

# Recognition and Path Planning Strategy for Autonomous Navigation in the Elevator Environment

Jeong-Gwan Kang, Su-Yong An, Won-Seok Choi, and Se-Young Oh

**Abstract:** This paper presents a robust and reliable method for a mobile robot to get on/off an elevator in a multistory building. Getting on/off the elevator requires the robot to perform two different tasks: a recognition task and a navigation task. First, we propose a recognition algorithm for the elevator buttons and status so that the robot reacts flexibly to the current elevator status. We first apply an adaptive threshold to the current image in order to get a binary image. Then we extract the candidates of the buttons and the floor number after preliminary filtering. Ambiguous candidates are rejected using an artificial neural network, and a matching method is applied to finally recognize the call buttons, destination floor buttons, moving direction and current location of the elevator. Second, we suggest a path planning algorithm to navigate into and out of the elevator without any collision. By constructing an occupancy grid map and computing a target function, we find the best position for the robot to get on the elevator. Then we plan an optimal path to the best position using a potential field method. Experiments were carried out in several simulated and real environments including empty, crowd and blocked scenarios. The approach presented here has been found to allow the robot to navigate in the elevator without collisions.

**Keywords:** Elevator, mobile robot, navigation, neural network, object recognition.

## 1. INTRODUCTION

Over the past ten years, the number of robots which have been deployed in museums, trade shows and exhibitions has grown steadily; they offer many kinds of services (e.g., guidance, delivery, cleaning, and patrol) with a high degree of reliability. A successful service robot must have complete autonomous capabilities. Hence, most services that the robot offers are based on autonomous navigation, one of the most important fields in the mobile robotics. A representative service robot is the Minerva [1]. Minerva was successfully deployed in a museum for two weeks offering tour guidance. Xavier [2] is a web-based autonomous robot. It estimates its location with a localization algorithm based on a partial observable Markov decision process (POMDP) [3] and avoids obstacles using a Lane-Curvature Method [4]. HERMES [5] was demonstrated in a long-term test in a museum (both office and exhibition areas) where it interacted and communicated with staff and visitors and also performed useful services for several hours a day for six months. RoboX [6] offered a tour guide service in

EXPO'02. The RobotX family was up and running for 9,000 hours, interacted with more than 500,000 visitors and drove an overall distance exceeding 2,500 km. In addition, many other service robots [7,8] were also introduced that successfully performed autonomous navigation in the museum or office environments. However, their navigation has been limited to only a single-floor environment.

Autonomous navigation in a multi story building requires that the robot can get on/off the elevator without any help. Several studies have been made on methods for the robot to get on/off the elevator. GRACE [9] was able to detect and enter/exit the elevator using a laser scanner. However, it could not recognize the floor where the elevator is. MOPS [10] detected the state of the elevator door using LIDAR sensors, whereas manipulation of the elevator (e.g., selecting the floor, obtaining the elevator floor information.) was realized with an infrared communication system. The robots in [11,12] could navigate in the elevator depends on the communication system between the robot and the elevator. Without this infrastructure, these robots cannot enter/exit the elevator. The Fujitsu Service Robot [13] can detect the elevator call button and door using the stereo vision system mounted on the robot. Another mobile robot [14] was able to learn how to get on/off the elevator from a user, and it can get on/off the elevator by itself.

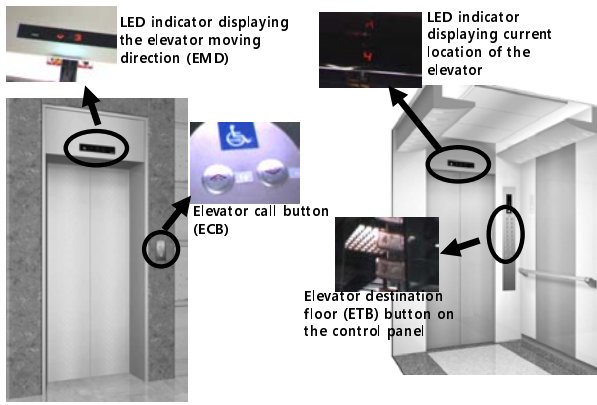
## 2. PROBLEM STATEMENTS

To use the elevator requires a coordinated execution of many component tasks. As a first step to use the elevator, the robot must call the elevator. To this end, the robot has

---

Manuscript received October 31, 2008; revised November 19, 2009; accepted February 2, 2010. Recommended by Editorial Board member Sooyong Lee under the direction of Editor Jae-Bok Song. This work was supported by National Strategic R&D Program for Industrial Technology of the Korean Ministry of Knowledge Economy.

Jeong-Gwan Kang, Su-Yong An, Won-Seok Choi, and Se-Young Oh are with the School of Electronic and Electrical Engineering, Pohang University of Science and Technology, San 31 Hyoja-dong, Nam-gu, Pohang 790-784, Korea (e-mails: {naroo1, grasshop, onlydol, syoh}@postech.ac.kr).



(a) Outside of the elevator. (b) Inside of the elevator.

Fig. 1. The caption must be shown after the figure.

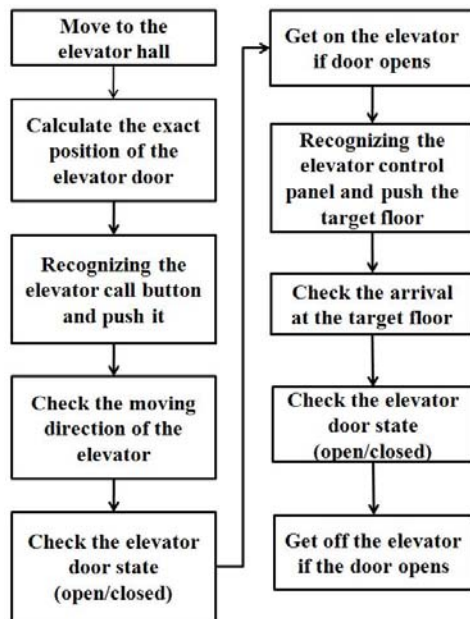


Fig. 2. The procedures for using an elevator.

to recognize the elevator call button (Fig. 1(a)) and press it. After calling the elevator, the robot should continuously inspect the LED indicator above the elevator door to check the moving direction of the elevator (Fig. 1(a)). If the direction coincides with the target direction, the robot gets into the elevator when its door opens. Within the elevator, as a second step, the robot stops in front of the elevator control panel (Fig. 1(b)) and then presses the button of the target floor. Next, the robot moves toward the center area if available and checks for arrival at the target floor using the number recognition method (Fig. 1(b)). Finally, when the elevator stops at the target floor and the door opens, the robot gets off the elevator. In summary, there are many coordinated maneuvers to carry out for the robot to use the elevator (Fig. 2).

In this paper, we proposed a framework for autonomous navigation in the elevator environment which is an extension of our previous work [15]. For getting on/off the elevator, the robot performs two different tasks. The first task is a recognition task. This is used to understand the elevator status including elevator call button, destina-

tion floor button on the control panel, moving direction, and current location of the elevator. Let us simply call each of these recognition targets a 'target object'. These four target objects are recognized through a stereo vision system, which enables the robot to calculate the 3D position of those objects. The second is a navigation task. It contains generation of a collision-free path to the elevator. The two tasks, the recognition and the navigation, are not separate but linked to each other in the overall framework.

To begin with, we suggest robust and consistent recognition methods to recognize four different target objects. Those methods are mainly based on adaptive thresholding, a neural network (NN) classifier, and template matching, but other additional techniques are also applied such as graph partitioning and on-line retraining to enhance recognition performance. Next, we propose a path planning algorithm to get on and off the elevator. In many cases, there may be some passengers in the elevator whose positions cannot be defined before the elevator door opens. Hence, we are not able to decide beforehand on the robot path into the elevator. Thus we make an occupancy grid map of the elevator and use it to compute a collision free path for getting on/off the elevator.

This paper is structured as follows. The next section presents the image processing algorithm for recognition, Section 4 presents the localization and map building and then Section 5 describes the path planning and navigation algorithm for entering/exiting the elevator. The last section shows the experimental results.

### 3. OBJECT RECOGNITION IN THE ELEVATOR

As mentioned above, there are four types of target objects. For recognizing each target object, we follow general procedures for segmenting and identifying object candidates. These common procedures include adaptive thresholding, a NN classifier, and template matching. In fact, however, target objects look different from each other in terms of size, shape, and lighting condition; therefore, the detailed steps for detection or rejection of candidates should also differ according to target objects. We can assume an uniform lighting condition when recognizing the elevator call button and moving direction, because these objects are placed outside the elevator. In this case, simple non-adaptive preprocessing is enough to extract candidates. In contrast, the inside of the elevator is a very reflective environment surrounded by mirror-like materials. Furthermore, elevator movement can cause immediate changes of lighting due to transparent glass walls.

Thus, we propose a more robust method for rejecting false candidates using graph partitioning and an online retraining NN when recognizing the control buttons. Recognition of current floor number also must deal with various lighting conditions. In this case, we use a stepwise thresholding technique for better segmentation of floor number. In this section, we first state common procedures and then describe the detailed strategy of detecting each target object and rejecting false candidates.

### 3.1. Common procedures

**Candidate Extraction:** Under a varying illumination condition, an adaptive thresholding [16] can be used to separate possible object regions from a background. It performs binary thresholding by analyzing each pixel with respect to its local neighborhood. In Table 1,  $I_{source}$  is an original input image and its size is  $H \times W$ . A neighborhood size for deciding the local threshold  $T(r, c)$  is  $m \times n$  and  $\sigma$  is an offset value subtracted from a local mean intensity.

Fig. 3(b) is the resulting binary image after applying the adaptive thresholding. A collection of white pixels represents possible object region. As can be seen in Fig. 3(b), there are too many possible regions, including those outside our interest as well as the target object regions such as the numbers from one to five. Then, the next problem is how to remove these undesirable regions. First of all, the white regions have to be divided into several independent groups.

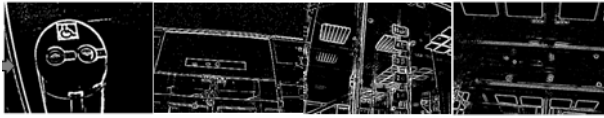
By grouping white pixels that are connected to each other using connected component labeling [17], a group of white pixels can be defined as a single object

Table 1. The adaptive thresholding procedure.

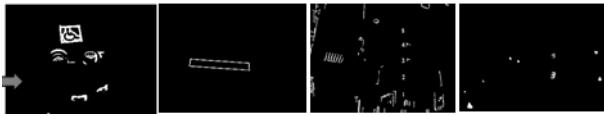
| <b>Adaptive_Thresholding (<math>I_{source}, H, W, m, n, \sigma</math>) :</b> |   |
|--|---|
| 1.   | For $r = 1$ to $H$  |
| 2.   | For $c = 1$ to $W$  |
| 3.   | $T(r, c) = \frac{1}{m \cdot n} \sum_{i=1}^m \sum_{j=1}^n I_{source}(i, j) - \sigma$                   |
| 4.   | $I_{binary}(r, c) = \begin{cases} 1 & I_{source}(r, c) > T(r, c) \\ 0 & \text{otherwise} \end{cases}$ |
| 5.   | EndFor  |
| 6.   | EndFor  |
| 7.   | return $I_{binary}$   |



(a) Original images.



(b) Adaptive thresholding results.



(c) Connected component labeling and size filtering results.



(d) Object candidates.

Fig. 3. The common procedures with corresponding result images for object candidate extraction.

Table 2. Extration of the object candidates.

| <b>Extracting_Candidates (<math>I_{source}</math>) :</b> |  |
|--|--|
| 1.   | $I_{binary} \leftarrow \text{Adaptive\_Threshold}(I_{source}, H, W, m, n, \sigma)$   |
| 2.   | $O = \{o_i \mid i=1, \dots, N\} \leftarrow \text{Connected component grouping}(I_{binary})$  |
| 3.   | $n \leftarrow 0$   |
| 4.   | $C \leftarrow \{\text{empty set}\}$  |
| 5.   | For all elements $o_i = (\beta_i, p_i, w_i, h_i, r_i, I_i)$ in the object candidate set $O$ do   |
| 6.   | If $\beta_i > \beta_{min}$ and $r_{min} < r_i < r_{max}$ Then<br>$c_n \leftarrow g_i = (\beta_i, p_i, w_i, h_i, r_i, I_i)$<br>$n \leftarrow n+1$ |
|  | EndIf  |
| 7.   | EndFor   |
| 8.   | return $C = \{c_i \mid i=1, \dots, n\}$  |

candidate  $o_i$  (Line 2 of Table 2). Then, we construct the object candidate set  $O$  which has  $N$  elements. Each candidate  $o_i$  in the object candidate set  $O$  is  $w_i \times h_i$  pixels, and it is represented by several properties: area  $\beta_i$  (i.e., the number of white pixels in it), a center position  $p_i$  on the image space, a width  $w_i$ , a height  $h_i$  and a width-to-height ratio  $r_i$  of its bounding box (line 5 of Table 2).

The number of candidates from the connected component labeling is reduced by a size filtering which uses geometric constraints such as area and width-to-height ratio of a candidate (Fig. 3(c)). In other words, a size filtering removes candidates which do not satisfy the following constraints:  $\beta_i$  should be larger than minimum area  $\beta_{min}$  and  $r_i$  should be between  $r_{min}$  and  $r_{max}$ , the lower and upper bounds of the width-to-height ratio (line 6-8 of Table 2). This preprocessing eliminates most false candidates while the target objects remain. However, this resulting image still has false candidates (Fig. 3(d)). In a subsequent step, a NN-based filter is applied to remove these false candidates.

**NN as a Binary Classifier:** The NN is a filter that receives a resized object candidate  $\hat{c}_i$  as input and generates an output ranging from 1 to -1, signifying the presence or absence of a target object [18].  $\hat{c}_i$  is a histogram-equalized and size-normalized version of  $c_i$ , with a pixel region. This size normalization reduces and fixes the number of connections between the input and hidden layer. Each image patch is converted into a vector whose elements represent pixel intensities which are normalized from zero to one, and fed into the NN. Each neuron in the input layer accepts corresponding normalized pixel intensity of  $\hat{c}_i$ , implying that the number of neurons in the input layer is identical to the size of  $\hat{c}_i$ ; thus there are 300 neurons in the input layer (Fig. 4).

To decide on the proper size of the network, many pruning techniques including the sensitivity method, penalty-term method, and pruning by a genetic algorithm have been proposed [19]. Our approach to choose the optimal network size, i.e., the number of neurons used in the hidden layer, is straightforward. We trained a number of networks of various sizes and selected the smallest one that shows a good generalization performance; we



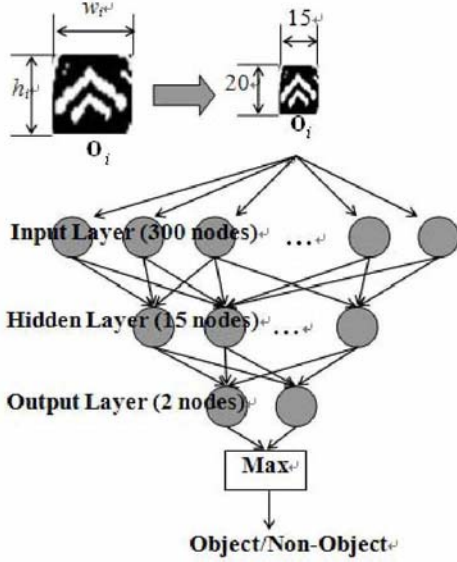


Fig. 4. Architecture of the neural network classifier.

found that utilization of 15 neurons in the hidden layer was the optimal choice in our classification problem. The number of neurons in the output layer corresponds to the number of output classes. We have two classes: object and non-object. Thus the output layer has two neurons. After input data was fed to the network, we observed the output values of the two neurons in the output layer, each of which represents a degree of activation. Let  $\mathbf{y} = [y_1, y_2]^T$  denote the activation vector of the output neurons, where each element of  $\mathbf{y}$  represents the corresponding activation value. If  $y_1$  is activated less than  $y_2$  and a predefined threshold  $\theta_a$ , then the input image patch is considered to belong to the non-object class (Fig. 5(a)). On the other hand, if  $y_1$  is more activated than  $y_2$  and  $\theta_a$ , it belongs to the object class (Fig. 5(b)). In this manner, each input candidate is classified as object/non-object according to the distribution of these activations in the output layer. Once a candidate is classified as a non-object, then it is removed from the object candidate set  $\mathbf{O}$ .

Although much work has been done on the topic of NN training, backpropagation (BP) is the most widely used training method [20]. BP training adjusts the neural weights in an iterative and gradient descent way, minimizing the overall error between the desired and actual outputs of the NN:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \Delta\mathbf{w}(k), \quad (1)$$

$$\Delta\mathbf{w}(k) = -\eta \frac{\partial \mathbf{e}(k)}{\partial \mathbf{w}(k)}, \quad (2)$$

where  $k$  is an iteration number within training,  $\eta$  is a learning rate,  $\mathbf{e}$  is error at the output layer, and  $\mathbf{w}$  is a collection of weights associated with all layers. The initial weights of the NN are distributed according to [20]. Here, note that we have four NN classifiers, each of which covers a different target object group.

**Object Identification by Template Matching:** To identify an object image from the NN output, we use the Template Matching method. It is a technique for

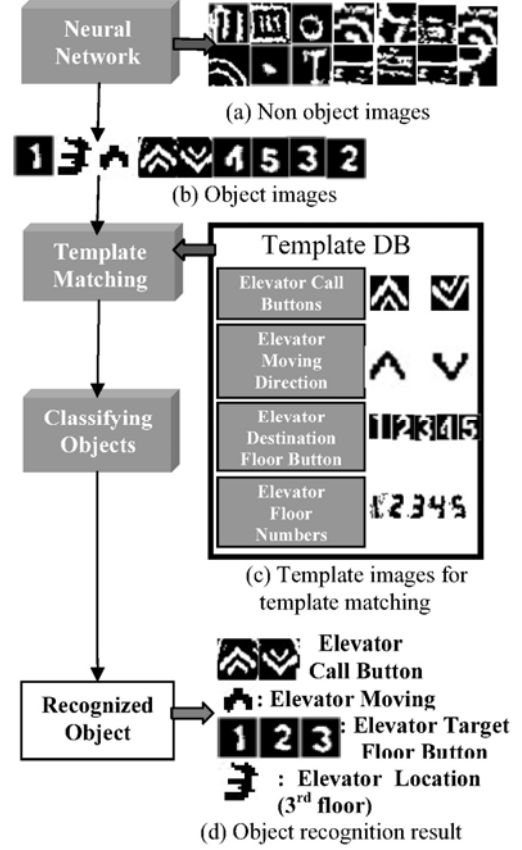


Fig. 5. The common procedures with corresponding result images for object recognition.

estimating a correlation between a query image and a given template, where a high correlation value means that the query image is similar to the given template. To begin with, we make a Template DB which consists of 14 templates, each of which corresponds to a target object (Fig. 5(c)). The Normalized Cross-Correlation (NCC) [21] is used to compute the correlation value between the object image and the  $i^{\text{th}}$  template in the Template DB. Let  $o(x, y)$  denote the intensity value of the object image  $\mathbf{o}$  of the size  $M_x \times M_y$  at point  $(x, y)$ . The  $i^{\text{th}}$  template is represented by  $T_i$  of the size  $N_x \times N_y$ . In this research, the size of  $\mathbf{o}$  is 15 by 20 and  $T_i$  is 10 by 14; thus, the search space is limited to 6 by 7. Let  $\rho(u, v)$  be the NCC value at point  $(u, v)$ :

$$\rho(u, v) = \frac{\sum_{x,y} (o(x, y) - \bar{o}_{u,v}) (T_i(x-u, y-v) - \bar{T}_i)}{\sqrt{\sum_{x,y} (o(x, y) - \bar{o}_{u,v})^2 (T_i(x-u, y-v) - \bar{T}_i)^2}}, \quad (3)$$

where  $\bar{o}_{u,v}$  is the mean value of the template  $T_i$  area in the object image  $\mathbf{o}$  shifted to  $(u, v)$ , and  $\bar{T}_i$  is the mean value of the  $i^{\text{th}}$  template  $T_i$ .

The template matching result  $\rho_i$  with the template  $T_i$  is a maximum value of  $\rho(u, v)$ . The index of maximum value of  $\rho_i$  is considered to be the object image. If the index is denoted by  $k$ , the object image from NN is regarded as the  $k^{\text{th}}$  object in the Template DB (Fig. 5(d)).

$$\rho_i = \max_{u,v}(\rho(u,v)), \quad (4)$$

$$k = \arg \max_i(\rho_i). \quad (5)$$

### 3.2. Different strategies for recognizing target object

**Elevator Moving Direction (EMD):** The Moving direction of the elevator is displayed above the elevator door. We assume that the robot is placed around the elevator door but does not know its exact position. Thus the robot first finds the display panel using the pan-tilt module which is mounted on its head (Fig. 6(a)). Here, we also use the adaptive thresholding to extract the display panel and filter out false candidates (Fig. 6(b)). This gradual approach to target object recognition reduces the computational cost of searching the whole image. The detected display panel can be rotated, so we align the panel image parallel to the x-axis of image coordinate. If several display panel candidates are admitted by size filtering (Fig. 6(c)), then we apply common procedures to these display panel images to detect and recognized the target object in the usual way as described in Section 3.1 (Fig. 6(d)). In this stage, we can calculate the relative position of the robot (i.e., with the origin at the center of the display panel) using the stereo camera by the matching the four corner points of the display panel.

**Elevator Call Button (ECB):** If the robot knows its relative position by detecting the display panel or the elevator door, it can navigate and locate the ECB based on odometric information. Instead of searching for the target object directly, using the pan-tilt module, the robot first finds a mark which is easily detected (Fig. 7). For this step, we convert the color space of the input image from RGB to HSV so as to bring the blue-colored mark into relief. Based on detection, two possible regions of

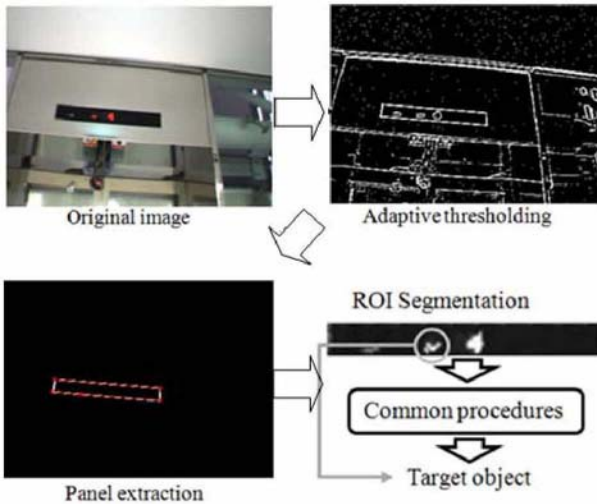


Fig. 6. Adaptive thresholding is applied to an original image to extract the display panel. Straight lines (red lines) are extracted using Hough transform to compensate for image rotation. Target objects are found within the segmented ROI image to reduce computation time.

interest (ROI) can be obtained; one is for the up button and the other is for the down button. Then, like the case of detecting the moving direction, we follow common procedures (except for the NN classifier) to detect our final target objects. The reason that the NN is excluded that setting two ROIs can effectively prune a large portion of the search space, meaning that a very small number of candidates is extracted for the two ROIs.

**Current Floor Number:** In this stage, the robot may not be placed exactly in front of the LED display panel, so the raw image can be rotated. We compensate for this rotation by extracting the dominant line segment from the edge image. The lighting condition varies with movement of the elevator, so more consideration should be given to segment the floor number. Here, we do not apply adaptive thresholding for initial segmentation, but

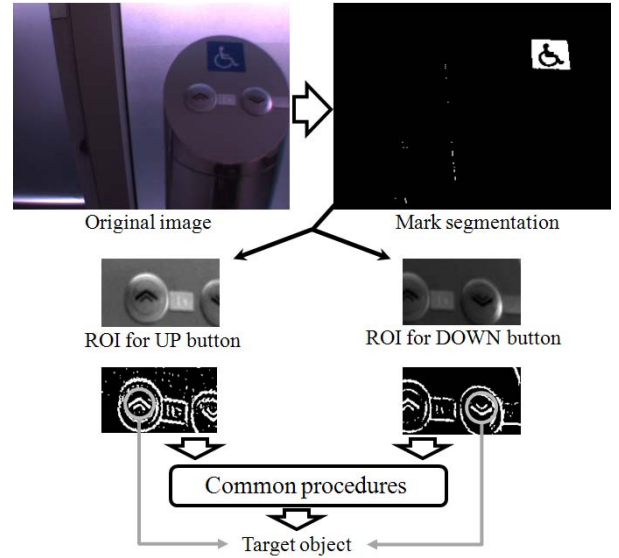


Fig. 7. A saturation channel of the HSV color space is used to segment the mark. Based on this estimated mark position in the image space, two ROIs can be obtained. Target objects are found within ROI image through common procedures.

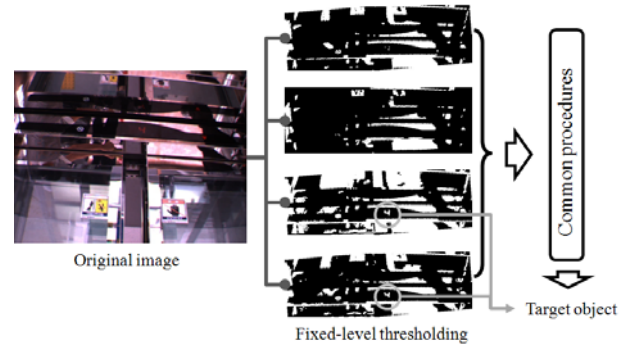
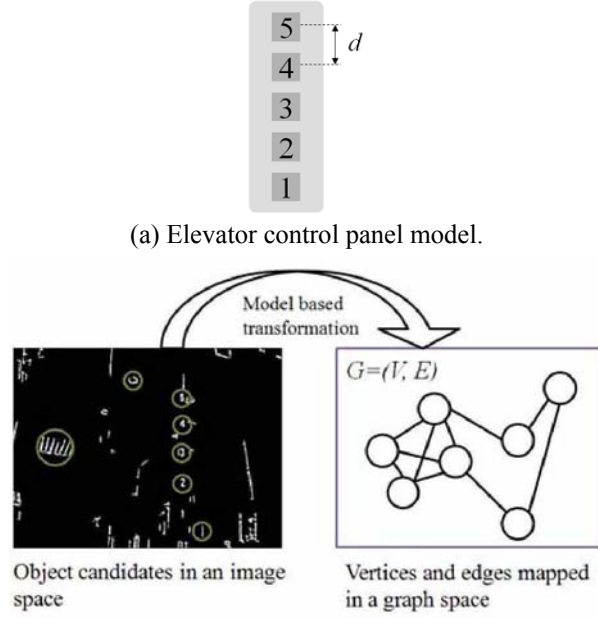


Fig. 8. Only the R channel of the original image is used in fixed-level thresholding. Different thresholding levels yield different binary images. The reason for multiple thresholding is that the overall intensity of input image varies with movement of the elevator, so the optimal thresholding level may change.



(b) Graph representation of object candidates based on control panel model [15].

Fig. 9. Graph representation for recognizing the elevator control panel model.

multiple fixed-level thresholding is used to detect floor number candidates (Fig. 8). The multiple thresholding (i.e., repetitive thresholding with different threshold levels) prevents a loss of information when the optimal binary thresholding parameter is not known. We know that the floor number is displayed in red, so we deal only with the R channel from the RGB color space and threshold this image. According to the thresholding level, the floor number may be or may not be visible in the resulting binary image. Applying common procedures to all resulting binary images, we can identify the target floor number. One may speculate that repetitive thresholding could cause a computational burden, but, we found that this does not deteriorate performance in real-time applications.

**Elevator Target Floor Button (ETB):** Since the ETBs are aligned vertically in the control panel, we can detect these buttons together [15]. We modeled the elevator control panel as in Fig. 9(a), where  $d$  is a constant proportional to the distance between two adjacent buttons, which will be used as the scale factor for estimating the real distance in image coordinates. With this control panel model and a graph representation of the object candidates, we can further eliminate false candidates.

Usually a graph  $G$  is expressed by a set of vertices  $V$  and edges  $E$  connecting them [22]. An image space can be transformed into a graph space; object candidates and their relationships in the image space are mapped to vertices and edges, respectively (Fig. 9(b)). In our application to rejection of false candidates, all vertices are connected with each other via edges which have different connection weights. The connection weight  $a_{ij}$  between the  $i^{th}$  and the  $j^{th}$  vertices is calculated based on

the control panel model as follows:

$$a_{ij} = \frac{1}{\sqrt{2\pi}} \exp(-(d_{ij}^O - d_{ij}^M)^2 / 2), \quad (6)$$

where  $d_{ij}^O$  is the distance between two object candidates observed in a scene image and  $d_{ij}^M$  is the reference distance estimated by the control panel model. If two candidates match the model well, then the connection weight will be high, otherwise low.

In this manner, we can assign connection weights to all possible pairs of two vertices. A collection of connection weights is represented in the form of an  $N$  by  $N$  symmetric matrix,  $\mathbf{A} = [a_{ij}]$ , defined as an adjacency matrix, where  $N$  is the number of candidates. A false candidate in a graph is defined as a vertex which has a relatively low connection weight; thus an average connection weight  $m(G)$  of a graph  $G$  is maximized when such false candidates are removed from the graph  $G$ .  $m(G)$  is calculated as follows:

$$m(G) = \frac{\sum_i \sum_j a_{ij}}{N}, \quad (7)$$

Let us now define  $\mathbf{u}$  as an indicator whose  $i^{th}$  element  $u_i$  represents whether an object candidate is a false candidate or not:

$$u_i = \begin{cases} 0 & \text{if } o_i \text{ is a false candidate} \\ 1 & \text{otherwise.} \end{cases} \quad (8)$$

Consequently  $\mathbf{u}$  expresses a subgraph  $H = (Y, F)$  of a graph  $G=(V, E)$ , which satisfies the following conditions:

$$Y \subseteq V \text{ and } F \subseteq E \cap (Y, 2), \quad (9)$$

where  $(Y, 2)$  means an edge set which consists of edges formed by any two vertices in  $Y$ . Then,  $m(H)$  of a subgraph  $H$  is represented by a matrix form:

$$m(H) = \frac{\mathbf{u}^T \mathbf{A} \mathbf{u}}{\mathbf{u}^T \mathbf{u}}, \quad (10)$$

where the numerator is a summation of all connection weights except those of false candidates, and the denominator is the number of elements whose values are one.

Then, the problem is to find a subgraph  $H$ , or  $\mathbf{u}_{\max}$  that maximizes  $m(H)$ :

$$\mathbf{u}_{\max} = \arg \max_{\mathbf{u}} \frac{\mathbf{u}^T \mathbf{A} \mathbf{u}}{\mathbf{u}^T \mathbf{u}}. \quad (11)$$

We seek  $\mathbf{u}_{\max}$  by differentiating (10) with respect to  $\mathbf{u}$ , giving:

$$\mathbf{A} \mathbf{u} = m(H) \mathbf{u}. \quad (12)$$

Fortunately,  $m(H)$  and  $\mathbf{u}$  are the eigenvalues and eigenvectors of the adjacency matrix  $\mathbf{A}$ , respectively. We can get a maximum eigenvalue  $m(H)$  and its

corresponding eigenvector  $\mathbf{u}_{\max}$  with the help of linear algebra. Then, each element of  $\mathbf{u}_{\max}$  indicates a degree of false candidate. We convert each element of  $\mathbf{u}_{\max}$  into a binary value so as to indicate either a false candidate or not as follows:

$$u_i = \begin{cases} 1 & \text{if } u_i > \theta_u \\ 0 & \text{otherwise,} \end{cases} \quad (13)$$

where  $\theta_u$  is a predefined threshold value. Thus, rejection is simply done by a thresholding eigenvector.

**On-line retraining of NN:** For the NN retraining, we prepared two different data storages, the Object Image Queue (OIQ) and Non-object Image Queue (NIQ) (Fig. 10). Once the object candidate is determined to be a non-object by graph partitioning, it is stored in the NIQ and used as negative training data at the next retraining step. The data storage mechanism of the NIQ is analogous to First In First Out (FIFO); thus whenever new training data comes in, the oldest training data is deleted from the NIQ. In the same way, the recognized object candidate is stored in the OIQ. These newly stored data in OIQ and NIQ form the new training data set for next retraining. The decision of when to retrain the NN is related to the NIQ, especially the number of deleted non-objects. If this number is greater than a pre-defined value, then the NN is retrained. The generation of new training data and decision of when to retrain are inherent in the existing recognition process, so real-time retraining is guaranteed without any extra process.

#### 4. LOCALIZATION AND MAPPING WITHIN THE ELEVATOR

As mentioned earlier, we cannot decide the path for entering in the elevator before the door opens. So whenever the robot gets into the elevator, an occupancy grid map (OGM) of the elevator [23] is generated and a collision free path based on the OGM is computed. Construction of the OGM requires that the robot knows its own position. In this research, we calculate the robot pose using deadreckoning which is improved by the gyroscope and the Extended Kalman Filter (EKF) [24,25] localization method.

##### 4.1. Improved deadreckoning using a gyroscope

If the walls or door of the elevator are detected, the odometric error of the robot can be corrected by EKF localization. However, if the walls or door are occluded by obstacles or passengers in the elevator. In this case, the robot pose and OGM of the elevator are computed based on deadreckoning only.

Deadreckoning is the most widely used technique for estimating the pose of a mobile robot, taking into account prior position and amount of travel distance derived from encoder readings. However, the pose estimated from encoding readings deteriorates rapidly with movement due to gearbox play, wheel imbalance, slippage and many other real world factors. So we can rely on the deadreckoning ability of the robot for just

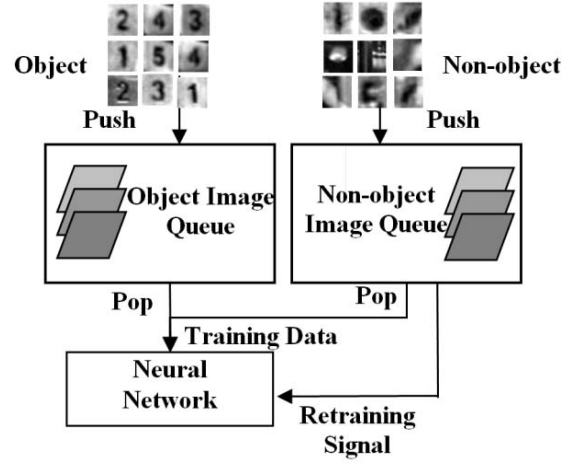


Fig. 10. Training data examples and retraining mechanism.

short range on the order of several meters, depending on the surface. The robot may not successfully get on/off the elevator if it computes its pose and the OGM of the elevator based on deadreckoning only. A gyroscope is usually used to increase accuracy of the deadreckoning by providing a reference direction. In this research, the heading information of the robot from the gyroscope is fused to odometry by the EKF to minimize the deadreckoning error. The heading angle from a gyroscope with bias drift is represented as follows:

$$\theta_t = \theta_m + \varepsilon_{drift} + \varepsilon_{random}, \quad (14)$$

where  $\theta_t$  is the robot's heading information based on the gyro,  $\theta_m$  is the robot's actual heading,  $\varepsilon_{drift}$  is the gyro bias drift error and  $\varepsilon_{random}$  is the associated white noise. The drift error is influenced by the temperature and the rate of rotation and can be approximated by the third-order polynomial,

$$\varepsilon_{drift}(\dot{\theta}_m) = a_0 + a_1 \cdot \dot{\theta}_m + a_2 \cdot (\dot{\theta}_m)^2 + a_3 \cdot (\dot{\theta}_m)^3. \quad (15)$$

In this research, we find the drift error parameters,  $[a_0, a_1, a_2, a_3]$ , using particle swarm optimization (PSO) [26]. Candidate solutions for the drift error parameters are encoded into the particles and the optimal error parameters are estimated by adjusting the particles in the PSO process. The approximated values of the drift error parameters,  $[a_0, a_1, a_2, a_3]$ , were set to 0.0005, 0.0023, 0.0025 and 0.0118, respectively.

Let  $\bar{\mathbf{x}}_t^+ = [\bar{x}_t, \bar{y}_t, \bar{\theta}_t]^T$  and  $\bar{\mathbf{P}}_t^+$  be the fused robot pose and the uncertainty covariance corresponding to the robot pose. The EKF recursively updates  $\bar{\mathbf{x}}_t^+$  and  $\bar{\mathbf{P}}_t^+$  by combining the predicted pose  $\bar{\mathbf{x}}_t^-$  and covariance  $\bar{\mathbf{P}}_t^-$  with current measurement  $\theta_t$  of the robot heading from the gyro. The EKF [25] is divided into two steps: a prediction step and a correction step. When the fused robot pose at time  $t-1$  is  $\bar{\mathbf{x}}_{t-1}^+ = [\bar{x}_{t-1}^+, \bar{y}_{t-1}^+, \bar{\theta}_{t-1}^+]^T$  and the control command to the robot is  $\mathbf{u}_t = [v_{t,l}, v_{t,r}]$ , the



predicted robot pose and covariance at time  $t$  are computed as:

$$\bar{\mathbf{x}}_t^- = \mathbf{f}_v(\bar{\mathbf{x}}_{t-1}^+, \mathbf{u}_t) = \begin{bmatrix} \bar{x}_{t-1}^+ + \frac{(v_{t,l} + v_{t,r})}{2} \Delta t \cos\left(\bar{\theta}_{t-1}^+ + \frac{(v_{t,r} - v_{t,l})}{2b} \cdot \Delta t\right) \\ \bar{y}_{t-1}^+ + \frac{(v_{t,l} + v_{t,r})}{2} \Delta t \sin\left(\bar{\theta}_{t-1}^+ + \frac{(v_{t,r} - v_{t,l})}{2b} \cdot \Delta t\right) \\ \bar{\theta}_{t-1}^+ + \frac{(v_{t,r} - v_{t,l})}{b} \Delta t \end{bmatrix}, \quad (16)$$

$$\bar{\mathbf{P}}_t^- = \nabla \mathbf{f}_x \bar{\mathbf{P}}_{t-1}^+ \nabla \mathbf{f}_x^T + \nabla \mathbf{f}_u \mathbf{Q} \nabla \mathbf{f}_u^T, \quad (17)$$

$$\nabla \mathbf{f}_x = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\bar{\mathbf{x}}_{t-1}^+}, \quad \nabla \mathbf{f}_u = \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \Big|_{\mathbf{u}=\mathbf{u}_t}, \quad (18)$$

$$\mathbf{Q} = \begin{bmatrix} |\sigma_l|^2 & 0 \\ 0 & |\sigma_r|^2 \end{bmatrix}, \quad (19)$$

where  $\mathbf{f}_v(\cdot)$  is a motion model of mobile robot,  $\nabla \mathbf{f}_x$  and  $\nabla \mathbf{f}_u$  are jacobians of the robot motion model relative to the robot pose vector and control command, respectively.  $\mathbf{Q}$  represents the error covariance matrix of the motion model. Here, the values of  $\sigma_l$  and  $\sigma_r$  were set empirically to 0.3.

After the heading angle of the robot,  $\theta_t$ , is measured from the gyroscope,  $\bar{\mathbf{x}}_t^+$  and  $\bar{\mathbf{P}}_t^+$  are computed in the update step of the EKF.

$$\nabla \mathbf{h} = [0 \ 0 \ 1], \quad \mathbf{R} = \sigma_\theta^2, \quad (20)$$

$$\mathbf{K}_t = \bar{\mathbf{P}}_t^- (\nabla \mathbf{h})^T (\nabla \mathbf{h} \bar{\mathbf{P}}_t^- (\nabla \mathbf{h})^T + \mathbf{R})^{-1}, \quad (21)$$

$$\bar{\mathbf{x}}_t^+ = \bar{\mathbf{x}}_t^- + \mathbf{K}_t (\theta_t - \bar{\theta}_t^-), \quad (22)$$

$$\bar{\mathbf{P}}_t^+ = (\mathbf{I} - \mathbf{K}_t \nabla \mathbf{h}) \bar{\mathbf{P}}_t^-. \quad (23)$$

Here,  $\nabla \mathbf{h}$  is a jacobian matrix of the observation model relative to the robot pose,  $\mathbf{R}$  is a measurement noise

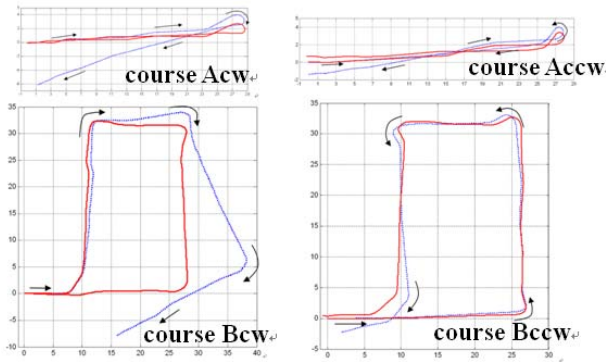


Fig. 11. Mobile robot trajectory with the improved deadreckoning using a gyroscope (blue dotted line: odometry trajectory, red solid line: improved deadreckoning trajectory).

covariance matrix whose element  $\sigma_\theta$  is set to  $(0.1 \cdot \pi / 180)$  radians, and  $\mathbf{K}_t$  is a Kalman gain. Then  $\bar{\mathbf{x}}_t^+$  and  $\bar{\mathbf{P}}_t^+$  are updated by (22) and (23), respectively. The improved deadreckoning with gyroscope was tested at two different courses with clockwise and counter clockwise rotation and the position errors of the improved deadreckoning were less than 1.65m (Fig. 11).

#### 4.2. Line feature based EKF localization

Although the accuracy of odometry is improved by using a gyroscope, small errors still occur which will cause unbounded accumulation errors in the integrated measurements. Because of these errors, especially in the robot's heading, the robot might not approach the elevator door when it opens. Thus, EKF localization is applied to compensate the unbounded accumulation errors based on the assumption that the map is represented by a collection of features. The geometric feature type in this research is line segment, because it is simple and easy to extract from laser scan data in the vicinity of the elevator.

**Laser Scan Data Segmentation:** Range data provided by a laser scanner in a single scan is typically in the form of:

$$\mathbf{M} = \{\mathbf{m}_n = (r_n \cos \phi_n, r_n \sin \phi_n) \mid n = 1, \dots, N\}, \quad (24)$$

where  $(r_n, \phi_n)$ , the polar coordinate of the  $n$ -th range reading, represents the measured distance and direction to an obstacle. A single scan data set  $\mathbf{M}$  is obtained from the laser scanner and then divided into several groups, each of which can be associated with different structures of the environment. This is called a segmentation process to extract a line feature. Range readings belong to the same segment if the distance between them is less than a given threshold. The threshold value for segmentation is determined by an adaptive break point detector [27]. In this method, two consecutive range readings belong to different segments if

$$\|\mathbf{m}_n - \mathbf{m}_{n-1}\| > r_{n-1} \cdot \frac{\sin(\Delta\phi)}{\sin(\lambda - \Delta\phi)} + 3\sigma_r, \quad (25)$$

where  $\Delta\phi$  is the angular resolution of laser scanner, the parameter  $\lambda$  corresponds to the worst case of incidence angle of the laser scan ray w.r.t a line for which the scan points are still reliable and  $\sigma_r$  is the standard deviation of measured distance from the laser scanner. In this research, the parameter values are  $\sigma_r = 0.03\text{m}$  and  $\lambda = 10^\circ$ .

**Line Extraction:** In the line extraction step, the measurement from the laser scanner is divided into several line segments using an *Iterative-End-Point-Fit* (IEPF) method as in [28]. The fitted line  $L_i$  resulting from IEPF is represented as follows:

$$L_i = \{\rho_i, \alpha_i, (x_1^i, y_1^i), (x_e^i, y_e^i)\}, \quad (28)$$

$$\alpha_i = \tan^{-1} \left( -\frac{x_n - x_1}{y_n - y_1} \right), \quad (29)$$



$$\rho_i = \frac{x_n + x_1}{2} \cos \alpha_i + \frac{y_n + y_1}{2} \sin \alpha_i,$$

where  $(\rho_i, \alpha_i)$  are the polar form parameters to represent a line model  $x \cos \alpha_i + y \sin \alpha_i = \rho_i$ ,  $(x_1^i, y_1^i)$  is the start point and  $(x_e^i, y_e^i)$  is the end point of  $s_i$ .

#### Elevator Door Extraction using a Laser scanner:

When the robot is placed as shown in Fig. 12(a), it will make laser readings like those in Fig. 12(b). In order to find the left and right end positions of the elevator door,  $P_{right}$  and  $P_{left}$  in Fig. 12(c), we need to carry out the following:

- Cluster the laser scan points (Fig. 12(c)), where  $C_i$  represents the  $i^{\text{th}}$  cluster.
- Find the clusters which are longer than 0.5 m.
- Among the above long clusters, identify the cluster pairs  $C_i$  and  $C_j$  whose distance between  $P_{i,end}$  and  $P_{j,start}$  are within 1.0 ~ 1.2 m, because the length of the door is 1.1m.

Compute the angle differences of each pair,  $|\theta_i - \theta_j|$ ,  $|\theta_i - \theta_{ij}|$ , and  $|\theta_j - \theta_{ij}|$ .  $\theta_i$  is the angle between the x-axis of the robot coordinate and the cluster  $C_i$ ,  $\theta_{ij}$  is the angle between the x-axis of the robot coordinate and the line which passes through the points  $P_{i,end}$  and  $P_{j,start}$ . If  $\theta_{ij}$  is lower than a specified threshold, we consider  $P_{i,end}$  to be the right end of the elevator door,  $P_{right}$ , and  $P_{j,start}$  to be the left,  $P_{left}$ .

**Line Feature based EKF Localization:** Let point and  $\hat{\mathbf{x}}_t^-$  and  $\hat{\mathbf{P}}_t^-$  be the predicted robot pose and uncertainty covariance of the robot pose, respectively. When the control command to the robot is  $\mathbf{u}_t = [v_{t,l}, v_{t,r}]$ , the robot pose at time  $t$  based on the improved deadreckoning is computed by fusing odometry and gyro measurement. So  $\hat{\mathbf{x}}_t^-$  and  $\hat{\mathbf{P}}_t^-$  in the prediction step of EKF localization are obtained from improved deadreckoning.

$$\hat{\mathbf{x}}_t^- = \bar{\mathbf{x}}_t^+, \quad \hat{\mathbf{P}}_t^- = \bar{\mathbf{P}}_t^+. \quad (30)$$

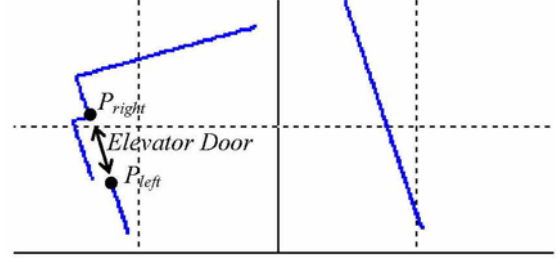
Suppose that the robot measures the  $m$ -th line feature  $\mathbf{z}_m = \{\rho_m, \alpha_m\}$ , whose distance and orientation are  $\rho_m$  and  $\alpha_m$  in the robot centered coordinate. If the line feature in the global feature map is  $l_m = \{r_m, \phi_m\}$  which corresponds to  $\mathbf{z}_m$ , the estimated line feature in the robot centered coordinate,  $\hat{\mathbf{z}}_m(t)$ , can be expressed as a function of  $\mu(\hat{\mathbf{x}}_t^-, l_m)$  [29]:

$$\hat{\mathbf{z}}_m(t) = \mu(\hat{\mathbf{x}}_t^-, l_m) + v_m(t), \quad (31)$$

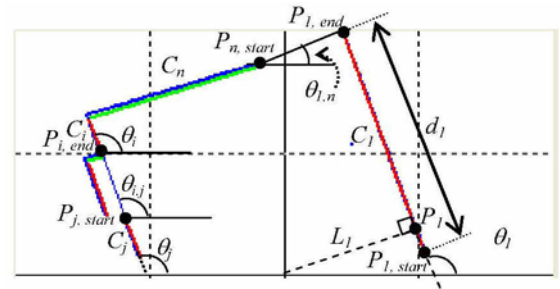
where  $v_m(t)$  models the noise affecting the  $m$ -th measurement. However, the measurement function  $\mu(\hat{\mathbf{x}}_t^-, l_m)$  is defined differently if the line which connects the robot to the origin intersects with the observed line segment in the global coordinate (Fig. 13) [29]:



(a) Simulation environment.



(b) Raw laser points around the elevator door.



(c) Clustering result of the raw laser scan of the elevator.

Fig. 12. Detecting the elevator door using the laser scanner.

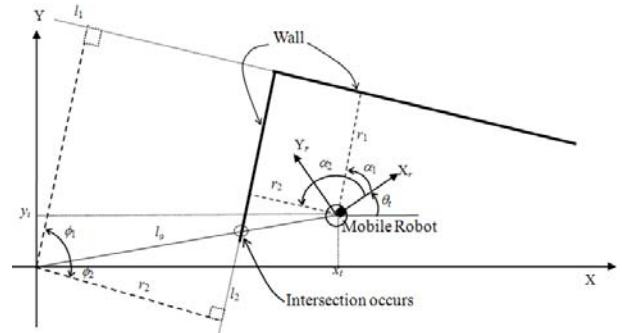


Fig. 13. Line parameter representation w.r.t the global and robot coordinate system.  $l_2$  intersects with  $l_o$  which connects the robot to the origin [29].

$$\mu(\hat{\mathbf{x}}_t^-, l_m) = \begin{cases} \begin{bmatrix} -r_m + \hat{x}_t^- \cos(\phi_m) + \hat{y}_t^- \sin(\phi_m) \\ \phi_m - \hat{\theta}_t^- + \pi \end{bmatrix} & \text{if intersecting} \\ \begin{bmatrix} r_m - \hat{x}_t^- \cos(\phi_m) - \hat{y}_t^- \sin(\phi_m) \\ \phi_m - \hat{\theta}_t^- \end{bmatrix} & \text{otherwise.} \end{cases} \quad (32)$$

Then, the robot pose is updated in the correction step of EKF localization.

$$\mathbf{K}_t = \hat{\mathbf{P}}_t^- (\nabla \mu)^T (\nabla \mu \hat{\mathbf{P}}_t^- (\nabla \mu)^T + \mathbf{R}_{l_m})^{-1}, \quad (33)$$

$$\hat{\mathbf{x}}_t^+ = \hat{\mathbf{x}}_t^- + \mathbf{K}_t (\mathbf{z}_m - \hat{\mathbf{z}}_m(t)), \quad (34)$$

$$\hat{\mathbf{P}}_t^+ = (\mathbf{I} - \mathbf{K}_t \nabla \mu) \hat{\mathbf{P}}_t^-. \quad (35)$$

Here  $\nabla \mu$  is the jacobian matrix of the observation model relative to the robot pose and  $\mathbf{R}_{l_m}$  is an uncertainty covariance matrix of the measured line feature.

#### 4.3. Probabilistic mapping for generating a collision free path

A collision free path is computed using an OGM of the elevator. The occupancy grid  $\mathbf{G}$  consists of cells  $g_{ij}$  which hold part of the environment information, which gathered with sensors (e.g., in our case, the laser scanner) mounted on the robot, and the cell  $g_{ij}$  is updated according to its location whether  $g_{ij}$  lies within the measurable range of the sensors  $\mathbf{S}$  or not. Details of building the occupancy grid map are described in [23].

The elevator interior is a changeable environment because the passengers always get on/off. So the surroundings when the robot recognizes the ETB in the elevator are not the same as when the robot gets on. However the OGM from [23] can not represent this change. When the cell  $g_{ij}$  is out of the sensor readings, we applied the time-decay-constant  $\lambda$  to the  $p(g_{ij}=O)$ . Thus we made the occupancy probability of the cell which is not in the sensor readings decrease according to the time. In this way, our robot can cope with passenger's movement in the elevator. Finally, if  $\mathbf{S}$  denote the measurable range of the sensor, the occupancy grid map in the elevator is updated as follows:

$$p(g_{ij} = O | z_t) = \begin{cases} \frac{p(z_t | g_{ij} = O)p(g_{ij} = O)}{p(z_t | g_{ij} = O)p(g_{ij} = O) + [1 - p(z_t | g_{ij} = O)][1 - p(g_{ij} = O)]} & g_{ij} \in \mathbf{S} \\ p(g_{ij} = O) - \lambda p(g_{ij} = O) & g_{ij} \notin \mathbf{S} \end{cases} \quad (40)$$

Let three passengers already exist in the elevator in the

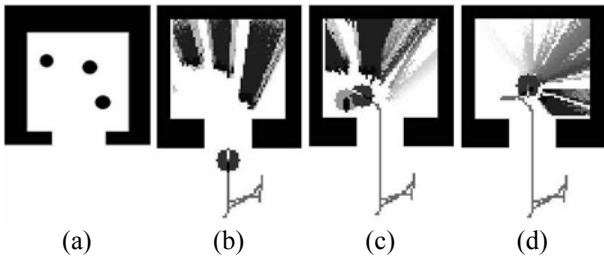


Fig. 14. An occupancy grid map with the time-decay-constant  $\lambda$  applied in the elevator. (a) simulation environment. (b) The occupancy grid map when the robot gets on the elevator. (c) Decayed occupancy grid map when the robot approaches to the elevator destination floor. (d) Decayed occupancy grid map when the robot gets off the elevator.

simulator (Fig. 14(a)). When the robot gets on the elevator, these three passengers are represented in the occupancy grid map of the elevator (Fig. 14(b)). However, when the robot turns left after getting on the elevator, the cell values in the occupancy grid map which represent the passengers standing behind the robot decay (Fig. 14(c) and (d)). Because of this decaying effect, the passenger's movement in the elevator can be represented almost in real time.

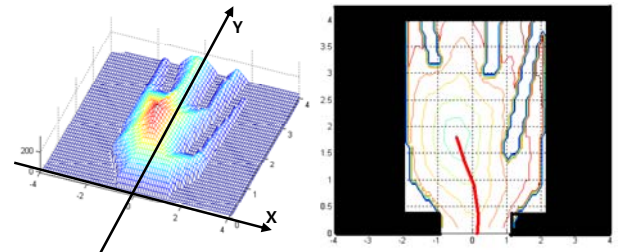
## 5. COLLISION FREE NAVIGATION BASED ON THE SAFETY EVALUATION

### 5.1. Safety evaluation and path plan in the elevator

When no obstacle exists in the elevator, the center area is the best position for the robot to get on, because it is the best place for checking elevator position and door state. However, in many cases, there are some passengers in the elevator whose positions cannot be preestimated before the elevator door opens. Hence, we are not able to decide beforehand on the robot's stop position in the elevator. So we defined the safety evaluation function  $S(i, j)$  which represents how for the cell  $g_{ij}$  in the OGM is from the nearest obstacle and how close to the center of the elevator. When the robot gets into the elevator (Fig. 15(a)(b)),  $d_{min}$  is the distance between  $g_{ij}$  and the closest occupant, and  $d_{offset}$  is a predefined safety margin for collision prevention.  $d_{door}$  is the distance between  $g_{ij}$  and the door center and  $d_{goal}$  is the distance from  $g_{ij}$  to a stop position  $T$  in the elevator. The stop position  $T$  is different depending on the robot position *w.r.t* the elevator door center. In the getting on phase, the stop position of the robot is the center of the elevator. After getting on, the robot changes its stop position to in front of the elevator control panel. Lastly, the stop position is changed to in front of the door for checking the position of the elevator.

Eq (40) is the safety evaluation function used to determine the robot's destination position, which is far from obstacles and close to the stop position. The coefficients  $K_1$ ,  $K_2$  and  $K_3$  change for each phase to compute the safety evaluation function properly. Finally, we decide the robot's destination position for each phase using (41) (Fig. 15(a)).

$$d_{occ} = \min(d_{min}, d_{offset}), \quad (41)$$



(a) Computing result of the Safe Evaluation Function for path planning. (b) Path planning result using the potential field method.

Fig. 15. Safety evaluation and path plan in the elevator.

$$S(i, j) = K_1 \cdot d_{occ} + K_2 \cdot d_{door} + K_3 \cdot d_{goal}, \quad (42)$$

$$G = \arg \max_{i,j} (S(i, j)). \quad (43)$$

To move to the destination position which has the maximum safety value, we compute an optimal path using a Potential field path planning [30]. The potential field method generates a smooth shortest path to the target position in real time and can be dynamically updated according to environmental changes, so it is good for path planning and navigation in the elevator. Its main disadvantage is a local minima problem. However it does not occur in the elevator environment, because the destination position is always selected from the empty cells of OGM. The robot's final path for getting on the elevator is shown in Fig. 15(b). The potential field method is described in detail in [30].

## 6. EXPERIMENTAL RESULTS

### 6.1. System overview

The proposed method was implemented on our mobile robot, MORIS, which was a ActiveMedia Pioneer DX2 equipped with a CruzCore R1001E gyroscope, a HOKUYO URG-04LX laser range finder and a BUMBLEBEE stereo camera (Table 3).

The CruzCore R1001E, a MEMS digital gyroscope, measured heading angles to improve the accuracy of the deadreckoning (Table 4).

MORIS senses the environment with a laser range finder which detects distance to obstacles in a 180 degree angular range out to 4 meters. (Originally, URG-04LX could detect a 240 degree angular range, but it is limited to 180 degree because of the equipment stand on the robot.) (Table 5). The BUMBLEBEE stereo camera mounted on a pan-tilt module is used for recognizing

Table 3. Pioneer DX2 mobile robot used in the experiments.


| MORIS (MOBILE Robot for Intelligent Service)  |          |  |
|---|----------|--|
|  | Platform | • Pioneer DX2<br>• Pan-tilt module   |
|   | Sensors  | • HOKUYO URG-04LX laser range finder<br>• Bumblebee stereo camera<br>• CruzCore R1001E gyroscope |
|   | CPU      | • Pentium IV 1.86 GHz  |

Table 4. Specification of CruzCore R1001E.

|  |   |
|--|---|
| Input range                                | $\pm 100^\circ/\text{sec}$ (Continuous)<br>$\pm 150^\circ/\text{sec}$ (Instantaneous) |
| Rate Noise<br>( $1\sigma$ @50Hz bandwidth) | $< 0.1^\circ/\text{sec}$  |
| Bandwidth                                  | 10Hz ~ 50Hz   |
| Bias drift                                 | $10^\circ/\text{hr}$  |
| Angle resolution                           | $0.36^\circ$  |
| Weight                                     | Approx. 5g  |
| External dimension                         | 25 x 20 x 3 mm  |

Table 5. Specification of URG-04LX.

|                    |   |
|--------------------|---|
| Detection Distance | 0.02 ~ 4m   |
| Accuracy           | Distance 0.02 ~ 1m : $\pm 0.01\text{m}$<br>Distance 1 ~ 4m : $\pm 1\%$ of measurement |
| Resolution         | 1mm   |
| Scanning angle     | $240^\circ$   |
| Angle resolution   | $0.36^\circ$  |
| Scanning time      | 100 msec/scan   |
| Weight             | Approx. 160g  |
| External dimension | 50x50x70mm  |

objects and computing their 3D positions. The experiment was carried out in several simulated and real environments (the hallways of the Pohang Institute of Intelligent Robotics (PIRO) and Pohang City Hall).

### 6.2. Recognition

For experiments, we prepared all possible scene images that might appear in the real environment. Scene images were obtained from a moving camera (i.e., a stereo camera mounted on the pan-tilt module), so these images are not static but dynamic in terms of intensity variation, especially the images obtained from the inside of the elevator. For each target object, we chose 100 representative test images to evaluate our proposed methods. The NN performance and recognition result are listed in Table 6.

True-True (T-T) is the number of correct identifications of target objects (i.e., true positive), while False-True (F-T) is a false identification, or false positive. If the robot fails to catch a target object (i.e., Miss) in the current image, it still has opportunity to recognize a target object from subsequent images. However, when identifying a non-target object as a target (i.e., false positive), the robot may, for example, get off the elevator at the wrong floor. To prevent this undesirable situation, we designed the NN with careful selection of the initial training set to reduce false positives. NN performance is measured by the following equation:

$$p = \frac{\text{number of correct classification}}{\text{total number of test pattern(image)}} \times 100. \quad (47)$$

Table 6. The recognition statistics.

|                           | Elevator call button | Elevator moving direction | Elevator floor number | Elevator control button |
|---------------------------|----------------------|---------------------------|-----------------------|-------------------------|
| True-True                 | 84                   | 89                        | 96                    | 97                      |
| False-True                | 0                    | 0                         | 1                     | 0                       |
| Miss*                     | 16                   | 11                        | 4                     | 3                       |
| NN performance            | (-)                  | 89.4                      | 95.9                  | 97.5 (80.4)**           |
| Avg. processing time(sec) | 0.069                | 0.035                     | 0.247                 | 0.115                   |

(-) = Non-applicable; we do not use neural network for call button recognition.

\*Miss = Recognition failure where a target object is not identified, even though it existed in image.

\*\*() = Performance of non-retrained NN.

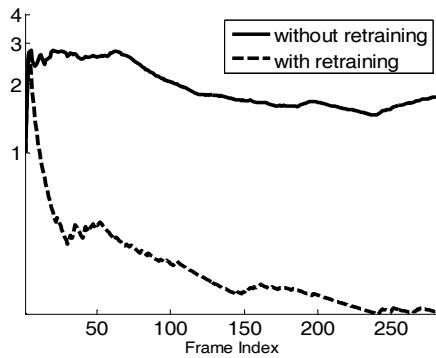


Fig. 16. Performance comparison between non-retrained and retrained cases.

If a certain pattern was classified as an object, however, the template matching stage may reject this pattern due to its correlation value with the best matching template. Thus, the NN classifier just reduced the number of candidates for the template matching stage. The average processing time of recognition was acceptable for real-time application, though one case (i.e., the floor number recognition) took a relatively long time. This was the case where repetitive thresholding of  $640 \times 480$  images was used to segment the floor number region, implying that many candidates were tested by common procedures. However, it still could not affect the system performance because at least three images could be processed per second.

For a special case of using graph partitioning and on-line retrainable NN, we also compared the performance of each retrained and non-retrained case. The performance was measured by the number of candidates which are rejected by graph partitioning per image. A lower value indicates that the NN classifies well enough to simplify the role of graph partitioning. Without retraining, this value is almost constant; however, it decreased with increasing frame index when retraining was performed during processing (Fig. 16). This graph definitely shows the effectiveness of retraining.

### 6.3. Navigation

Suppose that three obstacles exist in the right region of the elevator (Fig. 17(a)). Before getting on the elevator, since the robot cannot see the obstacles in the elevator, the obstacles are not represented in the occupancy grid map (Fig. 17(b)-(d)). Only once door opens can the robot detect the obstacles in the elevator, which are then shown in the occupancy grid map of the elevator (Fig. 17(e) and (f)). In case of Fig. 17, the robot turns left as soon as it passes the elevator door because of the right hand obstacles.

In the PIRO building, the elevator walls are made of glass (Fig. 18), so the laser scanner mounted on the robot cannot detect the elevator wall properly. Thus, the robot first found the display panel mounted on the elevator door after approaching door. The display panel could be found by edge detection and connected component labeling with size filtering (Fig. 18(a)). Then the position of the ECB was estimated from the position of door and

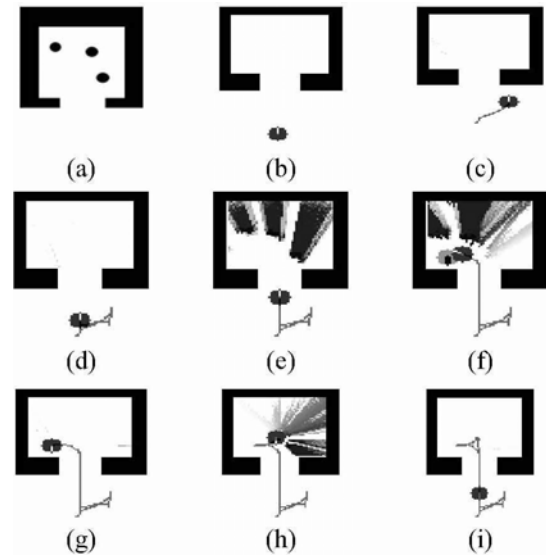
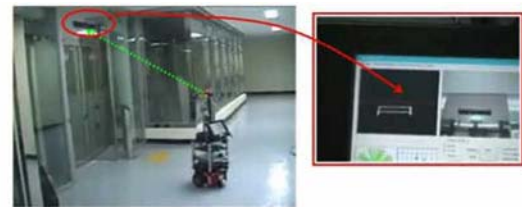


Fig. 17. Navigation procedures for getting on/off the elevator.



(a) Approaching the elevator and recognizing the position of the elevator door.



(b) Recognizing the elevator call button.



(c) Recognizing the elevator moving direction.



(d) Getting on the elevator when the door open.

Fig. 18. Real experiments in the PIRO building (from approaching the door to getting on the elevator).





(a) Recognizing the elevator control button.



(b) Recognizing the position of the elevator.



(c) Getting off the elevator. (d) Moving toward target position.



Fig. 19. Real experiments in the PIRO building (from recognizing the elevator control button to getting off the elevator).

the robot approaches to the ECB to recognize it (Fig. 18(b)). After finding the ECB, the robot found the display panel again and started to check the EMD (Fig. 18(c)). If the EMD coincided with the target direction and the elevator door opened, the robot got into the elevator (Fig. 18(d)). Inside the elevator, the robot approached the ECB on the left side because of person in the elevator and recognized the ECB (Fig. 19(a)). Then, it returned to the center of the elevator and checked the current position of the elevator (Fig. 19(b)). When the robot arrived at the target floor, it got off the elevator (Fig. 19(c)). At this time, the behavior “*GetOnOff\_Elevator*” was completed, and the behavior “*NavigationToTarget*” was activated again (Fig. 19(d)).

## 7. CONCLUSION

In this paper, we proposed a set of behaviors and algorithms which allow the robot to move to another floor by an elevator. In order to use the elevator, the robot needs two different abilities: recognition and navigation. In recognition, our robot could perceive a state of the elevator and a location of the buttons using an algorithm based on an adaptive threshold and an artificial NN. This way, our robot could decide when to get on/off the elevator. Even when the robot’s working environment changed, our robot could still get on/off the new elevator by changing the templates and retraining the NN for this new environment. For navigation, we constructed an OGM of the inside of the elevator in real time. This map was used to compute the safety

evaluation function for planning the path within the elevator and find suitable subgoal positions corresponding to each motion phase. Consequently, our robot could get on/off the elevator without any collision.

Our robot followed a preplanned sequence of motions for getting on/off the elevator, so our experiments were executed in a kind of robot friendly environments (less than two people in the elevator, a prior-knowledge map of the working elevator, etc.). However, we tried to consider every possible situation that the robot may encounter while getting on/off the elevator in the Petri net based control architecture of the robot. So our robot did not always successfully to get on/off the elevator. Sometimes, the robot failed to get on the elevator when unplanned situation occurred like path blockage, preoccupation of the target position or a crowded elevator. However, in most cases, our mobile robot worked under control of the behavior “*GetOnOff\_Elevator*”. For example, if the position error of the robot was more than 2m because of localization error in the navigation process, the robot could not approach to the elevator correctly. So the robot tried to find the elevator and reported “failure to approaching door” after a certain searching time. In this case, although the robot failed to navigate to the elevator, he still remained under control of the behavior “*GetOnOff\_Elevator*” and this kind of fail are in the robot navigation could be recovered by an additional error recover process.

Our robot could recognize the elevator buttons but could not push them, because it was not currently equipped with an arm. So, our robot now simply asked someone to press the button. In the future, we might consider adding an arm to the robot for complete autonomy.

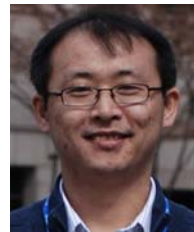
## REFERENCES

- [1] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, and D. Fox. “MINERVA: A second-generation museum tour-guide robot,” *Proc. of Int. Conf. on Robot. and Autom.*, pp. 1999-2005, 1999.
- [2] R. Simmons, J. Fernandez, R. Goodwin, S. Koenig, and J. O’Sullivan, “Xavier: an autonomous mobile robot on the web,” *Robotics and Automation Magazine*, 1999.
- [3] R. Simmons and S. Koenig, “Probabilistic robot navigation in partially observable environments,” *Proc. of the Int. Joint Conference on Artificial Intelligence*, pp. 1080-1087, 1995.
- [4] R. Bischoff and V. Graefe, “HERMES - a versatile personal robotic assistant,” *Proc. of the IEEE*, vol. 92, no. 11, pp. 1759-1779, 2004.
- [5] N. Y. Ko and R. Simmons, “The lane-curvature method for local obstacle avoidance,” *Proc. of Int. Conf. on Intelligent Robots and Systems*, pp. 1615-1621, 1998.
- [6] R. Siegwart *et al.*, “Robox at Expo.02: A large-scale installation of personal robots,” *Robot. Auton. Syst.* vol. 42, pp. 203-222, 2003.
- [7] I. R. Nourbakhsh, R. Powers, and S. Birchfield,

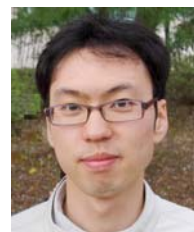
- "Dervish: an office-navigating robot," *AI Magazine*, vol. 16, no. 2, pp. 53-60, 1995.
- [8] W. Chung, G. Kim, and M. Kim, "Development of the multi-functional indoor service robot PSR systems," *Auton. Robot.*, vol. 22, pp. 1-17, 2007.
- [9] R. Simmons *et al.*, "Grace: an autonomous robot for the AAAI robot Challenge," *AAAI Magazine*, vol. 42, no. 2, pp. 51-72, 2003.
- [10] N. Tschichold-Gürman, S. J. Vestli, and G. Schweitzer, "The service robot MOPS: first operating experiences," *Robot. Auton. Syst.*, vol. 34, pp. 165-173, 2001.
- [11] Y. Shimosasa *et al.*, "Security service system using autonomous mobile robot," *Proc. of IEEE Int. Conf. Sys., Man, and Cyber.*, pp. 825-829, 1999.
- [12] F. Capezio *et al.*, "Mobile robots and intelligent environments," *Lect. Notes in Comput. Sci.*, vol. 4733, pp. 781-788, 2007.
- [13] K. Sakai, T. Asada, Y. Yasukawa, and Y. Murase, "Developing a service robot with communication abilities," *Proc. of IEEE Int. Workshop on Robotics and Human Interactive Comm.*, pp. 91-96, 2005.
- [14] J. Miura, K. Iwase, and Y. Shirai, "Interactive teaching of a mobile robot," *Proc. of IEEE Int. Conf. on Robot. and Autom.*, pp. 3378-3383, 2005.
- [15] S.-Y. An, J.-G. Kang, W.-S. Choi, and S.-Y. Oh, "A neural network based Retractable framework for robust object recognition with application to mobile robotics," *Appl. Intell.*, Published online, 26 February 2010.
- [16] F. H. Y. Chan, F. K. Lam, and H. Zhu, "Adaptive thresholding by variational method," *IEEE Trans. Image Processing*, vol. 7, no. 3, pp. 468-473, 1998.
- [17] L. D. Stefano and A. Bulgarelli, "A simple and efficient connected components labeling algorithm," *Proc. of Int. Conf. on Image Analysis and Processing*, pp. 322-327, 1999.
- [18] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23-38, 1998.
- [19] R. Reed, "Pruning algorithms-A survey," *IEEE Trans Neural Net*, vol. 4, no. 5, pp. 740-747, 1993.
- [20] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice-Hall, New Jersey, 1999.
- [21] K. Briechele and U. D. Hanebeck, "Template matching using fast normalized cross correlation," *Proc. of SPIE*, vol. 4387, pp. 95-102, 2001.
- [22] D. B. West, *Introduction to Graph Theory*, 2nd edition, Prentice-Hall, 2001.
- [23] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46-57, 1989.
- [24] S. Kim and S.-Y. Oh, "SLAM in indoor environments using omni-directional vertical and horizontal line features," *Journal of Intelligent & Robotic Systems*, vol. 17, no. 2, pp. 161-173, 2008.
- [25] S. Thrun, W. Burgard, and D. Fox, *Probabilistic ROBOTICS*, MIT Press, Cambridge, 2005.
- [26] H. Chung, L. Ojeda, and J. Borenstein, "Accurate

mobile robot dead-reckoning with a precision-calibrated fiber optic gyroscope," *IEEE Trans. on Robotics and Automation*, vol. 17, no. 1, pp. 80-84, 2001.

- [27] G. A. Borges and M. Aldon, "Line extraction in 2D range images for mobile robotics," *Journal of Intelligent & Robotic Systems*, vol. 40, pp. 267-297, 2004.
- [28] V. Nguyen *et al.*, "A comparison of line extraction algorithms using 2D range data for indoor mobile robotics," *Autonomous Robots*, vol. 23, pp. 97-111, 2007.
- [29] A. Garulli *et al.*, "Mobile robot SLAM for line-based environment representation," *Proc. of IEEE Conf. Decision and Control and European Control Conference*, pp. 2041-2046, 2005.
- [30] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Proc. IEEE Int. Conf. on Robot. and Autom.*, pp. 500-505, 1985.



**Jeong-Gwan Kang** received his B.S. degree in KyungPook University, Daegu, Korea in 2002 and his M.S. degree in Electrical Engineering from Pohang University of Science and Technology (POSTECH), Pohang, Korea in 2004. He is currently working toward a Ph.D. degree in the Department of Electrical Engineering, POSTECH. His research interests include soft computing technology (neural networks, fuzzy logic, and evolutionary computation) and its application to robotics.



**Su-Yong An** received his B.S. degree in POSTECH in 2006. He is currently working toward a Ph.D. degree in the Department of Electrical Engineering, POSTECH. His research interests include vision based SLAM and autonomous exploration of mobile robot.



**Won-Seok Choi** received his B.S. and M.S. degrees in POSTECH in 2005 and 2007, respectively. He is currently working toward a Ph.D. degree in the Department of Electrical Engineering, POSTECH. His research interests include vision based SLAM and intelligent vehicle.



**Se-Young Oh** received his B.S. degree in Electronics from Seoul National University, Seoul, Korea, and his M.S. and Ph.D. degrees in Electrical Engineering from Case Western Reserve University, Cleveland, Ohio, USA in 1974, 1978 and 1981, respectively. He is currently a professor at POSTECH, Korea. His research interests include soft computing technology (neural networks, fuzzy logic, and evolutionary computation) and its application to robotics, face detection and recognition, and intelligent vehicles. He is a senior member of IEEE.