A Monte Carlo Based Framework for Multi-Target Detection and Tracking Over Multi-Camera Surveillance System

Ching-Chun Huang, Sheng-Jyh Wang

A Monte Carlo Based Framework for Multi-Target Detection and Tracking Over Multi–Camera Surveillance System

Ching–Chun Huang and Sheng–Jyh Wang

Department of Electronics Engineering, National Chiao Tung University, Taiwan.

Abstract. In the paper, we proposed a system for automatic detection and tracking of multiple targets in a multi–camera surveillance zone. In each camera view of this system, we only need a simple object detection algorithm, such as background subtraction. The detection results from multiple cameras are fused into a posterior distribution, named TDP, based on the Bayesian rule. This TDP distribution indicates the likelihood of having some moving elements on the ground plane. To properly handle the tracking of multiple moving targets over time, a sample–based framework, which combines Markov Chain Monte Carlo (MCMC), Sequential Monte Carlo (SMC), and Mean–Shift clustering, is proposed. The MCMC is used to handle the occurrence of new targets. The SMC is used to track existing targets over time. The Mean-Shift clustering is adopted to automatically identify new comers. With the Monte Carlo based framework, the detection and tracking of multiple targets can be achieved in a unified and seamless manner. The detection and tracking accuracy is evaluated by both synthesized videos and real videos. The experimental results show that the proposed system can successfully track a varying number of people accurately.

1 Introduction

With the increasing demands for security and safety, multi–camera video surveillance systems are gaining popularity. To assist users in efficiently summarizing information from multiple cameras, we design a multi–camera system that can fuse information from a set of calibrated cameras to perform detection and tracking of multiple targets. To build such a system, we focus on three major issues: (1) this system shall provide reliable data fusion from multiple cameras; (2) this system can simultaneously detect and track multiple moving targets; and (3) the computational complexity of this system shall be affordable.

So far, a few multi–camera surveillance systems have already been proposed for multi–target tracking. These systems can be roughly classified into two categories. In the first category, moving objects are tracked in each 2–D camera view; object correspondences are built among cameras; and finally 2–D detection and tracking results are fused together to support surveillance over the 3–D space. For

Supported by National Science Council with No. NSC95-2221-E-009-106-MY2.

example, Utsumi [1] proposed the adoption of intersection points, which are the intersections of the 3–D lines emitted from the 2–D tracking results of different camera views. In this approach, authors use a mixture of Gaussian distributions to describe the possible positions of moving objects in the 3–D space. By projecting these 3–D Gaussian distributions onto individual 2–D image plane, the object correspondence among camera images can be derived in a probabilistic manner. In [2] and [3], Mittal and Davis establish the object correspondence by matching segmented regions along the epipolar lines in pairs of camera views. The corresponding mid–points are then projected onto the 3–D space to yield a 3–D probability distribution map for the description of people's positions. In [4], the authors propose a principal axis–based correspondence among multiple cameras for data fusion. This principal axis–based method may offer robust results and can allow a certain level of defects in the motion detection and segmentation processes of each camera view. Moreover, the commonly required camera calibration step is not a necessity in their system.

In the second category, researchers attempt to simplify 2–D analyses by directly tracking moving targets in the fused 3–D space. For example, Fleuret et al. [5][6] adopt a simple blob detector in 2–D analysis and introduce a Bayesian generative model to fuse data from multiple views. In their system, a discrete occupancy map is defined to describe whether an individual target is standing at a specific ground location in the 3–D space. The most likely trajectory of each individual over the 3–D ground plane is then traced via a Viterbi algorithm. Since the 3–D ground plane is divided into a lattice of discrete locations, a huge number of discrete locations will be required if we ask for a finer spatial resolution or a broader surveillance scope.

In this paper, we proposed a new multi–camera surveillance system, which also performs detection and tracking in the fused 3–D space. Similar to Fleuret's approach, we only require a simple object detection algorithm, such as background subtraction, for 2–D analysis. However, instead of defining the occupancy map over a lattice of discrete ground locations, we formulate a posterior distribution, named target detection probability (TDP), to indicate the likelihood of having a moving target at a continuous–valued ground location. To systematically manage the whole system in a unified manner, the multi–target detection and tracking problems are treated as a probabilistic sample management problem, which includes the generation of samples, the prediction and updating of samples, the labeling of sample IDs, and the elimination of samples. Here, we propose a Monte Carlo–based framework to combine together Markov Chain Monte Carlo (MCMC), Mean–Shift clustering, and Sequential Monte Carlo (SMC). The MCMC sampler is used to draw samples from the TDP distribution; the Mean–shift clustering is used to group iso–samples for ID labeling; and the SMC is adopted to iteratively update and predict the positions, weights, and IDs of samples. Under the proposed scheme, the probabilistic distribution of the moving targets' 3–D locations can be robustly tracked over time. Moreover, any newcomer can be automatically detected, tagged, and tracked until it eventually moves out of the scene.

2 Multiple–Camera Data Fusion

2.1 2–D Analysis on Single Camera

In the proposed system, the main goal is to integrate the detection results from a set of 2–D camera views to offer a global 3–D view. With the integration of 2–D information, the requirement of accurate 2–D analyses can be somewhat alleviated. In our system, we use static cameras and the geometric relationships among these cameras are well calibrated beforehand. For each camera, we build its reference background based on the Gaussian mixture model. The foreground image is calculated by computing the frame difference between the current image and the reference background in a pixel–wise manner. That is, we only apply a simple background subtraction method for 2–D detection. In Figure 1(a) and (b), we show a camera view and its binary foreground image.

2.2 Target Detection Probability (TDP) Distribution

In the proposed system, the integration of 2–D information is accomplished by maintaining a Target Detection Probability distribution over time. This TDP distribution expresses the likelihood of having a moving target at a ground location given a set of foreground images from multiple cameras. In our approach, we formulate the TDP as a posterior distribution and use the Bayes rule to estimate its distribution, which is expressed as

$$p(X|F_1, ..., F_N) \propto p(X)p(F_1, ..., F_N|X). \tag{1}$$

In (1), X represents a location (x_1, x_2) on the ground plane. N is the number of static cameras in the multi–camera system. F_i denotes the foreground image acquired from the i–th camera view. Assume (m, n) denote the coordinates of a pixel on the foreground image, then

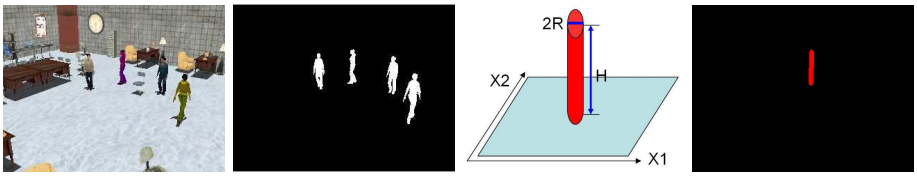
$$F_i(m, n) = \begin{cases} 1 & \text{if (m, n) } \in \text{foreground regions} \\ 0 & \text{if (m, n) } \notin \text{foreground regions} \end{cases} \tag{2}$$

Moreover, given the location X on the ground plane, we assume the foreground images from different camera views are conditionally independent of each other. That is, we assume (1) can also be written as

$$p(X)p(F_1, ..., F_N|X) = p(X) \prod_{i=1}^{N} p(F_i|X). \tag{3}$$

On the other hand, to simplify the formulation, we approximate a moving target at the ground position X by a cylinder, with height H and radius R, as shown in Figure 1(c). Based on the pre–calibrated projection matrix of the i–th camera, we project the cylinder onto the i–th camera view to get the projected image M_i, as shown in Figure 1(d). Mathematically, we have

$$M_i(m, n) = \begin{cases} 1 & \text{if (m, n) } \in \text{projected regions} \\ 0 & \text{if (m, n) } \notin \text{projected regions} \end{cases} \tag{4}$$



(a) (b) (c) (d)

Fig. 1. (a) A camera view. (b) Its binary foreground image F. (c) A cylinder on the ground plane, with height H and radius R. (d)The cylinder's projection image M.

The overlapped region of M_i and F_i with perspective normalization offers a reasonable estimate about $p(F_i|X)$. That is, if the projected region M_i is greatly overlapped with the detection result F_i, it is very likely that there is a moving target standing on the ground location X. Hence, we define

$$p(F_i|X) \equiv \frac{e^{\Omega_i}}{1+e^{\Omega_i}}, \tag{5}$$

$$where \quad \Omega_i \equiv \Big(\int\int F_i(m,n)M_i(m,n)dmdn\Big) \Big/ \Big(\int\int M_i(m,n)dmdn\Big). \tag{6}$$

Moreover, $p(X)$ in (1) indicates the prior belief of finding a moving person at X. In our system, two different prior probabilities, $p_1(X)$ and $p_2(X)$, are considered. The first choice, $p_1(X)$, assigns an equal probability to each ground location inside the surveillance zone. That is,

$$\mathrm{p_1(X)} = \begin{cases} 1/\mathrm{W} & \text{if X} \in \text{surveillance zone} \\ 0 & \text{if X} \notin \text{surveillance zone} \end{cases} \tag{7}$$

where W is the total area of the ground plane inside the surveillance zone. This surveillance zone is decided by the overlapped fields of view of all cameras. The second choice of prior function, $p_2(X)$, is defined as $p(X^t|F^{t-1})$, where $F^{t-1} = \{F_1^{t-1}, F_2^{t-1}, ..., F_N^{t-1}\}$. This prior function is the estimated probability of X at the current time based on the belief propagated from the previous observations. Essentially, this prior function considers the property of temporal continuity and can be formulated via the Chapman–Kolmogorov equation.

$$p_2(X) \equiv p(X^t|F^{t-1}) = \int p(X^t|X^{t-1})p(X^{t-1}|F^{t-1})dX^{t-1} \tag{8}$$

Note that in (8), $P(X^t|X^{t-1})$ is a motion prediction model and $P(X^{t-1}|F^{t-1})$ is the TDP distribution at time $t-1$.

Both $p_1(X)$ and $p_2(X)$ have their pros and cons. If $p_1(X)$ is used, each ground location is equally probable and (1) is mainly dominated by the likelihood function $p(F_1, \ldots, F_N|X)$. With this prior probability, any newly appearing target can be easily detected, while the crucial temporal information between successive frames is not properly included. In contrast, $p_2(X)$ focuses mainly on the temporal correlation, but may cause poor performance in detecting new comers.

To compromise between these two choices, we propose a hybrid prior probability and define the TDP distribution to be

$$\begin{aligned} TDP &\equiv (p_1(X) + p_2(X))p(F_1, .., F_N|X) \equiv G_1(X) + G_2(X) \\ \text{where } & G_1(X) \equiv p_1(X)p(F_1, ..., F_N|X); G_2(X) \equiv p_2(X)p(F_1, ..., F_N|X) \end{aligned} \tag{9}$$

Typically, the TDP distribution is composed of several clusters, with each cluster indicating a moving target on the group plane. Hence, the detection of multiple moving targets can be treated as a clustering problem over the TDP distribution, while the tracking of these targets can be thought as the temporal association of clusters at different time instants. In Figure 2(a), we show an example of TDP distributions, which are fused from the detection results of four cameras. Later in Section 3, we'll explain how to apply the TDP distribution to the detection and tracking of multiple targets.

Occasionally, some fake clusters may occur in the TDP distribution. This happens when the projection of a cylinder at an incorrect location accidentally matches the foreground masks on the camera views. An example is illustrated in Figure 2(b), where the TDP distribution includes not only the four real targets but also two extra fake clusters. Fortunately, these fake clusters can be effectively discriminated from true clusters by checking their temporal properties. In general, a fake target either has some temporally unstable characteristics or has a short life time. These two properties enable us to get rid of them.

3 Multi–Target Detection And Tracking

As mentioned in Section 1, the detection and tracking of multiple targets can be treated as a sample management process. Figure 3 illustrates the block diagrams of the proposed sample management process. In this process, the sample management process includes four major modules: sample generation, sample labeling, identification of new targets, and target updating. The details of each module will be explained as follows.

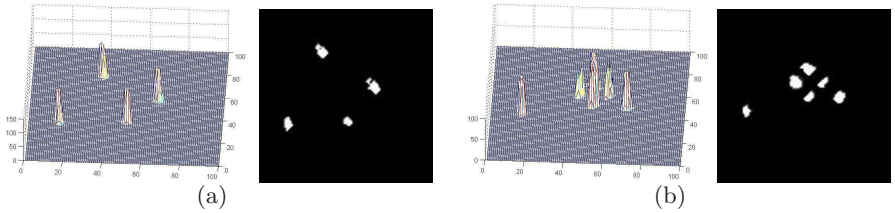


Fig. 2. Two examples of TDPs and their brid's eye views. (a) The TDP of four moving targets. (b) The TDP of four moving targets with the presence of two fake clusters.

3.1 Sample Generation

Since two different types of prior probabilities are considered, the generation of samples is implemented via two mechanisms. The sample generation of $G_1(X)$ is implemented by the mixture Markov Chain Monte Carlo (MCMC) method, while the sample generation of $G_2(X)$ is implemented by the Sequential Monte Carlo (SMC) method. The details of the samplers will be explained as follows.

Mixture–MCMC sampler MCMC is a popular technique for the sampling of a probabilistic model. In our system, we use MCMC to generate samples from $G_1(X)$, which summarizes the 2–D detection results of multiple camera views. Since $G_1(X)$ usually contains narrow peaks, the commonly used Metropolis sampler is not an appropriate choice. Instead, a mixture hybrid kernel, which includes two Metropolis samplers with different proposal distributions [7], is adopted. In our system, we choose two different proposal distributions, $N(\mu,\ \sigma_1^2)$ and $N(\mu,\ \sigma_2^2)$. The larger standard deviation, σ_1, is chosen to be large enough so that the sampler can explore between peaks; while the smaller standard deviation, σ_2, is chosen to be small enough to discover fine details. The mixture weights are denoted as V_1 and $(1 - V_1)$, respectively. To draw K fair samples from $G_1(X)$, the process is detailed in Algorithm 1 and Algorithm 2. In addition, the ID of each sample is labeled as "undecided" in this step.

Algorithm 1: Mixture–MCMC sampler (MMS)
Assume the distribution is $G_1(.)$, and the mixture weights are V_1 and $(1 - V_1)$.

1. Randomly select the first sample $X^{(0)}$
2. For $i = 0$ to K
   Generate a random value U from a uniform distribution over $(0, 1)$
   - If $U < V_1$
     Get a sample $X^{(i+1)}$ by the Metropolis sampler based on $G_1(.), N(X^{(i)},\ \sigma_1^2)$, and $X^{(i)}$. That is, $X^{(i+1)} = MS(G_1(.), N(X^{(i)},\ \sigma_1^2), X^{(i)})$
   - else
     Get a sample $X^{(i+1)}$ by the Metropolis sampler based on $G_1(.), N(X^{(i)},\ \sigma_2^2)$, and $X^{(i)}$. That is, $X^{(i+1)} = MS(G_1(.), N(X^{(i)},\ \sigma_2^2), X^{(i)})$
3. Discard the first d samples.

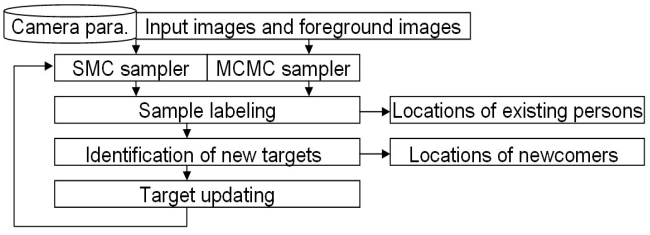


Fig. 3. System flow of the proposed algorithm.

Algorithm 2: Metropolis sampler (MS)
$MS(G(.), f(.), X)$
- Randomly generate a candidate sample X^* based on the probability density function $f(.)$
- Randomly generate a sample W from the uniform distribution over $[0, 1]$
- If $W < \min\{1, \frac{G(X*)}{G(X)}\}$, output $= X^*$; otherwise, output $= X$.

SMC sampler Sequential MC is a technique for representing posterior distribution by a set of samples with different weights [7]. To generate samples, this method usually chooses a proposal distribution which is relative to the previous posterior distribution. This mechanism makes SMC suitable for representing sequential distributions through recursive prediction and updating over time. A practical algorithm called sequential importance sampling (SIS), which is an extension from important sampling (IS) algorithm [7], is adopted in our system for implementing the SMC sampler.

Assume we have already obtained a set of S samples $\{X^{(i),t-1}\}_{i=0\sim S-1}$ based on the TDP, $p(X^{t-1}|F^{t-1})$, at the previous moment. These samples have equal weights $\{q^{(i)}\}_{i=0\sim S-1}$, while the distribution of the samples basically follows $p(X^{t-1}|F^{t-1})$. For each sample $X^{(i),t-1}$, we assume its motion model follows the uniform distribution. That is, we assume

$$p(X^{(i),t}|X^{(i),t-1}) = \begin{cases} \frac{1}{\pi R^2} & \text{if } \left|X^{(i),t} - X^{(i),t-1}\right| < R \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where R is a pre–defined radius. By randomly moving each sample based on this motion model, we obtain a new set of samples $\{X^{(i),t}\}_{i=0\sim S-1}$, which basically follow the temporal prior distribution $p_2(X) \equiv p(X^t|F^{t-1})$, as expressed in (8). Note that the new samples still have equal weights $\{q^{(i)}\}_{i=0\sim S-1}$ at this stage.

Then we apply a so–called "importance sampling" process over the new set of samples. The details of importance sampling can be found in [7]. In this process, for a sample $X^{(i),t}$, we adjust its weight to

$$w^{(i)} = q^{(i)} \cdot p(F_1, .., F_N|X^{(i),t}) \tag{11}$$

Since $q^{(i)}$'s are equal–weighted, $w^{(i)}$'s are proportional to the likelihood function $p(F_1, \ldots, F_N|X^{(i),t})$. Hence, after random movement and importance sampling, we have obtained a new set of unequal–weight samples $\{X^{(i),t}\}_{i=0\sim S-1}$ to carry the information about the $G_2(X)$ distribution. The $p(F_1, \ldots, F_N|X^t)$ part of $G_2(X)$ is represented by sample weights, while the $p(X)$ part of $G_2(X)$ is represented by the sample distribution.

Finally, to avoid the degeneracy problem, we further use the re–sampling process to convert these unequal–weight samples into a new set of equal–weight samples. Samples with larger weights are converted to more equal–weight samples, while samples with smaller weights are converted to fewer equal–weight samples. After the re–sampling process, the sample weights become constant again and the sample distribution carries the whole information about $G_2(X)$. More details of the re–sampling process can be found in [7].

On the other hand, these samples generated by the mixture–MCMC sampler are also assigned constant weights. By properly mixing the samples generated by the mixture–MCMC sampler with the samples generated by the SMC sampler, we form a set of equal–weight samples to carry the information about the TDP distribution $p(X^t|F_1, \ldots, F_N)$. The aforementioned procedure can thus be repeated again to estimate the TDP distribution at the next moment, based on these equal–weight samples.

3.2 Sample Labeling

In our system, we also assign an ID for each sample X. If at time t, the ID of the sample X is assigned to H_k, it means the target H_k may have an opportunity to appear at the location X at that moment. In sample generation, samples generated by the mixture MCMC method are marked as "undecided" and their ID's are to be labeled. On the other hand, except a few samples with small color weights, most samples generated by the SMC method will be marked as "decided" and their ID's inherit from their parent samples. In this sample labeling module, the major functionality is to assign a suitable ID for each of these "undecided" samples. In our system, a group of samples with the same ID represent a probabilistic distribution of a target's location on the ground plane. At successive time instants, those samples with the same ID reveal the traces of that target over time.

Assume we have already identified M targets $\{H_0, H_1, \ldots, H_{M-1}\}$ on the ground plane inside the surveillance zone at the previous moment. If we treat the ground plane as a 2–D feature space and the ground position X as a feature point, the ID assignment problem can be treated as a typical classification problem. Also, since there could be some newly appearing targets at the current moment, we add one extra target H_M to handle these samples caused by new comers.

To label these "undecided" samples, we first construct the likelihood function $p(X|H_k)$ for $k = 0, 1, \ldots, M$. For $k = 0, 1, \ldots, M-1$, we model $p(X|H_k)$ as a Gaussian function. At time t, assume there are R "decided" samples $\{X_{k,0}, X_{k,1}, \ldots, X_{k,R-1}\}$ with their IDs being labeled as H_k. To help in removing inappropriate samples, we calculate the "color weight" for each of these "decided" samples. Assume at the previous time instant, the position of H_k was estimated to be at $\mu^{k,t-1}$. That is, we assume the corresponding cylinder of H_k is standing at $\mu^{k,t-1}$ at time $t-1$. By projecting that cylinder onto all camera views, we can get N projected regions. Based on the RGB values of these pixels inside these N projected regions, we can generate a color histogram for that target at $t-1$, denoted as $CH(b; \mu^{k,t-1})$, where b is the bin index of histogram. Similarly, for each sample $X_{k,j}$, we generate a cylinder at $X_{k,j}$ and collect its color information based on its projections over all camera views. This forms another color histogram $CH(b; X_{k,j})$. By calculating the Bhattacharyya distance between $CH(b; \mu^{k,t-1})$ and $CH(b; X_{k,j})$, we define the color weight of the sample at $X_{k,j}$. That is, we define

$$CW(X_{k,j}) \equiv \sum\nolimits_b \sqrt{CH(b; \mu^{k,t-1}) CH(b; X_{k,j})} \tag{12}$$

For the robustness of tracking, the status of a "decided" sample will be switched back to "undecided" if its color weight is smaller than a pre–defined threshold.

Based on these qualified "decided" samples, we can estimate the likelihood function $p(X|H_k)$ for each target H_k. As mentioned above, $p(X|H_k)$ is modeled as a Gaussian function. For these samples $\{X_{k,0}, X_{k,1}, \ldots, X_{k,R-1}\}$ that belong to H_k, we calculate their color weights $CW(X_{k,j})$. The mean vector and covariance matrix of $p(X|H_k)$ are then estimated based on these color weights

$$\mu^{k,t} = \sum_{j=0}^{R-1} CW(X_{k,j}) X_{k,j} \Big/ \sum_{j=0}^{R-1} CW(X_{k,j}) \tag{13}$$

$$\mathbf{C}^{k,t} = \sum_{j=0}^{R-1} CW(X_{k,j})(X_{k,j} - \mu^{k,t})(X_{k,j} - \mu^{k,t})^T \Big/ \sum_{j=0}^{R-1} CW(X_{k,j}) \tag{14}$$

On the other hand, for H_M, we define

$$p(X|H_M) = \begin{cases} \frac{1}{W} & \text{if } X \in \text{surveillance zone} \\ 0 & \text{if } X \notin \text{surveillance zone} \end{cases}$$

This likelihood function implies that a new person may uniformly appear at any location within the surveillance zone.

Based the maximum likelihood decision rule, an "undecided" sample X is then classified as H_k if $p(X|H_k) > p(X|H_j)$ for j = 0, 1,.., M but j ≠ k.
Note that if a sample is classified as H_M, that sample belongs to a new comer.

3.3 Identification of New Targets

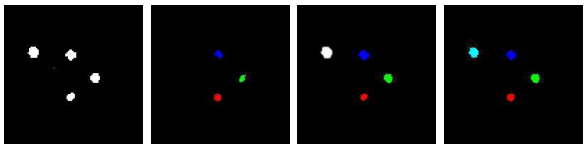
For those samples that are assigned to H_M, we further cluster them based on the mean–shift clustering technique [8]. This mean–shift clustering method is efficient and robustness. It doesn't require the prior knowledge about the number of new targets. Assume the set of samples $\{X_{M,0}, X_{M,1}, \ldots, X_{M,U-1}\}$ have been assigned to H_M. By iteratively calculating the next position y_{j+1} based on the previous position y_j, as expressed below:

$$y_{j+1} = \sum_{i=0}^{U-1} X_{M,i} \exp(\left\| \frac{y_j - X_{M,i}}{h} \right\|^2) \Big/ \sum_{i=0}^{U-1} \exp(\left\| \frac{y_j - X_{M,i}}{h} \right\|^2) \tag{15}$$

we can easily find a few convergence points. In (15), h is a parameter controlling the kernel size. Those samples that converge to the same convergence point are grouped together to form a new target. That convergent point is then defined to be the initial ground location of the new target.

3.4 Target Updating

Since there could be some new comers and some leaving people, we update the number of targets in the fourth module. For a new target, our system assigns



(a) (b) (c) (d)

Fig. 4. Illustration of System Flow.

a unique ID to it and increases the number of targets by one. In contrast, as a target leaves the scene, the samples assigned to that target will become fewer and fewer. Once if the number of samples is lower than a pre–defined threshold, that target is regarded as "absent" and we decrease the number of targets by one. All the samples assigned to that target will also be removed.

In Fig. 4, we show an example of the whole system flow. In this example, there were 3 people at the previous moment and a new person enters the scene at the current moment. In Fig. 4(a), we show the "undecided" samples generated by the mixture–MCMC sampler. We can see that samples have been successfully generated around all four targets. In Fig. 4(b), we show the samples generated by the SMC sampler. Samples assigned to the same target are plotted in the same color. As expected, no samples are generated around the new target. In Fig. 4(c) and (d), we show the result after sample labeling and the final result after target updating. The newly appearing target has been successfully detected and is colored in cyan. The other three targets are also successfully tracked.

4 Experiments And Results

In our experiments, we use both synthesized and real video sequences to test our system. The synthesized videos are generated by ObjectVideo Virtual Video (OVVV). By using OVVV, we can easily design various kinds of scenario and camera setups. We can also obtain the ground truth of the moving targets for performance evaluation. In the example shown in Fig. 5, we create a virtual scene with 10.78 m long and 8.82 m wide. Around the scene, four virtual static cameras are set up. In this example, we test the robustness of our system in tracking varying number of people. In the upper row of Fig. 5, we show four frames captured by these virtual cameras. In the lower row of Fig. 5, we show the bird's eye view of our result. Different colors correspond to different targets. The colored circles indicate the current target locations, while the colored tails indicate the traces of these targets in the previous moments. It can be easily seen that the proposed system can robustly detect and track multiple targets.

To objectively evaluate the performance of our system, we compare the estimated locations of all individuals with the ground truth provided by OVVV over 600 successive frames. As shown in Table 1, the maximum, minimum, and mean estimation errors have been calculated over two different test sequences to measure the tacking accuracy of our system. Besides, we also set up four static

Fig. 5. Upper Row: Four synthesized camera videos. Lower Row: The tracking result at frame 27. Check `http://archer.ee.nctu.edu.tw/~chingchun#new`.

cameras in our lab to capture real videos for testing. In Fig. 6, we show the captured images in the first row, and the 2–D detection results in the second row. It can be seen that there are plentiful errors in the detection results. However, based on the proposed system, reliable tracking results can still be obtained, as shown in the third row of Fig. 6. Up to now, the whole system has been implemented in C++ with a 2.4GHz Pentium–4 CPU. The color image size is 320 by 240. For one fusion, it takes about 0.3 seconds to perform the whole system steps, except the background subtraction process. Generally, the subtraction process can be executed at the camera side for a client–server surveillance architecture. Hence, the speed of the proposed system is reasonably fast for practical usage.

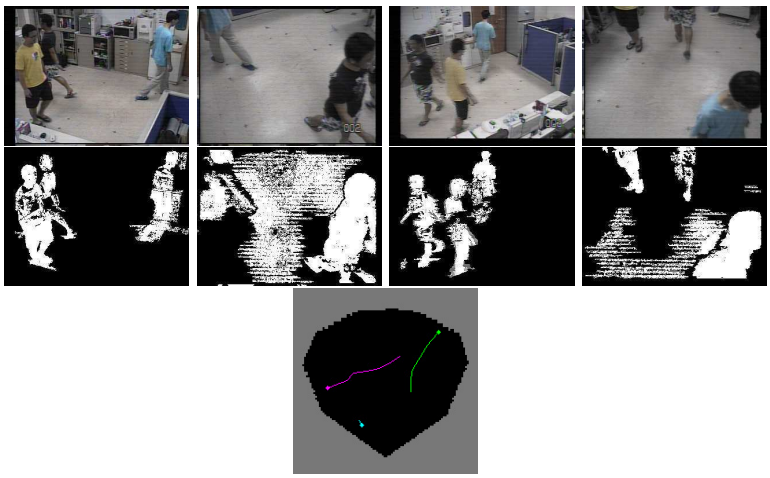


Fig. 6. Top Row: Four camera views from the real videos at Frame 1164. Middle Row: Foreground images. Bottom Row: The tracking result. Please browse our website.

Table 1. Estimation Errors of Two Synthesized Videos.

	Maximum error	Minimum error	Mean error
Seq 1	0.287 m	0.001 m	0.068 m
Seq 2	0.362 m	0.001 m	0.008 m

5 Conclusion

In this paper, we propose a multi–camera surveillance system for multi–target detection and tracking. Based on a Monte Carlo framework, we fuse the 2–D detection results from multiple cameras to establish a TDP distribution. The 2–D detection technique can be as simple as the background subtraction method. This approach avoids directly solving the complicated tracking and corresponding problems in the 2–D domains. Instead, we perform detection and tracking in the fused 3–D domain. Here, we have successfully converted the detection and tracking processes into a sample management process, which includes the generation o samples, the labeling of samples, the identification of new targets, and the updating of targets. With the proposed Monte Carlo framework, these four modules have been effectively combined into a unified framework.

References

1. Utsumi, A., Mori, H., Ohya, J., Yachida, M.: Multiple–human tracking using multiple cameras. Third IEEE International Conference on Automatic Face and Gesture Recognition (1998) 14–16
2. Mittal, A., Davis, L.: Unified multi–camera detection and tracking using region–matching. In: IEEE Workshop on Multi–Object Tracking Recognition, Vancouver, BC, Canada (2001) 3–10
3. Mittal, A., Davis, L.: M2tracker: A multi–view approach to segmenting and tracking people in a cluttered scene. International Journal of Computer Vision **51** (2003) 189–203
4. Hu, W., Hu, M., Tan, T., Lou, J., Maybank, S.: Principal axis-based correspondence between multiple cameras for people tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence **28** (2006) 663–671
5. Fleuret, F., Lengagne, R., Fua, P.: Fixed point probability field for complex occlusion handling. IEEE International. Conference on Computer Vision (2005)
6. Fleuret, F., Berclaz, J., Lengagne, R., Fua, P.: Multi–camera people tracking with a probabilistic occupancy map. IEEE Transactions on Pattern Analysis and Machine Intelligence **30** (2008) 267–282
7. Andrieu, C., de Freitas, N., Doucet, A., Jordan, M.I.: An introduction to mcmc for machine learning. Machine Learning **50** (2003) 5–43
8. Georgescu, B., Shimshoni, I., Meer, P.: Mean shift based clustering in high dimensions: A texture classification example. IEEE International Conference on Computer Vision (2003)