Empirical learning in mobile robot navigation

SHINJI KOTANI1, KAZUHIRO NISHIKAWA2 and HIDEO MORI1

1 Department of Electrical Engineering and Computer Science, Yamanashi University, 4-3-11 Takeda, Kofu, Yamanashi 400, Japan
2 KITO Co. Ltd., 2000 Tsukiji Arai, Syouwa Nakakoma Yamanashi 400, Japan

Received 25 March 1997; accepted 26 March 1998

Abstract—The purpose of this study is to improve the locomotion performance for autonomous mobile robots in outdoor environments. In this paper improvement of an environment model is called empirical locomotion performance learning. A system avoids wasting time of observations and actions by analyzing data from the last run. We propose a method of empirical learning. The method is expressed by rewriting the rules on the trajectory data. Brief route information for navigating a robot is represented with motion directions at intersections and metric distances between intersections. The behavior of our robot is based on a locomotion strategy ‘sign pattern-based stereotyped motion’. The behaviors are implemented on our mobile robot HARUNOBU-4 and tested at our university campus. Experimental results show a robustness of our proposed behaviors under dynamic environments with existing obstacles. Furthermore, they showed that our proposed rewriting rules improved the locomotion performance. In particular, searching time was shortened by 87% (from 453 to 61 s) and the travel distance was shortened by 10% (from 173.8 to 157.5 m).

1. INTRODUCTION

Autonomous navigation of mobile robots is a difficult problem because of the noise and variability associated with dynamic worlds.

When a robot which has metric information of an environment moves from a starting point to a destination point at the first time, the robot will waste time by searching for a road or moving forward or backward in order to find a target intersection.

Assume that the robot moves along the same course again. If the robot wastes the same time as the last run, we will think that the robot is not intelligent. Thus, travel time may be a good measure of the locomotion performance of robots. The robot must acquire information of that environment and must learn the information for the next action in order to improve the locomotion performance.
There are many outdoor mobile robots, e.g. VaMoRs [1] and Navlab [2]. Subsumption architecture [3] is a famous behavior-based approach. Arkin [4] proposed a behavior system adding knowledge about the environments. Our proposed behavior-based locomotion strategy is to utilize a simple feature.

We have also developed outdoor mobile robots [5]. Our locomotion strategy is based on ‘sign pattern-based stereotyped motion’ [5]. A ‘sign pattern’ (SP) [6] is a simple feature of an environment or object. SP is used to guide a fixed action pattern. In our university campus, for example, gutters, boundaries and line marks become SP.

Many researchers have performed research on ‘learning’ for many years. ‘Concept learning’ is divided into two fields: ‘learning from examples’ and ‘learning by observation’. The latter is also called ‘empirical learning’. One purpose of empirical learning is to improve the performance of a solution when we meet the same conditions as the conditions for which we solved the same problems previously.

The ARMS [7] system was developed in order to improve performance of a robot assembly. The system improves actions of a new robot assembly plans by analyzing the series of the last plans using ‘explanation-based learning’ (EBL) [8]. Our study develops this approach on autonomous mobile robots.

As to mobile robot learning, the main fields of learning are as follows: ‘learning numerical functions’, ‘learning about the world’, ‘learning how to co-ordinate behaviors’ and ‘learning new behaviors’. There are many studies on ‘learning about the world’. Martric [9] developed a behavior-based robot that builds an environment model by learning. He showed good experimental results in terms of path planning and navigation in indoor environments. Kaelbling [10] proposed a ‘learning in embedded systems’ approach considering the complexity of an environment. Connell and Mahadevan [11] discussed ‘learning in dynamic worlds’ using knowledge, neural networks, reinforcement learning and semantics. Pomerleau [12] developed the ALVINN (Autonomous Land Vehicle In a Neural Network) system. The system was implemented on Navlab II [2] and allowed it to drive in a variety of circumstances at speeds of up to 55 miles per hour. Zhu and Shi [13] developed an interactive learning system using a pattern-matching method and a genetic rule-modification schema. Goel [14] developed an experience-based approach based on case memory and implemented the method on the ROUTER system. However, there are few studies that build on the environment model for improving locomotion performance in real worlds.

In this paper we propose a method of learning to improve locomotion performance and the discuss behaviors of our robot. We implemented the method and the behaviors on our mobile robot HARUNOBU-4 [15] and experimented on our campus road. Experimental results showed the robustness of our proposed behaviors under dynamic environments with existing obstacles. Furthermore, they showed that our proposed rewriting rules improved the locomotion performance: 87% of searching time (from 453 to 61 s) became shortened and 10% of travel distance (from 173.8 to 157.5 m) became shortened.
2. ENVIRONMENT MODEL

The environment where our robot moves is an asphalt paved road on our university campus. There are gutters on both sides of the road. When the gutters are partially covered by fallen leaves and dust, their boundaries are not so clear. Moreover buildings behind gardens often make shadows on the road. It is difficult to distinguish the boundaries from the influence of shadow.

A photograph of our campus near a hall (see Figs 12 and 13) is shown in Fig. 1.

2.1. Metric map

2.1.1. Node. An intersection is the place where two or more roads meet and cross each other. The intersection is denoted by node $N_k$. In particular, we define an interpolation point $P_k$. The interpolation point is not an intersection but a point where a road changes its direction, more than $\varepsilon$, see Fig. 2. The nodes and the interpolation points have a $(x, y)$ value of a world co-ordinate system. They also have some links that are connected to the next node or the next interpolation point. For a mobile robot to navigate safely and effectively, the robot needs an environment model from the start to the goal. Different types of the model are presented in a hierarchy [16]. The advantages and disadvantages of each type are very closely linked to the purpose for which the model is being used and the cost of map-building. Our robots uses ‘full metric maps’. In our ‘full metric maps’, paths and objects locations are specified in a fixed co-ordinate system. Metric information can be recovered about any object in the map. However, the position and the location information are not precisely correct because the correct information entails a very high cost. The error distribution of our map is of the order of some meters.

Figure 1. A photograph of our campus environment.
2.1.2. Link. A link is represented by a list of road segments that are denoted by vector $M_j$.

$$M_j = (x_s, y_s, \lambda, d\lambda, w, \theta, next, stm),$$

where $(x_s, y_s)$ is the value of the starting point of this link; $\lambda$ is the distance from the starting node; $d\lambda$ is the length of this link; $w$ is the width of this link; $\theta$ is the direction of this link and $next, stm$ is the relation of the connected link and short-term memory.

An example of our environment model is shown in Fig. 2.

2.2. SP-based model

2.2.1. Short-term memory. During robot navigation, data of the time series of the robot’s trajectory and the obtained SP of each link are stored in short-term memory. An element $X_i$ of data of the time series shows the obtained data.

$$X_i = (x, y, \phi, \lambda, \theta, d, next),$$

where $(x, y)$ is the current position of the robot; $\phi$ is the current heading of the robot; $\lambda$ is the distance from the starting link; $\theta$ is the direction of the detected SP; $d$ is the distance of the detected SP from the robot and $next$ is the relation of the connected short-term memory.

2.2.2. SP-based model. ‘SP-based model’ is generated by our learning system to avoid waste observations and waste actions from the last run. The robot can improve locomotion performance at the next run using this SP-based model. The data structure is the same as that of the short-term memory.
3. AUTONOMOUS MOBILE ROBOT HARUNOBU-4

3.1. Hardware

Figure 3 shows HARUNOBU-4, the autonomous mobile robot that was developed during this research. HARUNOBU-4’s configurations are shown in Table 1.

The internal sensors are two encoders that are attached to driving wheels and an optical fiber gyroscope (Hitachi Cable Ltd., OFG3). External sensors are a video camera (Sony Handy Cam V30) and two IR sensors (Sunx Ltd., Protection Sensor). The video camera and one IR sensor are mounted on a camera platform. The platform can rotate at maximum speed of 65°/s and its resolution is about 0.007°.

<table>
<thead>
<tr>
<th>Items</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimensions</td>
<td>45 (W) 105 (L) 107 (H) (cm)</td>
</tr>
<tr>
<td>E-generator</td>
<td>100 V 3.0 A 50 Hz × 2</td>
</tr>
<tr>
<td>CPU (image processing)</td>
<td>68030, 25 MHz (OS-9 System)</td>
</tr>
<tr>
<td>CPU (motion control)</td>
<td>80286, 10 MHz (MS-DOS System)</td>
</tr>
<tr>
<td>Image memory</td>
<td>640 × 484 RGB 8-bit</td>
</tr>
<tr>
<td>Video camera</td>
<td>CCD auto-focus, auto-iris</td>
</tr>
</tbody>
</table>

Figure 3. Autonomous mobile robot HARUNOBU-4: 1. video camera; 2. IR sensor (rotatable); 3. IR sensor (fixed); 4. image processing computer; 5 and 6. right and left driving wheels; 7. TV monitor; 8. motion control computer; 9. terminal.
The robot calculates its current position and direction based on the data from the encoders and the optical fiber gyroscope by the conventional dead-reckoning method. The robot can detect obstacles and search for free space using two IR sensors. Communication between the image processing computer and motion control computer is via a RS-232C. HARUNOBU-4 can travel autonomously for 2 h.

3.2. Software

Figure 4 is a schematic representation of HARUNOBU-4's control architecture. In this robot there are main three controllers: behavior controller, route controller and obstacle detection controller.

3.2.1. Behavior controller. We have developed a behavior-based locomotion strategy 'sign pattern-based stereotyped motion' [6]. A 'sign pattern' is a simple feature of an environment or object which is used to guide a fixed action pattern. A 'stereotyped motion' is a fixed action pattern which fits the mechanics of a robot and produces a skillful action. In our robot, five stereotyped motions are defined. They are Searching, Moving Along, Dead Reckoning, Turning and Avoiding Obstacles. The transitions of these five stereotyped motions are shown in Fig. 4.

Searching. A video camera is angled to look at the ground from 2 to 5.5 m in front of the robot. In Searching, the robot pans the video camera 13 times. As a result, the robot can acquire information over a range of 240° around the robot. One example area of Searching is shown in Fig. 5. In every panning motion, our image processing system takes an image of an environment and detects elongated
edges by the $\gamma - \omega$ Hough transformation method [17] and selects edges that satisfies conditions in environment model. Its calling sequence is formalized as follows:

$$X_{ji} = \text{Searching}(M_j), \quad i = i + 1.$$  \hspace{1cm} (3)

where $X_{ji}$ are $i$th trajectories data of the robot and detected SP data of the $j$th link, and $M_j$ is the $j$th link.

**Moving_Along.** After an appropriate SP is searched for, the video camera is directed to the SP and the robot detects the SP. Moving_along is repeated until the robot fails in the SP detection. An example of our edge detection for the Moving_Along behavior is shown in Fig. 6. Moving_Along is formalized as follows:

$$\text{while (the SP is detected) } \{$$

$$X_{ji} = \text{Moving}_\text{Along}(X_{ji-1}, M_j); \quad i = i + 1;$$

$$\}$$

While the a SP is detected, its orientation $\theta_{ji}$ and distance $d_{ji}$ are calculated using the inverse perspective transformation method. Some methods are proposed for steering control method [18]. The method which we used is based on the ‘Look Ahead Model’ [18] and PID (Proportional–Integral–Derivative) control considering our control characteristics, performance of our CPU, required specifications.

![Figure 5. One searching area for the searching behavior.](image)
and dynamics of our robot. We obtained parameters of PID control applying the ‘Ziegler Nichols Formulas’ method [19] using step response.

**Dead_Reckoning.** The robot can move referring to a distance and a direction which are given by the route controller. An optical gyroscope is used in order to correct direction errors caused by robot slippage. Dead_Reckoning is formalized as follows:

\[
\text{Dist} = \text{distance}; \\
\text{while } (\text{Dist} > 0) \{ \\
\quad X_{ji} = \text{Dead}\_\text{Reckoning}(X_{ji-1}, \text{direction}, \text{Dist}); \\
\quad \text{Dist} = \text{Dist} - \text{traveled\_distance}; \quad i = i + 1; \\
\}
\]

**Turning.** The robot can do a spin turn of $\Delta \phi$ is given by the route controller. Turning is formalized as follows:

\[
X_{ji} = \text{Turning}(X_{ji-1}, \Delta \phi), \quad i = i + 1.
\]

**Avoiding Obstacles.** In Avoiding Obstacles, an IR sensor mounted on a rotatable video camera platform searches for areas free from obstacles. The robot
moves to the free area. After that the robot activates Searching. The hardware configuration is shown in Fig. 7 and an example of Avoiding_Obstacles behavior is shown in Fig. 8. Avoiding_Obstacles is formalized as follows:

\[ X_{ji} = \text{Avoiding_Obstacles}(X_{ji-1}) \quad i = i + 1. \]  

(7)

3.2.2. Obstacle detection controller. This module has the highest priority and is interrupt-driven. If the IR sensor which is fixed on the front of the robot detects something, the current movement is stopped immediately. If the sensor again detects the object after a few seconds, the robot assumes that the object is not a pedestrian but a stationary object and the robot activates Avoiding_Obstacles.
3.2.3. Route controller. This module ensures the robot does not deviate from the route using the environment model and short-term memory. As our environment model has some error distribution, the robot may have extra locomotive motion. If this manager judges that the robot has deviated from the route, this module activates Turning.

4. EMPIRICAL LEARNING

One purpose of empirical learning is to improve the performance of a solution when we meet the same conditions as the conditions for which we solved the same problems previously. When applying this definition to autonomous mobile robots, the purpose is to improve locomotion performance by avoiding wasted observations and actions based on the analysis of the data of the last run. Short-term memory is revised in order to improve the locomotion performance. This is called locomotion performance learning in this paper. After the run, our navigation system analyzes \( \{X_{ji}\} \) in the short-term memory, and avoids the waste observations and waste actions of the last run. After that the system generates a SP-based model by the following algorithms. Examples of an improvement of wasted observations are shown in Fig. 9. An example of an improvement of wasted action is shown in Fig. 10.

We describe the learning algorithm by rewriting rules. A function \( \text{side}(X) \) is defined as follows:

\[
\begin{align*}
\text{side}(X) &= \begin{cases} 
L & \text{when } d > 0 \\
R & \text{when } d < 0 \\
Z & \text{when SP is not detected}
\end{cases}
\end{align*}
\]

where \( d \) is the distance of the detected SP from the robot.

4.1. Rewriting rules

Rule 1

\[
\text{if } (\text{side}(X_i) == \text{side}(X_{i+1})) \{ \\
\text{ rewrite } X_i X_{i+1} \text{ by } X_i \\
\}
\]

Applying this rule in repetition, the short-term memory that has the same SP or no SP is substituted by one element.

Rule 2

\[
\text{if } ((\text{side}(X_i) == \text{side}(X_{i+2})) \& (\text{side}(X_i)! = Z)) \{
\text{if } ((\text{side}(X_{i+1}) == Z) \& (\lambda(X_{i+2}) - \lambda(X_{i+1}) < \Delta \lambda)) \{
\]

rewrite $X_iX_{i-1}X_{i+2}$ by $X_i$,

Applying this rule, a short no SP part of trajectory less than $\Delta \lambda$ is removed. This improvement is shown in Fig. 9b. After applying Rule 1 and Rule 2, the following two rules are applied. These two rules are used in order to avoid wasted actions.

**Rule 3**

if $(\text{abs}\{(\theta(X_{i+1}) - \theta(X_i)) - 180 < \Delta \theta\})$

rewrite $X_iX_{i+1}X_{i+2}$ by $X_iX_{i+2}$

Applying this rule in repetition, the U turn part of the trajectory is avoided as shown in Fig. 10.

![Figure 9](image9.png)

**Figure 9.** Examples of an improvement of wasted observations.

![Figure 10](image10.png)

**Figure 10.** An example of an improvement of wasted actions.
Rule 4

\[
\text{if } \left( (\text{side}(X_i) = Z) \& \& (\lambda(X_{i-1}) - \lambda(X_i) < \Delta \lambda) \right) \{
\text{if } \left( (\text{side}(X_{i-1}) = \text{side}(X_{i+1})) \& \& (\text{side}(X_{i-1}) = Z) \right) \{
\text{if } \left( \text{side}(X_{i-2}) = \text{side}(X_i) \right) \{
\text{rewrite } X_{i-2}X_{i-1}X_iX_{i+1}X_{i+2} \text{ by } X_{i-2}X'_{i-1}X_{i+2} \quad (12)
\}
\}
\}
\]

Applying this rule, a wide intersection and a short crank part of the trajectory are avoided as shown in Fig. 11.

5. EXPERIMENTAL RESULTS

Figures 12 and 13 show the trajectories of the robot before learning and after learning in the campus environment. The starting point is ‘Factory’ and destination is ‘West gate’. The speed of the robot was up to 20 cm/s. A short-dashed line shows the trajectory of the robot. A solid line shows SP candidates detected by image processing. A closed circle shows the point of Searching.

In Fig. 12, the robot started at ‘Factory’ and activated Searching. Searching found a SP of the right side. The SP was an edge of a gutter. Then the robot moved a little while activated Moving_Along. After the robot lost the SP, it activated Searching again and then activated Moving_Along. At a corner, near
Node2, it lost the SP because the gutter came to an end. The robot activated Searching and detected a SP. However, it did not activated Moving_Along because the SP was too short to activate Moving_Along. So the robot repeated Searching and Dead_Reckoning four times. At the end of the corner the robot found an appropriate SP that fitted the direction of the link between Node2 and Node3. So the robot activated Moving_Along again. At point A, the robot lost the SP because of the broken gutter. At the next corner, near Node3, the robot could not detect the correct SP because the SP was out of camera range. So it followed a SP of the right side of the road and it moved forward too far. As the route controller managed the robot’s position and the length of the link between Node2 and Node3, the route controller activated Turning. After end of the Turning the robot activated Searching and Moving_Along. Therefore the robot reached its destination. The travel time is shown in Fig. 14. The total travel time includes Moving_Along, Dead_Reckoning and Searching.

After the first run, the learning system analyzed $X_{ji}$ in the short-term memory. Applying the rewriting rules, the navigation system generated a SP-based model. The result using this revised SP-based model is shown in Fig. 13. Judging from the robot’s behavior, the robot avoided waste observations and actions. In particular.
Figure 13. Trajectory and behaviors after learning.

Figure 14. Travel time in the environment before and after learning.
at the end of Link2 and Link3, as the robot knew the position and direction of the target SP, it moved to the target position by Dead_Reckoning.

We would like to consider Turning_Back. At the first run the robot passed through Node3 and then turned back. The reason that the robot turned back was the position information obtained by the dead-reckoning module of the robot. There are two reasons why the robot passed through the destination. One is that the error distribution of our map is of the order of some meters. We think very precise information is very good for robots, but it entails a very high cost in map-building. The other is that the robot has positioning error in the estimated position by dead-reckoning. Even if both types of error were taken into consideration, the robot judged that it already had passed through Node3 where it should turn left. After that the robot activated Turning_Back. As to ‘after learning’, the robot must take the latter’s error into consideration. However, the robot activated appropriate behavior (see Fig. 13), because the robot already has the metric map which is accurate.

The total travel time was shortened from 1534 to 951 s. After the learning, 87% of Searching time, 11% of Moving_Along time and 31% of Dead_Reckoning time became shortened.

As to the distance, the traveling distance was shortened by 10% (from 173.8 to 157.5 m).

6. DISCUSSION

Locomotion performance learning improves the environment model by analyzing wasted observations and actions of the last run.

The goodness of a SP depends on its type, sunlight and weather condition. For example, there are gutters which are separated from a little high garden area on both sides of the road in our campus. Assume that the road lies in north to south direction, when the sun is in the east, the left side gutter makes a shadow on the surface and it becomes a good SP, but the right side gutter does not. When the sun is in the west, the situation is reversed. Moreover, in autumn fallen leaves often cover the gutter, so the vision system often fails in its direction.

To our regret, the robustness depends on the environment. We must prepare for many rewriting rules and it will be important to use them properly corresponding to the environment. As a future subject, we would like to implement the automatic selection of the rewriting rules system on the robot.

REFERENCES


**ABOUT THE AUTHORS**

**Shinji Kotani** was born in Tokyo, Japan, on July 29, 1962. He received the BE degree in 1986 and ME degree in 1988 from Yamanashi University. He joined the Computer Center, Yokogawa Electric Corp., in 1988. In 1992 he joined Yamanashi University as a research associate. From 1997 to 1998 he was a visiting researcher at the University of Oxford, UK. His research interests include vision-based robotics and outdoor mobile robots. He is a member of the Robotics Society of Japan, the IPSJ and the IEICE.
Kazuhiro Nishikawa was born in Yamanashi, Japan, on April 7, 1967. He received the BE, ME and PhD degrees in 1990, 1992 and 1995, respectively, all from Yamanashi University. He joined the R&D Center, Kito Corp., in 1995. His research interests include vision-based robotics and outdoor mobile robots. He is a member of the Robotics Society of Japan and the IEICE.

Hideo Mori received BE and PhD degrees in Computer Science from Tokyo Institute of Technology in 1962 and 1990. From 1961 to 1965, he worked at Fujitsu Ltd. Since 1965 he has been teaching at Yamanashi University, from 1965 as a lecturer of Electrical Engineering, from 1972 as an Associate Professor of Computer Science, and since 1991 he has been Professor of Electrical Engineering and Computer Science. His main field of interest is behavior-based robotics and its application to the outdoor HARUNOBU robots. He has been applying it to a lane-mark drawing robot for road maintenance and a robotic travel aid for the visually impaired.