Proceedings of the 36th Chinese Control Conference
July 26-28, 2017, Dalian, China

Occupancy Measurement by Object Tracking at Building Entrances

Jianhong Zou, Qianchuan Zhao, *Senior Member, IEEE*, Rui Cong

Center for Intelligent and Networked Systems, Department of Automation, TNLIST, Tsinghua University, Beijing 100084, China
E-mail: zhaoqc@tsinghua.edu.cn

Abstract: Building occupancy measurement is becoming an increasingly important topic in the energy saving area. This paper proposes an occupancy measurement algorithm based on visual object tracking at entrances of buildings. The algorithm figures out variation of occupancy by comparing the size flags of the same pedestrian in adjacent frames. A CNN based deep learning tracker is designed to match multiple pedestrians in sequential frames. Its parameters are online uptated using the latest sample set after analysis of each frame. Five modes of occlusion are summarized and compensation mechanism is designed once the occlusion is detected by a Finite State Machine (FSM) model. The experimental result shows the good performance, especially in terms of robustness and accuracy. The algorithm is very valuable in measuring occupancy of buildings once it is integrated with the existing video surveillance software.

Key Words: Occupancy measurement, object tracking, deep learning, anti-occlusion tracking, sparse autoencoder

1 Introduction

Building occupancy measurement has attracted more and more attention in the area of energy saving. Past research shows that a good occupancy detection brings an energy saving potential up to 50% for lighting and 20% for HVAC [1].

Video analysis method is a promising approach to detecting occupancy and recognizing activity in buildings [2]. This method uses computer vision technology to process video images and gets information on persons, including the number, location and behavior. Video analysis is usually accomplished by detecting head, face and body contour.

Past research focuses on two approaches on vison-based methods. One is to detect and count heads in a region of interest relying on the surveillance videos. It easily fails once a head is blocked by background objects. Many cameras are required when the task is to work out the occupancy in a building. It demands to measure occupancy of buildings using the existing surveillance videos instead of new cameras. For example, Rougier *et al* used 3D ellipsoid to represent the head and tracked it with a hierarchical particle filter based on color histograms and shape information [3]. However, it is designed for one-person case and multiple people tracking have not been tested. Liu *et al* used multiple vision sensors in the room as well as at the entrance and measured occupancy by data fusion [4]. It detected heads by extracting the foreground pixels relying on an adaptive GMM and then splitting the moving body or head. The limitation includes deploying extra cameras and lack of real-time detection ability.

The other approach is to use a camera with top-down view to count people entering or exiting a room. This solution holds a relative high accuracy, but it suffers from extra expense of cameras and inevitable cumulative error. For instance, Cao *et al* hung a camera from the ceiling of the gate with a downward view and counted the number of people passing the gate [5]. They researched a strategy to solve the merging or splitting cases. However, this method needs extra cameras installed on the entrance. Moreover, it does not take into account occlusion cases, which often occur in the real surveillance videos and interfere the count of people.

*This work is supported by National Key Research and Development Project of China No. 2016YFB0901901, NSFC (Nos 61425027, 61174072, 61174105), and Special fund of Suzhou-Tsinghua Innovation Leading Action.

In this paper, we propose an occupancy measurement algorithm based on visual object tracking for entrances of buildings. The algorithm does not deploy eatra cameras and just makes use of the existing videos, which are recorded by the popular video surveillance system of buildings. Our algorithm takes into consideration various occlusion cases and variation of pedestrian appearance, which is very beneficial to its performance improvement. The occupancy is easily firgured out by measurements of occupancy variation at multiple building entrances.

2 Occupancy Measurement Based on Object Tracking

For entrances such as gate, floor and fire exit of buildings, occupancy viaration can be detected by tracking pedestrians and measuring their shifs. The boundary of an entrance is allowed to be marked manually on the video images. We can know that the occupancy in the building varies once a pedestrian passes through the boundary. The pedestrians can be found by object detection algorithms such as codebook model-based segmentation.

2.1 Pedestrian Feature Representation

In order to distinguish different pedestrians and quantify their motion, we use a feature vector ***v*** and a motion vector ***m*** to describe pedestrian static feature and dynamic feature, respectively.

A. Static feature representation

Denote p_i as the *i*-th pedestrian and *m* as the number of pedestrians, then the set of pedestrian areas detected in the *n*-th frame can be expressed as $P_n = \{p_1, \cdots, p_i, \cdots, p_m\}$.

Denotes w_i as the rectangular tracking window for p_i and its binary image is represented as

$$f_i(x,y) = \begin{cases} 1, (x,y) \in p_i \\ 0, (x,y) \notin p_i. \end{cases} \quad (1)$$

Denote (x_i, y_i) as the centroid coordinate of p_i, namely

$$\begin{cases} x_i = \frac{\sum_{x=1}^{N_i}\sum_{y=1}^{M_i} x f_i(x,y)}{\sum_{x=1}^{N_i}\sum_{y=1}^{M_i} f_i(x,y)} \\ y_i = \frac{\sum_{x=1}^{N_i}\sum_{y=1}^{M_i} y f_i(x,y)}{\sum_{x=1}^{N_i}\sum_{y=1}^{M_i} f_i(x,y)} \end{cases} \quad (2)$$

where N_i, M_i are the width and height of w_i, respectively. Denote S_i as the area of p_i (taking near-far effect into consideration), namely

$$S_i = (y_0 - y_i)^2 \sum_{x=1}^{N_i}\sum_{y=1}^{M_i} f_i(x,y) \quad (3)$$

where y_0 is the height of the video image.

The feature vector of the *i*-th pedestrian is defined as $\boldsymbol{v}_i =$

(x_i, y_i, S_i).

B. Dynamic feature representation

We use l_i (can also be written as $l(p_i)$) as the side flag of the i-th pedestrian. $l_i = 0$ means that p_i is located inside the building while $l_i = 1$ means that p_i is located outside the building. In addition, we introduce λ_i (can also be written as $\lambda(p_i)$) to represent the maximum internal that there is no match (MINM) for p_i. The relationship between λ_i and the location of p_i is:

- $\lambda_i = 0$: p_i is located inside the monitoring area and successfully detected;
- $0 < \lambda_i < \lambda_0$: p_i is located inside the monitoring area, but undetected because it is occluded by other pedestrians or objects;
- $\lambda_i \geq \lambda_0$: p_i has left the monitoring area.

The motion vector is defined as $\boldsymbol{m}_i = (l_i, \lambda_i)$.

2.2 Occlusion Model

Occlusion can be often observed in real surveillace videos when multiple pedestrians appear at near places. Common object tracking algorithms have a difficuty in anti-occlusion tracking and their effective is seriously affected. After careful observation of a large number of survellice videos, we summarize five modes of occulusion, namely ipsilateral occlusion, dispersive occlusion, converging occlusion, interim occlusion and merging occlusion, as shown in Fig. 1 .

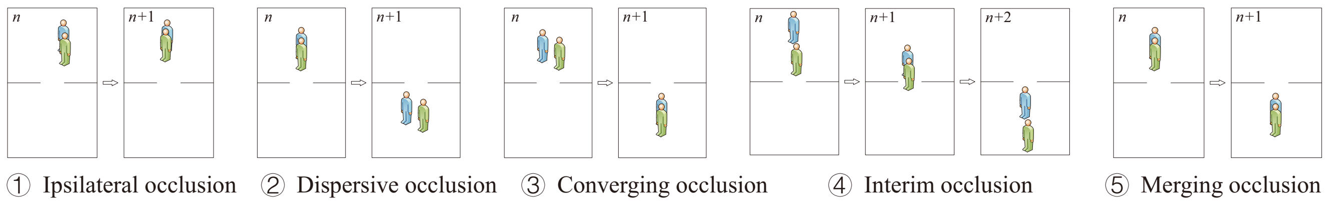

Fig. 1: Schematic diagram of five modes of occlusion in surveillance videos

Furthermore, we bulid model of the five modes of occlusion, as shown in Table 1. Each mode of occlusion has its decision criterion consisting of two or three relational expressions. In Table 1, (x_0, y_0) represents the coordinate of the lower right vertex of video image. x_t and y_t represent the minimum distance from a pedestrian to the boundary of the image in the horizontal and vertical direction, respectively. Our occupancy detection algorithm designs the anti-occlusion steps (in Section 2.3) based on the detection and compensation of occulsion.

Table 1: Model of common occlusion at entrances of buildings

Occlusion mode	Decision criterion
① Ipsilateral occlusion	a) $p_j^{(n)} \leftrightarrow p_i^{(n+1)}$ b) $l\left(p_j^{(n)}\right) == l\left(p_i^{(n+1)}\right)$ c) $S_i^{(n+1)} > S_\mathrm{u}$
② Dispersive occlusion	a) $m^{(n)} < m^{(n+1)}$ b) $\min\{\lvert x^{(n+1)}\rvert, \lvert x^{(n+1)} - x_0\rvert\} > x_\mathrm{t}$ c) $\min\{\lvert y^{(n+1)}\rvert, \lvert y^{(n+1)} - y_0\rvert\} > y_\mathrm{t}$
③ Converging occlusion	a) $m^{(n)} > m^{(n+1)}$ b) $\min\{\lvert x^{(n)}\rvert, \lvert x^{(n)} - x_0\rvert\} > x_\mathrm{t}$ c) $\min\{\lvert y^{(n)}\rvert, \lvert y^{(n)} - y_0\rvert\} > y_\mathrm{t}$
④ Interim occlusion	a) $p_{k+i}^{(n+1)} \leftarrow p_i^{\prime(n)}$ b) $p_{k+i}^{(n+1)} \leftrightarrow p_j^{(n+2)}$
⑤ Merging occlusion	a) $p_j^{(n)} \leftrightarrow p_i^{(n+1)}$ b) $l\left(p_j^{(n)}\right) \neq l\left(p_i^{(n+1)}\right)$ c) $S_i^{(n+1)} > S_\mathrm{u}$

We use a finite state machine (FSM) to model the MINM (Fig. 2). Denote Q as the state set, I as the input set, s as the initial state, F as the end state set, and δ as the transformation function, then the FSM is defined as FA = $\{Q, I, s, F, \delta\}$. In our model, Q = {0, 1, 2, …, λ_0, w} where λ_0 is the threshold of MINM and w represents pedestrian's exit, I = {m, n, o} where m, n, o represent match, no match and occlusion, respectively, s = 0, F = {w}, and δ *is defined as*

$$\begin{cases} \delta(q, m) = 0, \\ \delta(q, n) = q + 1, \\ \delta(q, o) = q \end{cases} \tag{4}$$

where $q \in \{0, 1, 2, \ldots, \lambda_0\}$.

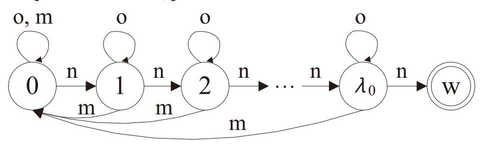

Fig. 2: Finite state machine model of MINM

2.3 Object Tracking Based Occpancy Detection

To measure the occupancy variation at entrances of builidngs,we propose an object tracking-based algorithm as shown in Algorithm 1. Algorithm 1 tries to find pedestrian rectangular area in the current frame matching to the rectangular area of the same pedestrian in the previous frame. Then it calculates the variation of occupancy by comparing the side flags of the pedestrain in the neighboring frames. Meanwhile, it tries to detect various modes of occlusion at several steps and compensates the occupancy measurement for occlusion. The static and synamic features play a important role in the match and the compensation.

Algorithm 1: Measurement of occupancy variation based on object tracking

```
1.  $P^{(n)} = \left\{p_1^{(n)}, \cdots, p_i^{(n)}, \cdots, p_k^{(n)}\right\}$, $n \leftarrow 1$.
    Compute $\boldsymbol{v}_j^{(n)}$ and $\boldsymbol{m}_j^{(n)}$ and set $\lambda_j^{(n)} \leftarrow 0, j = 1, \ldots, k$.
2.  $P^{(n+1)} = \{p_1^{(n+1)}, \cdots, p_i^{(n+1)}, \cdots, p_k^{(n+1)}\}$.
    Compute $\boldsymbol{v}_j^{(n+1)}$ and $\boldsymbol{m}_j^{(n+1)}$ ,$j = 1, \ldots, k$.
    **for** $i$ = 1 to $k$ **do**
      Find $p_j^{(n)}$in $P^{(n)}$ matching to $p_i^{(n+1)}$and set $\lambda_i^{(n+1)} \leftarrow 0$.
      **if** $P^{(n)} = \emptyset$ or there is no match, **then**
        **if** $p_i^{(n+1)}$ satisfies condition of occlusion mode ②,
        **then** dispersive occlusion occurs for $p_i^{(n+1)}$, and set
```

```
                $in \leftarrow in + l\left(p_z^{(n)}\right) - l\left(p_i^{(n+1)}\right)$
                where $z = \arg\max_{1 \le i \le k} S_i$.
            else $p_i^{(n+1)}$ is a new pedestrian.
            end if
        else if $p_i^{(n+1)}$ satisfies condition of occlusion mode ⑤,
        then merging occlusion occurs for $p_i^{(n+1)}$ and set
            $in \leftarrow in + \left[6S_i^{(n+1)}/S_{\mathrm{u}} - 4.5\right]\left(l\left(p_j^{(n)}\right) - l\left(p_i^{(n+1)}\right)\right)$
        else set $in \leftarrow in + l\left(p_j^{(n)}\right) - l\left(p_i^{(n+1)}\right)$
        end if
     end for
3.  In step 2, the elements matching to none of pedestrians in $P^{(n)}$
    are denoted as $p_1^{\prime(n)}, p_2^{\prime(n)}, \cdots, p_q^{\prime(n)}$. To compensate for
    potential interim occlusion, add $p_i^{\prime(n)} (1 \le i \le q)$ to $P^{(n+1)}$,
    namely, $p_{k+i}^{(n+1)} \leftarrow p_i^{\prime(n)}, \boldsymbol{v}_{k+i}^{(n+1)} \leftarrow \boldsymbol{v}\left(p_i^{\prime(n)}\right)$,
    $l_{k+i}^{(n+1)} \leftarrow l\left(p_i^{\prime(n)}\right), \lambda_{k+i}^{(n+1)} \leftarrow \lambda\left(p_i^{\prime(n)}\right) + 1, i = 1,2,\cdots,q$.
    Update $P^{(n+1)} \leftarrow \{p_1^{(n+1)}, p_2^{(n+1)}, \cdots, p_{k+q}^{(n+1)}\}$.
    for $i$ = 1 to $q$ do
        if $p_i^{\prime(n)}$ satisfies condition of occlusion mode ③,
        then converging occlusion occurs for $p_i^{\prime(n)}$, set
            $in \leftarrow in + l\left(p_i^{\prime(n)}\right) - l\left(p_z^{(n+1)}\right)$, and update
            $\boldsymbol{v}_{k+i}^{(n+1)} \leftarrow \boldsymbol{v}\left(p_z^{(n+1)}\right), l_{k+i}^{(n+1)} \leftarrow l\left(p_z^{(n+1)}\right), \lambda_{k+i}^{(n+1)} \leftarrow$
            $\lambda\left(p_i^{\prime(n)}\right)$ where $z = \arg\max_{1 \le i \le k} S_i^{(n+1)}$.
        end if
    end for
4.  Update $P^{(n+1)} \leftarrow \{p_i^{(n+1)} | p_i^{(n+1)} \in P^{(n+1)}, \lambda_i <$
    $\lambda_0, S_i^{(n+1)} \in [S_{\mathrm{l}}, S_{\mathrm{u}}], i = 1, \ldots, k+q\}$.
5.  $n++$. goto step 2.
```

Algorithm 1 has designed mechnism of anti-occlusion tracking to decrease error. If $p_j^{(n)}$ in P_n matches to $p_i^{(n+1)}$ in P_{n+1}, the occupancy variation *in* can be computed by the change of side flag:

$$in = l(p_j) - l(p_i). \tag{5}$$

If there is match to $p_i^{(n+1)}$, we further check whether merging occlusion has occurred for $p_i^{(n+1)}$. Or we check whether dispersive occlusion has occurred. Sometimes there are several pedestrians in P_n matching to none of pedestrians in P_{n+1}. The reason for no match is either occlusion or exiting in the *n*-th frame. These pedestrians as cache are added to P_{n+1} and their MINMs increase by one. If there is pedestrians in P_{n+2} matching to them, the interim occlusion is confirmed. Or the MINMs should increase by one again. Once the MINM reaches to the threshold, we consider that the pedestrian has left the building. We also compensate *in* for potential converging occlusion of $p_i^{\prime(n)}$.

The object tracking and matching algorithm used at step 2 of Algorithm 1 is shown as Algorithm 2. It is designed to find out the best match to a pedestrian in the (*n*+1)-th frame from sevaval pedestrians with similar features in the *n*-th frame.

Algorithm 2: Object tracking and matching

```
1.  $n \leftarrow 1$. Initialize the tracker T($W$).
    (i)   $P_1 = \{p_1, \ldots, p_{k'}\}$. Centroid coordinate of $p_i$ is denoted
          as $(x_i, y_i)$.
          for $i$ = 1 to $k'$ do
              Generate 17 samples with the same label $i$ by shifting
              $(x_i, y_i)$ to $((x_i)_l^d, (y_i)_l^d)$ where $d = 5, 10$; $l =$
              $\frac{r}{4}\pi, r = 0, \pm1, \pm2, \pm3, 4$.
          end for
          The $17k'$ samples compose sample set $S_1$.
    (ii)  Train the tracker T($W$) by the sample set $S_1$ and get the
          parameter set $W_1$.
2.  Detect pedestrians in the ($n$+1)th frame and get the pedestrian
    set $P_{n+1} = \{p_1, \cdots, p_i, \cdots, p_k\}$. Set $j \leftarrow 1$, $S_{n+1} \leftarrow S_n$.
3.  for $j$ = 1 to $k$ do
    (i)   Input $p_j \in P_{n+1}$ into the tracker T($W_n$) and get out $O_n =$
          $[o_1, o_2, \ldots, o_{k'}]$. $o_m = \max_{1 \le i \le k'} o_i$.
    (ii)  Compare $o_m$ with the upper threshold $\sigma_1$ and the
          lower threshold $\sigma_2$ $(\sigma_1 \ge \sigma_2)$.
          if $o_m < \sigma_2$,
          then $p_j$ is a new pedestrian in the ($n$+1)th frame.
              Shift $(x_j, y_j)$ to $((x_j)_l^d, (y_j)_l^d)$ where $d =$
              $5, 10$; $l = \frac{r}{4}\pi, r = 0, \pm1, \pm2, \pm3, 4$ and get 17
              pedestrian rectangular areas. Add them to $S_{n+1}$ as
              a new class of samples.
          else if $\sigma_2 \le o_m \le \sigma_1$,
          then $p_j \leftrightarrow p_m$.
              Shift $(x_j, y_j)$ to $((x_j)_l^d, (y_j)_l^d)$ where $d =$
              $5, 10$; $l = \frac{r}{4}\pi, r = 0, \pm1, \pm2, \pm3, 4$ and get 17
              pedestrian rectangular areas. Add them to the class
              of samples with label $m$ in $S_{n+1}$.
              if the number of samples with class label $m$ is great
              than the class volume V,
              then remove the first 17 samples with class label $m$.
              end if
          else if $o_m > \sigma_1$,
          then $p_j \leftrightarrow p_m$.
          end if
    (iii) $j++$.
    end for
4.  Update $S^{(n+1)} \leftarrow \{s_j^{(n+1)} | s_j^{(n+1)} \in S^{(n+1)}, \lambda_j < \lambda_0, S_j^{(n+1)} \in$
    $[S_{\mathrm{l}}, S_{\mathrm{u}}], j = 1, \ldots, |S^{(n+1)}|\}$.
5.  Train the tracker T($W$) by the sample set $S_{n+1}$ and get the
    parameter set $W_{n+1}$.
6.  $n++$. goto step 2.
```

$(Q_i)_l^d$ is the pixel whose D_8 distance in the direction l from Q_i (the centroid of $p_i^{(1)}$) equals d. The coordinate of $(Q_i)_l^d$ is $(Q_i)_l^d = ((x_i)_l^d, (y_i)_l^d)$ where

$$(x_i)_l^d = \begin{cases} x_i + d, l = \frac{n\pi}{4}, n = -3, -2, -1 \\ x_i,\ l = \frac{n\pi}{4}, n = -4, 0 \\ x_i - d, l = \frac{n\pi}{4}, n = 1,2,3 \end{cases} \tag{6}$$

$$(y_i)_l^d = \begin{cases} y_i + d, l = \frac{n\pi}{4}, n = -3, -2, -1 \\ y_i,\ l = \frac{n\pi}{4}, n = -4, 0 \\ y_i - d, l = \frac{n\pi}{4}, n = 1,2,3. \end{cases} \tag{7}$$

T(W) is a deep learning tracker whose design will be given in Section 3. After each frame is analyzed, the parameters of T(W) needs to be online updated by training with new set of samples. New samples are generated by the following rules:

- If a new pedestrian is detected, we creat a new class of pedestrians with a new label.
- If the apprance feature of a pedestrian varies, we add new samples with the latest feature. We set the maxmum volume of sample set as V to avoid too large sample set and low efficiency of network training. We use the FIFO rule to deal with the case that the number

of samples in a certain class exceeds V. That is to substitude the lateset samples for the first ones.

- If a pedestrain has left the monitoring area or is a false positive, we delete its samples.

We use new sample set S_{n+1} to train T(W) and get the parameters W_{n+1}. In the training process the initial values of parameters are W_n rather than random values. S_{n+1} differs from S_n but resembles S_n. Therefore, W_n makes the network close to convergence at the beginning of the training process. As a result, the training process takes a little time and the tracker has a good real-time performance. Parameter fine-tuning improves the adaptive capacity and the robustness under light variation, shape change and partial occlusion.

3 Deep Learning Tracker

3.1 Design of Deep Learning Tracker

The deep learning tracker used in Algorithm 2 is designed as Fig. 3.

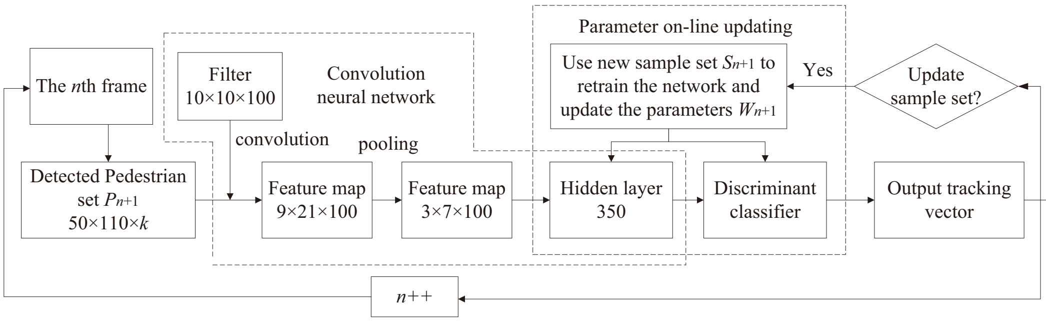
Parameter on-line updating
The nth frame
Filter 10×10×100
Convolution neural network
Use new sample set S_{n+1} to retrain the network and update the parameters W_{n+1}
Yes
Update sample set?
convolution
pooling
Detected Pedestrian set P_{n+1} 50×110×k
Feature map 9×21×100
Feature map 3×7×100
Hidden layer 350
Discriminant classifier
Output tracking vector
O_n
n++

Fig. 3: Structure of online tracker based on CNN feature extraction

The deep learning tracker works as the following process. Adjust all the pedestrian rectangle areas so that their sizes are 50 × 110 and then input them into the CNN for feature extraction. CNN computes the feature of the pedestrian area by convolution and pooling. The feature is input into the discriminant classifier and the classifier outputs a tracking vector representing the probabilities that the pedestrian is classified as different classes. If the tracking result indicates a new pedestrian, variation of pedestrian appearance, exiting the monitoring area or false positive, the samples are updated by the rules described in Section 2.3 and the network consisting of the hidden layer and the discriminant classifier is trained by the latest samples. Thus, the parameters of the tracker are redetermined and ready for analysis of the next frame.

The filter is a group of pre-trained feature vectors, which are used as convolution kernels. It is produced by off-line, unsupervised feature learning. In other words, it is obtained by training sparse autoencoder. Once the filter is determined, it is not updated any more in the procedures of tracking.

The CNN has an excellent ability to extract feature of pedestrian appearance. Because the convolution and pooling operation make the extracted feature very robust against shifting, scaling, rotation and deformation. In addition, the CNN is very effective in extracting the feature of the edge between the background and the foreground.

The discriminant classifier's mathematical model is a SoftMax function:

$$h_\theta\left(x^{(i)}\right) = \begin{bmatrix} P\left(y^{(i)} = 1|x^{(i)};\theta\right) \\ P\left(y^{(i)} = 2|x^{(i)};\theta\right) \\ \vdots \\ P\left(y^{(i)} = k|x^{(i)};\theta\right) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^{\mathrm{T}} x^{(i)}}} \begin{bmatrix} e^{\theta_1^{\mathrm{T}} x^{(i)}} \\ e^{\theta_2^{\mathrm{T}} x^{(i)}} \\ \vdots \\ e^{\theta_k^{\mathrm{T}} x^{(i)}} \end{bmatrix} \tag{8}$$

where $\{\theta_1, \theta_2, \dots, \theta_k \in R^{n+1}\}$ are the model parameters and $\{1,2,\dots,k\}$ are the class labels. $h_\theta\left(x^{(i)}\right)$ represents the probabilities that $x^{(i)}$ is classified as different classes.

3.2 Network Training

The partial network consisting of the hidden layer and the discriminant classifier needs to be retrained after analysis of each frame. The network's cost function is

$$J(\theta, W, b) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k} I\left(y^{(i)} = j\right)\lg\frac{e^{\theta_j^{\mathrm{T}} x^{(i)}}}{\sum_{i=1}^{k} e^{\theta_i^{\mathrm{T}} x^{(i)}}} + \frac{\lambda}{2}\sum_{i=1}^{k}\sum_{j=0}^{n}\theta_{ij}^2 + \frac{\lambda}{2}\sum_{i=1}^{2100}\sum_{j=1}^{350} w_{ji}^2 \tag{9}$$

where $x^{(i)}$ and $y^{(i)}$ are the feature map and the class label of the *i*-th image output by the CNN, respectively, w_{ji} is the weight that connects the *i*-th pixel in the feature map to the *j*-th node in the hidden layer, b is the bias term, and λ is the weight decay coefficient.

We combine the back propagation algorithm and the batch gradient descent algorithm to train the network. Our training algorithm is shown as Algorithm 3.

Algorithm 3: Training the cascade network consisting of the hidden layer and the disciminant classifier

```
1. Feedforward pass calculation. Calculate the feature map ($x$), weighted sum of the hidden layer ($z$), vector of activation value ($a$), and classification probability vector ($P$).
2. Calculate the error term
       $\delta = \frac{\partial J(\theta, W, b)}{\partial z} = -\theta^{\mathrm{T}}(I - P) \circ f'(z)$
   where $I$ is the vector of class labels, and $\circ$ is the sign of Hadamard product.
3. Calculate the partial derivative
       $\nabla_\theta J(\theta, W, b) = -x(I - P) + \lambda\theta$
       $\nabla_W J(\theta, W, b) = \frac{\partial J(\theta, W, b)}{\partial z} \cdot \frac{\partial z}{\partial W} = \delta x^{\mathrm{T}}$
       $\nabla_b J(\theta, W, b) = \frac{\partial J(\theta, W, b)}{\partial z} \cdot \frac{\partial z}{\partial b} = \delta$
4. Update the parameters
       $\theta = \theta - \alpha\nabla_\theta J(\theta, W, b)$
       $W = W - \beta\nabla_W J(\theta, W, b)$
       $b = b - \beta\nabla_b J(\theta, W, b)$
5. Repeat steps 1~4 until convergence.
```

4 Experimental Data and Setup

4.1 Experimental Data

We find through research that few open datasets of videos monitor entrances of buildings. Therefore, we build a video dataset using the videos exported from the video surveillance system in the Tsinghua Future Information Technology Building. We name this dataset Buildings' Entrance and Exit Surveillance Videos Dataset (BEESVD). The BEESVD contains 2100 video files whose total volume is 1965 GB and length is 720h. The frame rate is 25 fps and image resolution is 1280 × 720.

The BEESVD as a dataset is challenging to video analysis algorithms because it contains rich interference factors.

- Light varies obviously with the season and time.
- Mirror image is often formed on the glass door and on the floor tile with smooth surface.
- Pedestrians, leaves and vehicles outside the building are all moving background.
- Various modes of occlusion often occur near the gate.
- Pedestrian size and appearance change during his walk.
- Multiple pedestrians may have the similar appearances.

4.2 Experimental setup

The frame rate (25 fps) of the original videos in the BEESVD is very high. Therefore, we firstly pre-process the videos by reducing the frame rate to 1/30. Other parameters of the proposed algorithm are determined as Table 2.

We use C++ to program on a computer with an Intel Core i7-6700k CPU and a Kingston DDR4 2133 32G RAM. The IDE is Visual Studio 2013 and the CNN module is realized by the Caffe frame. The filter is produced by training the sparse autoencoder on the Tiny Images Dataset [6]. In the tracking process, the network parameters are online updated using Algorithm 3.

Table 2: Parameters of the proposed algorithm

Parameters	Value
Image resolution	1280 × 720
Threshold of pedestrian area (S_u)	25000
Size of pedestrian rectangular	50 × 110
Threshold of image edge (x_t, y_t)	(120, 120)
Threshold of MINM (λ_0)	8
Upper threshold of classification probability (σ_1)	0.9
Lower threshold of classification probability (σ_1)	0.6
Volume of sample set (V)	34
Step size of filter sliding	5
Size of filter	10 × 10

5 Experimental results and discussion

5.1 Experimental Results

We select a video in the BEESVD as a sample to test the proposed algorithm. The 274th frame is set as the initial frame. Run the algorithms and the experimental result is shown in Fig. 4. The red line indicates the entrance (gate) set by the user. The blue rectangle is the detected pedestrian window. The yellow line indicates the displacement figured out by the centroids of the same pedestrian in two adjacent frames with 1.2s interval. The values of feature vectors of pedestrians in video frames are listed in Table 3.

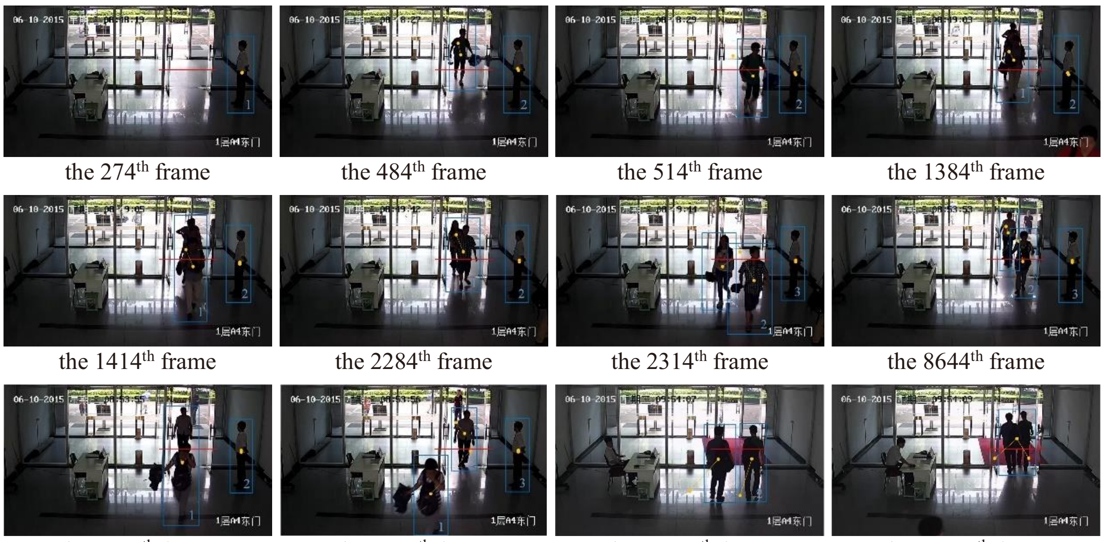
the 274th frame, the 484th frame, the 514th frame, the 1384th frame
the 1414th frame, the 2284th frame, the 2314th frame, the 8644th frame
the 8674th frame, the 8704th frame, the 99004th frame, the 99034th frame

Fig. 4: Results of object tracking in video frames

From Fig. 4 and Table 3 we can see that merging occlusion, dispersive occlusion, interim occlusion and converging occlusion occurred in the 1384th, 2284th, 8674th and 99034th frames, respectively. Because our algorithms compensated *in* for the occlusions, the measurement of occupancy variation were all finally accurate. It was unnecessary to compensate for ipsilateral occlusion, so our algorithms did not need to detect this kind of occlusion. Although the pedestrians varied in their sizes and appearances in the adjacent frames, our algorithms succeeded in tracking them continuously. This demonstrates good effectiveness and robustness of the algorithms owing to the CNN based feature extraction, online update of network parameters and anti-occlusion mechanism.

5.2 Performance and Comparison

The performance of the proposed algorithm is compared on the BEESVD with the well-known object tracking algorithms such as incremental visual tracking (IVT) [7], sparse collaborative model (SCM) [8] and multiple instance learning (MIL) [9], as shown in Table 4.

From Table 4 we can see that our algorithm peforms well in four respects:

- Precision in object tracking. The center location error (CLE) and success rate approach those of other algorithms while the overlap is a little smaller. This is because our algorithm has considered multiple modes of occlusion.

Table 3: Feature values of pedestrian areas

n		m	(x, y)	S	l	λ	in
n_1	274	1	(1132, 328)	15194	0	0	0
n_8	484	2	(849, 235)	13542	1	0	0
			(1126, 335)	15194	0	0	
n_9	514	2	(940, 335)	21518	0	0	1
			(1143, 325)	15195	0	0	
n_{38}	1384	2	(866, 260)	29260	1	0	1
			(1125, 328)	14343	0	0	
n_{39}	1414	2	(900, 340)	38328	0	0	5
			(1122, 327)	14404	0	0	
n_{68}	2284	2	(872, 268)	28437	1	0	9
			(1140, 320)	13553	0	0	
n_{69}	2314	3	(792, 342)	23424	0	0	11
			(941, 401)	30087	0	0	
			(1140, 318)	15610	0	0	
n_{280}	8644	3	(840, 180)	13975	1	0	39
			(911, 316)	18138	0	0	
			(1145, 320)	15613	0	0	
n_{281}	8674	2	(850, 370)	32256	0	0	39
			(1130, 320)	15621	0	0	
n_{282}	8704	3	(720, 519)	31298	0	0	39
			(881, 232)	18118	1	0	
			(1144, 321)	15616	0	0	
n_{3292}	99004	2	(776, 350)	27750	0	0	533
			(936, 349)	23310	0	0	
n_{3293}	99034	1	(882, 259)	36110	1	0	531

Table 4: Performance of the proposed algorithm and its comparison with other algorithms

Performance	Index	Proposed algorithm	IVT [7]	SCM [8]	MIL [9]
Precision of tracking	Center location error	6	5	5	7
	Overlap rate	0.82	0.90	0.88	0.85
	Success rate	44%	44%	45%	42%
Efficiency	FPS	9 fps	8 fps	8 fps	6 fps
Robustness	SRE	best	bad	bad	good
Accuracy of occupancy	Correctness	87%	76%	69%	81%
	Error	9%	27%	34%	18%

- Efficiency. The proposed algorithm takes about 110 ms to analyze one video frame, satistying the practical requirement. The time is mainly spent in the CNN feature extraction and the pramameter update. Our algorithm is more efficient than the other algorithms because a) the filter is produced offline, b) the CNN takes a simple structure, and c) just the hidden layer and the regression layer need to be retrained during the continous tracking process.
- Robustness. Our algorithm performs the best in terms of SRE. The robustness is mainly attributed to the CNN and the compensation for occulusion.
- Accuracy in occupancy measurement. We use correctness and error to evaluate accuracy of occupancy measurement. Correcteness is defiend as the propotion that the number of frames where pedestrains are exactly tracked accounts for of the total frame number. Error is defined as the propotion that the difference between the occupancy measurement and the ground truth accounts for of the ground truth. The correctness of the proposed algorithm is a little higher than those of other algorithms and the error is much lower. Although the presision of our algorithm approaches those of other algorithms, the algorithm is much more accurate in occupancy measurement than them.

We also tested all the algorithms on the Multiple Object Tracking Benchmark (MOT16) [10]. The result showed that their performances were close and our algorithm held no signifivant advantage over the others. It indicates that the proposed algorithm is most suitable for video scenes of building entrances.

6 Conclusion

In this paper, we propose an occupancy measurement algorithm based on visual object tracking for entrances of buildings. The algorithm figures out variation of occupancy by comparing the size flags of the same pedestrian in adjacent frames. A CNN based deep learning tracker is designed to match multiple pedestrians in sequential frames. Its parameters are online uptated using the latest sample set after it finishes analyzing each frame. Five modes of occlusion are summarized and their compensation mechanisms are implemented once the occlison is detected. The experimental result on the BEESVD demonstrates that the perforamce of the proposed algorithm exceeds some well-known algorithms, especially in robustness and accuracy. Our algorithm is valuable in measuring building occupancy. It can be easily integrated with the existing video surveillance software.

However, there is some work in the future. Sometimes mirror image of a pedestrian forming on the glass gate or on the smooth floor tile are also detected and matches to the rectangle area of the same pedestrian in the neighboring frames. We plan to add a feature to distingsuish a pedestrian and its mirror image. What’s more, it is necessary to test the proposed algorithm on video datasets involving other visual scenes of entrances and further enhance the robustness.

References

[1] R. Harle and A. Hopper, The potential for location-aware power management, in *Proc. 10th Int. Conf. on Ubiquitous Computing*, 2008: 302-311.

[2] B. Brumitt, B. Meyers, J. Krumm, A. Kern and S. Shafer, Easyliving technologies for intelligent environments, in *Proc. 2nd Int. Symp. on Handheld and Ubiquitous Computing*, 2000: 12-29.

[3] C. Rougier, J. Meunier, A. S. Arnaud, and J. Rousseau, 3D head tracking for fall detection using a single calibrated camera, *Image and Vision Computing*, 31: 246-254, 2013.

[4] D. Liu, X. Guan, Y. Du, and Q Zhao, Measuring indoor occupancy in intelligent buildings using the fusion of vision sensors, *Meas. Sci. Technol.*, 24: 074023 (13pp), 2013.

[5] J. Cao, L. Sun, M. G. Odoom, F. Luan, and X. Song, Counting people by using a single camera without calibration, in *The 28th Chinese Control and Decision Conference (CCDC 2016)*, 2016: 2048-2051.

[6] A. Torralba, R. Fergus, and W. T. Freeman William, 80 Million Tiny Images: A large data set for nonparametric object and scene recognition, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 30(11): 1958 – 1970, 2008.

[7] D. A. Ross, J. Lim, R. S. Lin, and M. H. Yang, Incremental learning for robust visual tracking, *International Journal of Computer Vision*, 77(1-3): 125-141, 2008.

[8] W. Zhong, H. Lu, and M. H. Yang, Robust object tracking via sparse collaborative appearance model, *IEEE Trans. on Image Processing*, 23(5): 2356-2368, 2014.

[9] B. Babenko, M. H. Yang, and S. Belongie, Robust object tracking with online multiple instance learning, *IEEE T. Pattern Anal.*,33(8): 1619-1632, 2011.

[10] http://motchallenge.net