1

Multi-View People Counting System – Pedestrian Representation

Jung-Ming Wang[1], Wan-Ya Liao[2], Sei-Wang Chen[2]
and Chiou-Shann Fuh[1]

[1]*Department of Computer Science and Information Engineering, National Taiwan University, Taiwan, ROC; E-mail: {d97030, fuh}@csie.ntu.edu.tw*
[2]*Department of Computer Science and Information Engineering, National Taiwan Normal University, Taipei, Taiwan, ROC; E-mail schen@csie.ntnu.edu.tw*

Abstract

In this chapter, a multi-view people counting system is presented. This system uses as the input data the video sequences acquired by a camcorder. The camcorder can be mounted anywhere (e.g., below a ceiling, on a side wall, or at a corner) with any viewing direction. In order to manage various appearances of people, a multi-view representation of pedestrian is introduced. This representation is characterized by a unit sphere, called the viewsphere. The viewsphere is composed of a number of nested spherical layers all centered at the core of the viewsphere. Each layer forms a 2D manifold of viewing directions (viewpoints), which are uniformly distributed over the layer. Pedestrian views generated according to the viewpoints of the viewsphere are clustered into so-called aspects. The pedestrian views within an aspect possess the same characteristics of silhouette.

Keywords: multi-view representation of pedestrian, sequential Monte Carlo method, static parameters, dynamic parameters, viewsphere.

P.S.P. Wang (Ed.), Pattern Recognition and Machine Vision – in Honor and Memory of the Late Professor King-Sun Fu, 1–16.
© 2010 *River Publishers. All rights reserved.*

1.1 Introduction

People counting system play an important role in a variety of applications regarding security, management and commerce. Considering a skyscaper, it is important for the security division of the building to know both the total number of people in the building and the number of people on each floor of the building. Such information becomes vital once an emergency, such as fire, explosion, and toxic gas, occurs in the building. Strategies for effectively evacuating and rescuing people from the spot of emergency heavily rely on the information of people count. Likewise, for the places, such as the public areas of transportation stations, stadiums, museums, and malls as well as the restricted areas of government buildingsm military camps, and construction sites, where the control of people count is essential, automatic people counters provide a reliable and persistent tool for governing the number of people in the regions.

People counters have also been considered to count the passengers getting in and out of transit carriages, such as buses and tains. The data provided by the counters can be used to schedule proper times and time intervals of carriage dispatch. Furthermore, the boarding and alighting behaviors at each station can be investigated based on the information of passenger count collected at the station. Accordingly, adequate utilities as well as facilities can be suggested for different stations.

Typically, three stages are involved in a people counter: (a) people detection, (b) people tracking, and (c) counting. The factors that can influence the implementation technique of a people counter include: (a) the scene to be considered, (b) the position and orientation of the camera, and (c) the goal of the application. The difficulties include occlusion, overlapping, and merge-split and shadow effect.

In detecting step, many detection methods, such as model matching [2], temporal differencing [8], and background subtraction [9], have been developed. After obtaining the foreground, we have to separate that if some human figures are connected with each other. Using a contour (ellipse or rectangular) to represent a pedestrian figure is a common idea [12], but it is too rough to detect while they are occluded. Level set [16] and snake model [3] can be applied to model a human's silhouette dynamically [20], but they are too precise to tolerate the imperfect observation. Detecting each part of a human body and combing them according to a predefined architecture can solve the above problems [11, 15], but there are many constrain required

to define the human architecture and the combinations may require many computation time to prove.

In the tracking step, we want to obtain the moving trajectory of each pedestrian. This can be done by matching the pedestrian silhouette between sequential frames [18], or changing the model's state to achieve the current state [20]. Because the detecting step may not be reliable, some prediction or filtering models are applied here to compensate that. Kalman filter [13] is a common filtering while the target has a stable moving. Since a pedestrian often moves at will, this method would have some problem in tracking people. Some updating versions of Kalman filter [4] have been proposed to solve the nonlinear system that could be approximated by a Gaussian distribution. Particle filter [7], unlike Kalman filter, is a non-parametric filtering model without predefining the distribution of the prior and posterior knowledge. Recently, some researches have shown that it is robust to track people [12].

After detecting and tracking, the people number can be obtained by counting the trajectory number. This counting can be applied in two kinds of area. In the first one, the area is closed and we want to count the people number in it. Such area often has an entrance that we can monitor that and it often requires to define a cross line for counting [1]. In an open area, since we cannot define inside or outside part, only the passing number can be counted. The passing number means the number of the people that have been detected [13], which will be the same as the trajectory number in the scene.

1.2 Multi-view Representation of Pedestrian

A multi-view representation of pedestrian is characterized by a unit sphere, which is called the viewsphere. The viewsphere is composed of a number of nested spherical layers all centered at the core of the viewsphere (Figure 1.1). Each layer forms a 2D manifold of viewing directions (viewpoints), which are uniformly distributed over the layer. A viewpoint located at longigudinal degree φ, latitudinal degree θ on layer d is specified by (φ, θ, d).

Let us consider one camera on the layer d. If the width of the observation object is w, we will have an observation angle γ to cover that. Those parameters would satisfy the following equation:

$$w = d\gamma, \quad d = w\gamma.$$

While we move the camera to the other layer d' under the same φ and θ angles, we may see the other object larger and behind the observation object.

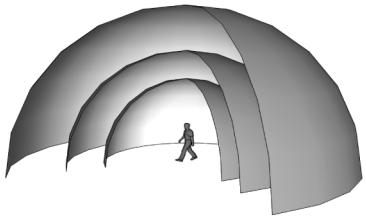


Figure 1.1 Viewsphere is composed of a number of nested spherical layers

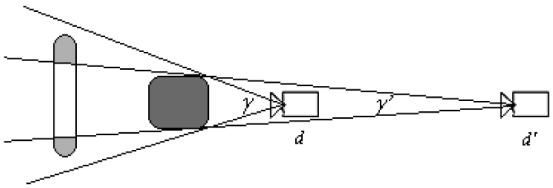


Figure 1.2 Viewing ranges of the cameras with different distances to the monitoring objects

Take for example Figure 1.2, where the background object will be viewed in the camera on the layer d' while the camera on the layer d cannot see that. If we want a significant change in the scene, there will be significant difference, $\Delta\gamma$, between γ and γ'. We have $d'/d = \gamma/(\gamma' - \Delta\gamma)$. Since the observation object is small in the most case, $\Delta\gamma$ would be large comparing with γ in our case. That means that we do not need to design too many layers in our model (there is only one layer in our research to reduce the following processing).

The above parameters can be regarded as the external parameters of the camera. In addition to those parameters, some local parameters, pan, tilt, and rotation angles, at each viewing point should be considered as well. Let we denote these angles as *viewing angle*. These parameters, however, influence the object position and shape distortion in the image plane. Take Figure 1.3, for example, where Image planes I_1 and I_2 have different viewing angles, so that they may have the same object shapes except for the different foreshortening result.

Suppose the reflecting light of the object project to the image plane at the position p_l with the distance l to the image center. We have the viewing angle a defined as

$$\alpha = \tan^{-1} \frac{l}{f},$$

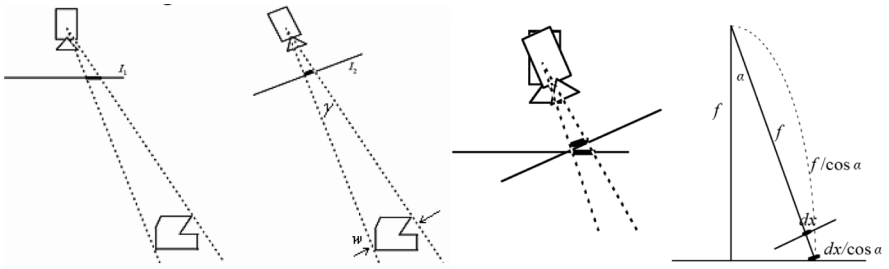


Figure 1.3 Image planes I_1 and I_2 have different viewing angles. (a)–(b) The projecting result on the image plane I_1 and I_2. (c)–(d) The relationship between the projecting results

where f is the focal length of the camera. Let dx be the projecting size on the center of the image plane. The same content projecting on the image position p_l will have size $dx/\cos\alpha$, since the focal length is $f\cos\alpha$. Due to the rotate angle of the plane, the projecting result will be $dx/\cos^2\alpha$.

Besides, model view may have different scale size, σ, and roataion angle, τ, because of image resolution, pedestrian distance to the camera, and camera setting. Finally, we have all parameters, $(\varphi, \theta, d, l, f, \sigma, \tau)$, in our model view. In a static camera, θ, f, and τ are static. Considering the specific position on the image plane, l also is static. If we could obtain the static parameter values before the system operating, we can reduce the searching space to the dynamic parameters, (φ, d, σ).

Our system consists of three components, which are training step, running step, and memory states. Image sequences with unoccluded pedestrians are applied to measure the static parameter values. This process, called as training step, is applied to define the parameter values using global search. After this process, the pedestrian views are clustered into so-called aspects. The pedestrian views within an aspect possess the possible silhouette on a specific image point. In the running step, the parameter space is much smaller than the original design, and we solve the dynamic parameters using sequential Monte Carlo (SMC) method. Finally, memory states are designed here to memory the distribution of the pedestrian state in the secene. They will support the information of the possible state of the pedestrian for detection and tracking in the previous two steps.

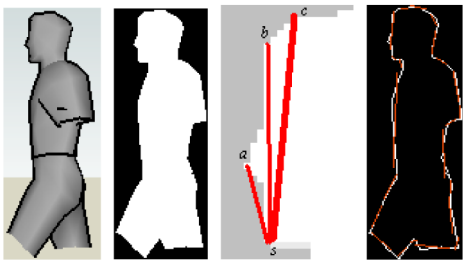


Figure 1.4 (a) Model shapes and (b) its binary image; (c) the process of detecting the line segment and (d) the polygonization result

1.3 Measurement of the Static Parameters

In each viewing direction, we have a model view as shown in the Figure 1.4a. Since a pedestrian shape has different contents while comparing with each other, we use the boundary of the shape to measure their similarity with the model shape. The model shape is extracted from the model view and represented using a binary image as shown in Figure 1.4b. After the foreground shape being extracted, it is compared with all of the model shapes to obtain the values of the parameters.

Comparing each boundary point between the foreground and the model shape can help to measure their matching degree. However, this method is too detail and may cause the comparing result too sensitive. Beside, we may need to compare with various poses of the model, so the search space is really large ($\varphi, \theta, d, l, f, \sigma, \tau$). Most of the camera is perspective camera with little viewing angle α. The distoration $dx/\cos^2\alpha$ would be too small to be ignored, and l and f can be ignored from the parameters.

Even l and f are ignored; such searching space is still intractable. We simplify the shape using a polygonization method. The boundary of the polygonization consists of some line segments which are represented as a graph using scale and rotation invariant feature values. This representation will reduce the searching space to fewer parameters space, (φ, θ, d). Assume the camera have a significant distance to the pedestrian, d can be set as one to reduce the searching space.

Our polygonization method is applied from the boundary point on the top left of a shape. Along the boundary under clockwise, the orientation from the start point to the current boundary point is calculated and recorded. While the difference between the maximum and minimum orientation values

is greater than a threshold, one line segment is assigned from the start point to the current one, and set the current one as the start point of the next line. Figure 1.4c shows the process. In this figure, s means start point, and we will get the orientations along the boundary. When the maximum orientation, a, and the minimum orientation, c, have a significant difference, we connect s to c as one line segment of the polygonization. Figure 1.4d shows the example of the polygonization result of a model shape.

Foreground objects are extracted using the method purposed in [19]. This method has the advantage of the complete shape comparing with the traditional background subtraction method. In this step, non-occluded pedestrians in the monitoring scene are used for training, where the training means that we want to locate the static parameters in this step. After the foreground object being extracted, the above polygonization method is applied and line segments also are defined as mentioned before.

Line segments of a shape are represented as a full connected graph $G = (V, E)$, where V is the set of the nodes showing the line segments, and E is set of the edges showing the relation between line segments. After representing line segments as graph, the matching process between the pedestrian model and foreground figures can be considered as graph matching problem. Both graphs may have different node noumbers, so this matching is an inexact matching. Our solution not only can apply to partial matching but also give the maching degree value.

We represent a graph G as a set of matrices $A_k, k = 1, 2, \ldots, m$, where m is the number of kinds of features. The elements of matrix A_k are the feature values calculated according for the k-th kind of feature for all nodes. For unary features, we construct a diagonal matrix whose element (i, i) contains the feature value of the i-th node. The binary feature values form a symmetric matrix with element (i, j) representing the binary feature between the i-th and j-th nodes.

Line segments for the pedestrian figure can also be represented using matrices A'_k. The correspondence between the feature points in different images can be obtained by solving the following equation to obtain the permutation matrix, P:

$$P = \min_P \left(\sum_{k=1}^{m} \| A_k - P A'_k P^T \| \right),$$

where P can be solved by the method proposed in [20], and $||\cdot||$ means some norm and can be computed by the square root of the sum of square of elements.

The above method, however, can be applied to A_k whose elements have been normalized. Here we modify this method by applying new measurement function to adapt to the variant types of feature values. P is solved using the following two steps: weighted matrix construction and optimal assignment. Each element $W(i, j)$ of the weighted matrix is computed by

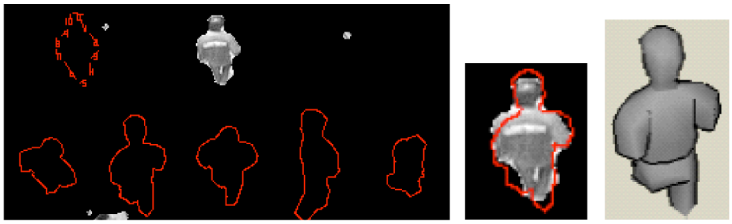
$$W(i,j) = \frac{\sum_{k=1}^{m} \max_s \left[\min_t \left|A_k(i,s) - A'_k(j,t)\right|\right] + \max_s \left[\min_t \left|A_k(s,i) - A'_k(t,j)\right|\right]}{\max[A_k(i,.), A'(j,.), A_k(.,i), A'_k(.,j)]},$$

where $A_k(.)$ and $A'_k(.)$ mean the element in A_k and A'_k respectively. The calculation result means the degree of the correspondence between node i and node j.

The graph with fewer nodes is assigned some null nodes to equal their node numbers. In the matrices, feature values of the null nodes and its corresponding edges are set as nulls. Null values are ignored in constructing the weighted matrix. After constructing the weighted matrix, we assign the optimal value for each element of P. Some Hopfield model can be applied here, for example, the Hopfield memory in the neural network. In this application, we use Hungarian algorithm [10] because of its polynomial processing time.

Binary feature values are assigned in calculating for size and rotation invariant. In this research, we use four feature values: difference of the orientations, ratio of the segment lengths, relative distance between line segments, and the angle from the center of the shape to the centers of the line segments. After this computation, each node will match one of the nodes in the other graph. The redundant nodes will match a null node or one redundant node of the other graph.

Matching degree is than computed according to the minimized term, $\sum_{k=1}^{m} \|A_k - P A'_k P^T\|$ in defining P. This term combines several value types so that cannot show the real degree value. In our application, one of the feature values, diference of the orientations, is assigned to compute. The model views with higher degree values are extracted as the comparing results (Figure 1.5). After obtaining the candidate model views, we compare their boundary points with the foreground object under various scale sizes and



(a) (b) (c)

Figure 1.5 The bottom row of (a) are the candidates of the model view and (b) is the best comparing result; (c) is the corresponding model view

rotation angles:

$$p^e(.) = \frac{1}{k}\sum_{i=1}^{k} g_i \times G\big(D(v_i, C(v_i))\big), \tag{1.1}$$

where $\{v_1, v_2, \ldots, v_k\}$ are the boundary points of the model, g_i is the corresponding weight, $G(.)$ is a Gaussian function, $D(.)$ is the distance two points, and $C(.)$ is the closet boundary point of the foreground object on the normal vector of v_i. The model view with the greatest $p^e(.)$ value (Figure 1.5b) is the most likely model at this image position. In most case, the shape of the human head is more reliable than the body, so we give the boundary points of the model head greater g_i values. This will help to locate the human shape more precisely (Figure 1.5c).

After training some pedestrian shape on this monitoring range, the distribution of the parameter values, $(\varphi, \theta, s, \tau)$, can be constructed. Among them, (θ, τ) are static parameters and their values are defined using the expected value. In addition, their distribution will be updated in running step while the human states have been detected. The values of (φ, s) will be computed in running step, because they are dynamic parameters.

1.4 Sequential Monte Carlo Method

In our application, we want to know the human behavior (hidden states) according to the monitoring image sequence (observations). Since the actual human behavior can not be known, its distribution of the state given the passing observations, $z_{0:t} = (z_0, \ldots, z_t)$, is defined as $p(s_t \mid z_{0:t})$, where s_t is the human behavior shooting at time t. In this system, each image frame is computed to the distribution of the human behavior $p(s_t \mid z_{0:t})$, and the

behavior s_t with a highest probability is determined as the pedestrian detecting result. The number of the behaviors along time, $s_{0:t} = (s_0, \ldots, s_t)$, is regarded as the detection result of one pedestrian.

Using the Bayesian rule, this prior probability can be updated to

$$p(s_t \mid z_{0:t}) = \frac{p(z_t \mid z_{0:t-1} s_t) p(s_t \mid z_{0:t-1})}{p(z_t \mid z_{0:t-1})}, \tag{1.2}$$

where $p(z_t \mid z_{0:t-1})$ is the predictive distribution of z_t given the past observation $z_{0:t-1}$, and it is a normalizing term in most case. Assume that $p(z_t \mid z_{0:t-1}, s_t)$ depends only on s_t through a predefined measurement model $p(z_t \mid s_t)$. Equation (1.2) can be rewritten as

$$p(s_t \mid z_{0:t}) = \alpha p(z_t \mid s_t) p(s_t \mid z_{0:t-1}), \tag{1.3}$$

where α is a constant.

Now, suppose s_t is Markovian, then its evolution can be described through a transition model, $p(s_t|s_{t-1})$. Based on this model, $p(s_t|z_{0:t-1})$ can be calculated using the Chapman–Kolmogorov equation:

$$p(s_t \mid z_{0:t-1}) = \int p(s_t \mid s_{t-1}) p(s_{t-1} \mid z_{0:t-1}) ds_{t-1}. \tag{1.4}$$

Equations (1.3) and (1.4) show that we can obtain $p(s_t \mid z_{0:t})$ recursively if we have the following requirements: the measurement model $p(z_t \mid s_t)$, the transition model $p(s_t \mid s_{t-1})$ and the initial distribution $p(s_0 \mid z_0)$.

The main problem is how to represent the distribution of the transition model $p(s_t \mid s_{t-1})$ and the unknown state given the observation $p(s_0 \mid z_0)$. This problem also induces the problem of the calculation of Equation (1.3). Particle filter is a Monte Carlo method that uses m particles, $s^{(i)}, i = 1 \ldots m$, and their corresponding weights, $w^{(i)}$, to simulate the distribution $p(s_t \mid z_{0:t})$. This simulation also can be applied to Equation (1.4):

$$p(s_t \mid z_{0:t-1}) = \sum_{i=1}^{m} p(s_t \mid s_{t-1}^{(i)}) p(s_{t-1}^{(i)} \mid z_{0:t-1}). \tag{1.5}$$

The distribution $p(s_t \mid z_{0:t-1})$ also can be simulated using the these particles if we have a new transition equation $s_t^{(i)} = f(s_{t-1}^{(i)}, u_{t-1})$ matching the transition model $p(s_t \mid s_{t-1})$, where u is a noise sequence with zero mean. The distribution of the propagating result

$$f(s_{t-1}^{(i)}, u_{t-1}) p(s_{t-1}^{(i)} \mid z_{0:t-1}), \quad i = 1, \ldots, m, \tag{1.6}$$

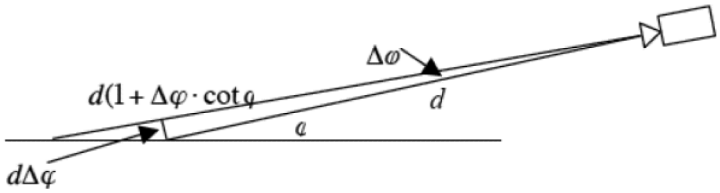


Figure 1.6 The relationship between the scale and the object distance

will be the same as $p(s_t \mid z_{0:t-1})$ if the particle number is infinite. For more details on the particle filter, the reader is referred to [7].

For each pedestrian moving in the monitoring range, we detect his behavior and denote the detecting result as $s_{0:t}$. State s_t is calculated to the MAP of $p(s_t \mid z_{0:t})$ using the mean shift method [6]. The distribution of $p(s_t \mid z_{0:t})$ can be obtained if we have the following requirements. The first requirement, initial state s_0, is defined according to the foreground object extracted using the method proposed in [19]. The second requirement, measurement model, is computed by comparing the current image with our predefined human model. The final requirement, transition model, would be defined by some prior knowledge, and updated according to the detecting result. People number is then calculated to the number of the moving trajectory $s_{0:t}$).

1.5 Measurement of the Dynamic Parameters

At the beginning, a model view requires parameters (φ, θ, d, l, f, σ, τ) to locate the foreground pedestrians. After the training step, they are reduced to (φ, σ) in running step. Scale value, σ, has different range according to φ value. Take Figure 1.6 for example. Let d be the distance from the camera to the pedestrian and φ is the tilt angle. If we limit the viewing direction of the model view in $\Delta\varphi$ angle, the distance from the pedestrian to the camera will be value from $d(1 - \Delta\varphi \cdot \cot\varphi)$ to $d(1 + \Delta\varphi \cdot \cot\varphi)$. The range of the scale value would be between $\sigma(1 - \Delta\varphi \cdot \cot\varphi)$ to $\sigma(1 + \Delta\varphi \cdot \cot\varphi)$, where σ is the scale value trained in the training step. In most case, $\Delta\varphi \cdot \cot\varphi$ is so small that we can ignore that, except that φ is very close to zero. Under the previous assumptions and training, we just need to define φ and σ (if φ is close to zero) values in system running.

Initial distribution $p(s_0|z_0)$ is the base of the sequential Bayesian estimation. Here we give particle set S_t using Markov Chain Monte Carlo (MCMC) method [22]. MCMC method has been proved [21] that it is more efficient

than the particle filter-based method when the dimensionality of the search space is high [17]. The state of a particle is set as the following features: shape, intensity histogram, and moving velocity. Among these features, the shape should be designed to match a pedestrian's silhouette. Here we use the multi-view representation to model this shape. The second feature, color histogram is constructed using the values of the pixels within the model view. The velocity includes the moving direction and moving distance that can be computed while we have two successive states.

Since people would be occlued with each other, their shape cannot be complete most of the times. Assume that their head and shoulder always be shown in the monitoring range. We use the models with head and shoulder parts to measure the $p(z_t|s_t)$ value, which will be given in a later section. In an image, the region not covered by any state is denoted as R, and the feature values of the first particle, $s^{(1)}$, is set on the boundary of R. The static features of the model view are extracted at the corresponding position. Scale σ and viewing angel φ are given as the expected value learned in the training step.

We sample a candidate state s' according to $s^{(i-1)}$ from the proposal distribution $q(s^{(i)}|s^{(i-1)})$, and then set $s_{(i-1)}$ as s' with the probability

$$p = \min\left\{1, \frac{p(s'|z_t)q(s^{(i-1)}|s')}{p(s^{(i-1)}|z_t)q(s'|s^{(i-1)})}\right\};$$

otherwise, set $s^{(i-1)}$ as $s^{(i)}$. Our proposal distribution is combined with random and data-driven proposal probability. Some feature values are proposed randomly, such as longigudinal degree and scale value, while others are proposed in data-driven, such as position value. In our application, $q(s'|s)$ is computed by $q(s'|s) = Ke^{-(x'-x)/2\sigma^2}$, where $\mathbf{x}'$ and $\mathbf{x}$ are the position of s' and s respectively, and K is a normalized term. To let the sampling more efficient, the positions of the samples are limited on the boundary of the foreground figures extracted by the method proposed in [19].

While applying the Bayesian rule to $p(s \mid z_t)$, we have $p(s \mid z_t) \propto p(z_t \mid s)p(s)$. Assume $p(s)$ is constant, we can computer $p(z_t \mid s)$ as $p(s \mid z_t)$. That also is the measurement model in sequential Monte Carlo model discussed in previous section. Our measurement model consists of two measurements, one is edge likelihood p^e and the other is color likelihood p^c. That is

$$p(z_t \mid s) = p^e(z_t \mid s) \times p^c(z_t \mid s),$$

where p^e is defined in Equation (1.1), and now we define p^c as below.

Human's face and hair are considered as our color information. At first, we detect the skin color [17] and hair color (black) to separate the input image into two binary images, I^s and I^h. The p^c is defined as

$$p^c(z_t \mid s) = p^c(I_t^s \mid s) \times p^c(I_t^h \mid s),$$

where $p^c(I \mid s)$ is defined according to the pixel number. As the s is given, we count the number of skin pixel falling on the face of the model, and denote this number as N_d^s. The number of non-skin pixel is denoted as N_f^s, and the number of hair pixel falling on the face of the model is denoted as N_n^s. Similarly, we calculate N_d^h, N_f^h, and N_n^h for the hair part of the model. Then we can define

$$p^c(I_t^{\{s,h\}} \mid s) = \frac{N_d^{\{s,h\}}}{(N_f^{\{s,h\}} + N_n^{\{s,h\}})}. \tag{1.7}$$

Unfortunately, Equation (1.7) will let the smaller model have larger measurement value, so we must refer to the pedestrian size in the scene. According to the position $\mathbf{x}$ and the scale σ values in the state s, we define a search region to calculate the skin area N_r^s and and hair N_r^h. Finally, the $p^c(I \mid s)$ value is defined as

$$p^c(I_t^{\{s,h\}} \mid s) = \frac{N_d^{\{s,h\}}}{(N_f^{\{s,h\}} + N_n^{\{s,h\}})} \times \min\left(\frac{N_s^{\{s,h\}}}{N_r^{\{s,h\}}}, \frac{N_r^{\{s,h\}}}{N_s^{\{s,h\}}}\right).$$

After sampling, the distribution of $\{s^{(i)}, i = 1 \ldots N\}$ can show the initial distribution $p(s_0 \mid z_0)$ (Figure 1.7a). However, there will be more than one peak in this initial distribution if there are more than one pedestrian in the foregound region. Mean shift with flat kernel [6] is applied here to locate the peak values. The radius of the kernel is designed as the half of the model length. After locating one cluster, we remove those samples closing to the center for the cluster, and do the mean shift algorithm again to locate other clusters.

For each cluster, we can compute an expected state, $\bar{s}$, and its corresponding measurement $p(z \mid \bar{s})$. A cluster with high $p(z \mid \bar{s})$ value is regarded as a human head, or we will stop to locate the other cluster. Figure 1.7c shows the detecting result. After we detect the human head, each human head is given a set of particles according to the distribution of the sample states. This set of particles model the distribution of $p(s_0 \mid z_0)$.

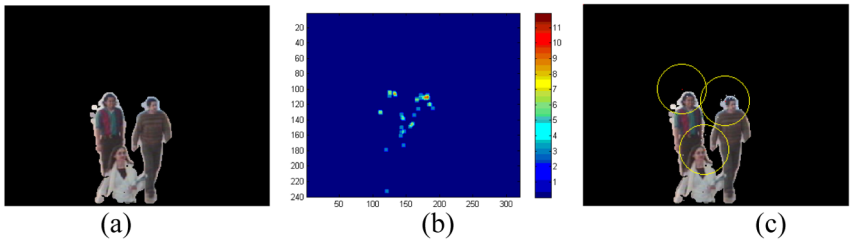


Figure 1.7 (a) The original foreground image, (b) the distribution constructed by MCMC method, (c) using mean shift to locate the pedestrians

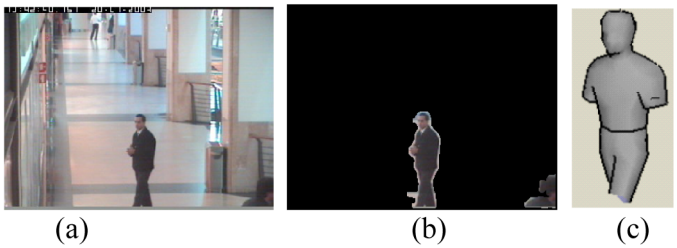


Figure 1.8 (a) The foreground object, (b) the corresponding model view

1.6 Experiments

We test our algorithm on the CAVIAR data set [5]. Each image is reduced to the size 320×240. Because foreground object detection and model comparing are time consumed, the processing to compute the static parameter values is 3.5 minutes on an Intel Core 2 Duo 2 GHz machine. Fortunately, it needs to compute only once after setting the monitoring system, and we can do it off-line. Figure 1.8 shows model view corresponding to the detection result.

The dynamic parameter values is computed on an Intel Core 2 Quad CPU 2.66 GHz machine. The size of the image is 320×240. The processing time is 4 seconds. Figure 1.9 shows some initialization results. Our algorithm can robust to many situations: (a) independent pedestrian, (b) connected pedestrians, and (c) occluded pedestrians.

1.7 Conclusion

Counting people cannot avoid detecting and tracking people. In this paper, we propose a people counting system based on the particle filter applied to track people. When the particle filter is applied, the initial state, measure-

(a) (b) (c)

Figure 1.9 The test result of (a) independent pedestrian, (b) connected pedestrians, and (c) occlued pedestrians

ment model and transition model need to be defined before hand. We give a processing method to define the initial state automatically and to construct the prior knowledge for measurement and transition model. Among those requirements, we define the initial state using our multi-view representation of the pedestrian with MCMC algorithm. The final calculation result is the probability of the human state; mean shift is applied here to locate the peak value instead of the expectation value.

In the future works, particle filter will be applied to track the pedestrians along time. The tracking results will show the human's trajectories, and we can obtain the people number by counting the trajectory number.

References

[1] A. Albiol, I. Mora, and V. Naranjo. Real-time high density people counter using morphological tools. *IEEE Trans. on Intelligent Transportation Systems*, 2(4):204–218, 2001.

[2] A. Broggi, M. Bertozzi, A. Fascioli, and M. Sechi. Shape-based pedestrian detection. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, Dearborn, pages 215–220, 2000.

[3] F. Buccolieri, C. Distante, and A. Leone. Human posture recognition using active contours and radial basis function neural network. In *Proceedings IEEE Conference on Advanced Vide and Signal Based Surveillance*, pages 213–218, 2005.

[4] O. Cappe, S.J. Godsill, and E. Moulines. An overview of existing methods and recent advances in sequential Monte Carlo. *Proceedings of the IEEE*, 95(5):899–924, 2007

[5] The CAVIAR Data Set, http://homepages.inf.ed.ac.uk/rbf/CAVIAR/, 2008.

[6] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.

[7] N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proc.-F Radar and Signal Processing*, 140(2):107–113, 1993.

[8] S. Jehan-Besson, M. Barlaud, and G. Aubert. A 3-step algorithm using region-based active contours for video objects detection. *Journal on Applied Signal Processing*, 2002(1):572–581, 2002.

[9] J.-W. Kim, K.-S. Choi, B.-D. Choi, and S.-J. Ko. Real-time vision-based people counting system for the security door. In *Proceedings International Technical Conference on Circuits/Systems Computers and Communications*, pages 1416–1419, 2002.

[10] H.W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[11] M.-W. Lee and I. Cohen. A model-based approach for estimating human 3D poses in static images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(6):905–916, 2006

[12] E. Maggio, F. Smerladi, and A. Cavallaro. Adaptive multifeature tracking in a particle filtering framework. *IEEE Trans. on Circuits and Systems for Video Technology*, 17(10):1348–1359, 2007

[13] O. Masoud and N.P. Papanikolopoulos. A novel method for tracking and counting pedestrians in real-time using a single camera. *IEEE Trans. on Vehicular Technology*, 50(5):1267–1278, 2001.

[14] E. Poon and D.J. Fleet. Hybrid Monte Carlo filtering: Edge-based people tracking. In *Proceedings Workshop on Motion and Video Computing*, Orlando, pages 151–158, 2002

[15] D. Ramanan, D.A. Forsyth, and A. Zisserman. Tracking people by learning their appearance. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(1):65–81, 2007

[16] J.A. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, 2nd edn. Cambridge University Press, 1999.

[17] J.-M. Wang, H.-W. Lin, C.-Y. Fang, and S.-W. Chen. Detecting driver's eyes during driving. In *Proceedings of the 18th IPPR Conference on CVGIP*, Taipei, Taiwan, 2005.

[18] J.-M. Wang, S.-W. Chen, S. Cherng, and C.-S. Fuh. People counting using fisheye camera. In *Proceedings of the IPPR Conference on CVGIP*, Mauli, Taiwan, 2007.

[19] J.-M. Wang, S. Cherng, C.-S. Fuh, and S.-W. Chen. Foreground object detection using two successive images. In *Proceedings of IEEE International Conference on Advanced Video and Signal-based Surveillance*, Santa Fe, pages 301–306, 2008.

[20] A. Yilmaz, X. Li, and M. Shah. Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(11):1521–1536, 2004.

[21] T. Zhao and R. Nevatia. Tracking multiple humans in crowded environment. In *Proceedings Computer Vision and Pattern Recognition*, Washington, Vol. 2, pages 406–413, 2004.

[22] T. Zhao, R. Nevatia, and B. Wu. Segmentation and tracking of multiple humans in crowded environments. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 30(7):1198–1211, 2008.