Article

Autonomous Surveillance for an Indoor Security Robot

Min-Fan Ricky Lee [1,2,*] and Zhih-Shun Shih [1]

[1] Graduate Institute of Automation and Control, National Taiwan University of Science and Technology, Taipei 106335, Taiwan
[2] Center for Cyber-Physical System Innovation, National Taiwan University of Science and Technology, Taipei 106335, Taiwan
* Correspondence: rickylee@mail.ntust.edu.tw

Abstract: Conventional surveillance for a security robot suffers from severe limitations, perceptual aliasing (e.g., different places/objects can appear identical), occlusion (e.g., place/object appearance changes between visits), illumination changes, significant viewpoint changes, etc. This paper proposes an autonomous robotic system based on CNN (convolutional neural network) to perform visual perception and control tasks. The visual perception aims to identify all objects moving in the scene and to verify whether the target is an authorized person. The visual perception system includes a motion detection module, a tracking module, face detection, and a recognition module. The control system includes motion control and navigation (path planning and obstacle avoidance). The empirical validation includes the evaluation metrics, such as model speed, accuracy, precision, recall, ROC (receiver operating characteristic) curve, P-R (precision–recall) curve, F_1-score for AlexNet, VggNet, and GoogLeNet, and RMSE (root-mean-square error) value of mapping errors. The experimental results showed that the average accuracy of VggNet under four different illumination changes is 0.95, and it has the best performance under all unstable factors among three CNN architectures. For the accuracy of building maps in real scenes, the mapping error is 0.222 m.

Keywords: mobile robots; face recognition; artificial intelligence; object detection; simultaneous localization and mapping; deep learning

check for updates

Citation: Lee, M.-F.R.; Shih, Z.-S. Autonomous Surveillance for an Indoor Security Robot. *Processes* **2022**, *10*, 2175. https://doi.org/10.3390/pr10112175

Academic Editor: Raul D. S. G. Campilho

Received: 2 August 2022
Accepted: 22 September 2022
Published: 24 October 2022

1. Introduction

Recent developments in AI (artificial intelligence) have led to a renewed interest in security robots. An autonomous indoor surveillance robot based on Raspberry Pi was designed and achieve face recognition and navigation [1]. A mobile robot of indoor monitoring and surveillance was proposed and achieved object recognition with an RGB-D camera [2]. A surveillance robot implemented object detection with CNN and YOLO (you only look once) algorithm [3]. An autonomous mobile robot for surveillance was proposed and achieved face detection and recognition [4]. A mobile robot was designed to implement people detection and automatic guard patrol [5]. However, conventional surveillance is severely limited by many factors. For example, at different places and viewpoints, different objects may look the same. In addition, significant changes in illumination and pose and random occlusion will greatly increase the recognition error rate. Several researchers have studied how to eliminate the effects of these factors on image recognition.

Significant changes in illumination, occlusion, expression, etc., make face recognition actually challenging, and many researchers have proposed methods to solve this problem. A novel deep learning network of circular symmetrical Gabor filter 2D PCA (principal component analysis) neural networks was proposed [6]. The method tested on various databases, and it is more robust to the variations mentioned above. A multitask CNN was proposed for face recognition [7]. Compared with the various datasets, the proposed method has better performance than existing technologies. A conditional generative adversarial network-based approach was presented to mitigate intraclass differences [8].

The experimental results showed that this method is more effective than the methods on the AffectNet and Real-world Affective Faces databases. A face recognition method based on a directional gradient dense-grid histogram was proposed [9]. Experimental results showed that this method is more suitable for changes in time and environment, and compared with local binary patterns, uses fewer dimensions to obtain a better recognition rate. A novel low-rank regularized general representation method was proposed to solve the problem [10]. The experimental results on the four face databases showed the robustness to expression, illumination, occlusion, and time-varying methods. Illumination variations, random occlusion, and surface textures also cause fruits and vegetables to be judged and positioned incorrectly. Faster R-CNN was used to locate the target in the fruit and vegetable images obtained by two CCD (charge-coupled device) cameras [11]. Experimental results showed that the method is robust and the average accurate recognition rate under six different conditions is 96.33%.

Severe illumination variations are considered as tough issues for the face images in the outdoor environment. DSP (diagonal symmetric pattern) was proposed for face recognition under severe illumination variation [12]. The experimental results on various databases indicate that the proposed methods are more efficient. A new method based on the Lambert reflectance model was proposed to improve the illumination invariants [13]. The experimental results showed that the model is insensitive to complex illumination variations. The frequency characteristic bases to construct a novel high-frequency facial feature were utilized [14]. Experiments showed that transforming the proposed faces into a GHSP (general high-frequency based sparse representation) model can eliminate the specific identity information of ordinary faces.

In addition to the factors mentioned above, human faces in surveillance often suffer from severe dramatic pose variations. TBE (trunk branch ensemble) CNN was proposed; it extracts features from patches cropped around facial components [15]. The results showed that TBE-CNN achieved the most advanced performance on various databases. A deep DSN (disentangling Siamese network) was proposed for frontal face [16]. Quantitative and qualitative evaluations of the proposed network on benchmarks showed that the proposed network performs better than the state-of-the-art methods.

However, the literature reviewed above might have been more useful if it considered more evaluation metrics, such as precision, recall, ROC curve, AUC value, F_1-score, etc. Only [11] considered the above evaluation metrics, and the rest only included the recognition rate and accuracy. In addition, since the security robot is not allowed to underreport any unauthorized person, this research also discusses sensitivity. In the face recognition algorithm part, the above literature was continued, using CNN and the Viola–Jones face detection algorithm, and continued [1] to achieve robot navigation and obstacle avoidance.

The proposed visual perception system includes a face detection module, face tracking module, and face recognition module. The face detection algorithm is a frontal face Viola–Jones cascade, because it is one of the more practical ways to detect human face. Three CNN architectures were used in this study: AlexNet, Vgg-16, and GoogLeNet, which are all winners and runners-up of the ILSVRC (ImageNet Large-Scale Visual Recognition Challenge), by pretraining images under different illuminations to solve the impact of environmental instability on the recognition rate. The evaluation metrics of the model include accuracy, precision, recall, ROC curve, P-R curve, and F_1-score, and compare the performance of the three CNN architectures. The control system includes navigation and obstacle avoidance. The mobile robot was equipped with the ROS (robot operating system) and LiDAR (light detection and ranging) modules and the Hector SLAM (simultaneous localization and mapping) algorithm. For the development tools, Python 2.7 and Keras deep learning libraries were used as the front end and Tensorflow as the back end to build the model. Different from other studies, the contribution of this study is to propose a deep learning and Hector SLAM algorithm that combines the LiDAR module and the popular ROS system and considers more evaluation metrics, such as precision, recall, ROC curve, AUC value, F_1-score, etc., as well as bring more possibilities to the research and development of autonomous security robots.

Section 2 presents the algorithms of face detection and face recognition, the configuration of a mobile robot under an ROS environment, navigation and obstacle avoidance of an autonomous robot using the SLAM algorithm and LiDAR module, and the program flow chart. Section 3 describes the experimental results of face detection, CNN model training, and the testing of three CNN models under an unstable environment and target. For each unstable factor, this paper presents the evaluation metrics, such as accuracy, precision, ROC curve, and P-R curve of three CNN models. For the results of the autonomous robot, this paper presents the RMSE value of mapping errors, and the results of the security robot performing the tasks of patrolling, path planning and surveillance in a real indoor environment. Sections 4 and 5 describe the discussion and conclusion of the research, and also present future work.

2. Materials and Methods

Since the accuracy of face recognition on the security robot will be affected under significant changes in illumination and pose, the purpose of this study is to propose a method of face recognition and reduce the impact caused by the above factors. A deep learning based face recognition is proposed for a wheeled mobile robot to reduce the impact from the variation of illumination and pose that affect the recognition accuracy. A Haar-like feature algorithm was used to achieve face detection, and CNN was used to achieve face recognition, and then deployed to the mobile robot to achieve face tracking. The research scenario is shown in Figure 1: if no human face is detected, keep patrolling and avoid obstacles until a human face is detected. For the part relating to the patrol of the security robot, the LiDAR module, Hector SLAM algorithm, DWA (dynamic window approach) algorithm, and navigation stack were used to achieve autonomous movement and obstacle avoidance, robot positioning, and mapping. When a human face is detected, the face is recognized through the trained CNN model. If the detected face is an authorized person, the security robot will track the face; otherwise, it will send a warning message to the mobile phone.

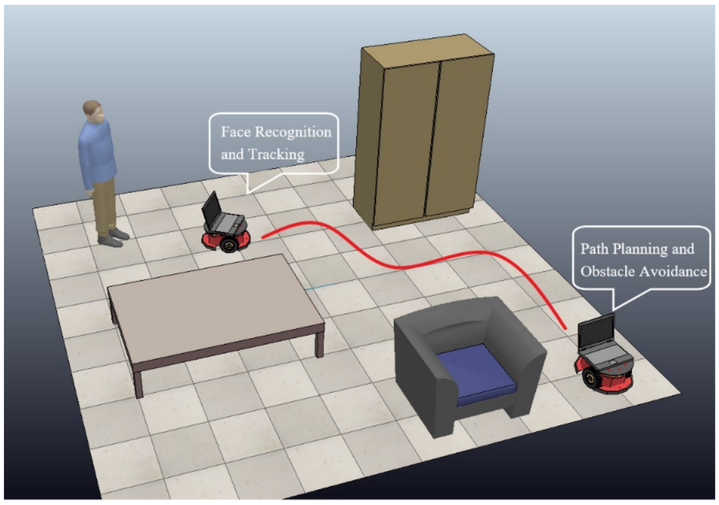

Face Recognition and Tracking
Path Planning and Obstacle Avoidance

Figure 1. Research scenario.

2.1. Proposed Architecture

The system architecture was divided into two parts. The first part is the architecture of deep learning, and the second is the system module architecture. As shown in Figure 2, deep learning of neural network architecture was used to realize face recognition and tracking. The Haar-like feature method was used to detect human faces and collect the face images to be trained; because the CNN model training can only focus on face images, it can reduce the number of parameters and improve the recognition rate. Deep learning training and validation of CNN were performed, the trained model was obtained to workstation, then the model was tested. The face images were detected by the robot in real time and sent movement commands to the robot to achieve face tracking.

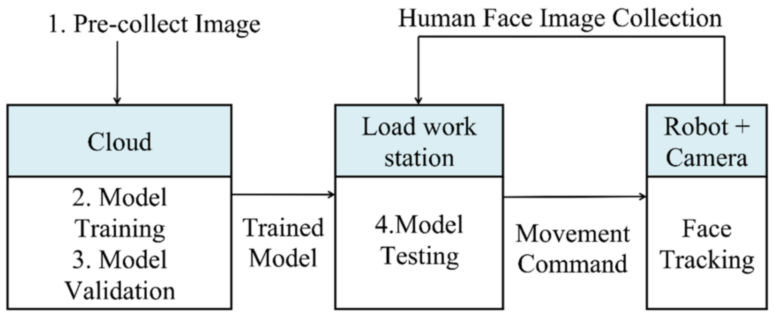


Figure 2. Deep learning architecture of face recognition and tracking.

The second part is shown in Figure 3, which includes three modules and two types of information collected from environment. The images obtained by PTZ (pan tilt zoom) camera are used to perform motion detection, feature extraction, and face tracking, and obtain the center point of target with face detection algorithm, via coordinate transformation, sending the angle information to motion control module to achieve face tracking. More information is obtained by LiDAR sensor and obtains the range from robot to target, using the navigation algorithm of obstacle avoidance module to obtain the angle and velocity, then sends the information to motion control module to achieve obstacle avoidance of mobile robot.

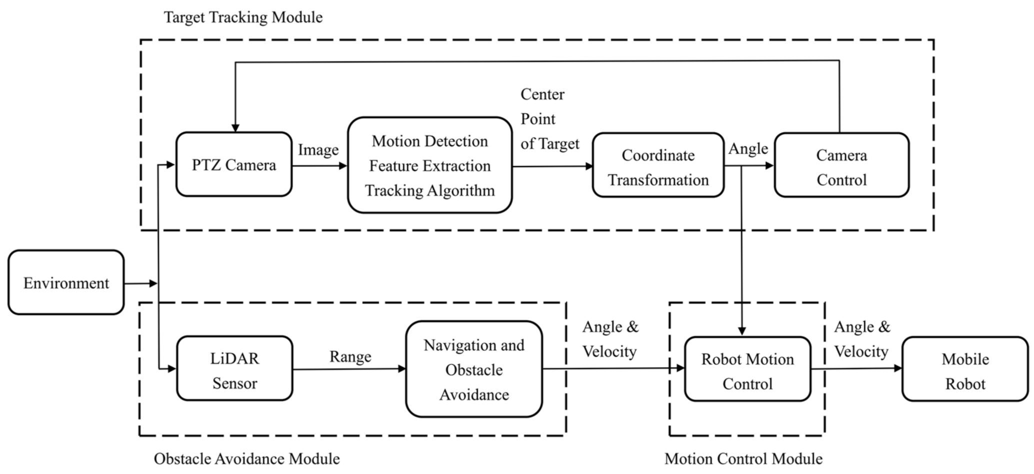


Figure 3. System module architecture.

2.2. Face Detection Algorithm

A machine learning method called adaptive boosting is proposed [17], it's often applied in the situation when there are some parameters that need to be adjusted, andthe output function can be expressed as in (1).

$$H(x) = \text{sign}(\sum_{n=1}^{N} w_n f_n(x)), \quad (1)$$

where $H(x)$ is the output function of weak classifier, x is input data of weak classifier, N is the number of classifiers, w_n is the weight assigned to the output function, f_n is the output of weak classifier. AdaBoost and Haar-like features were applied to implement object detection [18];

However, because there are too many features on the face, and there will be multiple rectangles combined to form a feature, an integral image was proposed to represent any point in the image, which is the sum of all pixels from the point to the origin of the upper left corner, which can easily calculate the eigenvalue of a specific area in the image, such as the 4 rectangular features shown in Figure 4, and the integral image at location (x, y) can be expressed as in (2).

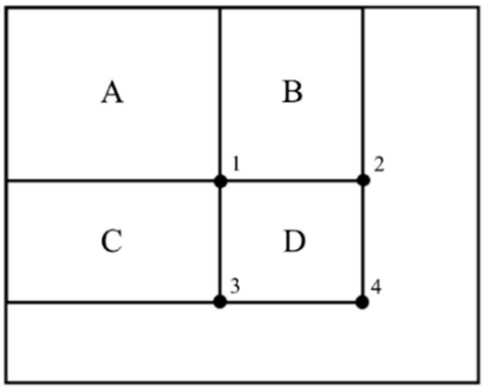


Figure 4. The eigenvalue of area of Viola–Jones algorithm. A to D is the eigenvalue area, and 1 to 4 mean the coordinates.

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x',y'), \tag{2}$$

where ii (x, y) is the integral image at location (x, y), x and y are the coordinates of images, i (x, y) is the original image, the cumulative row sum can be expressed as in (3), and (2) can be expressed as in (4). According to the above equations, the eigenvalue of designated area D can be expressed as in (5), and the pseudo code of Viola–Jones face detection algorithm is shown as Algorithm 1.

$$s(x,y) = s(x,y-1) + i(x,y), \tag{3}$$

$$ii(x,y) = ii(x-1,y) + s(x,y), \tag{4}$$

$$D = ii(4) - ii(2) - ii(3) + ii(1), \tag{5}$$

where s (x, y) is the cumulative row sum of integral image and D is the eigenvalue of area in Figure 4.

Algorithm 1 Viola–Jones

```
1:  $image_i$ ← downsampling of input image
2:  for $j$ = 1 to number of shift steps do
3:    for $k$ = 1 to number of stages do
4:      for $l$ = 1 to number of filters do
5:        filters detect subwindow of $image_i$
6:        $w_i$ ← accumulation filter outputs
7:        if $w_i$ < $threshold$ then
8:          reject subwindows as face
9:        end if
10:     end for
11:   end for
12:   if pass all stages then
13:     accept subwindows as face
14:   end if
15: end for
```

The object detector was extended, and eventually generated Viola–Jones Cascade Classifier [19], and it was used in several studies. Viola–Jones algorithm and support vector network were used to recognize faces [20]. Neural network and Viola–Jones algorithm were used to detect center of eye [21]. A firefighting robot was designed and used Viola–Jones to recognize human faces [22].

The Viola–Jones algorithm was applied because it is one of the more practical ways to detect human faces and crop face pictures. Different from the above research, the face photos obtained by the Viola–Jones algorithm were input into CNN, and the technology of face recognition was directly applied to the autonomous mobile robot. In order for the robot to recognize specific target under different viewpoints and illumination changes, the human face images of various viewpoints and illuminations in an indoor environment were collected, then input to the CNN architecture.

2.3. CNN Architecture

Several CNN architectures performed well in the ILSVRC classification competition, and many researchers used CNN to achieve object recognition. CNN architectures were used for emotion recognition [23]. Deep learning library was used to identify criminal suspects [24]. An example of CNN architecture is shown in Figure 5, and three CNN architectures were used: AlexNet, Vgg-16, and GoogLeNet, and these three CNN architectures have different numbers of layers and connection methods. AlexNet is composed of 5 convolution layers and 3 fully connected layers, and uses the relu activation function and dropout to prevent overfitting. This architecture is shown as Algorithm 2. Vgg-16 is composed of 13 convolution layers and 3 fully connected layers. Compared with AlexNet, VggNet contains several consecutive 3 × 3 convolution kernels instead of the larger convolution kernels in AlexNet, which can improve the depth of the neural network, reduce the number of parameters, and more effectively maintain the image properties. The architecture is shown as Algorithm 3. GoogLeNet contains a total of 22 layers, of which 21 convolutional layers are composed of Inception architecture; each contains four 1 × 1 convolution layer, one 3 × 3 convolution layer, one 5 × 5 convolution layer, and one 3 × 3 max pooling layer. This modular structure is adopted, which greatly increases the flexibility and reduces the number of parameters. The architecture is shown as Algorithm 4.

Algorithm 2 CNN architecture of AlexNet

```
1:  Convolution (96, 11, 11, padding = 'same', activation = 'relu')
2:  MaxPooling (pool_size = (3, 3))
3:  Convolution (256, 5, 5, padding = 'same', activation = 'relu')
4:  MaxPooling (pool_size = (3, 3))
5:  Convolution (384, 3, 3, padding ='same', activation = 'relu')
6:  Convolution (384, 3, 3, padding ='same', activation = 'relu')
7:  Convolution (384, 3, 3, padding ='same', activation = 'relu')
8:  MaxPooling (pool_size = (3, 3))
9:  FullConnected (4096, activation = 'relu', dropout = 0.5)
10: FullConnected (4096, activation = 'relu', dropout = 0.5)
11: FullConnected (1000, activation = 'relu', dropout = 0.5)
12: Classification (classes = 2, activation = 'softmax')
```

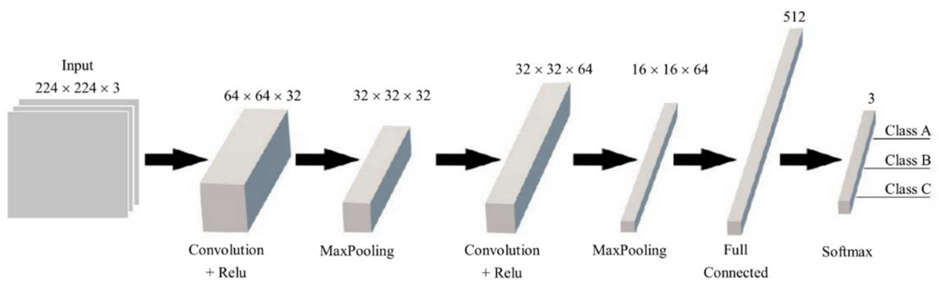


Figure 5. CNN architecture.

Algorithm 3 CNN architecture of VggNet

```
1:  Convolution (64, 3, 3, padding = 'same', activation = 'relu')
2:  Convolution (64, 3, 3, padding = 'same', activation = 'relu')
3:  MaxPooling (pool_size = (2, 2))
4:  Convolution (128, 3, 3, padding = 'same', activation = 'relu')
5:  Convolution (128, 3, 3, padding = 'same', activation = 'relu')
6:  MaxPooling (pool_size = (2, 2))
7:  Convolution (256, 3, 3, padding = 'same', activation = 'relu')
8:  Convolution (256, 3, 3, padding = 'same', activation = 'relu')
9:  Convolution (256, 3, 3, padding = 'same', activation = 'relu')
10: MaxPooling (pool_size = (2, 2))
11: Convolution (512, 3, 3, padding = 'same', activation = 'relu')
12: Convolution (512, 3, 3, padding = 'same', activation = 'relu')
13: Convolution (512, 3, 3, padding = 'same', activation = 'relu')
14: MaxPooling (pool_size = (2, 2))
15: Convolution (512, 3, 3, padding = 'same', activation = 'relu')
16: Convolution (512, 3, 3, padding = 'same', activation = 'relu')
17: Convolution (512, 3, 3, padding = 'same', activation = 'relu')
18: MaxPooling (pool_size = (2, 2))
19: FullConnected (4096, activation = 'relu', dropout = 0.5)
20: FullConnected (4096, activation = 'relu', dropout = 0.5)
21: FullConnected (1000, activation = 'relu', dropout = 0.5)
22: Classification (classes = 2, activation = 'softmax')
```

Algorithm 4 CNN architecture of GoogLeNet

```
1:  Conv_Batch_normalization (64, 7, 7, padding = 'same')
2:  MaxPooling (pool_size = (3, 3))
3:  Conv_Batch_normalization (192, 3, 3, padding = 'same')
4:  MaxPooling (pool_size = (3, 3))
5:  Inception ((64), (96, 128), (16, 32), (32))
6:  Inception ((128), (128, 192), (32, 96), (64))
7:  MaxPooling (pool_size = (3, 3))
8:  Inception ((192), (96, 208), (16, 48), (64))
9:  Inception ((160), (112, 224), (24, 64), (64))
10: Inception ((128), (128, 256), (24, 64), (64))
11: Inception ((112), (144, 288), (32, 64), (64))
12: Inception ((256), (160, 320), (32, 128), (128))
13: MaxPooling (pool_size = (3, 3))
14: Inception ((256), (160, 320), (32, 128), (128))
15: Inception ((384), (192, 384), (48, 128), (128))
16: AveragePooling (pool_size = (7, 7))
17: FullConnected (1000, activation = 'relu')
18: Classification (classes = 2, activation = 'softmax')
```

Input the face images into the input layer, next, the convolutional layer of same padding was used to extract the features from pictures: if the size of each patch was $m \times n$, then the sum of the $m \times n$ data multiplied by the filter data was computed; so that the features of the pictures are more obvious, the equation can be expressed as in (6), and the dimensions of the output matrix can be expressed as in (7). The relu activation function was added behind each convolution layer; the data can be subtracted from the negative value by the relu function, as expressed in (8). The shape of the object can be more obvious.

$$conv(I,K)_{x,y} = \sum_{i=1}^{n_H} \sum_{j=1}^{n_W} \sum_{k=1}^{n_C} K_{i,j,k} I_{x+i-1,y+j-1,k}, \tag{6}$$

$$n_{out} = \frac{n_{in} + 2p - n_k}{s} + 1, \tag{7}$$

$$f(r) = \max(0, r), \tag{8}$$

where *conv* (*I*, *K*) is the convolution equation, *I* is feature map, *K* is kernel, *x* and *y* are the row and column of feature map, n_{H} is the size of the height of the image, n_{W} is the size of the width of the image, n_{C} is the number of channels, n_k is the kernel size, *p* is the convolution padding size, *s* is the convolution stride size, n_{in} is the number of input features, n_{out} is the number of output features, *f*(*r*) is the relu function, and *r* is the input data of relu function.

In order to retain important features, reduce the parameters, and prevent overfitting, max pooling layer was added behind the activation function layer; the kernel size was 2 × 2, the size of data was half after the pooling layer. At the end of the CNN, fully connected layer was added, which can flatten the previous results, and the number of dimensions of fully connected layer can be expressed as in (9).

$$n_{i-1} = {n_{\mathrm{H}}}^{i-1} \times {n_{\mathrm{W}}}^{i-1} \times {n_{\mathrm{C}}}^{i-1}, \tag{9}$$

where n_{i-1} is the number of dimensions of fully connected layer; *i* is the number of layers; n_{H}, $n_{\mathrm{W},}$ and n_{C} are as mentioned in (6). The last layer is softmax activation function layer, as expressed in (10) to (12); this is a Gaussian function layer used to obtain the probability of class, and classify the result to achieve object recognition.

$$y_r(x) = \frac{\exp(a_r(x))}{\sum\limits_{j=1}^{k} \exp(a_j(x))}, 0 \le y_r \le 1, \tag{10}$$

$$\sum_{r=1}^{k} y_r = 1, \tag{11}$$

$$P(c_r|x) = \frac{\exp(a_r(x))}{\sum\limits_{j=1}^{k} \exp(a_j(x))}, 0 \le P(c_r|x) \le 1, \tag{12}$$

where y_r is the softmax function, *x* is the input data of conditional probability, *k* is the number of classes, a_r and a_j are the element of each vector of specified layer, c_r is specified class to be calculated, and $P(c_r \mid x)$ is the class prior probability. For the error of multiclass classification, the cross-entropy loss can be expressed as in (13).

$$\text{loss (cross-entropy)} = \sum_{i=1}^{n} \sum_{j=1}^{k} t_{ij} ln(y_{ij}), \tag{13}$$

where *n* is the number of samples, *k* is the number of classes, t_{ij} is the indicator that the i^{th} sample belongs to the j^{th} class, y_{ij} is the output for sample *i* for class *j*. The regression layer is also used to add behind the fully connected layer, and the loss of HMSE (half-mean-squared error) of regression layer can be expressed as in (14).

$$\text{loss (HMSE)} = \frac{1}{2} \sum_{i=1}^{R} \frac{(t_i - y_i)^2}{R}, \tag{14}$$

where *R* is the number of responses, t_i is the target output, and y_i is the prediction of network for response *i*. After building the entire CNN structure, training of the CNN model starts. The most important thing is to minimize the value of the loss function. There are many methods to optimize the training process, instead of inputting all the data into the model at once, adjust the learning speed and number of trainings. Gradient descent method requires all training samples when updating each parameter; the training process becomes abnormally slow as the number of samples increases. SGD (stochastic gradient descent) is proposed to solve the disadvantage of gradient descent.

SGD optimizer was used to improve the training process of neural network and iteratively update the weights by calculating the gradient of the loss function on the small batch of data; the weight will be updated with (15).

$$\Delta w^{(l)} = -\eta \frac{\partial E(m)}{\partial w^{(l)}}, \tag{15}$$

where w is the weight value, E is the loss function, m is the training mode, η is learning rate, l is specified layer. In addition to using pooling layer to reduce parameters to prevent overfitting, dropout layer was used to discard some neurons randomly; the equation can be expressed as in (16).

$$z_i = \frac{1}{p}\sum_{j=1}^{N} w_{i,j}\left(a_j \times \text{Bernoulli}(p)\right), \tag{16}$$

where z_i is the output of dropout function, p is the specified probability, $w_{i,j}$ is the weight between the element and each element of previous layer, a_j is the element of each vector of specified layer, N is the number of vectors of specified layer, p is the specified probability, Bernoulli function takes the value 1 with probability p and the value 0 with the probability $1 - p$. The pseudo code is shown as in Algorithm 5.

Algorithm 5 CNN training

```
1: **input**: dataset
2: **output**: score of CNN trained model on testing dataset
3: **for** $f$ in dataset **do**
4:     $f_{\text{train}}, f_{\text{validate}}, f_{\text{test}} \leftarrow$ split dataset into training subset,
       validate subset, and test subset
5:     $M \leftarrow$ CNN ($f_{\text{train}}, f_{\text{validate}}$)
6:     $score \leftarrow$ evaluate ($f_{test}$, $M$)
7:     **return** $score$
8: **end for**
```

2.4. *Two-Wheeled Mobile Robot*

In order to realize the autonomous movement of the security robot, two-wheeled mobile robot was used. To achieve the best control, the mathematics model of two-wheeled robot was explored [25,26]. As shown in Figure 6, the coordinate system of the two-wheeled robot can be divided into two types. In inertial coordinate system, the coordinates can be expressed as in (17), and in robot coordinate system, the coordinates can be expressed as in (18). The relationship between the two coordinate systems and the orthogonal rotation matrix can be expressed as in (19) and (20).

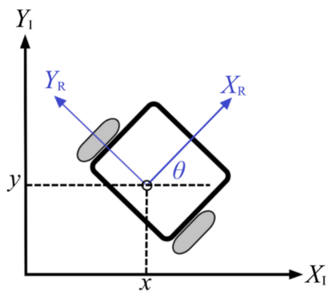


Figure 6. Two coordinate systems of two-wheeled robot.

$$\dot{\xi}_R = \begin{bmatrix} X_R \\ Y_R \\ \theta_R \end{bmatrix}, \tag{17}$$

$$\dot{\xi}_I = \begin{bmatrix} X_I \\ Y_I \\ \theta_I \end{bmatrix}, \tag{18}$$

$$\dot{\xi}_I = R(\theta)\dot{\xi}_R, \tag{19}$$

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{20}$$

where $\dot{\xi}_R$ is the robot coordinate system, $\dot{\xi}_I$ is the inertial coordinate system, X_I and Y_I are the inertial coordinates, X_R and Y_R are the robot coordinates, θ is the orientation angle, and R is the orthogonal rotation matrix. The kinematic modeling is the motion of mechanical systems without considering the forces that affect the motion; for the differential-drive mobile robot, as shown in Figure 7, the linear velocity and angular velocity in the robot coordinate can be expressed as in (21) and (22).

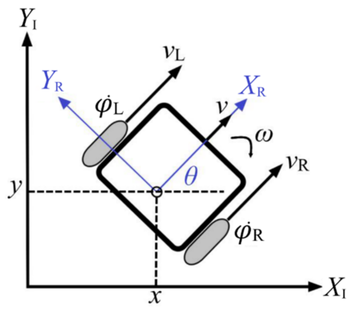


Figure 7. Forward kinematic model of two-wheeled robot.

$$v = \frac{v_R + v_L}{2} = r\frac{(\dot{\varphi}_R + \dot{\varphi}_L)}{2}, \tag{21}$$

$$\omega = \frac{v_R - v_L}{L} = r\frac{(\dot{\varphi}_R - \dot{\varphi}_L)}{2}, \tag{22}$$

where v is average speed of two rounds, ω is the angular velocity, v_R is the speed of right wheel, v_L is the speed of left wheel, L is the distance between left wheel and right wheel, r is the radius of the wheel, $\dot{\varphi}_R$ and $\dot{\varphi}_L$ are the angles of rotation of left wheel and right wheel. The velocities in the robot coordinate system can be expressed as in (23) and (24), and it can be also expressed in matrix form in (25).

$$\dot{X}_R = r\frac{(\dot{\varphi}_R + \dot{\varphi}_L)}{2}, \dot{Y}_R = 0, \tag{23}$$

$$\dot{\theta} = r\frac{(\dot{\varphi}_R + \dot{\varphi}_L)}{L}, \tag{24}$$

$$\begin{bmatrix} \dot{X}_R \\ \dot{Y}_R \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ 0 & 0 \\ \frac{r}{L} & -\frac{r}{L} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_R \\ \dot{\varphi}_L \end{bmatrix}, \tag{25}$$

For the partial moments of inertia, if the moment of inertia and mass of the two-wheeled robot are considered; the dynamic model, the partial moments of inertia, the matrix of Coriolis forces, the total mass m of the robot, and the moment of inertia J can be expressed as in (26) to (30).

$$M\begin{bmatrix} \dot{\varphi}_R \\ \dot{\varphi}_L \end{bmatrix} + V\begin{bmatrix} \varphi_R \\ \varphi_L \end{bmatrix} = B\tau, \tag{26}$$

$$M = \begin{bmatrix} J_\omega + \frac{r^2}{L^2}(m\frac{L^2}{4} + J) & \frac{r^2}{L^2}(m\frac{L^2}{4} - J) \\ \frac{r^2}{L^2}(m\frac{L^2}{4} - J) & J_\omega + \frac{r^2}{L^2}(m\frac{L^2}{4} + J) \end{bmatrix}, \tag{27}$$

$$V = \begin{bmatrix} 0 & \frac{r^2}{L} m_c d\omega \\ \frac{-r^2}{L} m_c d\omega & 0 \end{bmatrix}, \tag{28}$$

$$m = m_c + 2m_\omega, \tag{29}$$

$$J = m_c d^2 + m_\omega \frac{L^2}{2} + J_c + 2J_m, \tag{30}$$

where M is matrix of the moments of inertia, V is matrix of Coriolis forces, B is matrix of input, τ is matrix of motor torques, m is the mass, J is the moment of inertia, L is the distance between left wheel and right wheel, r is the radius of the wheel, m_c is the mass of the chassis, m_ω is the mass of the wheel, J_ω is the moment of inertia of each driving wheel with a motor about the wheel axis, J_c is the moment of inertia of the chassis, and J_m is the moment of inertia of the wheel. According to the kinematic model, the wheel angular velocities can be expressed as in (31), and the torque of the two wheels can be expressed as in (32) and (33).

$$\begin{bmatrix} \omega_{\mathrm{R}} \\ \omega_{\mathrm{L}} \end{bmatrix} = \begin{bmatrix} \frac{1}{r} & 0 \\ 0 & \frac{1}{r} \end{bmatrix} \begin{bmatrix} v_{\mathrm{R}} \\ v_{\mathrm{L}} \end{bmatrix}, \tag{31}$$

$$\tau_R = J_\omega \omega_R + \frac{rmv}{2} + \frac{rJ}{L}\omega + \frac{r^2}{L} m_c d \omega \dot{\varphi}_{\mathrm{L}}, \tag{32}$$

$$\tau_L = J_\omega \omega_L + \frac{rmv}{2} - \frac{rJ}{L}\omega + \frac{r^2}{L} m_c d \omega \dot{\varphi}_{\mathrm{R}}, \tag{33}$$

where ω_{R} *and* ω_{L} are the angular velocities of two wheels, and τ_{L} and τ_{R} are the torque of the two wheels.

2.5. Navigation and Hector SLAM Algorithm

In order to realize the navigation and obstacle avoidance of the robot, LiDAR was used to measure the range, direction, point cloud, and other parameters of target by irradiating a pulse of laser light to the target. As shown in Figure 8, the laser ranging is a triangulation measurement system.

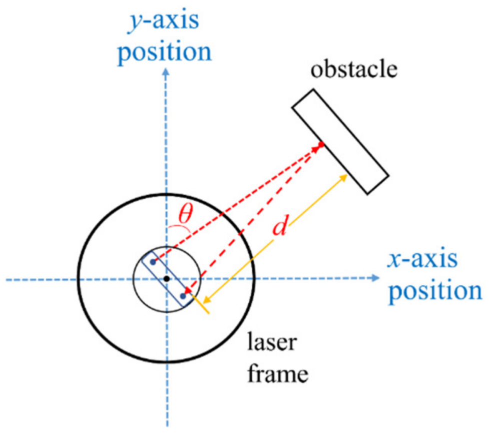


Figure 8. Triangulation measurement system of LiDAR.

SLAM technology enables the robots to localize in unknown environment. During the movement process, the LiDAR features such as walls and pillars are obtained by repeated scanning through LiDAR and self-localized to build a map to achieve simultaneous positioning and map construction. This technique was used in [27–29]. Hector SLAM is one of the most popular SLAM methods [30], and it is widely used in the research of mobile robots [31–35].

Hector SLAM algorithm was used to obtain the distance between the robot and the surrounding obstacles through the LiDAR sensor module. It uses a grid with a unit length of 1 to represent the surrounding environment, and each grid contains a current state value, which represents whether the grid is idle or occupied. The occupancy grid map is shown in Figure 9 as in [30]. The occupancy value and the gradient can be expressed as in (34) and (35).

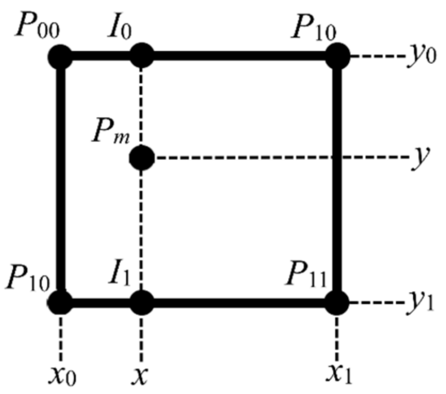


Figure 9. The occupancy grid map.

$$\begin{aligned} M(P_m) &\approx \tfrac{y-y_0}{y_1-y_0}\left(\tfrac{x-x_0}{x_1-x_0}M(P_{11}) - \tfrac{x_1-x}{x_1-x_0}M(P_{01})\right) \\ &+\tfrac{y_1-y}{y_1-y_0}\left(\tfrac{x-x_0}{x_1-x_0}M(P_{10}) - \tfrac{x_1-x}{x_1-x_0}M(P_{00})\right) \end{aligned} \tag{34}$$

$$\nabla M(P_m) = \left(\frac{\partial M}{\partial x}(P_m), \frac{\partial M}{\partial y}(P_m)\right), \tag{35}$$

where P_m is a single point of continuous map coordinate, $M(P_m)$ is the occupancy value of P_m, x and y are the coordinates in Figure 9, and P_{00}, P_{01}, P_{10}, P_{11} are the coordinates of grids surrounding. The derivatives of (34) can be expressed as in (36) and (37).

$$\begin{aligned} \tfrac{\partial M}{\partial x}(P_m) &\approx \tfrac{y-y_0}{y_1-y_0}(M(P_{11}) - M(P_{01})) \\ &+\tfrac{y_1-y}{y_1-y_0}(M(P_{10}) - M(P_{00})) \end{aligned} \tag{36}$$

$$\begin{aligned} \tfrac{\partial M}{\partial y}(P_m) &\approx \tfrac{x-x_0}{x_1-x_0}(M(P_{11}) - M(P_{10})) \\ &+\tfrac{x_1-x}{x_1-x_0}(M(P_{01}) - M(P_{00})) \end{aligned} \tag{37}$$

According to (34) to (37), the pose of the robot, the scan endpoint, and the pose of the robot in world coordinates can expressed as in (38) to (40).

$$\xi = (p_x, p_y, \theta)^T, \tag{38}$$

$$S_i = (S_{i,\,x}, S_{i,\,y})^T, \tag{39}$$

$$S_i(\xi) = \begin{pmatrix} \cos\theta - \sin\theta \\ \sin\theta \ \ \cos\theta \end{pmatrix}\begin{pmatrix} S_{i,\,x} \\ S_{i,\,y} \end{pmatrix} + \begin{pmatrix} p_x \\ p_y \end{pmatrix}, \tag{40}$$

where ξ is the rigid transformation, p_x and p_y represent the position of the robot, θ is the orientation angle, S_i is scan endpoint, $S_{i,\,x}$ is the scan endpoint of x-axis, $S_{i,\,y}$ is the scan endpoint of y-axis. Then, the future pose $\Delta\xi$ is estimated according to (41). By first-order Taylor expansion, (41) can be expressed as in (42), then the Gauss-Newton equation is used to obtain the optimum alignment with the map, and the new optimized pose of robot and updated scanning map are obtained.

$$\textstyle\sum_{i=1}^{n}(1 - M(S_i(\xi + \Delta\xi)))^2 \to 0, \tag{41}$$

$$\textstyle\sum_{i=1}^{n}\left(1 - M(S_i(\xi)) - \nabla M(S_i(\xi))\dfrac{\partial S_i(\xi)}{\partial \xi}\Delta\xi\right)^2 \to 0, \tag{42}$$

where n is the number of world coordinates. Using LiDAR on mobile robot and detecting the range of the environment around the robot, Hector SLAM algorithm was used to obtain the real-time position and orientation of the robot and the scanned map at the same time, to achieve robot navigation and obstacle avoidance. Navigation means that the robot can move steadily in space and effectively avoid obstacles by establishing the surrounding map environment.

The entire navigation stack architecture is shown in Figure 10. Through the scanning and mapping of the LiDAR sensor module, the dynamic map is obtained and input into the global planner. According to the specified goal of navigation, the global path is calculated. Since there may be dynamic obstacles on the estimated route, it is necessary to use the local planner of DWA algorithm to calculate a path that can avoid obstacles based on the local cost map and achieve robot navigation and obstacle avoidance. DWA is an algorithm of path planning [36], assuming that the objective function can be expressed as in (43).

$$G(v,\omega) = \sigma(\alpha \times angle(v,\omega) + \beta \times dist(v,\omega) + \gamma \times vel(v,\omega)), \tag{43}$$

where G is the objective function of DWA, v is velocity, ω is angular velocity, *angle* is measure of progress towards the goal location, *dist* is the distance to the closet obstacle on the trajectory, *vel* is the forward velocity of the robot, σ is a function that smooths the weighted sum of the three components and results in more side clearance from obstacles, and α, β, γ are the weighting factors for each one of those terms. The set V_a of admissible velocities can be expressed as in (44), and the set V_d of all reachable velocities in a time can be expressed as in (45).

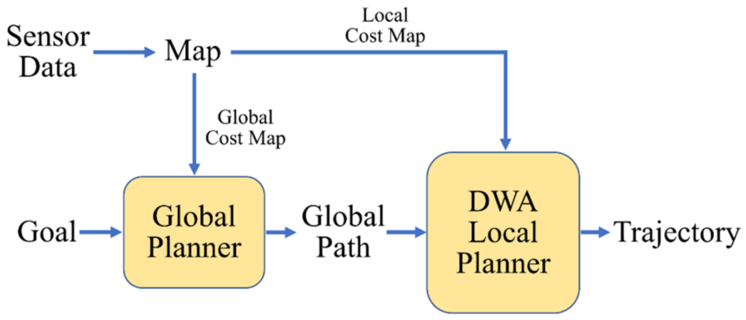


Figure 10. Navigation stack architecture.

$$V_a = \left\{ (v,\omega) \middle| \begin{array}{l} v \leq \sqrt{2 \cdot dist(v,\omega) \cdot \dot{v}_b} \\ \omega \leq \sqrt{2 \cdot dist(v,\omega) \cdot \dot{\omega}_b} \end{array} \right\}, \tag{44}$$

$$V_d = \left\{ (v,\omega) \middle| \begin{array}{l} v \in [v_a - \dot{v}t, v_a + \dot{v}t] \\ \omega \in [\omega_a - \dot{\omega}t, \omega_a + \dot{\omega}t] \end{array} \right\}, \tag{45}$$

where V_a is the admissible velocities, $\dot{v}_b$ is acceleration for breakage, $\dot{\omega}_b$ is angular acceleration for breakage, V_d is the reachable velocities, v_a is actual linear velocity, ω_a is actual linear angular velocity, $\dot{v}$ is acceleration, $\dot{\omega}$ is angular acceleration, and t is the time interval during the acceleration will be applied. Finally, the resulting search space can be expressed as in (46).

$$V_r = V_s \cap V_a \cap V_d, \tag{46}$$

where V_r is the intersection of the restricted areas, V_s is the space of possible velocities, V_a and V_d are mentioned as in (44) and (45). As shown in Figure 11, a goal is sent to the navigation stack to let the robot autonomously patrol and avoid obstacles in the indoor environment, then human faces are detected and the map is built in the process. If no human face is detected, patrolling continues until a human face is detected. If a face is detected, the obtained human face image is input into the trained CNN model to perform face recognition. If the result is an authorized person, face tracking is performed. Otherwise, a warning notification is sent to mobile phone. The pseudo code of program procedure is shown as Algorithm 6.

Algorithm 6 Program procedure of proposed security robot.

```
1:  while robot_is_running do
2:    if face_detect () = true then
3:      if face_recognize () > threshold then
4:        action = face_track ()
5:      else
6:        action = send_warn ()
7:      end if
8:    else
9:      action = robot_navigation ()
10:   end if
11: end while
```

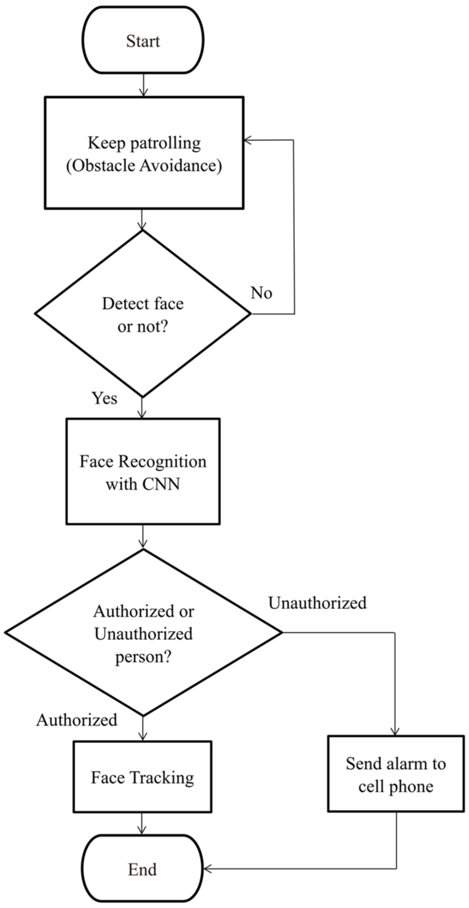

Start
Keep patrolling (Obstacle Avoidance)
Detect face or not?
No
Yes
Face Recognition with CNN
Authorized or Unauthorized person?
Unauthorized
Authorized
Face Tracking
Send alarm to cell phone
End

Figure 11. Program flow chart.

3. Results

Conventional surveillance is severely limited by many factors, such as different viewpoints and different objects potentially looking the same. In addition, significant changes in illumination and pose will greatly affect the accuracy of face recognition. However, several studies were tested in a static environment, but while a security robot is moving, there are more unstable factors to image recognition. The purpose of this study is to propose a visual perception system on a security robot that can identify specified persons during patrol under changes in illumination and pose and identify the target as the person to be tracked or a suspicious person. In this research, a deep learning facial recognition method was proposed, and an algorithm on the two-wheeled mobile robot and ROS environment was performed. The patrol part of the security robot combines the LiDAR sensor module to realize navigation and obstacle avoidance of an autonomous robot.

In order to solve the factors mentioned above that affect image recognition, many researchers have proposed many methods to improve it, such as circular symmetrical Gabor filter 2D PCA neural networks; a face recognition method based on directional histogram of oriented gradient; DSP for face recognition; the Lambert reflectance model; TBE-CNN, which extracts features from patches cropped around facial components; and deep DSN. Among them, Faster R-CNN and two CCD cameras were used; the experimental results showed that the average recognition rate is 96.33% under six different illumination and occlusion situations [11]. The DSN face recognition method was proposed; the experimental results showed that under 20 illumination changes and 6 different angle changes, compared with 8 different algorithms, the average rate of facial recognition of the DSN algorithm is 91.0% under different illuminations and different poses, which is better than other algorithms [16].

In the part relating to robot navigation and obstacle avoidance, many researchers have also proposed SLAM methods. Stereo VIL (visual-inertial LiDAR) SLAM was proposed; the experimental results showed that the ATE (absolute trajectory error) is less than 0.2 m [29]. An original 2D LiDAR SLAM based on point-and-line features was proposed; the experimental results showed that the absolute error of mapping is about 5% on average [33]. A visual-inertial SLAM was proposed. The experimental results showed that compared with ORB-SLAM (oriented fast and rotated brief) and OKVIS (open keyframe-based visual-inertial SLAM), the absolute trajectory of the root-mean-square error of the proposed SLAM is less than 0.1 m, which is better than other algorithms [28].

The Viola–Jones face detection algorithm was used to collect face images under various illumination changes, then three CNN models were trained, including AlexNet, VggNet, and GoogLeNet; the average accuracy of face recognition under four different illumination changes is 0.95, 0.95, and 0.6. The accuracy under a 15-degree viewpoint is 0.39, 0.7, and 0.63, respectively. In order to realize the navigation and obstacle avoidance of a mobile robot, the Hector SLAM algorithm and ROS navigation stack were used, so that the mobile robot can locate and build a map at the same time. In the first indoor environment, the RMSE of the mapping error was 0.134 m (0.957%), and in the second indoor environment it was 0.222 m (0.653%). The experimental flow diagram is shown in Figure 12.

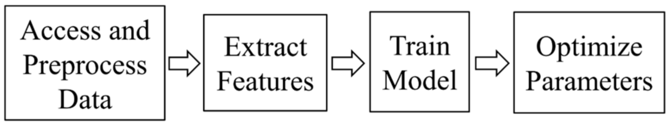


Figure 12. Experimental workflow diagram.

3.1. Face Detection and CNN Model Training

In the first experiment, the face detection of the Haar-like feature algorithm was tested in an indoor room environment. Human face images under different illumination changes were collected through the switching of different illuminations; there were a total of six fluorescent lamps in the scene—three lamps are directly above the target, and three lamps are located above the target, each with 20 watts. The top left figure shows six lights fully on, a total of 120 watts; the top right figure shows the three lights on the upper right that are fully on, a total of 60 watts; the bottom left figure shows three lights on the upper right that are fully on, a total of 60 watts; the bottom right figure shows that the lights are all off. As shown in Figure 13, some images of four different kinds of illumination changes were collected: 50 images under each illumination, and a total of 200 images of each target; as shown in Figure 14, images of 10 targets were collected. After obtaining the face images, the images must be preprocessed. First, the unclear face images were deleted, then the face images were cropped through the Viola–Jones algorithm, and finally the images were resized to a specific size. The images for the models of AlexNet and VggNet needed to be resized to 224 × 224, and the images for GoogLeNet were resized to 299 × 299.

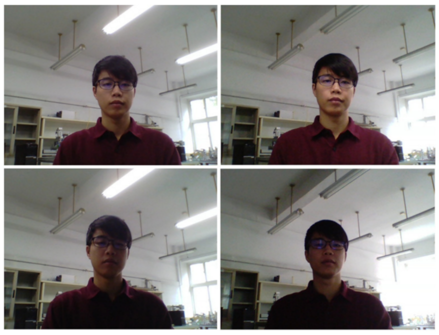


Figure 13. Different illumination images. (**Top left**) Normal; (**top right**) high illumination; (**bottom left**) middle illumination; (**bottom right**) low Illumination.

Figure 14. Collected images of ten targets.

A total of 200 images of each target were preprepared, including four kinds of different illumination changes, and containing five authorized people and five unauthorized people. Using the Keras model class, the three different CNN architectures of AlexNet, VggNet, and GoogLeNet were constructed, and then machine learning was realized through the sk-learn library. The machine learning model of the cross-validation method and the SGD optimizer were used. The images of the faces were saved in two different folders, and then the tags of the two folders were set. After the labeling was completed, the training parameters were set. The training data parameter was 70%, the validating data parameter was 30%, the test data parameter was 50%, the batch size is 20, the learning rate is 0.001, and the parameter of dropout is 0.5. After setting the parameters, the model is compiled and then training of the model starts, and finally the trained CNN model is obtained. Figures 15 and 16 show epoch to accuracy and epoch to loss of three CNN models. The comparison of the parameters of the three CNN models is shown as Table 1. The results of the experiment showed that AlexNet takes the least time, and has the fewest parameters, so the execution speed is faster. VggNet spends the most time and has the most parameters, and GoogLeNet is somewhere in between.

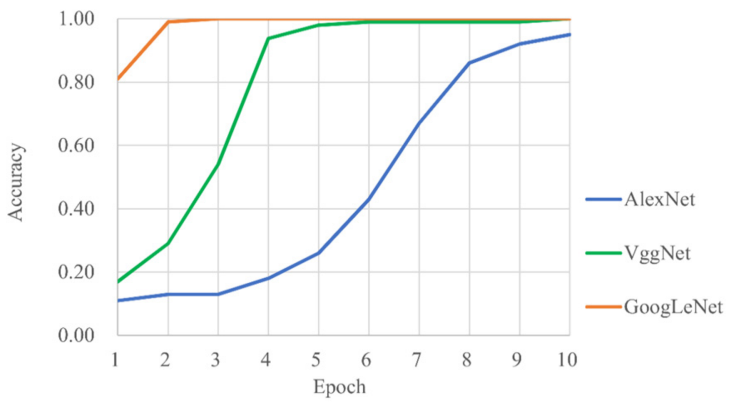


Figure 15. Accuracy–epoch curve of three CNN models.

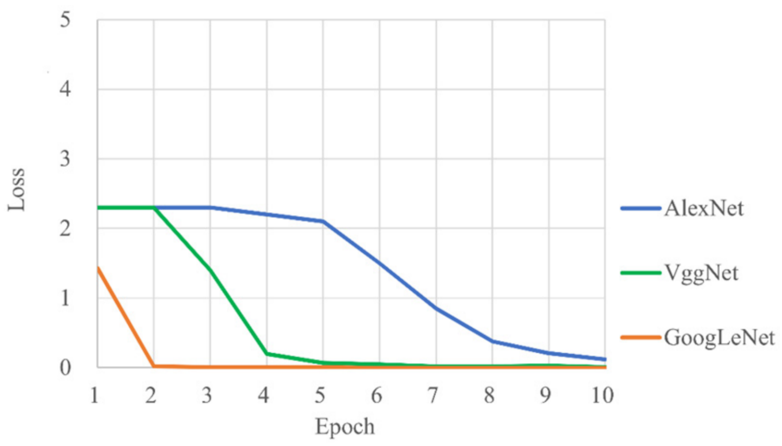


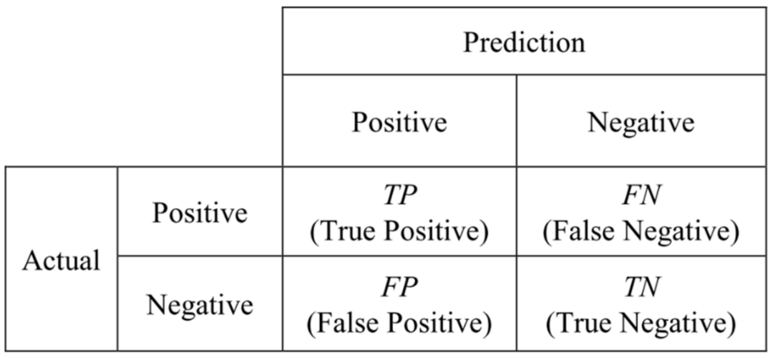
Figure 16. Loss–epoch curve of three CNN models.

Table 1. Parameters of three CNN models' training.

Specification	AlexNet	VggNet	GoogLeNet
Total Layer	8	16	22
Convolution Layer	5	13	21
Pooling Layer	3	5	5
Fully Connected Layer	3	3	1
Parameters	62,388,354	110,406,986	23,851,784
Training Accuracy	0.997	0.998	0.999
Cross Entropy Loss	0.12	0.006	0.004
Root-Mean-Squared Error	0.09	0.08	0.07
Training Time	22 min	210 min	120 min

3.2. Model Testing under Unstable Environment

For the model evaluation part, various perceptual algorithm evaluation indicators were used, including the speed, accuracy, complexity, ROC curve, precision and recall rate, P-R curve, and F_1-score of the CNN model. As shown in Figure 17, the confusion matrix, including the true positive rate, false negative rate, false positive rate, and true negative rate, is obtained first. According to (47), the accuracy is calculated. To further evaluate the model, the precision, specificity, recall (sensitivity), TPR (true positive rate), and FPR (false positive rate) were calculated; the equations can be expressed as in (48) to (52). To neutralize the error in precision and recall, the F_1-score was also calculated, which is the harmonic mean of precision and recall, and can be expressed as in (53).



		Prediction	
		Positive	Negative
Actual	Positive	*TP* (True Positive)	*FN* (False Negative)
	Negative	*FP* (False Positive)	*TN* (True Negative)

Figure 17. Confusion matrix.

$$\text{Accuaracy} = \frac{TP + TN}{TP + FP + FN + TN}, \tag{47}$$

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{48}$$

$$\text{Recall (Sensitivity)} = \frac{TP}{TP + FN}, \tag{49}$$

$$\text{Specificity} = \frac{TN}{TN + FP}, \tag{50}$$

$$\text{TPR} = \frac{TP}{TP + FN}, \tag{51}$$

$$\text{FPR} = \frac{FP}{FP + TN}, \tag{52}$$

$$\text{F}_1\text{-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{53}$$

where *TP*, *FP*, *TN*, *FN* are true positive, false positive, true negative, and false negative in the confusion matrix. After training the three CNN models, the model is first tested in an unstable environment where the illumination changes, and four illumination changes are achieved through the switching of the indoor illumination. The classification layer of the CNN model is 2, so that the robot can recognize the authorized and unauthorized people. According to the confusion matrix of Figure 17, the identification results of the three CNN models under four illumination changes are shown in Figures 18–20; the number of testing samples for each illumination is 100, including 50 unauthorized people and 50 authorized people; and the identification threshold is set to 0.8. The results of the experiment show that AlexNet and VggNet have higher recognition rates under different illumination, but GoogLeNet has worse performance. According to (47)–(53), the average metric core of model evaluation under four illumination changes is obtained; the comparison is shown in Table 2.

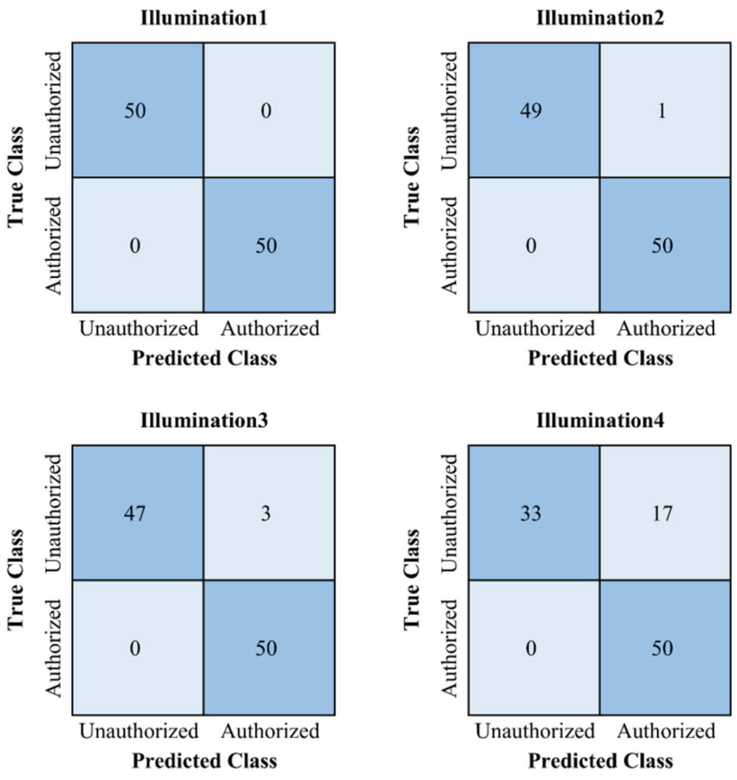


Figure 18. The confusion matrix of AlexNet under different illuminations.

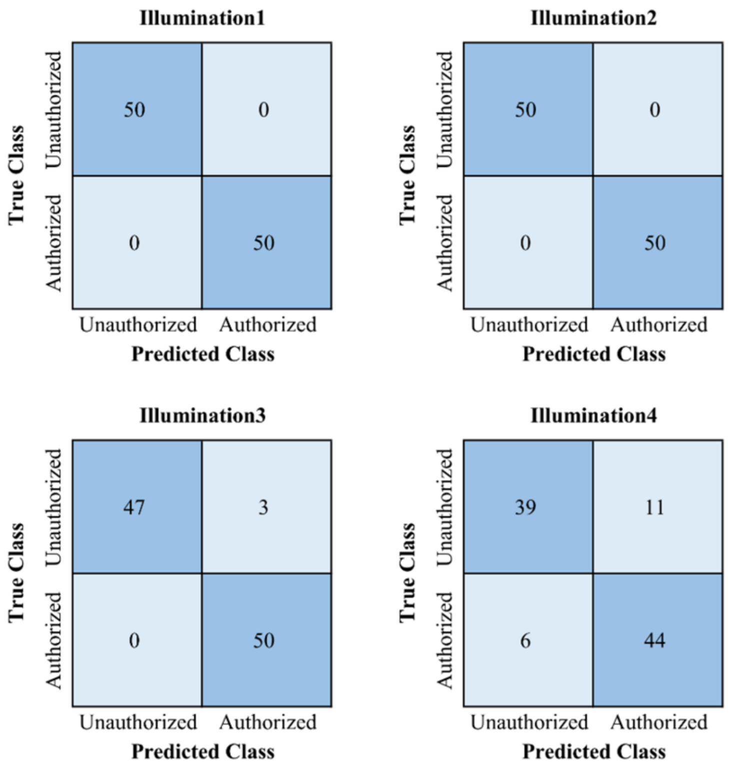


Figure 19. The confusion matrix of VggNet under different illuminations.

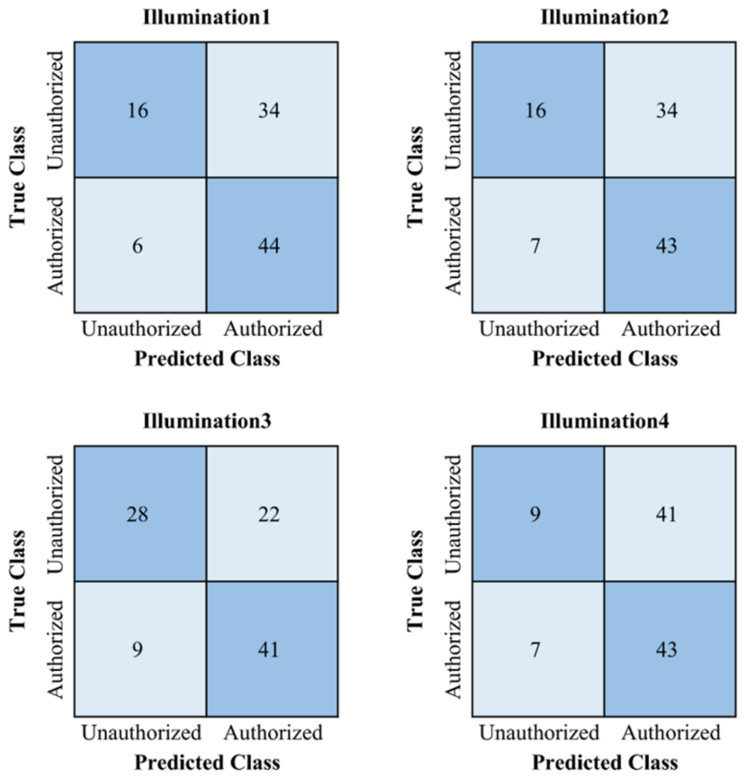


Figure 20. The confusion matrix of GoogLeNet under different illuminations.

Table 2. Metric comparison under different illuminations.

Specification	AlexNet	VggNet	GoogLeNet
Accuracy	0.95	0.95	0.60
Precision	1.00	0.97	0.69
Recall	0.90	0.93	0.35
F_1-score	0.94	0.95	0.45
Sensitivity	0.90	0.93	0.35
Specificity	1.00	0.97	0.86

In order to evaluate the recognition rate under different thresholds, as shown in Figures 21–26, the ROC curves and P-R curves of three CNN models under four illumination changes were drawn, and the ROC curves were further evaluated to find the AUC. The comparison of various models is shown in Table 3. The results of the experiment showed that AlexNet and VggNet can maintain good recognition rates under different illuminations. GoogLeNet has poor recognition rates under certain situations of illumination, so the AUC will be significantly different under four illumination changes.

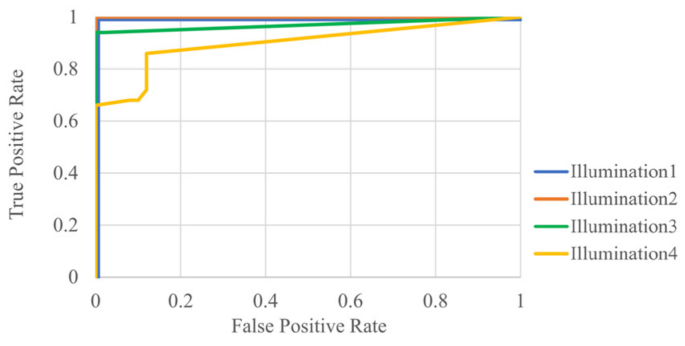


Figure 21. The ROC curve of AlexNet under different illuminations.

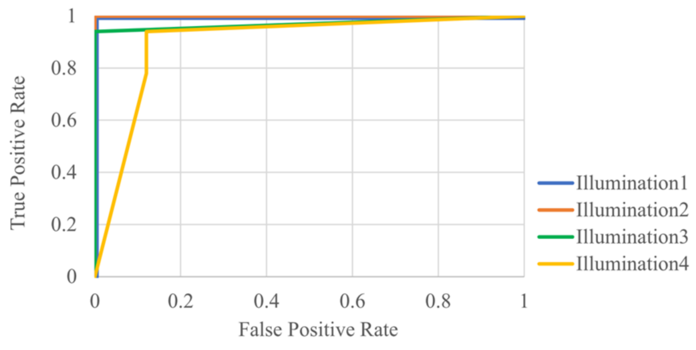


Figure 22. The ROC curve of VggNet under different illuminations.

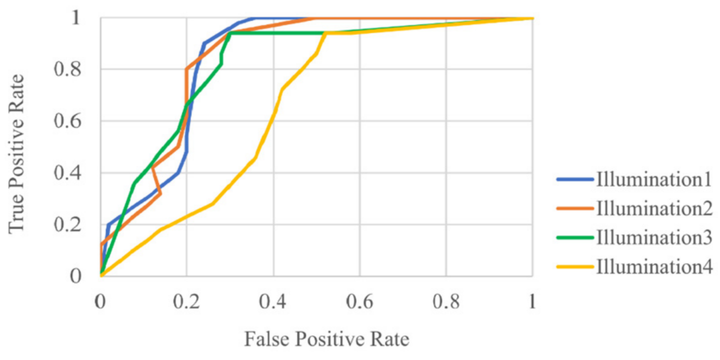


Figure 23. The ROC curve of GoogLeNet under different illuminations.

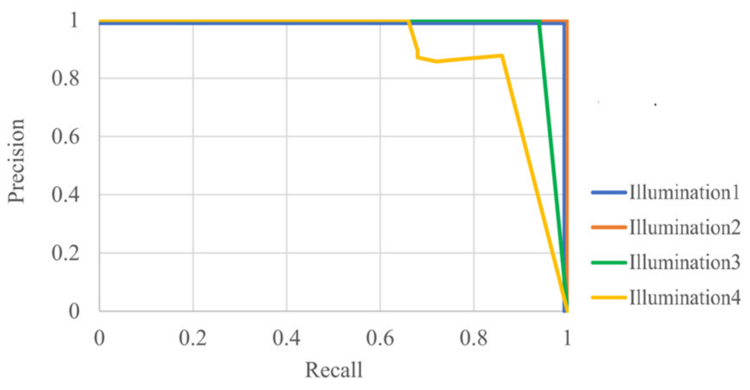


Figure 24. The P-R curve of AlexNet under different illuminations.

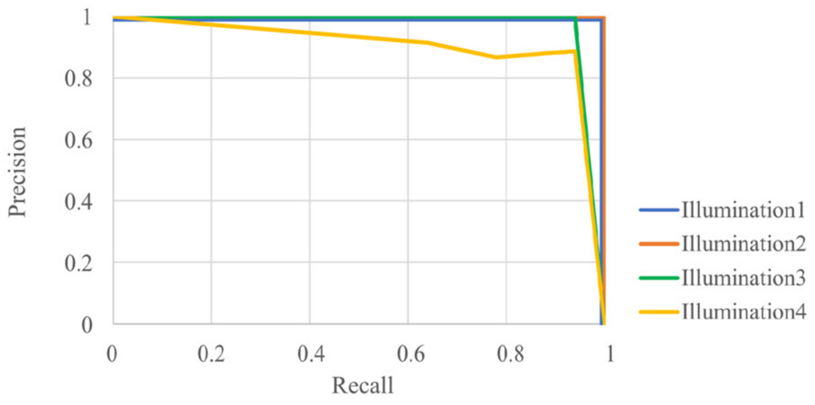


Figure 25. The P-R curve of VggNet under different illuminations.

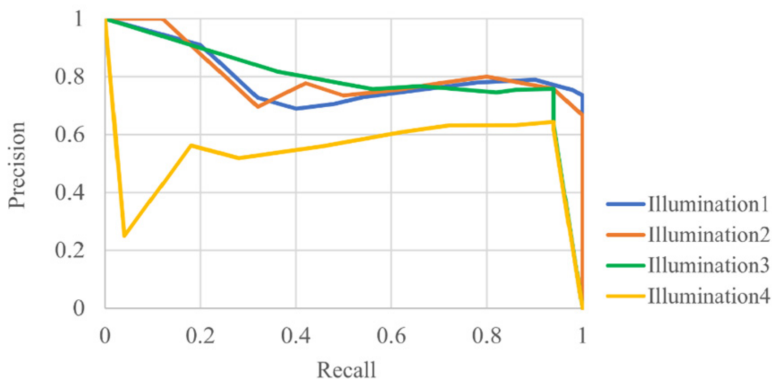


Figure 26. The P-R curve of GoogLeNet under different illuminations.

Table 3. AUC comparison under different illuminations.

Illumination	AlexNet	VggNet	GoogLeNet
Illumination1	1.00	1.00	0.84
Illumination2	1.00	1.00	0.84
Illumination3	0.96	0.97	0.82
Illumination4	0.87	0.90	0.66
Average	0.96	0.97	0.79

The second unstable environment to test is under different viewpoints; since the algorithm of face detection used in the research is the Haar-like frontal face cascade, the experimental unstable factor of viewpoint can only be up to 15 degrees, and the face will not be detected if this is exceeded. The confusion matrix of the three CNN models is shown in

Figure 27; the threshold is set to 0.7. The result showed that the accuracy of face recognition under different viewpoints is not as good as under different illuminations. The metric comparison of the three CNN models is shown in Table 4; the ROC curve and P-R curve of three CNN models are shown in Figures 28 and 29; the AUC value is shown in Table 5. The AUC values of AlexNet, VggNet, and GoogLeNet are 0.44, 0.73, and 0.71, respectively. The results showed that VggNet has the best recognition performance under viewpoints and different thresholds. GoogLeNet is second, while the AUC of AlexNet is only 0.44, and the average accuracy is only 0.39, which cannot achieve face recognition under uncertain factors of viewpoint.

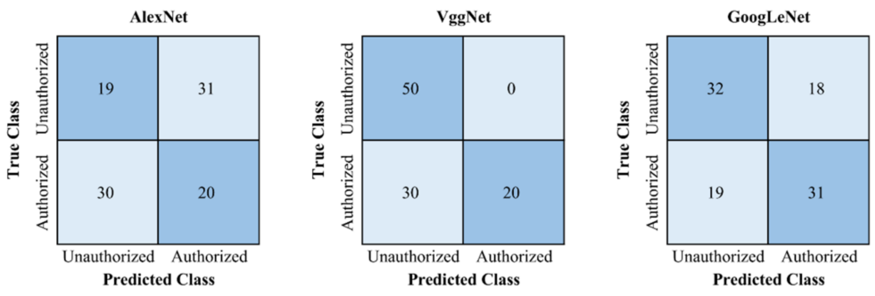


Figure 27. The confusion matrix of three CNN models under different viewpoints.

Table 4. Metric comparison under different viewpoints.

Metrics	AlexNet	VggNet	GoogLeNet
Accuracy	0.39	0.70	0.63
Precision	0.39	0.63	0.63
Recall	0.38	1.00	0.64
F_1-score	0.38	0.77	0.63
Sensitivity	0.38	1.00	0.64
Specificity	0.40	0.40	0.62

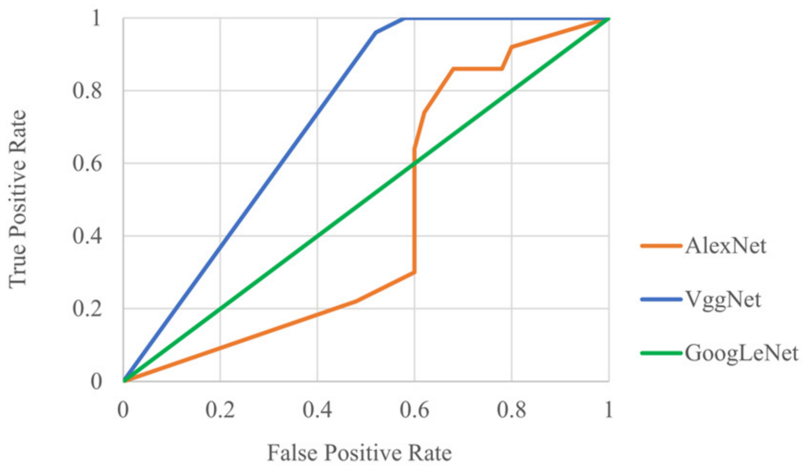


Figure 28. The ROC curve of three CNN models under different viewpoints.

Table 5. AUC comparison under different viewpoints.

AlexNet	VggNet	GoogLeNet
0.44	0.73	0.71

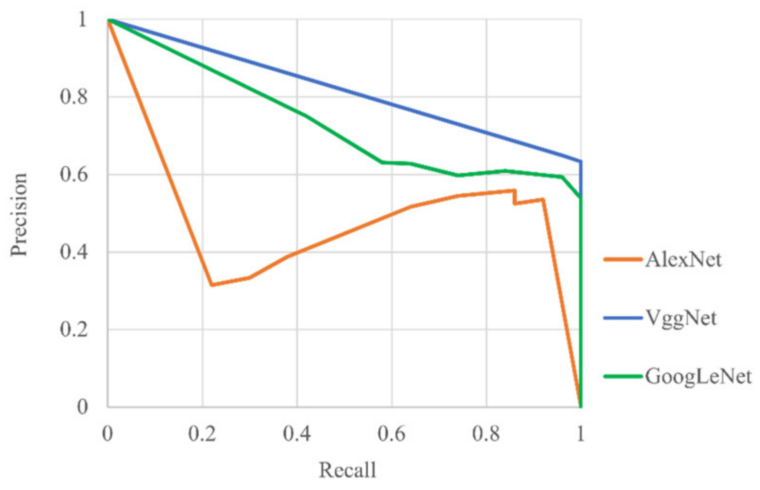


Figure 29. The P-R curve of three CNN models under different viewpoints.

The third unstable environment to test is the change in distance between robot and target; the average metrics of face recognition of three CNN models under different ranges are shown in Table 6, the threshold is set to 0.8, and the comparison of AUC of three CNN models is shown in Table 7. The results showed that AlexNet has a good performance in face recognition under the change of distance between robot and target, while the recognition rate of GoogLeNet will be greatly reduced when the range is far away, and the AUC is less than 0.8 and the average accuracy is only 0.56.

Table 6. Metric comparison under range change.

Metrics	AlexNet	VggNet	GoogLeNet
Accuracy	0.82	0.82	0.56
Precision	1.00	0.84	0.65
Recall	0.64	0.79	0.24
F_1-score	0.77	0.81	0.34
Sensitivity	0.64	0.79	0.24
Specificity	1.00	0.85	0.87

Table 7. AUC comparison under range change.

Range	AlexNet	VggNet	GoogLeNet
Near	0.99	0.91	0.82
Far	0.96	0.88	0.76
Average	0.98	0.89	0.79

For unstable environments under different occlusion changes, because the face detection algorithm used in this research can only detect the complete front human face, when the face is occluded, it will not be able to achieve face recognition, so the unstable factor of different occlusion conditions will not be discussed in this experiment. To improve this problem, other face detection algorithms must be used, such as the YOLO detection algorithm, which will be discussed in later sections.

3.3. Model Testing under Unstable Target

The first unstable target to test is a target without glasses; the average metrics of face recognition of the three CNN models when the target does not wear glasses are shown in Table 8, and the comparison of AUC of the three CNN models is shown in Table 9. The results showed that when the target is not wearing glasses, the face recognition performance of AlexNet and VggNet were both very good and the AUC of both is greater than 0.9, while the average accuracy of GoogLeNet is less than 0.6, which leads to poor evaluation metrics, and an AUC of only 0.73.

Table 8. Metric comparison when target is without glasses.

Metrics	AlexNet	VggNet	GoogLeNet
Accuracy	0.90	1.00	0.58
Precision	1.00	1.00	0.72
Recall	0.80	1.00	0.26
F_1-score	0.89	1.00	0.38
Sensitivity	0.80	1.00	0.26
Specificity	1.00	1.00	0.90

Table 9. AUC comparison when target is without glasses.

AlexNet	VggNet	GoogLeNet
1.00	1.00	0.73

The second unstable factor of the target to test is hairstyle change; the average metrics of face recognition of the three CNN models are shown as Table 10, and the comparison of AUC of the three CNN models is shown as Table 11. The results showed that if the hairstyle of the target is greatly changed, the face recognition rate will be greatly reduced. The average accuracy of face recognition of AlexNet is only 0.47, and the AUC is only 0.55; however, the performance of GoogLeNet is better than AlexNet and VggNet, with the AUC of GoogLeNet under hairstyle change being 0.99.

Table 10. Metric comparison under hairstyle change.

Metrics	AlexNet	VggNet	GoogLeNet
Accuracy	0.47	0.64	0.71
Precision	0.40	0.60	0.42
Recall	0.47	0.65	1.00
F_1-score	0.43	0.63	0.59
Sensitivity	0.47	0.65	1.00
Specificity	0.47	0.63	0.63

Table 11. AUC comparison under hairstyle change.

AlexNet	VggNet	GoogLeNet
0.55	0.79	0.99

The last unstable target to test is when authorized and unauthorized people appear at the same time. When two human faces are detected at a time, both targets will be framed, and the faces will be recognized through the CNN models, respectively, and the recognition rate of the two targets will be obtained. Therefore, it can still maintain the original performance of face recognition under the unstable factors of multiple targets appearing at the same time. To summarize the above, the average AUC values under all uncertain factors of the three CNN models were compared, as shown in Table 12; the result showed that VggNet has the best performance under uncertain factors among the three CNN models, followed by GoogLeNet, and finally AlexNet.

Table 12. AUC comparison under all unstable factors.

Uncertain Factors	AlexNet	VggNet	GoogLeNet
Illumination change	0.96	0.97	0.79
Viewpoint change	0.44	0.73	0.71
Different range	0.98	0.89	0.79
Without glasses	1.00	1.00	0.73
Hairstyle change	0.55	0.79	0.99
Average	0.79	0.88	0.80

3.4. Hector SLAM and Navigation of ROS Turtlebot

ROS is an open-source robot development platform. Under this common platform, robot developers can easily share robotic research such as navigation, perception, cognition, and motor driver algorithm, and also easily develop robot applications with the stable simulation test platform of ROS. As shown in Figure 30, there is also a good simulation environment, RVIZ, for easy development.

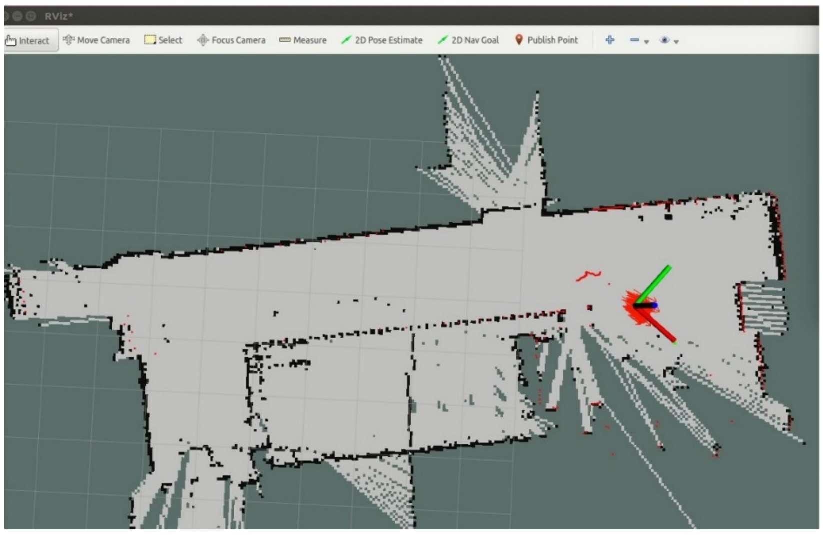


Figure 30. RVIZ simulation environment of ROS.

The ROS platform is often used in mobile robot control systems (navigation and obstacle avoidance). A dynamic obstacle avoidance method was proposed for mobile manipulators, which was verified in the ROS gazebo simulation environment [37]. DWA was used on the ROS two-wheeled robot to experiment with obstacle avoidance and path planning [38]. In addition, many researchers also developed robotic image recognition systems under ROS environments. Mobile robots and neural networks were used in ROS environments to realize the navigation and license plate recognition of autonomous mobile robots [39]. ROS robots and RGB-D cameras were used to achieve target location detection [40].

Turtlebot was equipped with RP-LiDAR-A1 with a sweep frequency of 5.5 Hz, range of 0.15 to 6 m, measurement frequency of 2000 Hz, and laser triangulation ranging technology. Turtlebot was also equipped with a USB webcam; the hardware configuration is shown in Figure 31. The program was run via PC and connected to Turtlebot via USB. The ROS environment was used to execute the program, and the ROS launch file was opened for face detection, face recognition, face tracking, RP-LiDAR, Hector SLAM, and navigation stack.

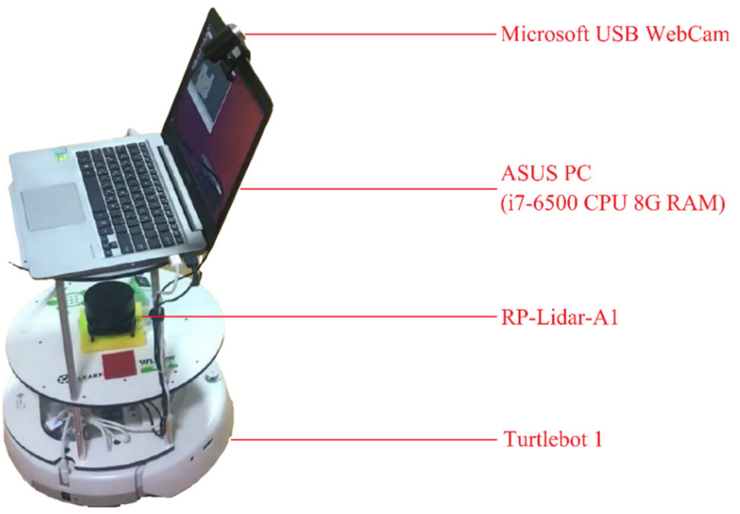


Figure 31. Turtlebot hardware configuration.

Through the navigation, obstacle avoidance, and map building of the mobile robot, the map of the indoor environment is obtained, and the result of the mapping is 5 cm per pixel. The experimental result of the mapping of the first indoor environment is shown in Figure 32, and the mapping error is shown in Table 13; through (54), the RMSE of the mapping error is 0.134 m and the percentage value is 0.957%. In order to improve the authenticity of the experimental environment, the second indoor environment of the experiment is shown in Figure 33. The mapping error is shown in Table 14; the RMSE of mapping error is 0.222 m, and the percentage value is 0.653%.

$$\text{Mapping Error (RMSE)} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(E_i - T_i)^2}, \tag{54}$$

where n is the number of side lengths to be calculated, E_i is the estimated value of the side length, and T_i is the true value of the side length.

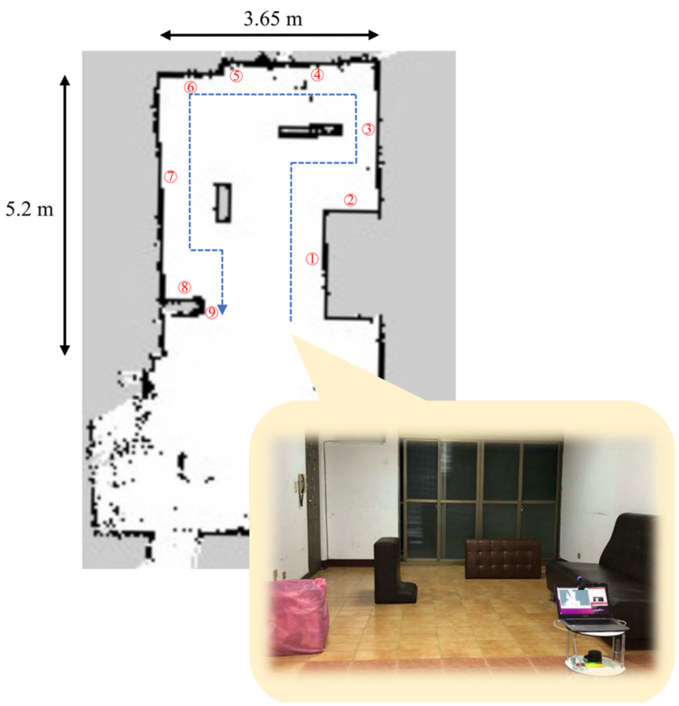


Figure 32. Mapping results of first environment, number 1–9 mean the length of each side.

Table 13. Mapping errors of first environment.

Number	Estimate Value (cm)	True Value (cm)	Absolute Error (cm)
1	185	180	5
2	85	90	5
3	200	210	10
4	230	255	25
5	15	13	2
6	110	110	0
7	445	474	29
8	25	25	0
9	45	43	2

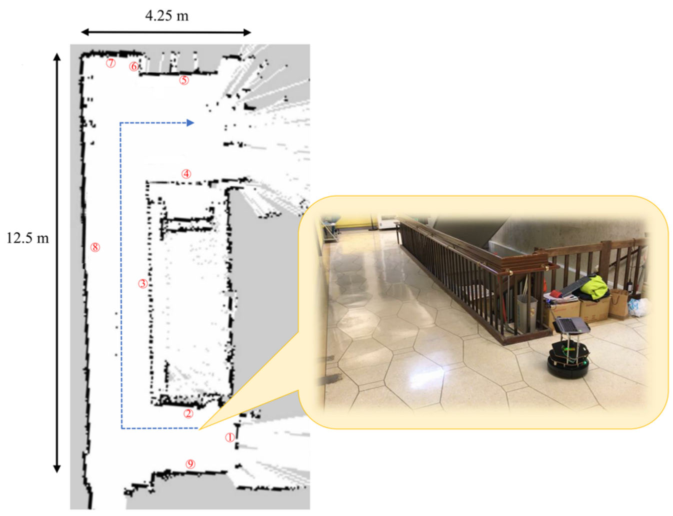


Figure 33. Mapping results of second environment, number 1–9 mean the length of each side.

Table 14. Mapping errors of second environment.

Number	Estimate Value (cm)	True Value (cm)	Absolute Error (cm)
1	185	190	5
2	255	260	5
3	650	680	30
4	235	245	10
5	245	245	0
6	75	90	15
7	155	180	25
8	1200	1250	50
9	255	260	5

Face detection, face recognition, navigation and obstacle avoidance of mobile robot are integrated, according to the system program procedure of Algorithm 6. When a human face is detected, it will start to perform face recognition and tracking; the experimental result of the first indoor environment is shown in Figure 34. If the recognition result is an unauthorized person, a warning message will be sent through the Line Notify service to the phone; the result is shown in Figure 35. When no human face is detected, the robot will perform an autonomous patrol and start navigation and obstacle avoidance. Because the odometer error of Turtlebot 1 was found to be relatively large during the experiment, the second indoor experimental environment was replaced with Turtlebot 2 for testing. The experimental result of the second indoor environment is shown in Figures 36–38; the green line represents the path planning of the patrol.

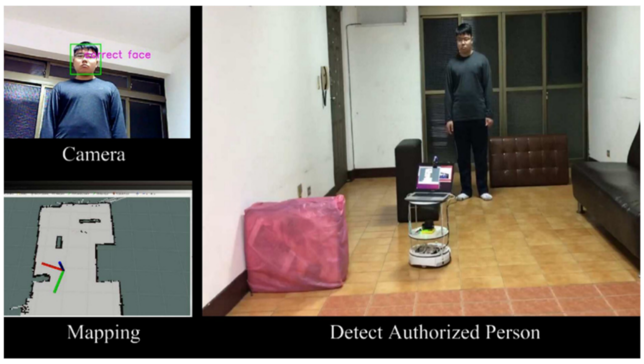
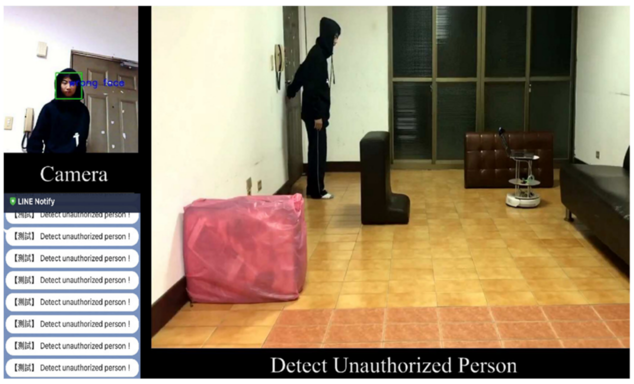


Figure 34. Face recognition of authorized target in first indoor environment.



【測試】 Detect unauthorized person !
【測試】 Detect unauthorized person !
【測試】 Detect unauthorized person !
【測試】 Detect unauthorized person !
【測試】 Detect unauthorized person !
【測試】 Detect unauthorized person !
【測試】 Detect unauthorized person !

Figure 35. Face recognition of unauthorized person in first indoor environment.

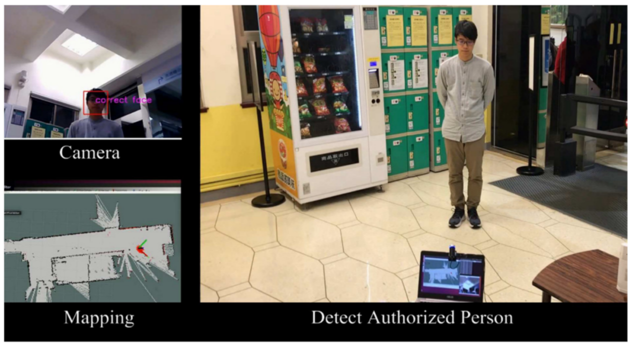
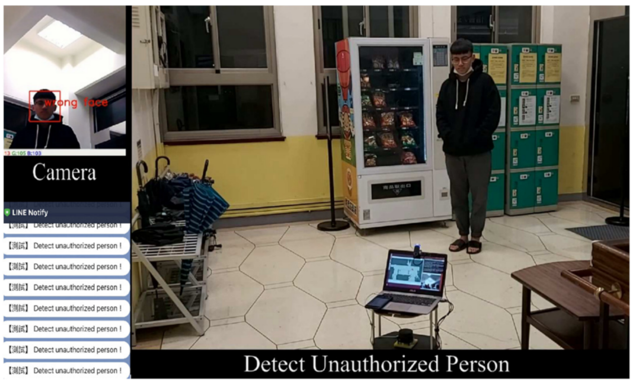


Figure 36. Face recognition of authorized target in second indoor environment.



【測試】 Detect unauthorized person !
【測試】 Detect unauthorized person !
【測試】 Detect unauthorized person !
【測試】 Detect unauthorized person !
【測試】 Detect unauthorized person !
【測試】 Detect unauthorized person !
【測試】 Detect unauthorized person !

Figure 37. Face recognition of unauthorized person in second indoor environment.

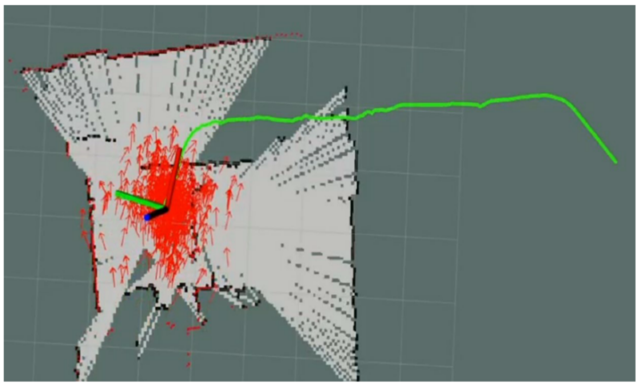


Figure 38. Path planning of the patrol of the second indoor environment.

4. Discussion

Summarizing the above results, the face recognition algorithm of the security robot proposed in this research showed that AlexNet and VggNet perform better under illumination changes. For home security, the main concern is sensitivity; the unauthorized target is not allowed to be regarded as the authorized target. As shown in Table 2, the sensitivity of AlexNet and VggNet are both greater than 0.9, while GoogLeNet is only 0.35. Therefore, considering the performance of the sensitivity, the most suitable algorithms are AlexNet and VggNet. However, if the viewpoint changes of security robots are considered, as shown in Table 4, the performance of AlexNet is poor, with the average accuracy, sensitivity, and specificity all being less than 0.5, which will greatly increase the chance of misjudging authorized people and missing unauthorized people. Under all uncertain factors, as shown in Table 7, VggNet has the best performance in face recognition, but from Table 1 it is shown that the CNN model of VggNet has the most parameters and takes the most time to train.

Compared with the experimental results of [11], the recognition rate under 6 different illumination and occlusion conditions is 0.96, while [16] has the recognition rate of 0.91 under 20 different illuminations and 6 different viewpoints. In this study, the average accuracy of face recognition under four different illumination changes was obtained; AlexNet and VggNet both reached 0.95, and the average accuracy of VggNet was 0.7 under a 15-degree viewpoint. However, when the target was under a viewpoint of more than 15 degrees, under occlusion, and when the hairstyle changes significantly, face recognition could not be achieved, because the face detection algorithm used in this study is a frontal face cascade. When under too-drastic changes to the face, face detection cannot be achieved. To overcome all of the above uncertain factors, a more complex algorithm of face recognition will be required, the required processor will be more demanding, it will be replaced by a GPU, and higher costs will be required.

In the part relating to robot navigation and mapping, the security robot can patrol autonomously through the LiDAR module and algorithms of Hector SLAM and ROS navigation. Different from the original algorithm, the map of the navigation stack was changed to a dynamic map, then the pose and position of the robot were obtained through the Hector SLAM algorithm, and finally a goal topic was published so that the robot could calculate a path to avoid obstacles. However, in the course of the experiment, it was found that the odometer of Turtlebot 1 has a large error. When the path is too tortuous, such as when it needs to turn or when the goal is too far, it will cause errors in the positioning of the robot, so it is replaced with Turtlebot 2, which greatly improves this problem, since no matter whether the path is very tortuous or far away, it can locate and navigate correctly.

In the part relating to building the map of the experimental results, compared with the experimental results of [29], the ATE of the proposed algorithm is 0.2 m, and the mapping error of the algorithm proposed in [33] is 5%; in this research, the experimental result of the RMSE of the mapping error is 0.134 m (0.957%) in the first indoor environment and 0.222 m (0.653%) in second indoor environment. The results showed that the map can be accurately built using the Hector SLAM algorithm and LiDAR module.

5. Conclusions

Routine surveillance of security robots is affected by many factors, including environmental instability factors, such as different illuminations, viewpoint, occlusion, and unstable factors in the target, such as changes of the hairstyle of target and whether they wear glasses. The above factors will affect the accuracy of face recognition. When the face recognition system is executed on a mobile robot, there will be more instability. This research proposes a deep learning algorithm of CNN on the security robot, which can identify authorized people under factors such as illumination changes and achieve autonomous movement and obstacle avoidance through the Hector SLAM and ROS navigation algorithms. When no human face is detected, it can autonomously patrol in the indoor environment and build a map.

Three CNN models, namely AlexNet, VggNet, and GoogLeNet, were used in this research. The evaluation indicators include accuracy, precision, recall, sensitivity, specificity, F_1-score, ROC curve, AUC value, and P-R curve. The first step of the experiment was to collect human face images under different illuminations, and then train three CNN models. The experimental results showed that AlexNet has the fewest parameters and the shortest training time, while VggNet has the largest number of parameters and training time, and GoogLeNet is somewhere in between. The next experiment is to test the performance of three CNN models under different unstable factors of environment. The experimental results showed that the average accuracy of the three CNN models is 0.95, 0.95, and 0.6 for face recognition performance under four different illuminations, respectively. The AUC of the ROC curve is 0.96, 0.97, and 0.79, respectively. Under a viewpoint change of 15 degrees, the average accuracy of the three CNN models is 0.39, 0.7, and 0.63, and the AUC of the ROC curve is 0.44, 0.73, and 0.71, respectively. Moreover, under the change of the distance between the robot and target, and when the target does not wear glasses, it can also have a good face recognition performance. However, when under occlusion, or when the hairstyle of target changes significantly, none of the three CNN models can achieve face recognition, because the face detection algorithm used is a frontal face cascade, so there are restrictions, as mentioned above.

In the part relating to the autonomous patrol of the security robot, in order to achieve navigation and obstacle avoidance of the robot, the LiDAR module and the Hector SLAM algorithm were used to achieve the robot locating and building a map at the same time. The experimental results showed that the robot moved autonomously and avoided obstacles in two indoor environments, while from measuring the error of the established map, the RMSE of the mapping error was 0.134 m and 0.222 m.

The experimental results showed that VggNet has the best performance under the influence of various uncertain factors; however, the model has the largest number of parameters, and the training speed and execution speed are the slowest. Although the proposed face recognition algorithm cannot achieve face recognition under all unstable factors, the combination of the frontal face detection algorithm and the two-dimensional CNN model is small in scale and relatively fast in execution. The algorithm can be applied on embedded systems, such as Raspberry Pi. If we want to improve the recognition rate, we need to use higher-level neural networks and object recognition algorithms; therefore, we need to greatly increase the cost and replace higher-level controllers. Otherwise, it will be impossible to identify strangers in time due to the large amount of computation, and it will not be possible to achieve an autonomous security robot. Therefore, this study adopts an algorithm with a lower computational amount and can achieve the expected effect.

The future extension of this research is to add a PTZ camera to allow the security robot to monitor all aspects, and to add a motion detection module, so that the monitoring system is not limited to human faces. When a dynamic change is detected, further face detection and recognition are performed. In addition, replacing the deep learning algorithm with a more complex model, such as replacing the Viola–Jones face detection algorithm with the YOLO object detection algorithm, will mean that the face recognition rate will not be restricted under occlusion or too-large changes in viewpoint, and the part relating to deep learning can also be replaced with models with higher accuracy, such as ResNet and Faster R-CNN, and the hardware can be replaced with a GPU controller, so that the security robot can achieve face recognition under more complex unstable factors.

Author Contributions: Conceptualization, M.-F.R.L.; methodology, M.-F.R.L. and Z.-S.S.; software, Z.-S.S.; validation, M.-F.R.L. and Z.-S.S.; formal analysis, M.-F.R.L.; investigation, M.-F.R.L.; resources, M.-F.R.L.; data curation, Z.-S.S.; writing—original draft preparation, Z.-S.S.; writing—review and editing, M.-F.R.L.; visualization, Z.-S.S.; supervision, M.-F.R.L.; project administration, M.-F.R.L.; funding acquisition, M.-F.R.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Ministry of Science and Technology (MOST) in Taiwan, grant number [108-2221-E-011-142-] and Center for Cyber-physical System innovation from the Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Dharmasena, T.; Abeygunawardhana, P. Design and Implementation of an Autonomous Indoor Surveillance Robot based on Raspberry Pi. In Proceedings of the 2019 International Conference on Advancements in Computing, Malabe, Sri Lanka, 5–7 December 2019.
2. Astua, C.; Crespo, J.; Barber, R. Detecting Objects for Indoor Monitoring and Surveillance for Mobile Robots. In Proceedings of the 2014 Fifth International Conference on Emerging Security Technologies, Alcala de Henares, Spain, 10–12 September 2014.
3. Kumari, V.R.; Sanjay, P.S. Smart Surveillance Robot using Object Detection. In Proceedings of the 2020 International Conference on Communication and Signal Processing, Chennai, India, 28–30 July 2020.
4. Mittal, S.; Rai, J.K. Wadoro: An autonomous mobile robot for surveillance. In Proceedings of the 2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems, Delhi, India, 4–6 July 2016.
5. Prykhodchenko, R.; Rocha, R.P.; Couceiro, M.S. People Detection by Mobile Robots Doing Automatic Guard Patrols. In Proceedings of the 2020 IEEE International Conference on Autonomous Robot Systems and Competitions, Ponta Delgada, Portugal, 15–17 April 2020.
6. Kong, J.; Chen, M.; Jiang, M.; Sun, J.; Hou, J. Face Recognition Based on CSGF (2D) 2 PCANet. *IEEE Access* **2018**, *6*, 45153–45165. [CrossRef]
7. Yin, X.; Liu, X. Multi-Task Convolutional Neural Network for Pose-Invariant Face Recognition. *IEEE Trans. Image Process.* **2017**, *27*, 964–975. [CrossRef] [PubMed]
8. Deng, J.; Pang, G.; Zhang, Z.; Pang, Z.; Yang, H.; Yang, G. cGAN Based Facial Expression Recognition for Human-Robot Interaction. *IEEE Access* **2019**, *7*, 9848–9859. [CrossRef]
9. Xiang, Z.; Tan, H.; Ye, W. The Excellent Properties of a Dense Grid-Based HOG Feature on Face Recognition Compared to Gabor and LBP. *IEEE Access* **2018**, *6*, 29306–29319. [CrossRef]
10. Liu, F.; Ding, Y.; Xu, F.; Ye, Q. Learning Low-Rank Regularized Generic Representation with Block-Sparse Structure for Single Sample Face Recognition. *IEEE Access* **2019**, *7*, 30573–30587. [CrossRef]
11. Wang, C.; Luo, T.; Zhao, L.; Tang, Y.; Zou, X. Window Zooming–Based Localization Algorithm of Fruit and Vegetable for Harvesting Robot. *IEEE Access* **2019**, *7*, 103639–103649. [CrossRef]
12. Hu, C.; Wu, F.; Yu, J.; Jing, X.; Lu, X.; Liu, P. Diagonal Symmetric Pattern-Based Illumination Invariant Measure for Severe Illumination Variation Face Recognition. *IEEE Access* **2020**, *8*, 63202–63213. [CrossRef]
13. Zhuang, L.; Guan, Y. Improvement of illumination-insensitive features for face recognition under complex illumination conditions. *J. Eng.* **2018**, *18*, 1947–1953. [CrossRef]
14. Hu, C.H.; Lu, X.B.; Liu, P.; Jing, X.Y.; Yue, D. Single Sample Face Recognition Under Varying Illumination via QRCP Decomposition. *IEEE Trans. Image Process.* **2018**, *28*, 2624–2638. [CrossRef] [PubMed]
15. Ding, C.; Tao, D. Trunk-Branch Ensemble Convolutional Neural Networks for Video-Based Face Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *40*, 1002–1014. [CrossRef]
16. Zhang, T.; Wang, H.; Dong, Q. Deep Disentangling Siamese Network for Frontal Face Synthesis Under Neutral Illumination. *IEEE Signal Process. Lett.* **2018**, *25*, 1344–1348. [CrossRef]
17. Freund, Y.; Schapire, R.E. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* **1997**, *55*, 119–139. [CrossRef]

18. Viola, P.; Jones, M. Rapid Object Detection using a Boosted Cascade of Simple Features. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Kauai, HI, USA, 8–14 December 2001.
19. Lienhart, R.; Maydt, J. An Extended Set of Haar-like Features for Rapid Object Detection. In Proceedings of the International Conference on Image Processing, Rochester, NY, USA, 22–25 September 2002.
20. Cristin, R.; Ananth, J.P.; Raj, V.C. Illumination-based texture descriptor and fruitfly support vector neural network for image forgery detection in face images. *IET Image Process.* **2018**, *12*, 1439–1449. [CrossRef]
21. Ibrahim, F.N.; Zin, Z.M.; Ibrahim, N. Eye Center Detection Using Combined Viola-Jones and Neural Network Algorithms. In Proceedings of the 2018 International Symposium on Agent, Multi-Agent Systems and Robotics, Putrajaya, Malaysia, 27–28 August 2018.
22. Uaday, M.A.; Shuzan, M.N.I.; Shanewaze, S.; Rakib, R.I.; Zaman, H.U. The Design of a Novel Multi-Purpose Fire Fighting Robot with Video Streaming Capability. In Proceedings of the 2019 IEEE 5th International Conference for Convergence in Technology, Bombay, India, 29–31 March 2019.
23. Talipu, A.; Generosi, A.; Mengoni, M.; Giraldi, L. Evaluation of Deep Convolutional Neural Network architectures for Emotion Recognition in the Wild. In Proceedings of the 2019 IEEE 23rd International Symposium on Consumer Technologies, Ancona, Italy, 19–21 June 2019.
24. Rasanayagam, K.; Kumarasiri, S.D.D.C.; Tharuka, W.A.D.D.; Samaranayake, N.T.; Samarasinghe, P.; Siriwardana, S.E. CIS: An Automated Criminal Identification System. In Proceedings of the IEEE International Conference on Information and Automation for Sustainability, Colombo, Sri Lanka, 21–22 December 2018.
25. Dhaouadi, R.; Hatab, A.A. Dynamic Modelling of Differential-Drive Mobile Robots using Lagrange and Newton-Euler Methodologies: A Unified Framework. *Adv. Robot. Autom.* **2013**, *2*, 107–113.
26. Jakub, C.; Anna, J.; Tomas, K. *Modeling and Control of Mobile Robot with Differential Chassis*; Faculty of Electrical Engineering and Informatics of the Technical University of Košice: Košice, Slovakia, 2015.
27. Jiang, B.; Zhu, Y.; Liu, M. A Triangle Feature Based Map-to-map Matching and Loop Closure for 2D Graph SLAM. In Proceedings of the 2019 IEEE International Conference on Robotics and Biomimetics, Dali, China, 6–8 December 2019.
28. Chen, C.; Zhu, H.; Wang, L.; Liu, Y. A Stereo Visual-Inertial SLAM Approach for Indoor Mobile Robots in Unknown Environments without Occlusions. *IEEE Access* **2019**, *7*, 185408–185421. [CrossRef]
29. Shao, W.; Vijayarangan, S.; Li, C.; Kantor, G. Stereo Visual Inertial LiDAR Simultaneous Localization and Mapping. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, Macau, China, 3–8 November 2019.
30. Kohlbrecher, S.; Von Stryk, O.; Meyer, J.; Klingauf, U. A flexible and scalable SLAM system with full 3D motion estimation. In Proceedings of the 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics, Kyoto, Japan, 1–5 November 2011.
31. Palacios, O.F.G.; Salah, S.H. Mapping marsian caves in 2D with a small exploratory robot. In Proceedings of the 2017 IEEE XXIV International Conference on Electronics, Electrical Engineering and Computing, Cusco, Peru, 15–18 August 2017.
32. Chen, X.; Zhang, H.; Lu, H.; Xiao, J.; Qiu, Q.; Li, Y. Robust SLAM system based on monocular vision and LiDAR for robotic urban search and rescue. In Proceedings of the 2017 IEEE International Symposium on Safety, Security and Rescue Robotics, Shanghai, China, 11–13 October 2017.
33. Wu, D.; Meng, Y.; Zhan, K.; Ma, F. A LiDAR SLAM Based on Point-Line Features for Underground Mining Vehicle. In Proceedings of the2018 Chinese Automation Congress, Xi'an, China, 30 November–2 December 2018.
34. Zhang, X.; Lu, G.; Fu, G.; Xu, D.; Liang, S. SLAM Algorithm Analysis of Mobile Robot Based on LiDAR. In Proceedings of the 2019 Chinese Control Conference, Guangzhou, China, 27–30 July 2019.
35. Wei, W.; Shirinzadeh, B.; Esakkiappan, S.; Ghafarian, M.; Al-Jodah, A. Orientation Correction for Hector SLAM at Starting Stage. In Proceedings of the 2019 7th International Conference on Robot Intelligence Technology and Applications, Daejeon, Korea, 1–3 November 2019.
36. Fox, D.; Burgard, W.; Thrun, S. The dynamic window approach to collision avoidance. *IEEE Robot. Autom. Mag.* **1997**, *4*, 23–33. [CrossRef]
37. Li, W.; Xiong, R. Dynamical Obstacle Avoidance of Task- Constrained Mobile Manipulation Using Model Predictive Control. *IEEE Access* **2018**, *7*, 88301–88311. [CrossRef]
38. Zhang, Y.; Xiao, Z.; Yuan, X.; Li, S.; Liang, S. Obstacle Avoidance of Two-Wheeled Mobile Robot based on DWA Algorithm. In Proceedings of the 2019 Chinese Automation Congress, Hangzhou, China, 22–24 November 2019.
39. Do, J.C.; Oh, J.M.; Lee, W.H. A License Plate Recognition using Neural Network and Autonomous Mobile Robot in Intelligent Parking Lot Management System. In Proceedings of the 2018 18th International Conference on Control, Automation and Systems, Pyeongchang, Korea, 17–20 October 2018.
40. He, H.; Wang, S.; Ma, S. Research on Object Location Detection based on ROS Machine Vision. In Proceedings of the 2019 Chinese Control And Decision Conference, Nanchang, China, 3–5 June 2019.