides. The seventh set includes "*dead starting grids*". Each symbol in this set corresponds to a starting grid with no way to exit. Starting grids are regarded to be different from other grids because each of them has no previous move. Note that the preprocessing unit generates starting grid symbols only when the ALV is in the start condition, and the starting grids are encoded as normal grids after the ALV leaves the start condition.

Without keeping track of the grids which have been explored, to explore the entire region of an environment including a loop, e.g., like that of G-H-I-J-G in Figure 4, becomes unfeasible because the ALV might navigate within the loop repeatedly. To solve this problem, a scheme is proposed for the ALV to handle a loop in a grid environment. In the proposed scheme, an explored grid is regarded as an ending grid with three walls when the ALV is in the exploring mode. This prevents the ALV from keeping moving in the loop and retreats the ALV to the entrance grid of the loop. However, keeping track of all grids needs a large memory if the number of grids is large. It is also unfeasible for the real-world environment because of non-existence of the grids. In the proposed scheme, only crossing grids are kept track of for memory saving. Furthermore, it is also reasonable to track the crossing in the real-world environment since crossings in a real-world environment are countable and limited.

The mappings between the environment features and the input symbols are listed in Figure 3. In the figure, the arrows show the previous move of the ALV; the thick lines represent *real walls* in navigation environments; the thin lines show the open sides of the grid; and the dash-lines represent the *false walls* generated by the preprocessing unit. The false walls are not real walls. They are generated to avoid dumb loop navigation, namely, to avoid moving back immediately to the previous grid, and also to indicate the previous move direction.

There are five output symbols for the proposed navigation transducer, and each output corresponds to an ALV action, as shown in Table 1. The states of the navigation transducer are listed in Table 2. The navigation transducer starts in the *exploring state*, and halts in the *final state*. The stack symbols are listed in Table 3. Each symbol in the stack corresponds to an unexplored path or a retreating path. For example, the symbol E_l corresponds to an unexplored path to the left-side neighbor grid.

The transition function determines the behavior of the transducer. It maps the input, the current state, and the current stack status into a new transducer state and a new stack status. For the proposed navigation transducer, there are fifty-two input symbols, nine stack symbols, and three states. So there are up to $52\times(9-1)\times3=1,560$ transition rules for this transducer and it is unfeasible to list all rules here. Instead, only several principles, named as '*super rules*', are described as follows. These super rules are induced from the principle of the transducer behavior mentioned previously.

1. For a "one-way grid" input or a "one-way starting grids" input, neither the state nor the stack content need be changed. An example which follows from this super rule is $\delta(q_e, G_1, E_b) = \{(q_e, E_b)\}$.
2. For an "ending grid" input:
   2.1. If the navigation transducer is in the "exploring state", change the state of the navigation transducer to the "retreating state" and keep the content of the stack unchanged. An example which follows this super rule is $\delta(q_e, G_{29}, E_b) = \{(q_r, E_b)\}$.
   2.2. If the navigation transducer is in the "retreating state" and the stack is empty, change the state to

the "final state". An example which follows from this super rule is $\delta(q_r, G_{29}, Z_0) = \{(q_r, Z_0)\}$.

3. For a "dead starting grid" input, change the state to the "final state". The only rule which follows from this super rule is $\delta(q_e, G_{52}, Z_0) = \{(q_{fin}, Z_0)\}$.
4. For a "crossing grid" input or a "multi-way starting grid" input, push the symbol which corresponds to the retreating path, i.e., the direction to the previous grid, onto the stack first; then scan the four sides of the current grid in the sequence of left, right, top, bottom; ignore the first open side, which is chosen to be the navigation direction ; and push the symbols which correspond to the other open sides onto the stack. In this case, the state of the navigation transducer remains in the "exploring state". An example which follows from this super rule is
   $\delta(q_e, G_{24}, E_b) = \{(q_e, E_l E_r R_b E_b)\}$.
   Note that it is impossible to have a "crossing grid" input when the navigation transducer is in the 'retreating state'.
5. For an "explored grid" input:
   5.1. If the navigation transducer is in the "exploring state", change the state of the navigation transducer to the "retreating state" and keep the content of the stack. An example which follows from this super rule is $\delta(q_e, G_{33}, E_b) = \{(q_r, E_b)\}$.
   5.2. If the navigation transducer is in the "retreating state", if the stack is empty, i.e., if the popped symbol is the start symbol of the stack, change the state into the "final state".
   $\delta(q_r, G_{33}, Z_0) = \{(q_{fin}, Z_0)\}$ is the only rule which follows from this case. If the stack is nonempty, pop a symbol from the top of the stack, and if the popped symbol corresponds to an unexplored path, switch the state to the "exploring state";
   $\delta(q_r, G_{33}, E_b) = \{(q_e, \varepsilon)\}$ is an example which follows from this case. If the popped symbol corresponds to a retreating path, keep the state in the "retreating state"; $\delta(q_r, G_{33}, R_b) = \{(q_r, \varepsilon)\}$ is an example which follows from this case.

The output function maps the input, the current state, and the current stack status into an output symbol. There are up to 936 output mapping rules here. Instead of listing all rules, the idea of '*super rules*' is employed again. The super rules for the output function are listed as follows.

1. For a "one-way grid" input or a "one-way starting grid" input, take as output the symbol which corresponds to the direction of the only open side implied in the input symbol. An example which follows from this super rule is $\lambda(q_e, G_1, E_b) = \{O_l\}$.
2. For an "ending grid" input:
   2.1 If the navigation transducer is in the "exploring state", take as output the symbol which corresponds to the direction of the false wall implied in the input symbol, i.e., the direction to the previous grid. An example which follows from this super rule is $\lambda(q_e, G_{29}, E_b) = \{O_l\}$.
   2.2 If the navigation transducer is in the "retreating state" and the stack is empty, take the "stop ALV" symbol as output. An example which follows from this super rule is $\lambda(q_r, G_{29}, Z_0) = \{O_s\}$.
3. For a "dead starting grid" input, take the "stop ALV" symbol as output. The only rule which follows from this super rule is $\lambda(q_e, G_{52}, Z_0) = \{O_s\}$.
4. For a "crossing grid" input or a "multi-way starting grid" input, scan the four sides of the current grid in the sequence of left, right, top, and bottom; and take as output the symbol which corresponds to the direction of the first open side. An example which follows from this super rule is
   $\lambda(q_e, G_{24}, E_b) = \{O_l\}$.
5. For an "explored grid" input:
   5.1. If the navigation transducer is in the "exploring state", take as output the symbol which corresponds to the direction to the previous grid. An example which follows from this super rule is
   $\lambda(q_e, G_{33}, E_b) = \{O_l\}$.
   5.2. If the navigation transducer is in the "retreating state", check if the stack is empty first. If the stack is non-empty, pop a symbol from the top of the stack, take as output the symbol which corresponds to the path implied in the popped symbol, no matter whether the popped symbol implies an unexplored path or a retreating path. An example which follows from this super rule is
   $\lambda(q_r, G_{33}, E_b) = \{O_b\}$.
   If the stack is empty, take the "stop ALV" symbol as output. An example which follows from this super rule is $\lambda(q_r, G_{33}, Z_0) = \{O_s\}$.

4. Proof of Correctness

In this section, several theorems are proven to show that the proposed transducer is able to explore the entire region of any closed navigation environment in finite cycles.

Theorem 1. The loops in the environment may be expanded as paths with terminals.

Proof :

The navigation transducer regards "explored grids" as "ending grids". It is trivial that a loop may be expanded as a path with an "explored grid" as a terminal. In the remainder of this paper, the term terminal is defined to be either an explored grid or an ending grid. □

From Theorem 1, the loops of the environment may be treated as paths with terminals. For any grid environment, an equivalent exploration tree may be derived to illustrate how the transducer explore the environment. The sequence to explore the environment is equivalent to the sequence to traverse the tree. The root of the exploration tree corresponds to the starting grid. The internal nodes are the crossing grids in the environments, and the leaf nodes are the terminals. The branch between two

nodes indicates the path between the corresponding grids. For example, the equivalent exploration tree for the grid environment in Figure 4 are shown in Figure 5.

The *level* of a grid G is defined to be the maximum number of crossing grids the ALV may encounter before the meeting a terminal along any exploring path expanded from G itself. In the equivalent exploration tree, the level of the corresponding grid G of a node is the maximum number of internal nodes in any exploring path expanded from itself. For example, in Figure 4 the level of Grid L is 0 because the only exploring path expanded from Grid L is L-F and there is no crossing grid in path L-F. The level of Grid C is 2. There are four exploring path expanded from Grid C. They are C-E-F, C-D, C-E-G-H-I-J-G, and C-E-G-J-I-H-G. In any of these paths, there are at most two crossing grids which will be encountered before a terminal is met.

Theorem 2. For any symbol in the stack, the grid in which the ALV is located when it is popped is the same as the one in which the ALV is located when it is pushed. And a retreating stack symbol, i.e., the symbol which corresponds the retreating path, is popped out only when the paths expanded from the grid where the symbol is popped are all explored.

Proof :

For the symbols which are pushed in a grid of level 0, it is trivial. When an unexplored crossing grid, say Grid X, of level 0 is encountered, the symbol which corresponds to the retreating path is first pushed into the stack. Then the transducer picks a path to go, and pushes the symbols corresponding to the remaining paths into the stack. Since Grid X is of level 0, no crossing grid will be encountered before the ALV arrives the terminal of the current path. The ALV then moves to the terminal of the picked path, switches into the "retreating mode", and then retreat to Grid X. A symbol corresponding to another exploring path is popped. Similarly, the ALV traverses along this path and then retreats to Grid X. Finally, the symbol corresponding to the retreating path will be popped after all paths expanded from Grid X have been explored.

Assume for symbols which are pushed in a grid of level smaller than or equal to k-1, the theorem holds. When an unexplored crossing grid, Grid Y, of level k is encountered, the symbol which corresponds to the retreating path is first pushed into the stack, then the transducer picks a path to go, and pushes the symbols corresponding to the remaining paths into the stack. When the ALV goes along the picked path, it may encounter either an unexplored crossing grid of level m, $m \le k-1$, or a terminal. If a terminal is encountered, no stack operations occur during traversing the picked path, and the ALV will retreat to Grid Y. If an unexplored crossing grid, say Grid Z, of level m is encountered, from the assumption, all symbols pushed in this grid will be popped in Grid Z after the ALV retreats from it. Thus, for either case, when ALV goes back to Grid Y, the condition of the stack is the same with the one before the ALV traverses the picked path. The symbols on the top of the stack are still the ones which are pushed in Grid Y. This situation also holds when the ALV traverse any of the other exploring paths. A symbol corresponding to another exploring path is then popped. Like traversing the first path, the ALV traverses along the new path and then retreats to the Grid Y. Finally, after the ALV traverses all exploring path expanded from Grid Y, the symbol corresponding to the retreating path is popped. The theorem hence also holds for symbols which are pushed in a grid of level k. By induction, it is concluded that the theorem is correct. □

Theorem 3. The ALV explores the entire environment region after the learning process is completed.

Proof:

We prove the theorem by contradiction. Assume that there exists an unexplored grid, say Grid U, after the learning process is completed. There must exist a path from some grid, Grid V, in the explored region to Grid U. When the ALV is in Grid V, the path V-U must either be picked to traverse along or be pushed into the stack. If V-U is picked to traverse along, Grid U will no longer be unexplored. This is a contradiction to the assumption. If the path V-U is pushed to the stack, the since the learning process will not be accomplished until the stack is empty, the symbol corresponding to the path V-U must be popped and the ALV will traverse along U-V sometime before the learning process is accomplished. This also causes a contradiction. Thus it is concluded that the ALV explores the entire environment region after the learning process is completed. □

Theorem 4. The ALV will stop in the starting grid after exploring the entire environment region.

Proof :

For the case that the starting grid is a "dead starting grid", the theorem is trivially true.

For the case the starting grid is a "one-way starting grids": The transducer pushes the first symbol when the first crossing grid is encountered. From Theorem 2, after exploring all paths expanded from this grid, all symbols in the stack are popped and the ALV keeps retreating toward the starting grid. From Super rule 2.2, the ALV stops when an ending grid is encountered, the transducer is in the "retreating state" and the stack is empty. The encountered ending grid is just the starting grid.

For the case that the starting grid is a "multi-way starting grids", the transducer first pushes all unexplored paths into the stack. After exploring all paths expanded from the starting grid, from Theorem 2, the ALV returns to the starting grid and the transducer will try to pop out the retreating symbol. Then, by Super rule 5.2, the empty stack causes the ALV stops in the starting grid. □

Theorem 5. The learning process will halt within finite learning cycles.

Proof :

The number of the exploring paths is finite, and each path has a terminal, i.e., there is no loop in the paths. Furthermore, the proposed algorithm ensures the ALV traverses each exploring path only once. Thus, it is con-

cluded that the ALV will explore the entire environment region within finite cycles. □

5. Experimental Results

An unsupervised learning simulation system for the grid environment has been developed to test the correctness of the proposed approach. The navigation grid environments were firstly generated manually with the aids of a graphical user interface program. Then, the simulator simulates the learning process in the grid environments. A simulation run is shown in Figure 6. The left-middle part of the figure shows the grid environment. The solid lines on the edges of the grids represent the walls. The little circles in the grid maps represent the starting grids. The grid with lighter borders represents one in which the ALV is located. The grids with x-marks represent "explored grids" and the grids with little spots represent those which the ALV has visited. The traces of the ALV are shown by animation in the simulation system and are not shown in Figure 6. The learned environment map is shown in the right-bottom corner of the figure. Several simulation results show that the ALV may explore the entire environment region, build an equivalent environment maps, and stop at the starting grid in finite learning cycles.

6. Conclusion and Future Works

In this study, we have developed a system with the capability of automatic exploration and unsupervised learning for ALV navigation in unexplored environments. The system is based on a pushdown transducer. With the capability of automatic exploration, the ALV may, as mentioned previously, work in the dangerous regions, serve as a safety guard, or perform a goal searching job, etc. With the unsupervised learning scheme, the ALV may collect the information and build a map of the navigation environment automatically without human involvement. The unsupervised learning scheme is especially valuable when the ALV works in the environment where human operator involvement is inappropriate. We have also proved several theorems showing the correctness of the proposed system.

The proposed learning system now works in the simulated grid model environment. Unsupervised learning for real-world environments based on the proposed approach is our current study topic. With certain modification, the principles developed in this study may be applied to learning of the real-world environment. How to collect the environment features of real-world environments and how to transfer the features into the input symbols to the navigation transducer are key problems of the future works.

References

[1] X. Lebègue and J. K. Aggarwal, "Extraction and Interpretation of Semantically Significant Line Segments for a Mobile Robot," *Proc. of 1992 IEEE Intl. Conf. on Robotics and Automation*, Nice, France, pp. 1778-1785, May 1992.

[2] X. Lebègue and J. K. Aggarwal, "Generation of Architectural CAD Models Using a Mobile Robot," *Proc. of 1994 IEEE Intl. Conf. on Robotics and Automation*, San Diego, California, U.S.A. , Vol. 1, pp. 711-717, May 1994.

[3] F. Nashashibi, M. Devy and P. Fillatreau, "Indoor Scene Terrain Modeling using Multiple Range Images for Autonomous Mobile Robots," *Proc. of 1992 IEEE Intl. Conf. on Robotics and Automation*, Nice, France, Vol. 1, pp. 40-46, May 1992.

[4] F. Nashashibi and M. Devy, "3D Incremental Modeling and Robot Localization in a Structured Environment using a Laser Range Finder," *Proc. of 1993 IEEE Intl. Conf. on Robotics and Automation*, Vol. 1, pp. 20-27, May 1993.

[5] H. Ishiguro, T. Maeda, T. Miyashita and S. Tsuji, "A Strategy for Acquiring an Environmental Model with Panoramic Sensing by a Mobile Robot," *Proc. of 1994 IEEE Intl. Conf. on Robotics and Automation*, San Diego, California, U.S.A. , Vol. 1, pp. 724-729, May 1994.

[6] A. Kurz, "Constructing Maps for Mobile Robot Navigation Based on Ultrasonic Range Data," *IEEE Trans. on System, Man, and Cybernetics - Part B: Cybernetics*, Vol. 26, No. 2, pp. 233-242, April 1996.

[7] T. Dean, D. Angluin, K. Basye, S. Engelson, L. Kaelbling, E. Kokkevis, and O. Maron, "Inferring Finite Automata with Stochastic Output Functions and an Application to Map Learning," *Machine Learning*, Vol. 18, pp. 81-108, 1995.

[8] F. M. Pan and W. H. Tsai, "Automatic environment learning and path generation for indoor autonomous land vehicle guidance using computer vision techniques", *Proc. of 1993 National Computer Symposium*, Chia-Yi, Taiwan, Republic of China, pp. 311-321, 1993.

[9] M. Dorigo, "Introduction to the Special Issue on Learning Autonomous Robots," *IEEE Trans. on System, Man, and Cybernetics - Part B: Cybernetics*, Vol. 26, No. 3, pp. 361-364, April 1996.

[10] J. Y. Donnart and J. A. Meyer, "Learning Reactive and Planning Rules in a Motivationally Autonomous Animat," *IEEE Trans. on System, Man, and Cybernetics - Part B: Cybernetics*, Vol. 26, No. 3, pp. 381-395, April 1996.

[11] B. Yamauchi and R. Beer, "Spatial Learning for Navigation in Dynamic Environments," *IEEE Trans. on System, Man, and Cybernetics - Part B: Cybernetics*, Vol. 26, No. 3, pp. 496-504, April 1996.

[12] L. Qiao, M. Sato, and H. Takeda, "Learning Algorithm of Environmental Recognition in Driving Vehicle," *IEEE Trans. on System, Man, and Cybernetics*, Vol. 25, No. 6, pp. 917-925, June 1995.

[13] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision, Volume 2*, Addison-Wesley, Reading, MA, U.S.A., 1993.

[14] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, MA, U.S.A., 1979.

Table 1. Output symbols of proposed navigation transducer.

Symbol	Corresponding ALV action
O_l	move to the left-side grid
O_r	move to the right-side grid
O_b	move to the bottom-side grid
O_t	move to the top-side grid
O_s	stop the ALV

Table 2. States of proposed navigation transducer.

Symbol	Meaning
q_e	exploring state
q_r	retreating state
q_{fin}	final state

Table 3. Stack symbols of proposed navigation transducer.

Symbol	Meaning
Z_o	stack start symbol
E_l	left-side, exploring mode
E_r	right-side, exploring mode
E_b	bottom-side, exploring mode
E_t	top-side, exploring mode
R_l	left-side, retreating mode
R_r	right-side, retreating mode
R_b	bottom-side, retreating mode
R_t	top-side, retreating mode

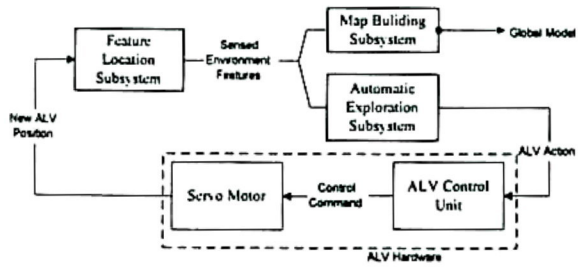

Figure 1. Interaction between three environment learning subsystems

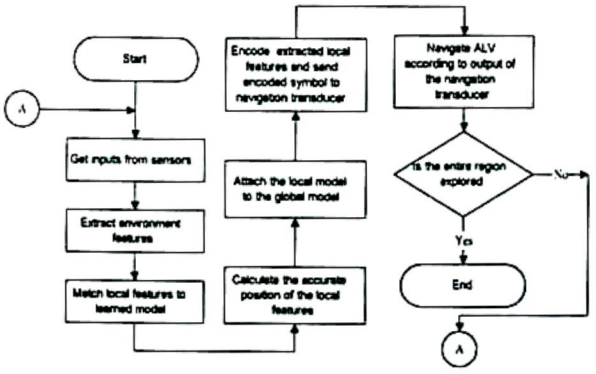

Figure 2. Flowchart of proposed learning system.

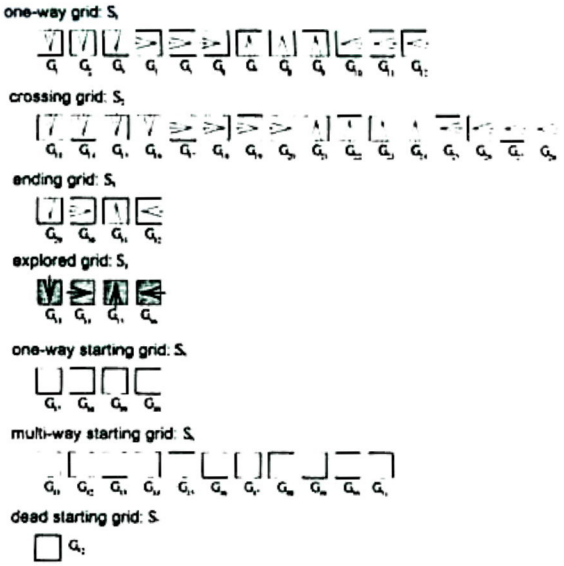

Figure 3. Inputs of navigation transducer.

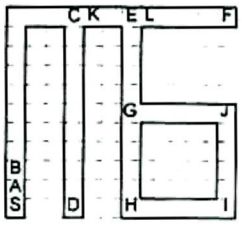

Figure 4. An example of grid maps.

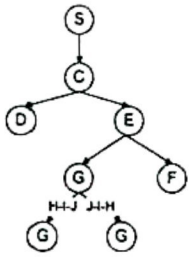

Figure 5. Equivalent exploring tree for the grid map in Figure 4.

Figure 6. An example of experimental results.