SIMPLE AUTOMATIC PATH LEARNING FOR AUTONOMOUS VEHICLE NAVIGATION BY ULTRASONIC SENSING AND COMPUTER VISION TECHNIQUES*

Shung-Yung Tsai (蔡雙圓)[1] and Wen-Hsiang Tsai (蔡文祥)[1, 2]
[1] Institute of Multimedia Engineering, College of Computer Science
National Chiao Tung University, Hsinchu, Taiwan 300
[2] Department of Information Communication
Asia University, Taichung, Taiwan 413
E-mail: whtsai@cis.nctu.edu.tw

ABSTRACT

A simple automatic path learning method for an intelligent autonomous vision-based vehicle system by person following and along-path object image matching is proposed. The vehicle can follow a person through a path in an indoor environment and navigates back to the start point by itself by the data learned during the following process. A technique of matching flat-surfaced object images using SIFT features is adopted for the vehicle to localize itself according to its learned path during the navigation process. In addition, a technique for dynamically refining the traversed path points which are learned during person following is proposed. Finally, a technique of odometer calibration, which uses a calibration model to reduce incremental mechanical errors the vehicle suffers, is proposed to increase path traversing accuracy. Good experimental results show the feasibility of the proposed techniques.

1. INTRODUCTION

In recent years, vision-based robots with ultrasonic sensors have played helpful roles in human life. However, the conventional way of path learning is inconvenient because the learning process is mostly *manual* and time-consumptive. It is desired to design an autonomous vehicle that can follow a *person automatically* to everywhere, and learn the path that we hope it to navigate as well as the along-path objects that we want it to monitor, just like a mother teaching her child to learn a route in daily life.

When the vehicle is navigating, it may seem not smart if it walks totally through the same the trajectory as it learned, because this trajectory could be rough and curved, and there is no need to step precisely on every spot of it. We hope the vehicle could reconstruct the trajectory points into a smooth path, but not too smooth to miss some important points we want it to patrol. On the other hand, it is inappropriate for the vehicle to navigate through a path by using only the learned odometer information because incremental mechanical errors might result in imprecise odometer data and so cause inaccurate path following or even wall collision, if the data are followed for later vehicle guidance. Hence, odometer calibration must be carried out to eliminate the errors.

To achieve the goal of person following in indoor environments, some methods have been proposed. Wang and Tsai [1] proposed a method which uses the color values of the clothes which a person wears to follow the person. Ku and Tsai [2] proposed a sequential pattern recognition method to decide the location of a person with respect to a vehicle and to detect a rectangular shape attached on the person's back to achieve smooth person following. A method of following a person who turns fast at a corner in a narrow path was proposed by Chen and Tsai [3]. Kwolek [4] proposed a method that localizes a mobile robot by laser readings.

For object monitoring, the vehicle has to learn the features of each concerned object and conduct feature matching to determine whether the object is the same as the previously learned one. Lowe [5] used a scale-invariant detector to find the extrema in the difference-of-Gaussian scale-space. He then created a scale-invariant feature transform (SIFT) descriptor to match key points using a Euclidean distance metric in an efficient best-bin first algorithm which can identify the nearest neighbors of points in high dimensional spaces. For the topic of robot navigation, Chen and Tsai [6] proposed a method with a simplified SIFT for monitoring objects and used the position and feature information of objects to localize the vehicle for vehicle guidance.

The goal of this study is to design an intelligent vehicle system with a simple learning scheme for indoor environment patrolling by the use of a vehicle equipped with a video camera and ultrasonic sensors. It is desired to design the system to be capable of performing three major tasks, described as follows.

* This work was supported by the Ministry of Economic Affairs under Project No. MOEA 96-EC-17-A-02-S1-032 in Technology Development Program for Academia.

(1) The vehicle will follow a person who walks through a path, and then navigate by itself back to the start point by the data learned during person following.
(2) When the vehicle follows the person, if he/she stands in front of a flat-surfaced object (like a picture on the wall) and faces to it for a while, the vehicle will go to the position where the person stands and pan the camera to the same direction as the person.
(3) The vehicle will navigate automatically with path correction to monitor the along-path objects viewed by the person while going back and during future navigation sessions.

In the following sections, we describe various techniques we propose to carry out these tasks.

2. SYSTEM CONFIGURATION

In the proposed system, we use Pioneer 3, a vehicle made by ActiveMedia Robotics Technologies Inc., as a test bed. A camera is equipped on the vehicle, which is AXIS 213 made by the AXIS company. It is IP-based with panning, tilting, and zooming (PTZ) capabilities and has a build-in web server through which we can adjust wirelessly some camera parameters with a graphical user interface, such as image resolution, image format, and so on.

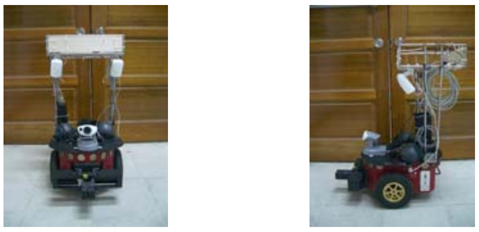
Figure 1. The vehicle Pioneer3 used in this study.

Figure 2. The AXIS 213 PTZ camera equipped on the vehicle.

3. LEARNING PROCEDURES

3.1. Learning 2D Objects Automatically

It is desired to design the vehicle system in such a way that when the vehicle follows a person, if he/she stands in front of a flat-surfaced object and faces to it for a while, the vehicle will go to the position where the person stands and pan the camera to the same direction as the person. To achieve this goal, at each node in the navigation process, an image I is taken from which the interesting region formed by a flat surface of a 2D object at the node and a horizontal line L parallel to the floor are found. Furthermore, the simplified SIFT algorithm proposed in [6] is applied to obtain a feature set of the interesting region and a location estimation algorithm also proposed in [6] is performed to find the relative position and the relative angle of the vehicle with respect to the start point of the horizontal line L in the 3D global coordinate system. Finally, we save the information into the data of the node. The detail process is described in the following algorithm.

Algorithm 1: *Learning of a monitored object at a path node automatically.*

```
Input: Consecutive images taken in navigation.
Output: Monitored object information data.
Steps:
Step 1.  Follow the user and take an image of the frontal scene.
Step 2.  Detect the user's facing direction by the relative positions of the face-skin and hair colors.
Step 3.  Use the result of the last step to decide if the user stops and turns to the lateral side for a while; if so, continue; else, go back to Step 1.
Step 4.  Go to the position of the user and pan the camera to the detected user's facing direction.
Step 5.  Take an image $I$ of the frontal scene which presumably includes an object to be monitored.
Step 6.  Find the interesting region of the 2D object and a horizontal line $L$ on the object surface, which is parallel to the floor in the image $I$.
Step 7.  Calculate the feature set of the 2D object and the posture of the vehicle with respect to the horizontal line $L$ (for details, see Section 4), and save the computed data.
```

3.2. Finding Regions of 2D Objects

To find the interesting region mentioned previously, we deal with each taken image by using image thresholding and region growing techniques. The details are described in the following algorithm.

Algorithm 2: *Finding interesting regions in images.*

```
Input: An image $I$ with a monitored object, and the gray version $I_g$ of $I$.
Output: Four corner points $P_{TopLeft}(i_l, j_t)$, $P_{TopRight}(i_r, j_t)$, $P_{BottomLeft}(i_l, j_b)$, and $P_{BottonRight}(i_r, j_b)$ of an interesting region, and a horizontal line $L$ in $I$.
Steps:
Step 1.  Calculate the threshold value $T_{threshoding}$ of the gray value which can differentiate background and foreground of $I$.
Step 2.  Reset the gray value $g_{pi}$ of every pixel $p_i$ in the image $I_g$ in the following way:
         If $g_{pi} > T_{thresholding}$, set $g_{pi}$ as a foreground point, else, a background point.
Step 3.  Conduct region growing from the center of $I_g$ and get the top and bottom vertical coordinate values $G_T$ and $G_B$, as well as the leftmost and rightmost horizontal coordinate values $G_L$ and $G_R$ of the resulting region.
Step 4.  Set the coordinate values $i_l$, $i_r$, $j_t$ and $j_b$ for corner points $P_{TopLeft}(i_l, j_t)$, $P_{TopRight}(i_r, j_t)$, $P_{BottomLeft}(i_l, j_b)$, and $P_{BottonRight}(i_r, j_b)$ of the interesting region as
```

$$i_l = G_L\ ;\ i_r = G_R\ ;\ j_t = G_T\ ;\ j_b = G_B\ .$$

3.3. Finding the Horizontal Line Automatically

To find the horizontal line which we use to compute the posture of the vehicle, we have to find the set of edge points of all possible lines by edge detection in the image first. For this, we use the Sobel operator with a threshold value T_{sobel}. We apply the Sobel operator on a pixel p_i, and let r_{pi} be the resulting value. If $r_{pi} > T_{sobel}$, then we mark the pixel p_i as an edge point. After checking every pixel p_i in the interesting region, we can get a set of edge points, EP.

We then find the desired horizontal line by using the *Hough transform*. It is a method of *line detection* by the use of a voting technique and the relationship between the parameter space and the normal distance-normal angle space. We use this method to find the horizontal line which is the longest one in the top quarter of the interesting region. At first, we divide the range of angles, [0, π], in to n part, Θ_0, Θ_1, …, Θ_n, and calculate the length r_{region} of a diagonal of the interesting region. We prepare next an accumulation array A, whose size is $n\times r_{region}$, and set the initial value of every cell to zero. Also, we substitute a point (x, y) in the edge point set EP and Θ_i into the equation below to get a value r and put a vote into the cell (r, Θ_i):

$$r = x\cos\theta_i + y\sin\theta_i\ .$$

We repeat this step of voting in the cells for every edge point and every Θ_i. After voting, we find the cell (γ, θ) with the maximum number of votes to obtain a line which has the intercept γ and the angle θ in the normal distance-normal angle space. This line is what we want.

There is a restriction in our method, that is, the heights of monitored objects must be of fixed values known in advance. By the method described above, we can find the interesting region of a concerned 2D object in the image I and an associated horizontal line L, which is parallel to the floor plane in the 3D global coordinate system in image I.

4. NAVIGATION BY VEHICLE LOCALIZATION USING FLAT-SURFACED OBJECTS

We use the posture of a flat-surfaced object and an associated horizontal line (obtained by Step 7 in Algorithm 1 above) for vehicle localization. For this, we set up a coordinate system, called *reference coordinate system* (*RCS*) on the horizontal line (called *calibration line* henceforth), with one of the two line ends as the system's origin.

In more details, in a navigation session (including going back to the start point in the learning stage), the vehicle moves from one node to another according to a planned path in the navigation environment. Vehicle localization at a node along the navigation path is based on the use of a monitored object B located there. It is implemented by applying 2D matching of some features of two images of B, one taken in the navigation stage, denoted as I_{N}, and the other taken in the learning stage, denoted as I_{L}. The features come from the SIFT of I_{N} and I_{L}, called SIFT features in the sequel, and the matching is carried out also by the Hough transform. More details are described in the following.

1. Apply the SIFT to I_{N} to obtain an SIFT feature set F_2, and take out the corresponding set F_1 of I_{L}, which was obtained in the learning stage.
2. Take every pair of similar features, one from F_1 and the other from F_2, to define a group of four parameters of an affine transformation from I_{L} to I_{N}, where the feature similarity is computed according to [7].
3. Put all found parameter groups into a Hough space and detect the peak in the space.
5. Find out the affine transform T_1 corresponding to the peak, as the relative transformation from I_{L} to I_{N}.
6. Use T_1 to transform the calibration line ℓ_{L}, which was detected in I_{L} in the learning stage, into the image space I_{N}, resulting in a new image line in I_{N}, denoted as ℓ_{N}.
7. Transform ℓ_{L} and ℓ_{N} into the RCS unambiguously according to an analytic 3D transformation T_2 [6] from the image space to the RCS to obtain two sets S_1 and S_2 of vehicle poses with respect to the RCS, one for the learning stage and the other for the navigation stage, respectively.
8. Use S_1 and S_2 to derive the translation V_{t} and orientation θ_{t} of the vehicle's current location with respect to its location planned in the learning stage.
9. Use (V_{t}, θ_{t}) to derive a sequence of vehicle commands to guide the vehicle to the correct path a continue the navigation session.

5. PATH PLANNING BY MINIMIZING MEAN SQUARE ERRORS USING ULTRASONIC SIGNALS

After learning the path by person following, the system will *refine* the collected path data before the vehicle navigates back to the start point and starts its regular navigation sessions. We call this refinement operation *path planning* in this paper. The path data are composed of many *path points* described by odometer data. We want to make the path to be smooth, but do not lose the original trend of the path points. This is accomplished by choosing from the path points some *critical* ones, and using a line segment to approximate those path points between every two critical points. The problem is how to choose the critical path points, and the technique adopted in this study is *binary cut by the LSE criterion*, as described next.

5.1. Path Planning by LSE Binary Cut

As a start, the system considers a given set of n path points as a smooth line segment and approximates the points by a line equation. It then computes the square error SE resulting from this line approximation of the n points. If the value of SE is larger than a threshold value T_{lse}, then the line approximation is considered not smooth enough and is cut into two portions at a certain intermediate point selected according to an LSE criterion. This process is repeated until no more unsmooth line approximation appears. The detailed process for computing the SE value for a set of path points is described as an algorithm in the following.

Algorithm 3: *Calculating the SE value.*
Input: m path points.
Output: The value of the corresponding square error SE.
Steps:

Step 1. Use the line fitting technique to find a linear equation of a line L with two variables, x and y as follows to approximate the input m path points:

$$L : Ax + By + C = 0,$$

where A, B and C are constants.

Step 2. Calculate the distance values d_i from every path point $P_i(x_i, y_i)$ to the line L by

$$d_i = \frac{|Ax_i + By_i + C|}{\sqrt{A^2 + B^2}}.$$

Step 3. Calculate the value of the induced square error SE of the line L by

$$SE = \frac{1}{m}\sum_{i=1}^{m}(d_i)^2.$$

If the SE value computed by the above algorithm for a path segment (may be the entire path) is larger than T_{lse}, then the path segment is considered to be unsmooth. The algorithm proposed for cutting an unsmooth path segment is as follows.

Algorithm 4: *Cutting an unsmooth path segment.*
Input: a path segment consisting of m points P_1, P_2, …, P_m, which, when approximated by a line segment, is unsmooth.
Output: a path point among the m input points, which cuts the path segment into two shorter ones with the *least square error*.
Steps:

Step 1. For each path point P_j of the m input ones except the first and the last, perform the following steps:
    consider P_j as a cut point, and find two line segments L_{1j} and L_{jm} by line fitting again;
    apply Algorithm 1 to calculate the values of the two square errors SE_{1j} and SE_{jm} respectively for L_{1j} and L_{jm};
    compute the square error SE_j of cutting on the point P_j as $SE_j = SE_{1j} + SE_{jm}$.

Step 2. Find the minimum value SE_k from the $m - 2$ SE_j values computed in the last step, with SE_k meaning that if we cut the path on point P_k, we can get the least square error value.

The system executes the above process recursively, until the original path segment is cut into many shorter segments which are *all* smooth. Finally, we save the start point of each of these segments as a node of the final path for navigation.

5.2. Dynamic Path Decomposition

The threshold value T_{lse} of the LSE binary cut described above is used to tolerate the value of the line approximation error. The larger the threshold, the cruder the path decomposed. The choice of the threshold should be adapted to the environment of the patrolled path. Moreover, the environment could be different at different parts of a path, and the path segment at each different part should be decomposed by a different degree of precision. We propose a technique here which chooses the threshold value dynamically.

In more detail, except the coordinates of path points, the width at each path point (such as the distance between two walls in a corridor) is also part of the information of path data which are learned in the learning process by using the ultrasonic sensors equipped on the vehicle. Specifically, in the learning process the system learned the left and right distances ($d_{l,i}$, $d_{r,i}$) of every path point P_i with respect to the obstacle or the wall around the point. We simplify such distance data to get a single distance value D_i of every path point by

$$D_i = \min\left(d_{l,i}, d_{r,i}\right).$$

Considering the correlation of the environments around the connected segments of the path, our system calculates the threshold value for one path segment by using the distance values of not only the path points in this segment but also five points before and five points behind this segment. Additionally, the composite square error for this segment is taken to be the *average* of the square errors computed in Step 3 of Algorithm 3. And the desired threshold value is derived to be as follows for a segment with n points:

$$Threshold = \frac{1}{k(5n+10)}\left(\sum_{i=-5}^{-1} D_i^{\,2} + \sum_{i=1}^{n} D_i^{\,2} + \sum_{i=n+1}^{n+5} D_i^2\right)$$

where k is a constant.

We decompose the path dynamically by recalculating the above threshold value in every cycle of a navigation session. In this way, the vehicle can be made to navigate more smoothly and safely with no collision with obstacles or walls in the navigation environment.

6. ODOMETER CALIBRATION

After planning the path, the last process of the system is vehicle patrolling from the end of the path to the start position of the path. The vehicle's moving direction is an important factor for guiding the vehicle to navigate. The direction information is provided by the odometer of the vehicle. However, the vehicle cannot navigate by using only the odometer information collected in learning because incremental mechanical errors might result in imprecise odometer data. Hence, in order to keep the navigation in the path precisely, odometer calibration must be carried out to eliminate the errors. In this study, we propose a technique to collect the data of *deviations* from correct paths, and analyze the data to build an odometer calibration model.

6.1. Odometer Calibration Model

Before building an odometer calibration model, we prepare some equipment for our experiment, including a sticky tape, a measuring tape, an autonomous land vehicle, and a computer. First, we fix the initial position O of the vehicle and mark the position by pasting a sticky tape on the ground. Second, we fix the initial direction of the vehicle and paste a straight line L along the direction on the ground by using a sticky tape. Third, we send commands to drive the vehicle forward on the straight line, and then commands to stop the vehicle. Fourth, we mark the terminal position T of the vehicle by pasting a piece of sticky tape on the ground. Fifth, we find the node P on the straight line L which is the vertical projection of the terminal position T. Sixth, we measure the distance D_1 between O and T which is the move distance of the vehicle, and the distance D_2 between T and P which is the *deviation* produced by mechanical errors. Seventh, we compute the angle Θ of the inverse sine value of D_2/D_1 which is the angle of the deviation. We repeat the steps at least twenty times and let the distance the vehicle moves be different every time. An illustration of the experiment is shown in Figure 3.

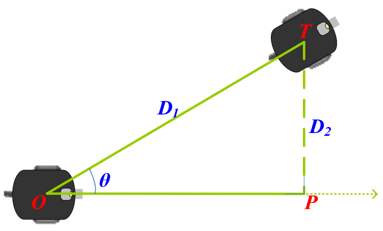

Figure 3. An illustration of the experiment.

Algorithm 5: *Building an odometer calibration model.*
Input: None.
Output: An odometer calibration model.
Steps:

Step 1. Fix the initial position O and initial direction line L of the vehicle.
Step 2. Send commands to let the vehicle move forward and then stop.
Step 3. Mark the terminal position T of the vehicle.
Step 4. Find the vertical projection node P of the terminal position T of the vehicle on the straight line L.
Step 5. Measure the distance D_1 between O and T and the distance D_2 between T and P.
Step 6. Compute the angle Θ of inverse sine value of D_2/D_1.
Step 7. Repeat Step 1 to Step 6 at least twenty times and let the distance the vehicle moves every time be different.

After measuring the values of the angles of deviations, we found out that the distribution of data has a trend which may be roughly described as a curve of the second order with respect to the vehicle move distance value. Therefore, we use an LSE fitting method to fit the data with a curve described as follows:

$$f(x) = 0.00000476 \times x^2 + 0.00592048\,x + 4.16437951\,. \qquad (6.1)$$

An illustration of the curve is shown in Figure 4.

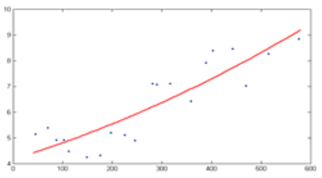

Figure 4. The results of line fitting of the angles of deviations.

6.2. Navigation by Odometer Calibration

We guide the vehicle most of the time while navigation by the command: “move to front.” But the vehicle we use for experiments always has a leftward deviation while moving forward. To deal with this problem, we use the odometer calibration model to balance the deviation. First, we have to know the distance D we want the vehicle to move forward. Second, we substitute the value D into Eq. (6.1) to get the angle of deviation Θ. That means if we command the vehicle to move to D centimeters ahead of the original position, then the vehicle should be instructed to deviate rightward for the angle of Θ. Therefore, we issue a command of right turn of angle Θ before commanding the vehicle to move forward. In this way, we can balance the deviation and so the mechanical error.

7. EXPERIMENTAL RESULTS

We show some experimental results of the proposed vehicle system with human following and patrolling capabilities. At first, the vehicle starts the person following mode to follow a person. If the person turns to the right or left for a while, the vehicle will go to the position where the person stands and turn to the same direction to see and “remember” the view seen by the person. Then, the system will analyze the image of the view and find the interesting region and the horizontal line of the 2D object, as shown in Figure 5 and Figure 6.

A record map of a navigation session is shown in Figure 7. After path planning, the vehicle will go back to the start point in the learning process and monitor the learned objects on the way, as shown in Figure 8.

8. CONCLUSIONS

An autonomous vehicle system with person following and object patrolling capabilities with a simple path learning scheme has been proposed. When the vehicle follows a person, it will remember the traversed path and the environment of the path. When the person stands in front of a 2D object and faces to it for a while, the vehicle will go to the position where the person stands and learn the information of the object automatically. Therefore, we can teach the vehicle to follow a path and monitor along-path objects easily, and there is no need to key in instructions into the system as done conventionally. We have also proposed a technique of path planning to refine the traversed path points into smooth path segments. In addition, a method of building an odometer calibration model has also been proposed, by which incremental mechanical errors suffered by the vehicle while patrolling can be reduced. Good experimental results indicate the feasibility of the proposed techniques. Future researches may be directed to conducting human following by different features, such as texture and shape; detecting the heights of monitored objects by using two pictures captured at different positions; supplying the position of the vehicle by using a top-view omni-directional camera, etc.

ACKNOWLEDGEMENT

The authors want to thank K. C. Chen for his help in supporting the algorithm described in Section 4, which became of part of the proposed system.

REFERENCES

[1] Y. T. Wang and W. H. Tsai, "Indoor security patrolling with intruding person detection and following capabilities by vision-based autonomous vehicle navigation," *Proceedings of 2006 Internat'l Computer Symposium (ICS 2006), Internat'l Workshop on Image Processing, Computer Graphics, and Multimedia Technologies,* Taipei, Taiwan, Dec. 2006.

[2] C. H. Ku and W. H. Tsai, "Smooth autonomous land vehicle navigation in indoor environments by person following using sequential pattern recognition," *Journal of Robotic Systems*, Vol. 16, No. 5, pp. 249-262, 1999.

[3] K. T. Chen and W. H. Tsai, "A study on autonomous vehicle guidance for person following by 2D human image analysis and 3D computer vision techniques," *Proceedings of 2007 Conf. on Computer Vision, Graphics & Image Process.,* Miaoli, Taiwan, 2007.

[4] B. Kwolek, "Face Tracking System Based on Color, Stereovision and Elliptical Shape Features," *Proc. of IEEE conf. on Advanced Video & Signal Based Surveillance,* Miami, Florida, USA, pp. 265-270, 2004.

[5] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Internati'l Journal of Computer Vision*, Vol. 60, No. 2, pp. 91-110, 2004.

[6] K. C. Chen and W. H. Tsai, "A study on autonomous vehicle navigation by 2D object image matching and 3D computer vision analysis for indoor security patrolling applications," *Proc. of 2007 Conf. on Computer Vision, Graphics & Image Process.*, Miaoli, Taiwan, 2007.

[7] K. Mikolajczyk, and C. Schmid, "A performance evaluation of local descriptors," *Proc. of Internat'l Conf. on Computer Vision & Pattern Recognition*, Vol. 2, pp. 257-263, June 2003.

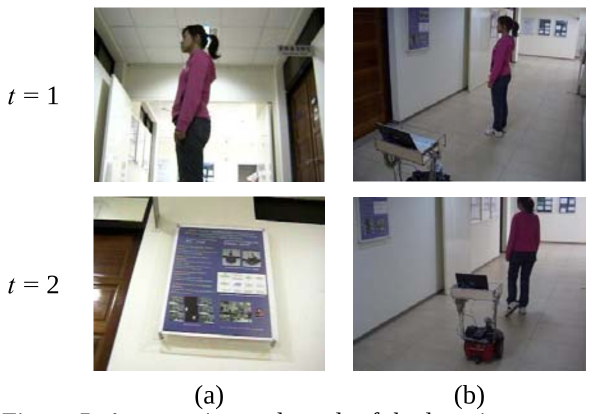

Figure 5. An experimental result of the learning mode. (a) Input image. (b) Position of vehicle.

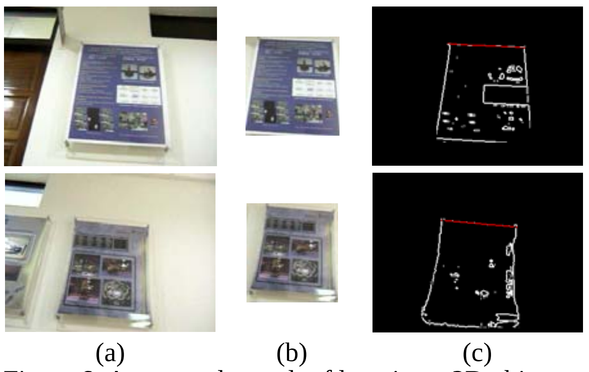

Figure 6. An example result of learning a 2D object. (a) Input image. (b) Region of 2D object. (c) Horizontal line of 2D object.

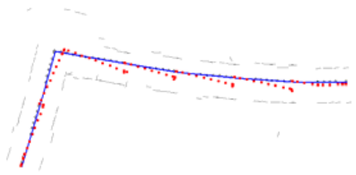

Figure 7. An experimental result of navigation with blue lines showing path planning result, and red points showing real patrolling path.

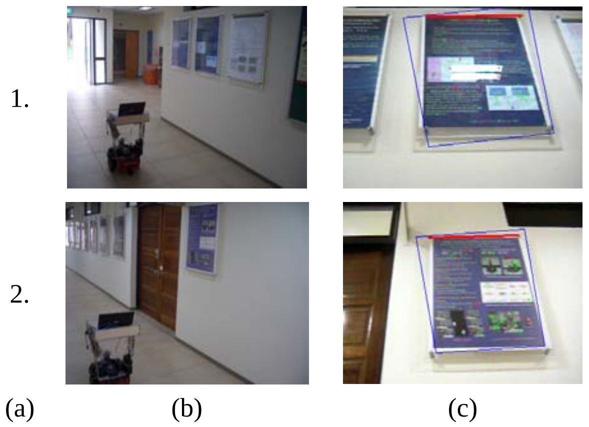

Figure 8. Experimental result of object monitoring and navigation path correction. (a) Numbers of monitored objects. (b) The vehicle monitors the objects. (c) The matching result and the horizontal line used for path correction.