Appl Intell (2011) 35:190–210
DOI 10.1007/s10489-010-0212-9

A neural network based retrainable framework for robust object recognition with application to mobile robotics

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

Published online: 26 February 2010

Abstract In this paper, we address object recognition for a mobile robot which is deployed in a multistory building. To move to another floor, a mobile robot should recognize various objects related to an elevator, e.g., elevator control, call buttons, and LED displays. To this end, we propose a neural network based retrainable framework for object recognition, which consists of four components—preprocessing, binary classification, object identification, and outlier rejection. The binary classifier, a key component of our system, is a neural network that can be retrained, the motivation of which is to adapt to varying environments, especially with illuminations. Without incurring any extra process to prepare new training samples for retraining, they are freely obtained as a result of the outlier rejection component, being extracted on-line. To realize a practical system, we adopt a parallel architecture integrating both recognition and retraining processes for seamless object recognition, and furthermore detect and cope with the deterioration of a retrained neural network to ensure high reliability. We demonstrate the positive effect of retraining on the object recognition performance by conducting experiments over hundreds of images obtained in daytime and nighttime.

S.-Y. An · J.-G. Kang · W.-S. Choi · S.-Y. Oh (✉)
Intelligent Robotics Lab, LG-302, Dept. of Electronic and Electrical Eng., Pohang University of Science and Technology, Pohang, Gyungbuk 790-784, South Korea
e-mail: syoh@postech.ac.kr

S.-Y. An
e-mail: grasshop@postech.ac.kr

J.-G. Kang
e-mail: naroo1@postech.ac.kr

W.-S. Choi
e-mail: onlydol@postech.ac.kr

Keywords Object recognition · Neural network · Graph partitioning · On-line retraining

1 Introduction

Mobile robots have been successfully employed in various applications including delivery services, tour guidance, and security services. In some cases, robots have been developed to assist partly motion-impaired users with the transportation of light objects in an office environment [1]; to guide tourists in a museum [2] and to interact and communicate with staffs and visitors in an exhibition area [3]. Unfortunately, most of these robots could navigate in a single story building successfully but extension to more realistic multistory buildings was not under consideration, bringing about a serious limitation in mobility. To navigate in a multistory environment, the robot should be able to exploit an elevator [4, 5]. From a mobile robot's point of view, elevator access is a very challenging task which has not yet been completely resolved. At the outset, a preliminary question which must be considered is an ability to recognize the status of an elevator, such as moving up or down.

Objects related to an elevator are the elevator call/control buttons, its moving direction indicator, the floor index indicator, all of which are assumed to lie on a planar surface. This assumption will turn the problem into a 2D object recognition. The scale invariant feature transform (SIFT) based object recognition has been accepted as the state-of-the art, showing an outstanding performance in recognizing objects even in circumstances where a partial occlusion or illumination change occurs [29, 30]. Since, however, it is based on extraction of dense keypoints, it suffers from a long processing time and actually ends up with few keypoints around the target object when applied to our problem. In addition, we need to devise a recognition scheme

that includes a robust outlier rejection method for reducing erroneous recognition as well as an adaptation mechanism for dealing with environmental changes. The outlier rejection is a basic problem in data mining literature, with a great deal of research on it [26]. Methods that have been proposed so far include a distance-based outlier rejection that is based on the distance of a point from its kth nearest neighbor [6], a density-based clustering method, which defines a local outlier factor for each object in a dataset to indicate its degree of outlier-ness [7], and the method of applying graph partitioning to identify the sets of outliers [8].

The neural network has been widely used in various recognition or classification problems. Rowley et al. used a neural network for detecting an upright frontal face in a static image [18]. Er et al. introduced a radial basis function neural network classifier to cope with face recognition [16]. To recognize the electrocardiographic (ECG) beat, Osowski et al. implemented a fuzzy hybrid neural network which consists of self-organizing subnetworks connected in cascade with the multilayer perceptron [15]. Moreover, to cope successfully with environmental changes in real time, a neural network that can be automatically adapted to the current environment has been proposed. An on-line retraining neural network was utilized to separate the foreground from the background of nonstationary image sequences [9], to classify multimodal feature-based emotion patterns into several representative emotions [10], and to detect and recognize malignant regions in colonoscopy video sequences [17].

The advantages of a neural network naturally come from its massively parallel distributed structure and its generalization ability by learning. Those advantages lie in adaptability, fault tolerance, dealing with certainty and contextual information, and model-free estimation by constructing input-output mapping [13]. Therefore, it can continuously provide intelligent perception to a mobile robot that navigates in varying environments. This paper presents an on-line retraining neural network and its performance improvement while dealing with the following problems: how to reorganize the new training set, how to determine when to retrain it, how to evaluate its efficacy, and finally how to achieve parallel execution of object recognition and retraining.

With this in mind, we introduce an on-line retraining neural network based framework to recognize objects related to an elevator. More specifically, we proposed a sequential framework, named as the Cascaded Processing Elements (CPEs). The CPEs include: preprocessing, binary classification, object identification, and outlier rejection. The first step of preprocessing extracts object candidates from an input image regardless of varying illumination conditions using an adaptive thresholding and a size filtering technique. The binary classifier, as a key component of the proposed method, is implemented with a neural network whose weights can adapt to changing environments. Then, object identification is accomplished by template matching, giving an identified index to each candidate. In the last step, graph partitioning in conjunction with a priori knowledge is utilized to reject outliers which were not removed by the preceding three steps. In brief, the main contribution of this paper is to provide a methodology for the relearning of a neural network to solve practical problems, and as its application we present a robust recognition system for mobile robots which adapt to varying environments through an on-line retrainable neural network.

This paper is organized as follows. The overall system description is given in Sect. 2, and Sect. 3 addresses the detailed procedures of object recognition. We consider four main problems in achieving on-line retraining in Sect. 4. Experimental results by the proposed method are demonstrated over daytime and nighttime images in Sect. 5.

2 System overview

The system configuration is divided into two parts. The left hand side of Fig. 1a is related to the system hardware which contains the mobile robot, a stereo camera module, and a pan-tilt module (Fig. 1b). The mobile robot provides the current position information for calculating a 3D position of the target object. The pan-tilt module is mounted on the head of the robot and controls stereo camera to look in arbitrary directions. The camera module captures stereo images, which are then analyzed by the CPEs so as to extract a 3D position of the target object.

2.1 Cascaded processing elements

The right hand side of Fig. 1a show the main structure of the proposed algorithm. The target object to recognize depends on the robot location. If it is in an elevator, then the target object may be an elevator control button or a floor number. As such, the Object Template Storage selects one of the relevant target templates when the robot location is given.

The CPEs consist of four components and two external image storages, defined as the Object Image Queue (ObjIQ) and the Outlier Image Queue (OutIQ) (Fig. 2). Each component has its unique function. In the preprocessing component, an adaptive thresholding technique is applied to the whole region of the input image to extract object candidates under varying illumination conditions. Then, the several extracted object candidates go through size filtering, using its geometric constraints. Next, a neural network that is retrainable during the recognition process, as a second component, can solve the binary classification problem, while reducing the number of object candidates by removing the ambiguous ones. Through the third component of template matching, each candidate is identified as one of the objects in the

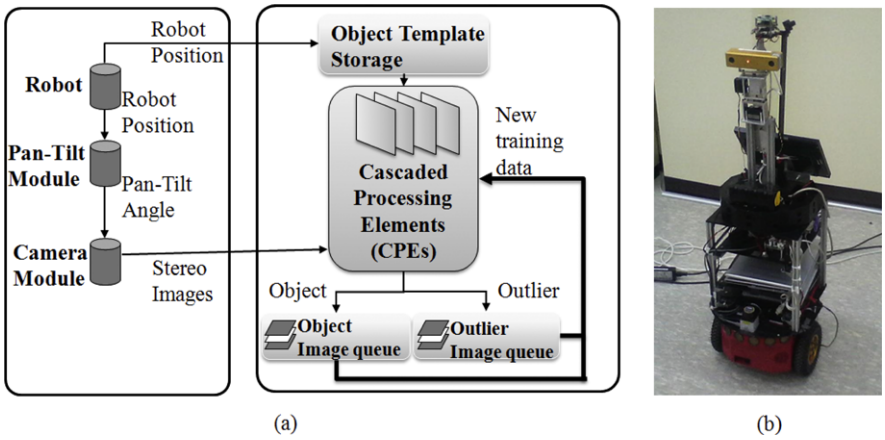

Fig. 1 Overall system configuration. (**a**) The system hardware and the simplified structure of the proposed method. (**b**) Our mobile robot

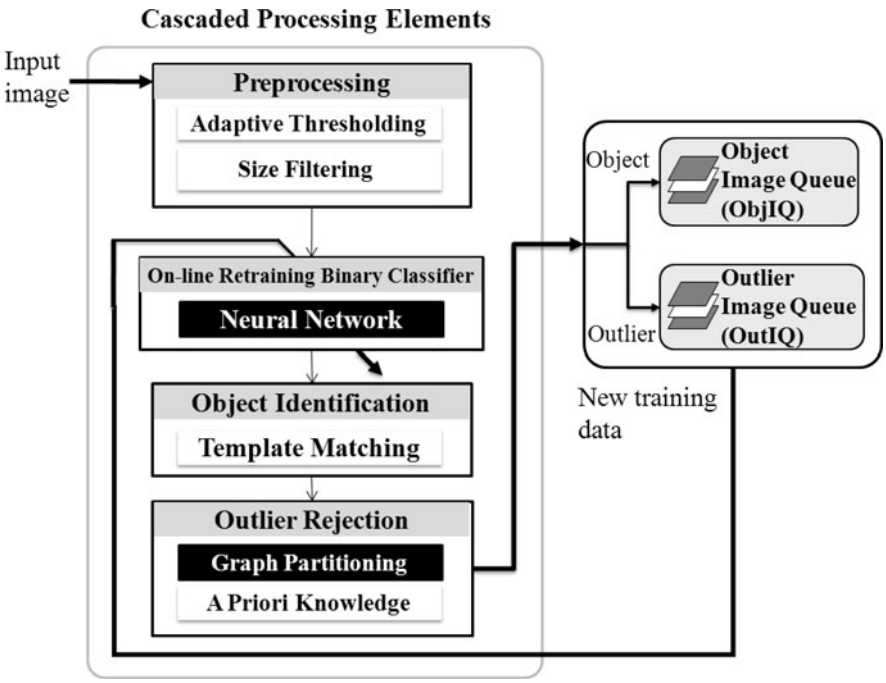

Fig. 2 Structure of the CPEs and two image queues. The components in *black boxes* are major processing units that allow the system to adapt to varying illumination condition

Object Template Storage with a proper template index. Finally, the outlier rejection component, using graph partitioning and a priori knowledge, further removes falsely identified objects from the candidate set by considering their relationships based on a model. Here the outlier is defined as a candidate which can pass through the neural network but is excluded by the outlier rejection component.

Through the CPEs, an ambiguous object is rejected sequentially whenever it goes through each component. Moreover, the rejected candidates are not just discarded but reused as negative training data to retrain the neural network in the second component; thus this retraining ability guarantees self-adaptability of the system. In our configuration, the CPEs have four components, yet more components can be added to reinforce rejection of ambiguous candidates. For instance, another classifier can be added in parallel with or after the neural network. Utilization of many components of the CPEs can lessen false positives, i.e., recognizing a non-target as a target object. On the other hand, false negatives, i.e., recognizing a target object as a non-target, may increase. This brings forth the problem of how many components of the CPEs to utilize, but it is beyond the scope of this study.

In summary, the proposed method is characterized by its self-adaptability and flexibility in structure; both are the core features for practical applications. In this paper, we implement the CPEs with four components and their primary roles will be discussed in more detail in Sect. 3.

2.2 Two image queues

There are two kinds of image queues, ObjIQ and OutIQ, which serve as data storage for training samples in our proposed system (Fig. 2). Once the objects and outliers are dis-

tinguished by the CPEs, then each object or outlier is stored in the corresponding image queue. The data storage mechanism of two image queues is analogous to First In First Out; whenever an outlier is detected, it is inserted to OutIQ as the form of an image patch by the push operation, whereas the oldest outlier is automatically ejected from OutIQ by the pop operation. This series of processes is also applied to ObjIQ in a similar way. Note that the size of each image queue is kept constant during the recognition process; however, the size of two image queues need not be the same. A number of image patches in OutIQ, used as negative training data, play a dominant role in suppressing the occurrence of outliers; hence the size of OutIQ, N_{out} is usually set to be larger than that of ObjIQ, N_{obj}, because reducing the occurrence of outliers is of vital importance when dealing with object recognition in a visually dynamic environment.

The existence of the two image queues is related to the retraining of the neural network, which is triggered whenever the current environment or its illumination condition is decided to be changed. In the retraining phase, two image queues perform important roles such as offering the new training data and determining when to start retraining. By retraining the neural network with the help of the two image queues, adaptability of the proposed method is guaranteed regardless of the environmental changes.

3 Object recognition

In this section, we address potential problems that are likely to occur when we use the SIFT based approach, one of the most widely used in object recognition, as a tool for solving our problem. Then, we explain the proposed method step by step including detailed preprocessing procedures for extracting object candidates, how the neural network can remove false candidates generated in the previous preprocessing step, how to identify object candidates, and how to reject outliers.

3.1 SIFT-based object recognition and its limitations

In the SIFT-based object recognition, A large number of features, or keypoints, are extracted from an image, and then the target object is recognized by matching these extracted keypoints with the registered keypoints in the robot's object database. This method has several advantages such as scale and rotation invariance, robustness to occlusion, affine distortion, and partial changes in illumination [29].

In spite of these advantages, use of the SIFT-based approach for recognition of our target objects encounters several problems:

- This method assumes that an image of a typical object has dozens of SIFT keypoints, but in our case, the target object is displayed as a tiny object in an input image (nearly 5% of the input image size). Thus, it may be hard to extract enough keypoints around the target object (Fig. 3a). We could adjust parameters so as to extract dense keypoints, but then, the processing time may increase, which is a failure seen from a real time processing point of view (Fig. 3b).
- When an elevator control button, as a target object, is pressed, the overall pixel intensities of the button change

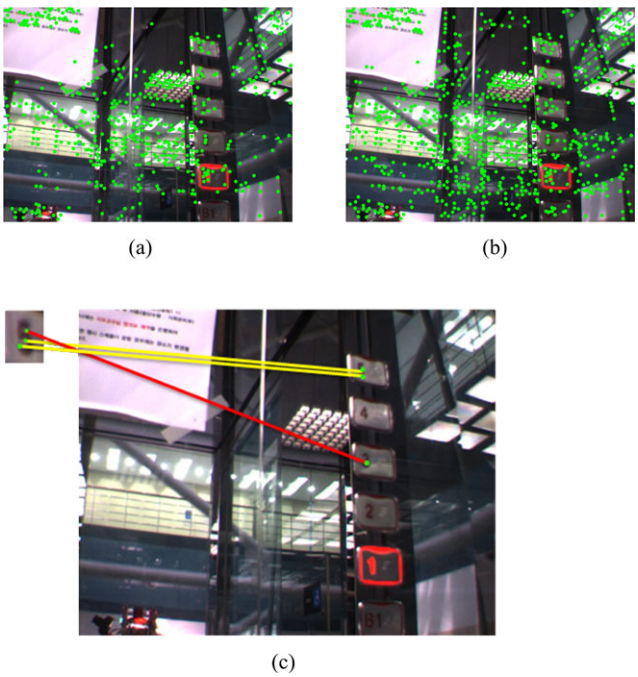

Fig. 3 (Color online) Extraction of SIFT keypoints by adjusting a contrast threshold parameter. The *green dots* represent extracted keypoints and *yellow* and *red lines* connect matched keypoints. (**a**) Contrast threshold 0.04; 623 keypoints are extracted from 640 × 480 image and the processing time is 2.15 s. (**b**) Contrast threshold 0.01; 930 keypoints are extracted and the processing time is 3.03 s. (**c**) There are two correct (*yellow line*) and one false (*red line*) keypoint matchings. The false matching happens in a similar object

significantly (Fig. 3). Therefore, multiple sets of keypoint descriptors may be necessary even for defining one target object.
- False keypoint matching may occur due to the high similarity between the target objects (Fig. 3c).
- From a visual scene perspective, the usual experimental environment is highly dynamic. However, there is no self-adaptation mechanism to cope with environmental changes such as illumination variations, even though SIFT feature is robust to partial illumination changes.

In the light of these problems, we might conclude that SIFT-based object recognition is not adequate for solving our problem and thus we need to find a different method.

3.2 Target object

Every time the robot is confronted with the task of an elevator, it has to recognize four kinds of target objects associated with the elevator. They may differ according to the current robot location or the task assigned to the robot as shown in Fig. 4.

The floor number offers the current elevator position, and the moving direction indicates whether an elevator is going up or down (Fig. 4a, b). These two objects show the moving state of the elevator. To call the elevator, the robot must recognize the up or down buttons (Fig. 4c). In order to move to the target floor, the robot must recognize and select an elevator control button to press within an elevator (Fig. 4d).

Identical CPEs except the outlier rejection are utilized to recognize all these objects. The graph partitioning step is used only to recognize the elevator control buttons, because it needs the relationship between the identified objects. Thus, the remainder of this paper deals with recognition of the elevator control buttons.

3.3 Preprocessing by adaptive thresholding and size filtering

The inside of the elevator where our experiment is to be carried out is a very reflective environment, surrounded by mirror-like surface (Fig. 5a). Furthermore, the lighting condition is constantly varying according to the vertical position of the elevator because the natural light penetrating transparent glass comes in directly through the inside of the elevator. Under such a varying illumination condition, an adaptive thresholding [11, 12] can be used to separate the possible object region from the background based on local pixel intensities. I_{source} is the input image and *param* is a parameter set related to the adaptive thresholding, which consists of the neighborhood size and the offset value subtracted from a local mean intensity to determine a threshold value. Figure 5b is the resulting binarized image after applying an adaptive thresholding, where the neighborhood size is nine and the offset is five, and a collection of white pixels represents possible object region. As can be seen in Fig. 5b, 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 that we face 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 [27], a group of white pixels can be defined as a single object candidate c_i. Each candidate c_i is assigned several attributes: area β_i (i.e., number of white pixels in the candidate), center position p_i in the image space, width w_i, height h_i, width-to-height ratio r_i of its bounding box, and size-normalized rectangular image patches $ImgO_i$ and $ImgB_i$ centered at p_i where the former is extracted from the input image and the latter is from the binarized image, respectively (Fig. 5d, e).

The number of candidates can be reduced by size filtering, which uses geometric constraints such as area and width-to-height ratio of a candidate. In other words, size filtering filters out candidates which do not satisfy the following constraints: β_i should be larger than a minimum area $\beta_{\min}$ and also r_i should be bounded between $r_{\min}$ and $r_{\max}$, which are the lower and upper bounds of the width-to-height ratio. All the preprocessing procedures are listed in Table 1. As a result of the above preprocessing, most false candidates were eliminated from Fig. 5b, whereas the target objects, or numbers in the control panel, still remained (Fig. 5c). These remaining candidates form the object candidate set C, which still includes a false positive. In the subsequent step, the neural network as a retrainable binary classifier can further filter out these remaining false candidates.

Fig. 4 Target objects. (**a**) Floor number display. (**b**) Moving direction display of the elevator. (**c**) Elevator call buttons. (**d**) Elevator control buttons

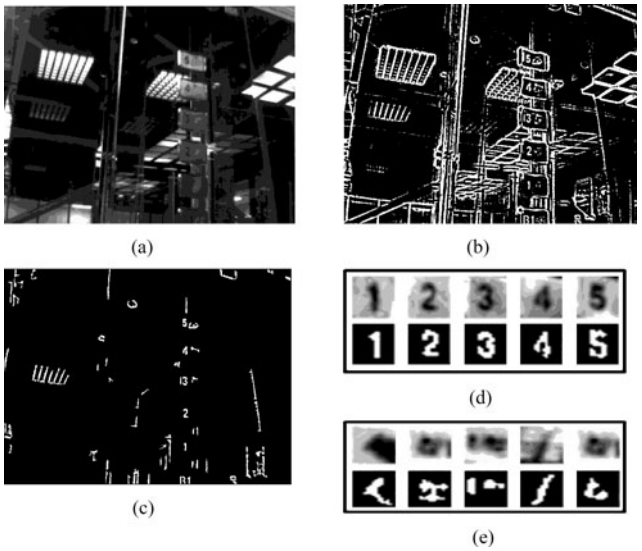

Fig. 5 Preprocessing. (**a**) A gray-scale input image. (**b**) After an adaptive thresholding. (**c**) After a connected component labeling and size filtering. (**d**) Example image patches of correct candidates. (**e**) Example image patches of false candidates. The *upper image patches* of (**d**)–(**e**) are extracted from the input image (**a**), followed by normalization in terms of size and pixel intensity. The *lower image patches* are from the binarized image (**b**)

Table 1 Preprocessing—extraction of object candidates

Preprocessing(I_{source})

1: $I_{binary} \leftarrow$ Adaptive thresholding ($I_{source}, param$)
2: $c_i = (\beta_i, p_i, w_i, h_i, r_i, ImgO_i, ImgB_i) \leftarrow$ Connected component labeling (I_{binary})
3: $n \leftarrow 0$
4: $C \leftarrow \emptyset$
//size filtering
5: **For** all detected candidates c_i **do**
6: **If** $\beta_i > \beta_{\min}$ and $r_{\min} < r_i < r_{\max}$ **then** $C \leftarrow C \cup \{c_i\}, n \leftarrow n + 1$
7: **Endfor**
8: $L \leftarrow n$
9: **return** object candidate set $C = \{c_1, c_2, \ldots, c_L\}$

3.4 Neural network as a binary classifier

Neural network based recognition and classification schemes have been widely investigated by researchers in a diverse range of research fields, such as pattern recognition [23, 24], image processing [22], and mobile robotics [19–21]. The key idea, adopting methods that are inspired by biological systems, is unchanged even though the motivation of research comes from different research areas. Although many types of the neural networks can be utilized in classification purposes [25], in our application, we choose a three-layer multilayer perceptron (MLP) that is most widely studied and used.

In this study, the neural network, as the second component of the CPEs, classifies each candidate c_i into an object or a non-object when an input image patch $Img\hat{O}_i$ of c_i is fed into it, where $Img\hat{O}_i$ is a histogram-equalized and size-normalized version of $ImgO_i$ (Fig. 6). Through histogram equalization, pixel intensities of each image patch $ImgO_i$ are mapped nonlinearly to other pixel intensities thereby expanding the range from 0 to 255. This compensates for differences in camera input gains, as well as improving contrast to some extent [18]. The width and height of $Img\hat{O}_i$ is also normalized to 15 by 20. This size normalization of the input image can reduce the number of connections between the input and the hidden layer.

Each image patch $Img\hat{O}_i$ is converted into a vector whose elements represent pixel intensities which are normalized from zero to one, and fed into the neural network. Each neuron in the input layer accepts corresponding normalized pixel intensity of $Img\hat{O}_i$, implying that the number of neurons in the input layer is identical to the size of $Img\hat{O}_i$; thus there are 300 neurons in the input layer.

To decide on a proper size of the network, many pruning techniques including the sensitivity method, penalty-term method, and pruning by a genetic algorithm have been proposed [28]. Our approach to choose the optimal network size, i.e., the number of neurons used in the hidden layer, is straightforward. We train a number of networks of various sizes and select the smallest one that shows a good general-

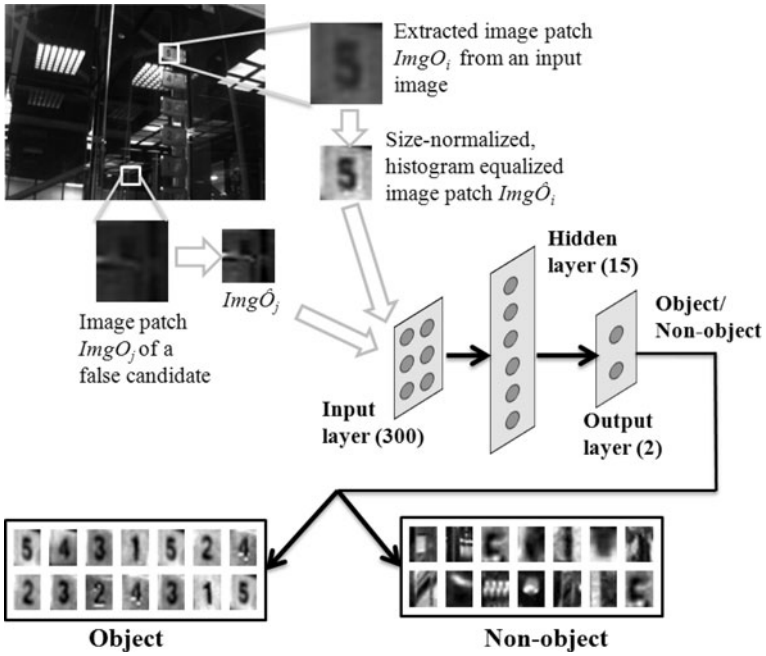

Fig. 6 A neural network used in a binary classification problem. Gray-scale image patch of each candidate is fed into the neural network so as to test whether it contains a number or not. The number in parentheses of each layer of the neural network is the number of neurons used

Table 2 Neural network as a binary classifier

Neural Network Classfier$(C = \{c_1, c_2, \ldots, c_L\}, \mathbf{w})$

1: **For** all object candidates $c_i = (\beta_i, p_i, w_i, h_i, r_i, ImgO_i, ImgB_i)$ **do**
2: Transform $ImgO_i$ into the input vector $\mathbf{x}$
3: $\mathbf{y} = [y_1 y_2]^T \leftarrow$ Neural network $(\mathbf{x})$
//removing non-object in the set C
4: **If** $y_2 > y_1$ **and** $y_2 > \theta_a$ **then** $C \leftarrow C - \{c_i\}$
5: **Endfor**
6: **return** object candidate set $C = \{c_1, c_2, \ldots, c_M\}$

ization performance; we found that utilizing 15 neurons in the hidden layer is 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 is fed to the network, we observe 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 corresponding activation value. If y_1 is activated more than y_2 and a predefined threshold θ_a, then the input image patch is considered as belonging to the object class. On the other hand, y_2 is more activated than y_1 and θ_a if belonging to the non-object class. In this manner, each input candidate is classified as object/non-object according to the distribution of these activations in the output layer (Fig. 6). Once a candidate is classified as a non-object, then it is removed from the object candidate set C (Table 2).

Although much work has been done on the topic of neural network training, backpropagation (BP) is most widely used training method [13]. Through BP training, the neural weights are adjusted in an iterative and gradient descent way, minimizing the overall error between the desired and actual outputs of the neural network:

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

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

where k is an iteration number within training, η is a learning rate, $\mathbf{e}$ is error at the output layer, and $\mathbf{w}$ is a collection of weights associated with all layers.

We have two classes of training samples: positive and negative samples. The positive samples are stored in ObjIQ in the form of a 15 by 20 gray scale image patch and the negative samples are stored in OutIQ in the same form, respectively. At the initial training step, where r is zero, these training samples are already provided to the system by manual selection, neural weights are then modified so as to learn these initial training samples. However, at the rth retraining phase, some initial training samples are not maintained

Table 3 Object identification by template matching

Object Identification($C = \{c_1, c_2, \ldots, c_M\}, T = \{tp_1, tp_2, \ldots, tp_5\}$)

1: **For** all object candidates $c_i = (\beta_i, p_i, w_i, h_i, r_i, ImgO_i, ImgB_i)$ **do**
2: $(cr_i, t_i) \leftarrow$ Template Matching (c_i, T)
//assign attributes
3: **If** $cr_i > \theta_c$ **then** add attributes (cr_i, t_i) to c_i
4: **Else** $C \leftarrow C - \{c_i\}$
5: **Endfor**
6: **return** object candidate set $C = \{c_1, c_2, \ldots, c_N\}$

but removed sequentially from the two image queues, giving room to newly acquired training samples. The concept as well as the detailed procedures of retraining will be discussed in Sect. 4.

3.5 Object identification by template matching

The camera viewpoint is related to the robot's pose in the elevator; moreover, we can assume that the robot can locate in front of the control panel within a reasonable bound. Under this assumption, we consider a skew and a slant factor small in each candidate image, so we chose to use a simple template matching to identify a target. There are five binarized templates corresponding to numbers from one to five; each template has a fixed size of 10 by 14. Although the size of a query image $ImgB_i$ is varying depending on the preprocessing results, it is normalized to 15 by 20; thus the template search space is limited to 6 by 7. Therefore, the processing time of template matching can be considerably reduced, and normalization of the query image allows template matching to be scale-invariant as well.

The most commonly used similarity measure is the normalized cross-correlation (NCC):

$$R_{ij}(x, y) = \frac{\sum_{x'} \sum_{y'} T_j(x', y') \cdot I_i(x + x', y + y')}{\sqrt{\sum_{x'} \sum_{y'} T_j(x', y')^2} \sqrt{\sum_{x'} \sum_{y'} I_i(x + x', y + y')^2}}, \quad (3)$$

where I_i is the ith query image, T_j is the jth template, and R_{ij} is the calculated correlation coefficient, respectively. In general, we can find several local maxima in the correlation surface, but owing to its limited size of 6 by 7, the assumption that the correlation surface is unimodal can be justified; so, only one peak with its correlation coefficient exists in a pair of the query image and a template.

Since we have five templates, multiple template matching processes are done sequentially to identify a query image. In other words, identification of an object candidate c_i using its query image $ImgB_i$ is accomplished by giving it a template index t_i which has a maximum value among five correlation coefficients. However, if the maximum correlation coefficient is lower than a predefined threshold value θ_c, then the object candidate c_i is discarded from the object candidate set C. Otherwise, the maximum correlation value cr_i and the template index t_i are included into c_i as additional attributes (Table 3).

3.6 Outlier rejection by graph partitioning and a priori knowledge

We have N number of candidates represented by a set $C = \{c_i = (\beta_i, p_i, w_i, h_i, r_i, ImgO_i, ImgB_i, cr_i, t_i) \mid i = 1, \ldots, N\}$. Until now, the rejection criterion is purely dependent on the attributes of each candidate, especially its segmented image patch, area, and width-to-height ratio. In other words, the relationship between the identified candidates is not taken into account in removing falsely identified objects from the candidate set C. Since the elevator control buttons are aligned vertically in the control panel, we can detect these buttons together. We modeled the elevator control panel as in Fig. 7a, where d is a constant proportional to the distance between two adjacent buttons, which will be used as the scale factor for estimating a real distance in image coordinates. There are five buttons, each of which contains a corresponding number in it, arranged in descending order. With this control panel model and a graph representation of the object candidates, we can further eliminate falsely identified candidates, also called outliers, from the set C.

3.6.1 Outlier rejection by graph partitioning

Usually a graph G is expressed by a set of vertices V and edges E connecting them. 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, in the graph space by the control panel model shown in Fig. 7b. The positions of the vertices mapped to the graph space have no relevance to those of the object candidates in the image space; however, their relationship is closely associated with an edge. Under the assumption that the transformed graph is a simple graph [14], a pair of any two vertices cannot hold more than one edge;

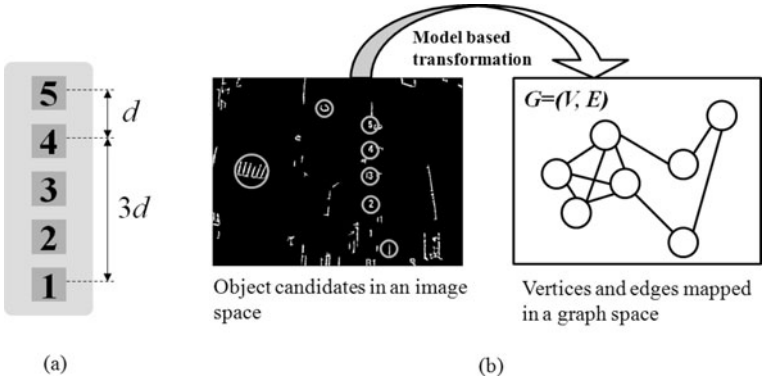

Fig. 7 Model based transformation from an image space to a graph space. (**a**) Elevator control panel model. (**b**) Graph representation of candidates by model based transformation. In the image space, seven candidates which are denoted by *white circles* are mapped to seven vertices one by one

thus if there are N vertices in a simple graph, then it has $N(N-1)/2$ edges at most.

In our application to outlier rejection, all vertices are connected with each other via edges, which have different connection weights. The connection weight a_{ij} between the ith and the jth vertices is calculated based on the control panel model as follows:

$$ED(p_i, p_j) = d_{ij}^O, \tag{4}$$

$$d(h_i + h_j)|t_i - t_j|/2 = d_{ij}^M, \tag{5}$$

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

where ED is a measure of the Euclidean distance, 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. In (4), a Gaussian function with zero mean and unity standard deviation is used to convert disparity between d_{ij}^O and d_{ij}^M into the connection weight. Therefore, if two candidates are matched well with the model, 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. In case of a simple graph in Fig. 8, the adjacency matrix is as follows:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}, \tag{7}$$

where a_{ij} is given by

$$a_{ij} = \begin{cases} 1 & \text{if } i \neq j,\ c_i \text{ and } c_j \text{ are connected,} \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

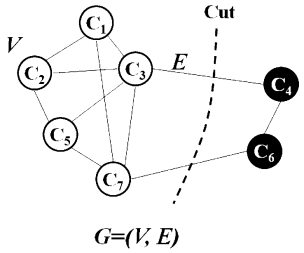

Fig. 8 Graph partitioning. The vertices having low connection weights, i.e., c_4 and c_6, are considered as outliers. Thus the graph G is partitioned into two parts by *dashed line*

However, in our case, an element of $\mathbf{A}$ is not binary value but continuous as defined in (6).

An outlier in a graph is defined as the vertex which has a relatively low connection weight; thus an average connection weight $m(G)$ of a graph G is maximized when such outliers are removed from the graph G. $m(G)$ is calculated as follows:

$$m(G) = \frac{\sum_i^N \sum_j^N a_{ij}}{N}, \tag{9}$$

where N is the number of vertices. Let us now define $\mathbf{u}$ as an indicator whose ith element u_i represents whether an object candidate c_i is an outlier or not:

$$u_i = \begin{cases} 0 & \text{if } c_i \text{ is an outlier,} \\ 1 & \text{otherwise.} \end{cases} \tag{10}$$

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

$$Y \subseteq V \quad \text{and} \quad F \subseteq E \cap (Y, 2), \tag{11}$$

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:

$$m(H) = \frac{\mathbf{u^T A u}}{\mathbf{u^T u}}, \tag{12}$$

where the numerator is a summation of all connection weights except the connection weights of outliers, 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 $m(H)$:

$$\mathbf{u}_{\max} = \arg\max_{\mathbf{u}} \frac{\mathbf{u^T A u}}{\mathbf{u^T u}}. \tag{13}$$

There are 2^N number of subgraphs, but it is tedious work to calculate all the $m(H)$ of 2^N subgraphs. Instead, we seek $\mathbf{u}_{\max}$ by differentiating (12) with respect to $\mathbf{u}$, and this gives:

$$\mathbf{Au} = m(H)\mathbf{u}. \tag{14}$$

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. Each element of $\mathbf{u}_{\max}$ indicates a degree of outlier-ness [7], i.e., small u_i implies a high outlier-ness. We convert each element of $\mathbf{u}_{\max}$ into a binary value to indicate it as either an outlier or not as follows:

$$u_i = \begin{cases} 1 & \text{if } u_i > \theta_u \\ 0 & \text{otherwise,} \end{cases} \tag{15}$$

where θ_u is a predefined threshold value. Now, we obtain the binarized indication vector $\mathbf{u}_{\max}$, which denotes a reduced candidate set $C = \{c_1, \ldots, c_P\}$, where P is the number of candidates in the set C. In the process of outlier rejection by graph partitioning, outliers are collected into the outlier set O.

3.6.2 Outlier rejection by a priori knowledge

Suppose that there are two candidates, c_i and c_j, whose template indexes t_i and t_j (respectively) are the same, then at least one of the two should be rejected as an outlier because only one floor number can exist in the set C; for instance, two or more buttons which correspond to 'floor 2' cannot appear in one control panel. So, we divide the set C into five subsets by defining $T_k = \{c_1^k, c_2^k, \ldots, c_{G_k}^k\}$ as a collection of candidates whose template indexes are k, where G_k is the number of elements in the subset T_k. We assign an evidence value to each c_i^k as follows:

$$ev_i^k = \eta cr_i^k + (1-\eta)\sum_{j=1}^{L} a_{ij}, \tag{16}$$

where η is a weighting constant ranging from zero to one. The first term of the right hand side of (16) represents self-evidence; it is determined by the correlation coefficient, indicating how well the candidate matches with a template. The second term, relational evidence, is determined by connection weights, signifying how well the candidate is in accord with the other candidates. If η is chosen small, we intend to give priority to the relationship between candidates rather than a candidate itself. Consequently, the weighting constant sets the balance between self-evidence and relational evidence. During this process, redundant candidates are also stored in the outlier set $O = \{o_1, o_2, \ldots, o_Q\}$, where Q is the number of outliers in the set O. Each o_i has the same attributes as c_i in the object set C.

Now, we have two separate sets as recognition results; the set C is a collection of identified objects, and the set O is a collection of rejected outliers. The elements of these two sets are used as a part of the positive and negative training data in subsequent neural network retraining. Table 4 sums up the whole procedure of outlier rejection.

4 Neural network revisited: from a retraining perspective

Up to now, we have looked at the whole object recognition procedures by stages. In this section, we again describe the neural network, but much attention is given to self-adaptability and retraining aspects rather than the neural network itself.

4.1 On-line retraining neural network: concept and related questions

In this study, the experimental environment is confined within the neighborhood of an elevator. This environment seems to be a small area in terms of the physical space in which the robot moves. However, even when the robot stands still, the environment is changing dynamically when viewed from the visual space where the object recognition process is carried out. The inside of the elevator is surrounded by transparent glass and reflective materials such as mirror. For this reason, a visual scene of the inside of the elevator is affected by the outer world and changes when the elevator is moving up and down.

In most cases, due to these conditions, an off-line training of the neural network is not enough to cope with all the possible variations of the environment, including illumination conditions. Hence it requires an on-line retraining behavior that can be performed in parallel to the recognition process. We now begin with a discussion of the four fundamental questions related to the on-line retraining neural network:

- Generation and reorganization of a new training set in real time for on-line retraining—this procedure has to be embodied in the recognition process so as to reduce extra processing time to generate new training data. It should also give a solution to the problem of forgetting some of the previously learned knowledge in the neural network.
- How to determine when to retrain; the decision is based on whether the environment changed or not.

Table 4 Outlier rejection by graph partitioning and a priori knowledge

Outlier Rejection($C = \{c_1, c_2, \ldots, c_N\}$, M_c[a])

$G \leftarrow \emptyset$
//transform candidates from an image space into a graph space
For all pairs of two candidates c_i and c_j with $i > j$ **do**
  $\{v_i, v_j, a_{ij}\} \leftarrow$ Transformation(c_i, c_j, M_c)
  $G \leftarrow G \cup \{v_i, v_j, w_{ij}\}$
Endfor
Define $\mathbf{A} = [a_{ij}]$ and $\mathbf{u} = [u_1, \ldots, u_N]^T$
$\mathbf{u}_{\max} \leftarrow \arg\max_{\mathbf{u}} \frac{\mathbf{u^T A u}}{\mathbf{u^T u}}$
//outlier rejection by the graph partitioning
$O \leftarrow \emptyset$
For all elements u_i of $\mathbf{u_{max}}$ **do**
  If $u_i < \theta_u$ **then** $C \leftarrow \{c_i\}$, $O \leftarrow O \cup \{c_i\}$
Endfor
//outlier rejection by a priori knowledge
For $k = 1$ to 5 **do**
  Let $T_k = \{c_1^k, c_2^k, \ldots, c_{G_k}^k\}$ be a collection of candidates whose t_i^k is k
  Calculate evidence value, $ev_i^k \leftarrow \eta cr_i^k + (1-\eta)\sum_{j=1}^{L} a_{ij}^k$
  Define $ev_{\max}$ as a maximum value among ev_i^k
  For all candidates in T_k **do**
    If $ev_i^k < ev_{\max}$ **then** $C \leftarrow C - \{c_i^k\}$, $O \leftarrow O \cup \{c_i^k\}$
  Endfor
Endfor
return recognized object set $C = \{c_1, c_2, \ldots, c_P\}$ and
  outlier set $O = \{o_1, o_2, \ldots, o_Q\}$

[a] M_c is the elevator control panel model

- Evaluation of the retrained neural network; it is necessary after every retraining to validate the retrained neural network, avoiding getting stuck in local minima.
- Simultaneous retraining and recognition process; the retraining process has to be done in parallel to recognition process.

We should bear in mind that real-time processing capability has to be considered; hence simplicity, or low-complexity, is worth pursuing when managing these questions. In the following sections, we suggest a simple but practical method to deal with the above questions.

4.2 How to organize a new training set

For generating a new training data, a maximum a posteriori estimation is used for selecting an optimal training data that represents the current environment [9]. In this paper, however, generation and reorganization of the new training data is implemented in a much simpler manner and is purposefully incorporated into the object recognition process, avoiding an extra process to acquire a new training data. Furthermore, it also enables the neural network to avoid forgetting previous knowledge by using a training set where some previous samples that are already learned are blended with the current ones to be learned. This forgetting phenomenon is called catastrophic interference, or catastrophic forgetting, that happens in a sequential learning task [31]. In our problem, it may also happen if we use only the newly obtained samples as the training data of next retraining; however, we have two image queues that serve as temporal storage for samples that are obtained through past processing steps so that we can exploit past samples as the next training set to adjust neural weights, thereby preventing catastrophic interference. This approach is analogous to the 'rehearsal' scheme, which relearns a subset of previously learned patterns at the same time that each new pattern is introduced [32].

The initial training data consisting of positive and negative object samples that are selected manually are already stored in the two image queues, ObjIQ and OutIQ; positive samples in ObjIQ and negative samples in OutIQ. Let S_{pos}^0 and S_{neg}^0 denote initial positive and negative sample sets in each image queue, respectively:

$$S_{pos}^0 = \{c_1^0 \cdots c_{N_{rep}}^0, c_{N_{rep}+1}^0 \cdots c_{N_{obj}}^0\}, \tag{17}$$

$$S_{neg}^0 = \{o_1^0 \cdots o_{N_{rep}}^0, o_{N_{rep}+1}^0 \cdots o_{N_{out}}^0\}, \tag{18}$$

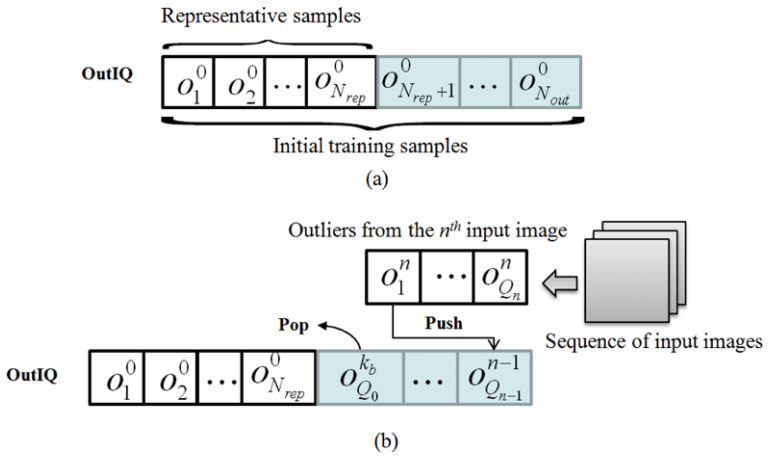

Fig. 9 Reorganization of the training data. (**a**) Initial training samples in OutIQ. (**b**) Reorganized training samples in OutIQ after processing the $(n-1)$th image. Representative samples are never removed so as to keep the initial knowledge

where N_{rep} is the number of representative samples which should be preserved during all retraining processes for the purpose of maintaining initial knowledge. After finishing the object recognition process of the nth input image, we have two sets; the identified object set $C^n = \{c_1^n, c_2^n, \ldots, c_{P_n}^n\}$, and the outlier set $O^n = \{o_1^n, o_2^n, \ldots, o_{Q_n}^n\}$, where P_n and Q_n are the numbers of identified objects and rejected outliers in the nth input image, respectively. Of course the elements in the outlier set O^n should have been removed as non-objects by the neural network classifier; unfortunately however, these are classified as objects. Therefore, each element in O^n is pushed into OutIQ as a negative training sample, whereas the oldest sample in S_{neg}^{n-1} except the representative ones are popped out from OutIQ. This procedure is depicted in Fig. 9. An identical procedure is applied to ObjIQ where c_i^n is pushed and the oldest sample is popped out. We can now express elements in ObjIQ and OutIQ after processing the nth input image as follows:

$$S_{pos}^n = \{c_1^0 \cdots c_{N_{rep}}^0, c_{N_P}^0 \cdots c_{N_{obj}}^0, \ldots, c_1^n \cdots c_{P_n}^n\}, \tag{19}$$

$$S_{neg}^n = \{o_1^0 \cdots o_{N_{rep}}^0, o_{N_Q}^0 \cdots o_{N_{out}}^0, \ldots, o_1^n \cdots o_{Q_n}^n\}, \tag{20}$$

where

$$N_p = \sum_{m=1}^{n} P_m + N_{rep} + 1,$$

and

$$N_Q = \sum_{m=1}^{n} Q_m + N_{rep} + 1.$$

This is the case where some initial training samples still remain in each image queue. However, if all initial training samples except the representative ones are removed from each image queue, then (19) and (20) are slightly modified as follows:

$$S_{pos}^n = \{c_1^0 \cdots c_{N_{rep}}^0, c_{P_0}^{k_a} \cdots c_{P_{k_a}}^{k_a}, \ldots, c_1^n \cdots c_{P_n}^n\}, \tag{21}$$

$$S_{neg}^n = \{o_1^0 \cdots o_{N_{rep}}^0, o_{Q_0}^{k_b} \cdots o_{Q_{k_b}}^{k_b}, \ldots, o_1^n \cdots o_{Q_n}^n\}, \tag{22}$$

where

$$P_0 = \sum_{m=k_a}^{n} P_m - N_{obj} + N_{rep} + 1,$$

and

$$Q_0 = \sum_{m=k_b}^{n} Q_m - N_{out} + N_{rep} + 1.$$

Each of k_a and k_b is an integer value that satisfies:

$$\sum_{m=k_a+1}^{n} P_m < N_{obj} - N_{rep} < \sum_{m=k_a}^{n} P_m, \tag{23}$$

$$\sum_{m=k_b+1}^{n} Q_m < N_{out} - N_{rep} < \sum_{m=k_b}^{n} Q_m. \tag{24}$$

Note that the number of elements in S_{pos}^n and S_{neg}^n, or the size of each image queue, is kept constant throughout the whole retraining process.

Now the training samples in each image queue represent the current and past knowledge of the environment. In (21), for example, the initial, former, and current knowledge corresponds to $c_1^0 \sim c_{N_{rep}}^0$, $c_{P_0}^{k_a} \sim c_{P_{n-1}}^{n-1}$, and $c_1^n \sim c_{P_n}^n$, respectively. Therefore, using S_{pos}^n and S_{neg}^n as training data, neural weights can be adapted to the current environment as well as the past one:

$$\mathbf{w}^c = \mathbf{w}^l + \Delta\mathbf{w}, \tag{25}$$

where each of $\mathbf{w}^c$ and $\mathbf{w}^l$ represents the matrix form of neural weights after current retraining and previous retraining is completed, respectively. $\Delta\mathbf{w}$ stands for a small incremental knowledge that should be supplemented through current retraining. In retraining process, a small number of iteration is expected to be sufficient to supplement $\Delta\mathbf{w}$ due to

the fact that $\mathbf{w}^l$ serves as initial weights that already learned the given training data to some extent.

In summary, the problem of reorganizing a new training set is nothing but a successive process of object recognition and outlier rejection. Hence the advantages of this method are that further processing is not needed to generate new training samples and due to its simplicity, it is also amenable to real time systems, avoiding catastrophic interference.

4.3 Decision of the time to retrain

In the preceding section, we have described how to organize a new training set, and then it is natural to ask how to decide on the time to retrain the neural network with this training set. Apparently, it is reasonable to retrain the neural network when the environmental changes including changes in illumination condition are believed to have occurred. The method that we adopt to detect environmental changes is related to the classification performance of the neural network and so straightforward. Suppose that the training set that is used for last retraining consists of the object recognition results of up to the kth image. It does not hold recent knowledge, or object recognition results from the $(k+1)$th to nth image; thus the lastly retrained neural network is likely to give an erroneous output when the current environment becomes different from the past ones. For example, we become aware of abrupt environmental changes in Fig. 13a, b through the occurrence of outliers. If this becomes frequent, then the neural network requires retraining to update with the recent knowledge of the changing environment. The degree of the necessity of retraining is inversely proportional to the degree of adaptability (DoA) of the neural network defined as follows:

$$\mathrm{DoA} = 1 \Big/ \left(\sum_{m=l}^{n} Q_m + 1 \right), \tag{26}$$

where n is the frame index of the current image, and l is the frame index where the last retraining occurred. A retraining process occurs when the following condition is satisfied:

$$\mathrm{DoA} < \theta_s, \tag{27}$$

where θ_s is a predefined threshold value that represents sensitivity to environmental changes, ranging from zero to one. It is associated with frequency of retraining. If θ_s is high, then the neural network is sensitive to environmental changes; thus retraining occurs frequently. However, it may be unreasonable and wasteful in terms of computing resources due to unnecessary retraining. On the other hand, if θ_s is low, then the adaptability of the neural network deteriorates, so one may not be able to cope with the current environmental changes immediately. In this study, we set θ_s to 0.2 as its best value. The calculation of DoA is carried out in conjunction with the object recognition similar to the case of generating a new training data.

4.4 Evaluation of the retrained neural network

Once retraining of the neural network is completed successfully, occurrence of outliers may be reduced at the next image frame. On the contrary, if retraining fails for some reason, the neural network may not work as a classifier properly. Consequently, whenever the retraining is done, the retrained network should be proven to be valid before being used in object recognition. Supposedly, the following are the reasons of failing in retraining:

- Catastrophic interference.
- Getting stuck in local minima.

The catastrophic interference problem, as suggested in Sect. 4.2, can be assumed to be resolved by the specially reorganized training set that preserves past training samples. The second problem is that the neural weights may get stuck in local minima due to periodic retraining. In this paper, attention was not directed to prevent neural weights from getting stuck in local minima but devoted to a method of detecting and handling the local minimum problem.

For every retraining, some criterion for terminating the retraining process must be established. Our stopping criterion involves the root mean square (RMS) error for a given training set and the number of iterations. More specifically, retraining stops if the RMS error for a given training set becomes lower than θ_e or if the number of iterations exceeds over θ_i. In general, the RMS error gets lower than θ_e within 20 iterations and so retraining comes to an end if the neural weights do not get stuck in local minima. By contrast, if retraining terminates after θ_i iterations in a situation where the RMS error is not lower than θ_e, then the neural network may yield unacceptable performance due to neural weights that are not correctly trained. In this case, we evaluate the retrained neural network to decide whether or not to utilize it as a classifier at the next object recognition process. Let $\mathbf{w}^c$ denote the neural weights after the current retraining (where (21) and (22) are used as training set) is completed. If one of the representative samples in the training set is presented to the retrained neural network, then the disparity between its actual output and desired output should be small:

$$\|(\mathbf{w}^c)^{\mathbf{T}}\mathbf{x}_i - \mathbf{o}_d\|_2 < \varepsilon_e, \quad i = 1, 2, \ldots, N_r, \tag{28}$$

where $\mathbf{x}_i$ is the input vector that corresponds to one of representative samples, $\mathbf{o}_d$ is desired output vector, and ε_e is a predefined small positive value. $\|\cdot\|_2$ denotes the l^2-norm. We randomly choose an N_r subset of the representative samples in (21), (22) and then apply (28) to each selected sample to see how many false classification will occur. We judge effectiveness of the retrained neural network by the rate of false classification p_f. In other words, if the rate of false classification is below θ_p, then we conclude that the retrained neural network is effective enough to be deployed at

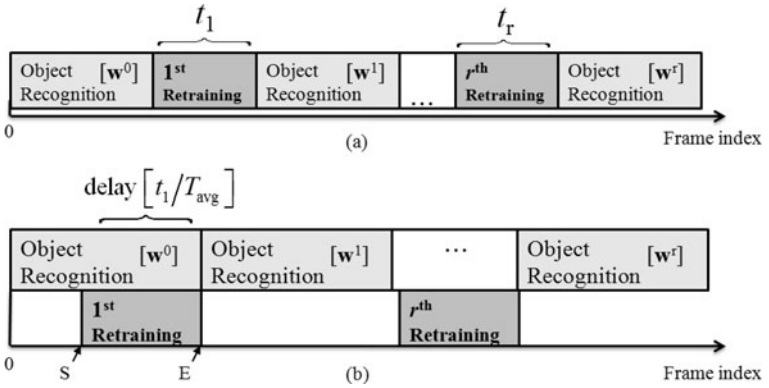

Fig. 10 A sequential and a parallel retraining structure. (**a**) Sequential retraining structure. (**b**) Parallel retraining structure. Each of S and E designates the frame index where the first retraining process is activated and terminated, respectively. Here, after finishing the rth retraining process, $\mathbf{w}^r$ is employed as new neural weights to reflect the recent environmental knowledge

next objection recognition process. In real situations, however, as discussed above, we have to deal with the problem of getting stuck in local minima where the rate of false classification is higher than θ_p. In this case, neural weights $\mathbf{w}^c$ are initialized again by the Nguyen-Widrow method [33] and are adjusted to learn the given training set as the case of initial training. On the surface, initialization of neural weights may seem to discard the past knowledge obtained so far, but this is not the case because the training set includes the past knowledge as well as the current one.

4.5 Parallel processing framework for object recognition and retraining

Retraining in parallel to object recognition is more reasonable than a sequential retraining viewed from seamless processing. In the case of sequential retraining, as shown in Fig. 10a, the object recognition process suspends during the retraining process and resumes after the retraining process is done. Let us assume that the total r number of retrainings occurred and the ith retraining required t_i time. Then the object recognition process is interrupted by the retraining process for a total of $\sum_{i=1}^{r} t_i$ time, which leads to degradation of the overall system performance. In contrast, in case of Fig. 10b, a constant object recognition process is achieved without being interrupted due to parallel retraining; each retraining is activated at the frame index where DoA of the current neural network falls below θ_s.

In an ideal case, the retrained neural network can be assumed to be deployed in the object recognition process soon after the frame index where a request for retraining is issued. To be specific, if the retraining process is initiated at the nth frame index, then the retrained network is expected to be employed immediately at the $(n+1)$th object recognition process, if retraining can be done within a very short time. However, as is often the case with real situations, each retraining requires a certain amount of time, which is likely to bring about a delay when applying the newly retrained network to object recognition. For this reason, the previously retrained network expressed by $\mathbf{w}^{r-1}$ is utilized for current object recognition process during rth retraining process. Let T_{avg} denote the average time required to perform object recognition per input image. The time requirement, or delay, for rth retraining amounts to $\lceil t_r / T_{\text{avg}} \rceil$ frames as the worst-case. Here, we define $\lceil x \rceil$ as the ceiling of x, which represents the smallest integer not less than x. Under the assumption that t_r is significantly shorter than T_{avg}, parallel retraining may show no big difference compared to a sequential one from the standpoint of seamless object recognition. Through various experiments, however, we found that t_r and T_{avg} turned out to be similar in most cases, except for some special cases arising from changes in parameters. The worst-case of average delay in real application was found to be five frames, still considered to be acceptable extent in a practical system. Table 5 summarizes the whole procedure of retraining.

5 Experimental results

In this section, we demonstrate the overall performance enhancement of the neural network through retraining. The performance was measured by using inner elevator images obtained in daytime and nighttime. We also present the effects of varying OutIQ size N_{out} on the performance of the neural network.

5.1 Image sets and parameter setting

For experiments, we prepared all possible scene images that might appear in real environments. These images are divided into two sets: 'DAY' image set which contains 271 images obtained in daytime and 'NIGHT' image set that contains 282 images obtained in nighttime. Because the natural light source cannot have influence on illumination in the nighttime, as shown in Fig. 11b, we assume that there is almost no illumination change in NIGHT, thereby considering only the effects of the artificial light source. Unlike NIGHT, illumination condition of DAY changes severely due to the natural

Table 5 Retraining phase

Retraining$(S^n_{pos}, S^n_{neg}, \mathbf{w}^l, \{Q_m \mid m = l, \ldots, n\})$

```
w^c ← w
Calculate DoA = 1/(Σ_{m=l}^{n} Q_m + 1)
If DoA < θ_s then
    BP learning using S^n = {S^n_pos, S^n_neg} as training set
    w^c ← w^l + Δw
    //evaluation of the retrained neural network
    Choose N_r test samples {x_i | m = 1, ..., N_r} among the representative ones in S^n
    p_f ← 0
    For all x_i do
        If ||(w^c)^T x_i − o_d||_2 > ε_e then p_f ← p_f + 1/N_r
    Endfor
    If p_f > θ_p then initialize weights in w^c by Nguyen-Widrow method and train again
return w^c
```

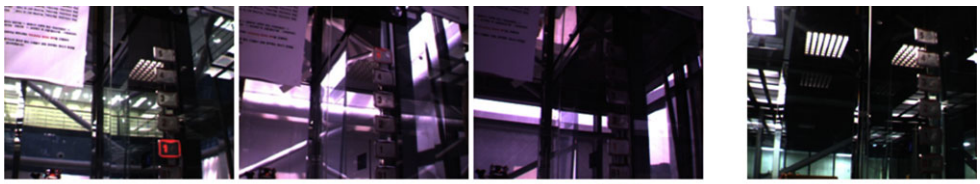

Fig. 11 Example images for the experiments. (**a**) Scene images from DAY. (**b**) A scene image from NIGHT

Table 6 General parameter setting for experiments

N_{obj}	N_{out}	N_{rep}	θ_s	θ_e	θ_i	θ_p	$r_{\min}$	$r_{\max}$	θ_a	θ_c	θ_u	η
50	70	10	0.2	0.01	100	0.3	0.1	1.5	0.8	0.65	0.3	0.4

light source and the movement of the elevator (Fig. 11a); thus some images of DAY showed great changes in overall pixel intensities without normalization techniques such as a white balancing. Table 6 shows the parameter set used in the experiments. Among those parameters, N_{out} is fairly associated with the suppression of outliers, so we examined the effects of N_{out} by varying its size. The other parameters which were determined empirically are not crucial factors that affect the system performance and kept constant during all experiments.

5.2 Performance enhancement of the neural network

Our experiment was carried out using the image set DAY for the purpose of demonstrating the performance enhancement by the neural network retraining. We measured the number of candidates passed through the neural network that were retrained during the recognition process. For performance comparison, we also measured the number of candidates passed through the neural network that were not retrained. Figure 12 shows the average number of candidates passed through each element of the CPEs. Through adaptive thresholding followed by connected component labeling, 1265 candidates were extracted initially, many of which were further eliminated by the size filtering, producing a result in which 82.05 candidates survived. Although the number of candidates entering the neural network was the same, however, there was a huge difference in the number of candidates surviving through the neural network according to whether or not retraining was done. With a retrained neural network, an average of 6.18 candidates were left, from which we could conclude that most ambiguous candidates were rejected by the neural network because there are actually five target objects corresponding to the numbers from one to five in a scene image. On the other hand, 24.66 candidates remained active when we employed the neural network without allowing the retraining process.

Table 7 shows precision and recall performance of the neural network tested by DAY image set. Both precision and recall were enhanced due to the on-line retraining; the precision, especially, was greatly improved. Here, we also present the F1 score, which can be interpreted as a harmonic mean of the precision and recall, where an F1 score reaches its best value at one and worst score at zero:

$$\mathrm{F1} = 2 \cdot (\mathrm{precision} \cdot \mathrm{recall}) / (\mathrm{precision} + \mathrm{recall}). \quad (29)$$

Consequently this revealed that the performance of the neural network as a classifier could be greatly enhanced

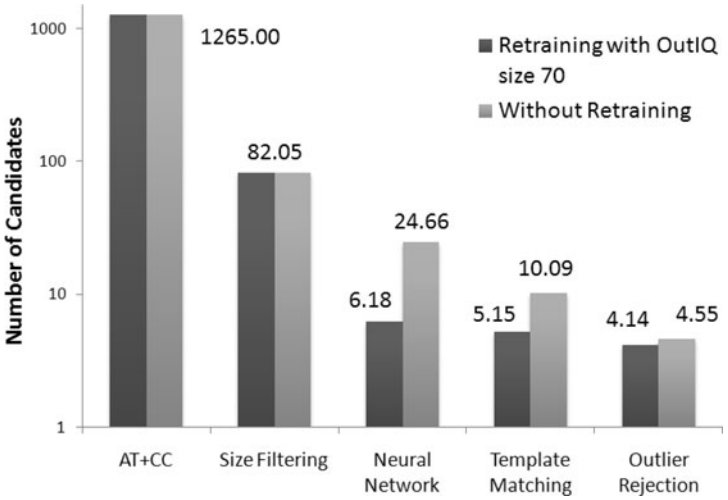

Fig. 12 The number of candidates surviving after passing each component in the experiment over the DAY image set

Table 7 Precision and recall performance

	True positive	False positive	False negative	Precision	Recall	F1
Non-retrained neural network	1294	1007	41	0.562	0.969	0.712
Retrained neural network	1309	60	26	0.957	0.981	0.968

Table 8 Rejection statistics

Frame index	AT + CC[a]	Size filtering	Neural network	Template matching	Outlier rejection
1	1187	76	27	12	5
2	1157	95	26	8	5
13	1169	79	4	4	4
22	1295	85	10	8	5
23	1252	80	13	7	4
44	1322	84	6	6	5
55	1227	80	6	6	5
103	1016	89	12	7	4
133	1273	97	6	5	4
160	1064	70	6	6	5
168	1117	60	6	6	4
218	1479	88	12	6	4
224	1504	75	6	5	4

[a]AT + CC = Adaptive thresholding + Connected component labeling

through the retraining process by allowing the system to adapt to varying environments.

We also examined the rejection status on a frame-by-frame basis in Table 8. The first column represents processing frame index and the second to the sixth column represent the number of candidates survived from each component in the CPEs. In the first frame where the retraining process was not yet performed, many candidates (i.e., 27 candidates) including false ones passed through the neural network without being filtered out. However, the number of candidates remaining was significantly reduced (i.e., 4 candidates) at the 13th frame after three retraining processes were done. In the 22nd frame, the number of candidates increased again due to environment or illumination changes; however, as shown in the rejection status of the 44th frame, in accordance with to our expectation it soon decreased by following several retraining processes. In subsequent frames, we could observe analogous phenomena, and this fluctuation in the number

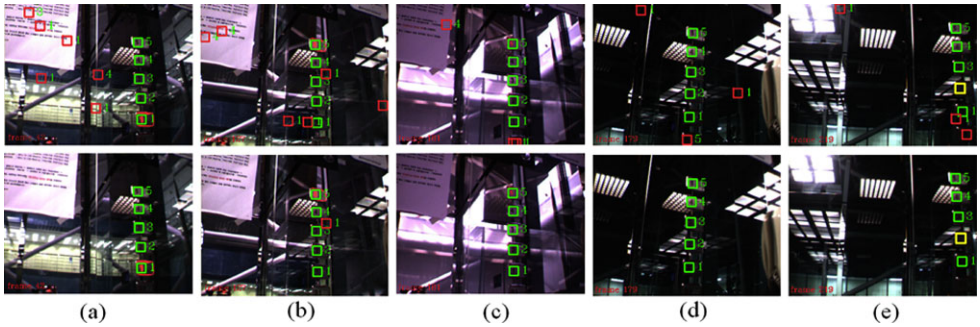

Fig. 13 (Color online) Performance improvement by the retrained neural network. *1st row*: non-retrained case, *2nd row*: retrained case. (**a**–**c**) DAY: Recognition results at frame 42, 127, 181. (**d**–**e**) NIGHT: Recognition results at frame 179, 219. Each of *red*, *green*, and *yellow square* indicates outlier, correctly recognized, and miss-recognized object, respectively

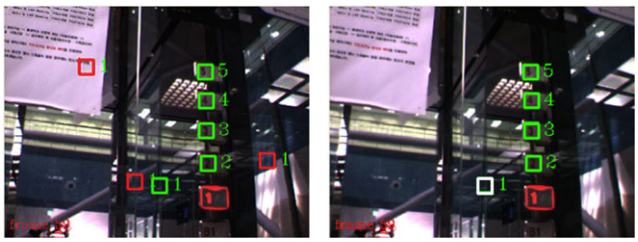

Fig. 14 False positives arising from an imperfect outlier rejection at frame 59 in a non-retrained case. The *white square in the right image* is falsely-recognized button

of candidates survived from the neural network is thought to be an inevitable thing accompanied by the environmental changes which are beyond our control.

Figure 13 directly illustrates the effect of retraining under varying lighting conditions and different view angles. The red and green squares correspond to candidates which go through the neural network and template matching, but red squares indicate the outliers that are removed by the outlier rejection component in the CPEs and the green ones are the recognized objects. The yellow square indicates a miss-recognized case, which is caused by over-filtering at preprocessing stage. One might think that the retraining process is unnecessary if the outlier rejection component works well enough to eliminate all the falsely classified candidates as shown in Fig. 13(a–d) (i.e., all target objects were correctly recognized in the 1st row). Unfortunately, however, we are not convinced that it works as perfectly as this in all situations, because it is predicated on the partial prior knowledge of the control panel model. As might be expected, without retraining, we could often find an instance in which the green square was not placed on the target object, as if the white square in Fig. 14. This corresponds to false positives. The total number of false positive N_f was listed in Table 10, showing a far better result when retraining process was allowed compared to the non-retraining case. Thus, practical advantage of retraining is that it can play an important role in alleviating the work load assigned to the outlier rejection component, thereby helping to reduce the false positives.

Table 9 Retraining statistics

	No. of iterations	Elapsed time (s)	p_f [a]
1	3	0.062	0
2	10	0.187	0
3	5	0.094	0
4	4	0.078	0.05
5	5	0.093	0
6	3	0.062	0
7	4	0.094	0
8	4	0.078	0
9	6	0.125	0
10	18	0.359	0
11	8	0.172	0
12	6	0.125	0
13	6	0.125	0
14	13	0.281	0.6
15	8	0.187	0
16	7	0.156	0
17	8	0.172	0
18	10	0.204	0

[a] p_f = The rate of false classification after retraining

5.3 Retraining time and DoA

As mentioned in Sect. 4.4, the efficacy of the retrained neural network was regularly evaluated on the basis of false classification p_f whenever each retraining process was completed. Table 9 shows retraining statistics acquired during the recognition process; total iterations and the elapsed time required to retrain the neural network; the rate of false classification p_f. At the 14th retraining, p_f of the retrained neural network showed a higher value 0.6 than a predefined

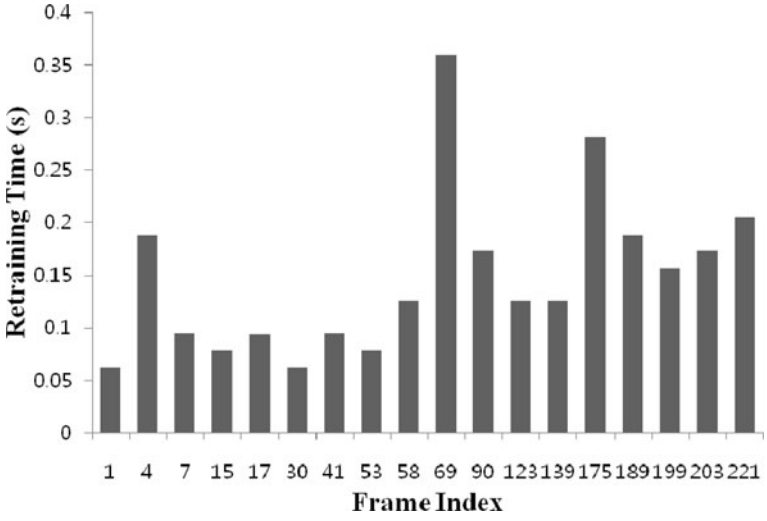

Fig. 15 Elapsed time for each retraining process. Maximum time is 0.359 s and minimum one is 0.062 s

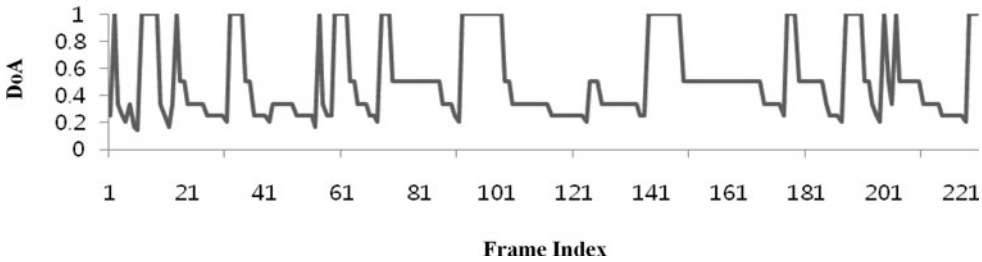

Fig. 16 The history of degree of adaptability over the DAY image set

threshold θ_p, and thus the retraining process turned out to be a failure, thereby having to be repeated with another set of weight initialization. Here, the neural weights do not preserve the former knowledge any more due to initialization; therefore, it was expected at the outset that more time would be required to learn a given training set, compared with the case of learning with the former neural weights as initial ones.

The results shown in Fig. 15 were consistent with our expectation except one special case, which corresponded to the 10th retraining process where 0.359 s was required. This undesirable case originated from the fact that the neural weights were placed around a gentle slope in the underlying error landscape, thus many iterations (i.e., total 18 iterations) were required to satisfy the RMS error criterion for terminating the retraining process. Nevertheless, the average of elapsed time to retrain the neural network can be concluded as being reasonable for a real-time system.

DoA, representing the effectiveness of the current neural network, was calculated every frame and its oscillating characteristic is shown in Fig. 16. This oscillation seems consistent with the fluctuation in the number of candidates shown in Table 8. Directly after each retraining process, DoA was observed to have an abrupt increment up to one, elucidating that the retrained neural network was adapted successfully to the current environment.

5.4 Effects of various OutIQ sizes on the total system performance

We now evaluate the total system performance by varying the OutIQ size with other parameters being fixed as in Table 6. One may speculate that it would be better for OutIQ to have a larger size in that the more negative examples are provided by OutIQ, the more outliers are likely to be suppressed. However, it may lead to a long training time due to increased training samples; therefore, it is desirable to establish an adequate size of OutIQ.

Experiments were carried out over the DAY and NIGHT with three different OutIQ sizes; (a) 50, (b) 100, and (c) 130. The most important thing is the outlier occurrence rate R_o, or the average number of outliers per frame in each 'no retraining' and 'retraining' case of DAY and NIGHT as shown in the second row in Table 10. This figure clearly shows the effects of varying OutIQ size and also effectiveness of retraining.

5.4.1 Experiment over the DAY set

Effects of the OutIQ size on the total system performance were verified by plotting the number of outliers occurring in DAY on a logarithmic scale (Fig. 17a). As might be expected, the three size instances could be sorted in descending

Table 10 Summary of the experimental results

	No retraining		N_{out} 50		N_{out} 100		N_{out} 130	
	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT
$N_{outlier}$	1002	493	110	69	95	55	88	49
R_o	3.70	1.75	0.41	0.24	0.35	0.20	0.32	0.17
N_{retr}	–	–	23	16	22	12	20	11
$T_r(s)$	–	–	0.13	0.08	0.16	0.09	0.17	0.11
N_f	27	1	3	0	2	0	0	0

DAY = The image set obtained in daytime
NIGHT = The image set obtained in nighttime
$N_{outlier}$ = The number of outliers
R_o = Outlier occurrence rate (i.e., $N_{outlier}$/total number of frames)
N_{retr} = The total number of retraining performed
T_r = Average elapsed time for retraining
N_f = Total number of false positives
(−) = Not applicable

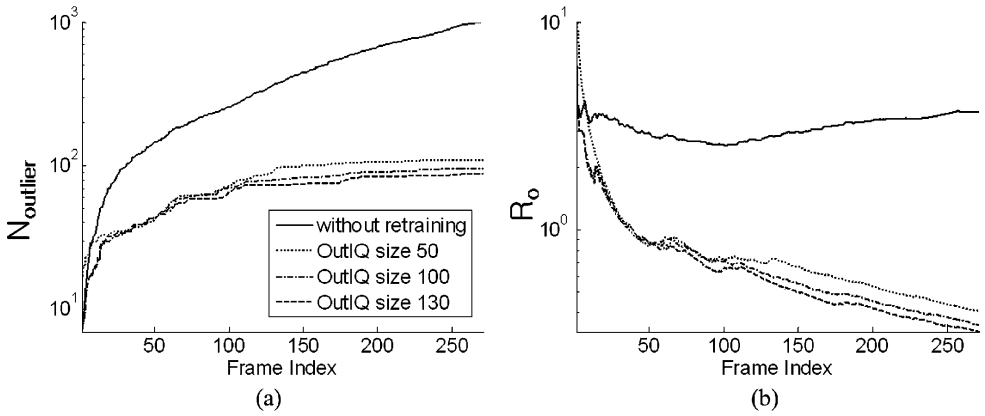

Fig. 17 Performance comparison with varying OutIQ size in the experiment over the DAY image set. (**a**) The number of outliers vs. frame. (**b**) Outlier occurrence rate vs. frame

order according to N_{outlier} (i.e., (a)–(b)–(c)). As a result, we found that:

- The outlier occurrence rate decreases with increasing OutIQ size.
- The average time required for the retraining process increases with increasing OutIQ size.
- The total number of retraining processes performed increases with decreasing OutIQ size.

As shown in Fig. 17b, R_o of case (c) was the smallest of the three cases only after about 50 frames, and R_o of case (b) was smaller than that of case (a) only after about 170 frames. In short, on the average 0.43, 0.35 and 0.32 outliers occurred per image in case (a), (b) and (c), respectively. It is natural to take a long time to retrain the neural network if the number of training samples increases due to an enlarged OutIQ size. The relationship between the total number of retraining processes and the OutIQ size can be interpreted in that more frequent retraining processes were needed to adapt to the current environment when the OutIQ size was small because the negative samples that were included in the training set were reduced.

5.4.2 Experiment over the NIGHT set

We also conducted an analogous experiment over the NIGHT image set and its result showed a consistent tendency with the result of the DAY (Fig. 18). However, since the natural light source could not affect the illumination condition at night, variation of the illumination condition was not serious compared to DAY. Thus we found that the total number of outliers was severely diminished up to half of the daytime case.

6 Conclusions

A retrainable framework approach to object recognition in a visually dynamic environment has been presented. First, in order to extract object candidates, basic image processing techniques including adaptive thresholding, connected component labeling, and template matching were applied to an input image. An on-line retraining neural network as a primary component of the proposed method, serving as a binary classifier, has then been employed as a practical tool

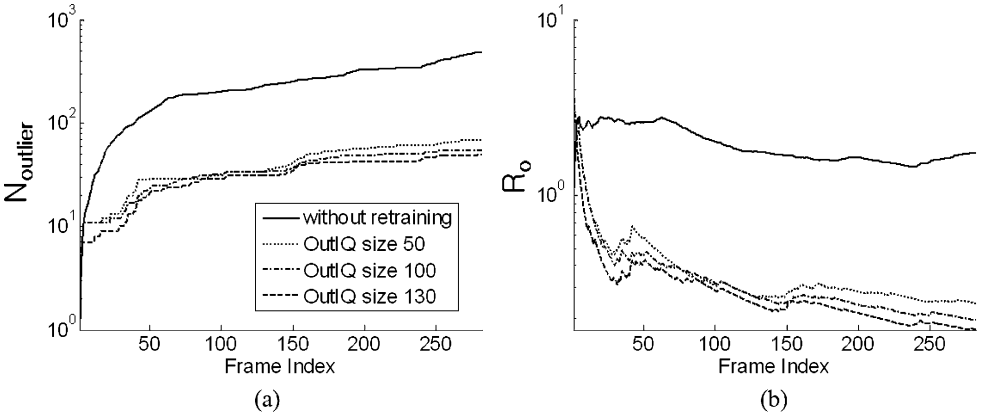

Fig. 18 Performance comparison with varying OutIQ size in the experiment over the NIGHT image set. (**a**) The number of outliers vs. frame. (**b**) Outlier occurrence rate vs. frame

for coping with varying illumination conditions. The new training samples for retraining the neural network were provided by graph partitioning, which can detect and eliminate outliers efficiently based on a predefined model, or a prior knowledge.

Furthermore, since we adopted a parallel retraining mechanism that can enable the system to perform object recognition and network retraining concurrently, the object recognition process was carried out seamlessly without being interrupted by the retraining process. We also took an immediate action on a failure of the retraining process for obtaining high reliability. The performance enhancement of the retrained neural network over a non-retrained one was verified by experiments over daytime and nighttime images, being represented by a quantitative difference in the number of outliers that occurred. The proposed method can also be applied to other problems of recognizing arbitrary 2D objects where the relationship between the target objects can be utilized as prior knowledge.

Acknowledgement This work was supported by Ministry of Knowledge Economy under Human Resources Development Program for Convergence Robot Specialists.

References

1. Hüttenrauch H, Eklundh KS (2002) Fetch-and-carry with CERO observations from a long-term study with a service robot. In: IEEE international workshop on robot and human interactive communication, pp 158–163
2. Thrun S, Bennewitz M, Burgard W, Cremers AB, Dellaert F, Fox D (1999) MINERVA: a second-generation museum tour-guide robot. In: IEEE international conference on robotics and automation, pp 1999–2005
3. Bischoff R, Graefe V (2002) HERMES—a versatile personal robotic assistant. Proc IEEE 92(11):1759–1779, Special issue on human interactive robots for psychological enrichment
4. Simmons R et al (2003) Grace: an autonomous robot for the AAAI robot Challenge. AAAI Mag 42(2):51–72
5. Tschichold-Gürman N, Vestli SJ, Schweitzer G (2001) The service robot MOPS: first operating experiences. Robot Auton Syst 34(2–3):165–173
6. Ramaswamy S, Rastogi R, Shim K (2000) Efficient algorithms for mining outliers from large data sets. In: ACM SIGMOD conference on management of data, Dallas, TX, pp 427–438
7. Breunig MM, Kriegel H, Ng RT, Sander J (2000) LOF: identifying density-based local outliers. In: Proceedings of ACM SIGMOD International conference on management of data, Dallas, TX, pp 93–104
8. Olson E, Leonard JJ, Teller S (2006) Robust range-only beacon localization. IEEE J Ocean Eng 31(4):949–958
9. Doulamis AD, Doulamis ND, Kollias SD (2000) On line retrainable neural networks: improving the performance of neural networks in image analysis problems. IEEE Trans Neural Netw 11(1):137–155
10. Ioannou S, Kessous L, Caridakis G, Karpouzis K, Aharonson V, Kollias S (2006) Adaptive on-line neural network retraining for real life multimodal emotion recognition. In: International conference on artificial neural networks (ICANN), pp 81–92
11. Chan FHY, Lam FK, Zhu H (1998) Adaptive thresholding by variational method. IEEE Trans Image Process 7(3):468–473
12. Gonzales R, Woods R (1992) Digital image processing. Addison-Wesley, Reading
13. Haykin S (1999) Neural networks: a comprehensive foundation. Prentice-Hall, New Jersey
14. West DB (2001) Introduction to graph theory, 2nd edn. Prentice-Hall, New Jersey
15. Osowski S, Linh TH (2001) ECG beat recognition using fuzzy hybrid neural network. IEEE Trans Biomed Eng 48(11):1265–1271
16. Er MJ, Wu S, Lu J, Toh HL (2002) Face recognition with radial basis function (RBF) neural networks. IEEE Trans Neural Netw 13(3):697–710
17. Magoulas GD, Plagianakos VP, Vrahatis MN (2004) Neural network-based colonoscopic diagnosis using on-line learning and differential Evolution. Appl Soft Comput 4(4):369–379
18. Rowley HA, Baluja S, Kanade T (1998) Neural network-based face detection. IEEE Trans Pattern Anal Mach Intell 20(1):23–38
19. Zou AM, Hou ZG, Fu SY, Tan M (2006) Neural networks for mobile robot navigation: a survey. In: International symposium on neural networks. LNCS, vol 3972. Springer, Berlin, pp 1218–1226
20. Chohra A, Benmehrez A (1998) Neural navigation approach for intelligent autonomous vehicles (IAV) in partially structured environments. Appl Intell 8(3):219–233
21. Han SJ, Oh SY (2008) An optimized modular neural network controller based on environment classification and selective sensor usage for mobile robot reactive navigation. Neural Comput Appl 17(2):161–173
22. Egmont-Petersen M, De Ridder D, Handels H (2002) Image processing with neural networks—a review. Pattern Recognit 35(10):2279–2301

23. Ou G, Murphey YL (2007) Multi-class pattern classification using neural networks. Pattern Recognit 40(1):4–18
24. Zhang ZP (2000) Neural networks for classification: a survey. IEEE Trans Syst Man Cybern, Part C, Appl Rev 30(4):451–462
25. Lippmann RP (1989) Pattern classification using neural networks. IEEE Commun Mag 27(11):59–64
26. Hodge VJ, Austin J (2004) A survey of outlier detection methodologies. Appl Intell Rev 22(2):85–126
27. Dillencourt MB, Samet H, Tamminen M (1992) A general approach to connected-component labeling for arbitrary image representations. J ACM 39(2):253–280
28. Reed R (1993) Pruning algorithms—a survey. IEEE Trans Neural Netw 4(5):740–747
29. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60(2):91–110
30. Lowe DG (1999) Object recognition from local scale-invariant features. In: Proceedings of the international conference on computer vision (ICCV), Corfu, Greece, pp 1150–1157
31. French RM (1999) Catastrophic forgetting in connectionist networks. Trends Cogn Sci 3(4):128–135
32. Robins A (1995) Catastrophic forgetting, rehearsal and pseudorehearsal. Connect Sci 7(2):123–146
33. Nguyen D, Widrow B (1990) Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. In: International joint conference of neural networks, vol 3, pp 21–26