SUMMARY  Emerging video surveillance technologies are based on foreground detection to achieve event detection automatically. Integration foreground detection with a modern multi-camera surveillance system can significantly increase the surveillance efficiency. The foreground detection often leads to high computational load and increases the cost of surveillance system when a mass deployment of end cameras is needed. This paper proposes a DSP-based foreground detection algorithm. Our algorithm incorporates a temporal data correlation predictor (TDPC) which can exhibit the correlation of data and reduce computation based on this correlation. With the DSP-oriented foreground detection, an adaptive frame rate control is developed as a low cost solution for multi-camera surveillance system. The adaptive frame rate control automatically detects the computational load of foreground detection on multiple video sources and adaptively tunes the TDCP to meet the real-time specification. Therefore, no additional hardware cost is required when the number of deployed cameras is increased. Our method has been validated on a demonstration platform. Performance can achieve real-time CIF frame processing for a 16-camera surveillance system by single-DSP chip. Quantitative evaluation demonstrates that our solution provides satisfied detection rate, while significantly reducing the hardware cost.

key words: foreground detection, DSP, frame rate control, multi-camera surveillance

1. Introduction

Video surveillance is the monitoring of the behavior, activities, or other changing information in order to ensure security. Content analysis functionalities such as foreground detection, visual object tracking, and event analysis are attempted to integrate with traditional surveillance to achieve intelligent and automated surveillance. This emerging surveillance system can automatically understand events happening at the site [1]. Development of robust content analysis functionalities for automated surveillance has been gaining substantial interest in recent years partially due to the progress in hardware technology scaling that enables more computation-intensive analysis tasks to be realized with reasonable performance.

Among content analysis tasks, foreground detection is an important and common early vision task in surveillance systems. A wide range of foreground detection algorithms have been proposed in the literatures. In [20]–[22], comparisons on segmentation qualities are made to evaluate the most widely implemented foreground detection approaches. Table 1 shows five algorithms that are cited by many literatures, namely frame difference (FD) [2], [19], linear predictive filter (LPF) [8], [20], mixture of Gaussian (MoG) [3]–[6], [21], kernel density estimation (KDE) [7], and histogram model (HM) [9]–[11], [17]. Using FD, background/foreground detection is achieved by the difference of the pixels between two adjacent frames. The simplicity of the algorithm comes at the cost of the detection quality. In general, FD is sensitive to noise and illumination change, and it fails to detect inner pixels of a large, uniformly-colored moving object. As more sophisticated algorithms are utilized aiming for improved robustness and detection quality, the complexity of realizing such systems increases.

We concentrate on the realization of foreground detection to meet real-time requirements on the multi-camera video surveillance system. For realistic implementation of such system, trade-offs have to be made between system robustness and system performance. From Table 1, the KDE and histogram approaches have the higher detection quality which comes at the cost of a high computation complexity and even to an increased memory requirement. The cost can be further magnified proportionally to the number of cameras in the system. These facts have led us to choose the MoG approach for the developed system. The advantage of the MoG is capable of handling numerous situations where some background objects are not perfectly static, with lower hardware cost as comparing with KDE or HM.

Various modifications to the MoG for potential improvements are reported in [4]–[6]. However, only few works address the issue of algorithm performance in terms of meeting real-time requirements. In [3], a frame rate of 11–13 fps is obtained for a small frame size of 160 × 120 on an SGI O2 workstation. In [17], the MoG is optimized
for a DSP platform and the frame rate of 23–25 fps is observed for video sequence with 352 × 240 resolution. As to meet the real-time requirements in a multi-camera video surveillance system, the straightforward implementation of the MoG can limit its performance. Therefore, possible algorithm modification that can lead to better computation efficiency is needed for such system.

In this paper, a low cost MoG solution for multi-camera surveillance system is presented. Our solution is based on a temporal data correlation predictor (TDCP) and an adaptive frame rate control. The TDCP estimates the correlation between temporal pixels and reduces its computational burden on background modeling against its correlation. The adaptive frame rate control further enhances the TDCP to automatically stabilize a specified frame rate when a mass deployment of cameras is needed. The rest of this paper is organized as follows. In Sect. 2, TDCP based MoG method is given. Section 3 details the TDCP based MoG and frame rate control realization. In Sect. 4, the experimental results and system performance are derived. Section 5 concludes this paper.

2. Predicted Statistics Based Foreground Detection Method

The foundation of the proposed foreground detection method is a multiple background maintenance method [5] and a statistic prediction method, namely TDCP. The multiple background maintenance method models the recent history of observation at a given position as a mixture of $K$ Gaussian clusters, and each Gaussian $\theta$ associates with weight $\omega$, mean $\mu$ and variance $\sigma^2$. The block diagram of the proposed method is shown in Fig. 1. It consists of three major stages: update, differentiation and prediction. The update stage estimates most significant background image from background model against the current frame. Then, the differentiation stage subtracts the current frame with the estimated background image to extract the foreground pixel. The prediction stage exhibits the temporal data correlation by the updated background model to control the incoming update procedure.

2.1 Update and Differentiation

The soft-partition update strategy [6] is adopted instead of winner-take-all as principal that in [3]–[5]. The soft-partition updates all Gaussians by an amount proportional to their posterior probabilities. With this strategy, the parameters can more precisely reflect true background if the background is time varied. Let $I_{s,t}$ be the pixel at position $s$ at time $t$. The posterior probability $p(\theta_i | I_{s,t})$ of $i$-th Gaussian $\theta$ is denoted as

$$P(\theta_i | I_{s,t}) = \begin{cases} 1, & \text{if } i = \text{SC} \\ \frac{g_i(I_{s,t})}{\sum_{i=1}^{SC} g_i(I_{s,t})}, & \text{else} \end{cases}$$

where SC is the most significant cluster in the mixture, and denoted as

$$SC = \arg\max \left\{ \omega_{i,s,t-1} \cdot g_i(I_{s,t}) \right\}$$

All weights are updated through the exponential-forgotten manner with its adapted learning rate $\eta$. The $\mu$ and $\sigma^2$ associated with SC cluster are updated with two possible situations: the long-term observation and the short-term observation. The short-term observation is selected if the temporal difference satisfying $I_{s,t} - I_{s,t-1} = 0$. In this case, the mean is updated only because the variance can be considered as temporal consistency in short term observation. The mean is updated as

$$\mu_{s,t,j} = I_{s,t}$$

Otherwise, parameters of SC are updated by long-term observation against its respective posterior probability with its adapted learning rate. The rest of mean and variance of each Gaussian are also updated by the long-term observation. Once all posterior probabilities in (1) are approximated to zero, the least weight cluster $L$ is replaced with a cluster with the current pixel $I_{s,t}$ as its mean, an initial high variance, and zero as its weight. The background image $EB_{s,t} = \mu_{s,t,B}$ is reconstructed by mean of class $B$ corresponding to the largest weight

$$B = \arg\max_i \{\omega_i\}$$

the foreground and background are separated by thresholding the difference value between estimated background image and current image. The binarized result $O$ is then denoted as

$$O_{s,t} = \begin{cases} \text{foreground}, & \text{if } |I_{s,t} - EB_{s,t}| > Th \\ \text{background}, & \text{otherwise} \end{cases}$$

where $Th$ is a threshold value. After thresholding, opening by partial reconstruction and closing by partial reconstruction [12] are applied to eliminate misclassified foreground pixel in $O$.

2.2 Prediction

In general, the background pixel will be frequently appeared in a 24/7 video surveillance. Updating the background pixel would become trivial if the background pixel is unchanged. On the other hand, updating the background pixel would become significant if the background is varied. Once the attri-
distribution of a certain pixel can be known in prior, the update can be done within an appropriate interval depending on its attribution instead of updating it at each time install. This paper presents TDPC to obtain the attribution of pixel. The TDPC examines the correlation between clusters in the mixture of a pixel. The attributions of a pixel is classified as four types: Non-correlated Mixture with Highest SC (NML), Non-correlated Mixture with Lowest SC (NML), Two correlated Mixture with Highest SC (TMH), and Two correlated Mixture with Lowest SC (TML). Figure 2 (a) shows the discrimination between NML type and NML type. The NML type contains a static background cluster where an unchanged background pixel frequently appears and the current pixel probably belongs to that background pixel. The NML type contains the same type of background cluster but it is different at the current pixel that probably belongs to an occluded foreground pixel. The discrimination between TMH type and TML type is shown in Fig. 2 (b). The TMH type contains a dynamic background cluster where two varied background pixels alternatively appear and the current pixel probably belongs to one of background pixel. The TML type contains the same type of background cluster but it is different in the current pixel that probably belongs to an occluded foreground pixel.

The cluster distance $d_c$ is used to determine whether two clusters are correlated or not

$$d_c = |\mu_i - \mu_j|, \forall i, j \notin L$$  \hspace{1cm} (6)

the weight distance $d_w$ is used to determine the effective observation diversity of two clusters

$$d_w = |\omega_i - \omega_j|, \forall i, j \notin L$$  \hspace{1cm} (7)

the definition of attributions is detailed in Table 2 in terms of cluster distance $d_c$, weight distance $d_w$ and the significant cluster SC. With the defined attribution of each pixel, the region update control can determine the appropriate update period for each macroblock (MB). Let $cnt_s$ correspond to the update counter of an MB at position $s$. Then an MB is considered as urgent priority if inside pixels are dominated by TMH attribution, and sets its $cnt_s = \beta_{TMH}$. An MB is considered as second priority if inside pixel is dominated by TML attribution, and sets its $cnt_s = \beta_{TML}$. If inside pixels of a MB are dominated by NML attribution, an MB is considered as third priority and sets its $cnt_s = \beta_{NML}$. Otherwise, an MB is considered as fourth priority and sets its $cnt_s = \beta_{NML}$. After each time install, each $cnt$ is incrementally decreased. Once the $cnt$ of an MB is counted to zero, pixels in that MB are applied for update stage as the notation shown in Fig. 1.

3. System Design by DSP Architecture

The background subtraction method shown in Fig. 1 can involve an acquisition task, a processing task and a visualization task while mapping to hardware realization. The acquisition and visualization tasks capture and display video stream respectively and should be synchronized by a pixel rate associated with the video stream. The processing task is dedicated to perform foreground detection and should be operated at a faster clock rate to facilitate the computation. In this paper, a hybrid architecture made of synchronous and asynchronous modules, which particularly adapts to such vision based task, is presented.

Figure 3 illustrates the hybrid architecture block diagram. In this system, acquisition and visualization are computed with two synchronous modules: standard definition encoder and standard definition decoder, such that a lot of problems related to data control by the DSP can be alleviated. The SDRAM enables asynchronous data access and ensures continuity of video data flow. In SDRAM, two sections are allocated as frame buffers. These two frame buffers store the needed frames which can be read/written from DSP by using DMA. Therefore, a frame-based pipeline scheme is applied to increase the throughput of the system. The frame-based pipeline scheme is processed as follow: the frame at time $t$ is captured while the frame at $t - 1$ is processing and

| Attribution     | Cluster distance $d_c$ | Significant cluster SC | Boost factor $\beta$ |
|-----------------|------------------------|-------------------------|----------------------|
| NML             | $d_c > 2.5\sigma_c$    | $B$ cluster             | $\beta_{NML} = 4$   |
| NML             | $d_c < 2.5\sigma_c$    | $B$ cluster             | $\beta_{NML} = 3$   |
| TMH             | $d_c > 2.5\sigma_c$    | $B$ cluster             | $\beta_{TMH} = 2$   |
| TML             | $d_c < 2.5\sigma_c$    | $B$ cluster             | $\beta_{TML} = 1$   |

Table 2: Conditions in attribution determination.
The presented hybrid architecture made of synchronous and asynchronous modules.

The frame at \( t - 2 \) is displaying. In addition, the data transferred to the data path of DSP is packed into a MB since the TDCP essentially evaluates and determines the update period according to the pixels in MB. The double buffering scheme is used to achieve pipelining process of MB. This scheme is processed as follow: one MB buffer reads the following MB from SDRAM and writes the processed MB to SDRAM by DMA transfer, while the data path of DSP reads/writes the required MB from the other MB buffer. A portion of SRAM is configured as MB buffers set A and B. Each buffer set involves that one buffer reads/writes to SDRAM by DMA transfer while the other reads/writes to the data path of DSP [15].

The processing task involves asynchronous modules: SRAM, data path of DSP, Timer and DMA controller. Pipeline and parallel design strategies are applied to the update and differentiation stages on the data path of DSP. The intrinsic functions dealing with multiple data in parallel are applied to differentiation stage. These functions include SUBABS4 and CMPGTU4 which can perform the subtraction with absolute value and the comparison for four pixels concurrently in each data path. Moreover, the image/video functions [14], which are based on 32-bit-wide bit-wise AND and OR, are applied to implement the post-processing. By the information generated by prediction stage, the TIMER hardware is utilized to further enhance the processing frame rate when a mass of deployment of cameras is needed. In the following, several details about DSP architecture are explained to understand the design strategies.

3.1 Data Path of DSP

The DSP is a fixed-point Very Large Instruction Word processor with two data paths [13]. Each path has a unit for multiply operations (.M), for logical and arithmetic operations (.L), for branch, bit manipulation, arithmetic operations (.S), and for loading/storing and arithmetic operations (.D). All the data transfers make use of the .D units. Addition and subtract operations can be performed by all the units. The floating-point multiplier can only be performed by .M units. Each path includes a set of 32-bit registers, and each functional unit can directly read from or write to the registers within its own path. In DSP architecture, parallelism is the major design issue to extremely high performance. High parallelism can be achieved by software pipeline manner [13]. In a single cycle, the DSP can reach the maximum performance where eight units are executed in parallel with software pipelined procedure.

3.2 Data Independency Procedure

By typical time proportion analysis for each stage, the update stage accounts for 89% of the background subtraction complexity. This is mainly because a large amount of update conditions and floating-point divider, square and exponential operations are required for computing posterior probabilities \( p(\theta|I) \) (i.e. pass 1) and updating Gaussians \( \theta \) by \( p(\theta|I) \) (i.e. pass 2) in the multiple background maintenance. However, these operations are inefficient for DSP software pipeline and always lead to worse DSP performance. Specifically, software pipeline attempts to generate efficient assembly code so that all functional units are utilized within one cycle. Considering the original processing flow shown in Fig. 4 (a), the computation of (1) requires obtaining each Gaussian PDF \( g_i \) and each of \( \omega_i \times g_i \) first. Then, each obtained \( g_i \) is accumulated. Until the accumulated value \( \sum_{i=1}^{T} g_i(I_{s_i}) \) is completed and stored in the buffer, the \( p(\theta|I_{s_i}) \) is obtained as 1 if \( i \)-th Gaussian is the most significant cluster in the mixture, and the rest as \( g_i/\sum_{i=1}^{T} g_i(I_{s_i}) \). The pass 1 mainly involves the branch operation and the
floating-point multiply operation which can take six pipeline phases and up to 10 phases respectively. However, due to the data dependency between (1) and (2) in pass 1, the related pipeline phases only utilize M units or S units within certain cycles until the branch or computations associated with divider, square and exponential operations are finished. This implies that most of functional units are idle when the pipeline phases associated with pass 1 are performed.

Pass 2 consists of a loop, in which parameters of all clusters are updated based on long-term observation or short-term observation. For both observations, the weights are updated by exponential-forgotten

$$\omega_{t,i,j} = (1 - \alpha \beta_j) \omega_{t-1,i,j} + \alpha \beta_j P(\theta|I_{t,i})$$

(8)

where \( \beta \) is the boost factor associated with j-th attribution. Updated mean and variance are conditionally different. In the short-term observation, only the mean is simply updated by (3). In the long-term observation, however, both the mean and the variance are updated by the adapted learning rate \( \eta \) to reflect the \( P(\theta|I) \). The adapted learning rate is denoted as

$$\eta_i = \left\{ \begin{array}{ll}
1 - \alpha \beta_j & , i = SC \\
\frac{P(\theta|I_{t,i}) \cdot ((1 - \alpha \beta_j)/\omega_i + \alpha \beta_j)}{\omega_i} & , i \neq SC
\end{array} \right.$$  

(9)

Then, mean and variance are successively updated as

$$\mu_{t,i,j} = (1 - \eta_i) \cdot \mu_{t-1,i,j} + \eta_i \cdot I_{t,i,j}$$

(10)

$$\sigma^2_{t,i,j} = (1 - \eta_i) \cdot \sigma^2_{t-1,i,j} + \eta_i \cdot (I_{t,i,j} - \mu_{t,i,j})^2$$

(11)

Despite the DSP can build software pipeline for the floating-point arithmetic, the conditional jumping will greatly decrease the pipeline efficiency. The total drawback of update procedure in Fig. 4 (a) is that the strong data dependency on long critical computation path prevents DSP to build efficient pipelining.

To split the data dependency, buffer is allocated for temporally storing \((I_{t,i,j} - \mu_{t,i,j})^2 \) value, which will be applied for (1) and (11). Then, a sorting procedure is added as additional pass 3 to further decrease the critical path on (2). The pass 3 rearranges weights in increasing order so that obtained SC can be replaced by sequentially finding match from cluster associated with the lowest weight to cluster associated with the highest weight. A match is found if the distance between pixel \( I \) and mean of a cluster is smaller than 2.5 standard deviation of that cluster. An update priority list is built prior to pass 2 to number the number of conditional jumping in the pass 2. The priority list rearranges the SC cluster and non-SC clusters. Since only SC cluster requires branch operations, the branch operations associated with non-SC clusters can be removed from the loop. The priority list, therefore, improves the software pipeline efficiency in the update loop in pass 2. The data independency procedure shown in Fig. 4 (b) provides a more clear description of the improved update stage.

The computation of Gaussian PDF \( g \), which involves the longest critical path, is considered also. The Gaussian PDF is given as follow

$$g(I_{t,i,j}) = (2\pi \sigma^2_{t,i,j})^{-1/2} \cdot \exp\left(\frac{(I_{t,i,j} - \mu_{t,i,j})^2}{2\sigma^2_{t,i,j}}\right)$$

(12)

The look-up-table (LUT) for linear interpolation of Gaussian PDF [18], which builds three tables for \( \sigma^2_{t,i,j} \), \( (2\pi \sigma^2_{t,i,j})^{-1/2} \), and \( \exp(.) \) respectively, is adopted. Two addresses of look-up table are generated by a particular value that is closed to the addresses in the table. Then, the resulted data of a particular value is linearly interpolated by the two indexed data. The required size of three LUTs is only 120 Bytes [18], and the exponential result by the 120 Bytes LUT is close to the result by floating-point values (see Fig. 5). Although, the truncation errors may disturb the detection result, the discussion of experimental results shown in Sect. 4 demonstrates that the effect on truncation errors can be satisfactorily removed by the post-processing.

3.3 Frame Rate Control

Without additional hardware resources for a multi-camera surveillance system, stabilizing the processing frame rate of background subtraction in real-time is difficult, since the processing data is increasing proportionally to the number of cameras. This paper introduces a self-frame-rate-control to stabilize the processing load once the processing frame rate is smaller than the target frame rate without additional hardware resources. The diagram of the proposed self-frame-rate-control is shown in Fig. 6. The selective control is handled as a problem of feedback control which consists of a control input and two control variables. Here the controlled variables are the processing frame rate of the overall system \( X \) (i.e., the number of processed frame per second) and the frequency of DSP processor. \( V \) is the target frame rate (i.e., the number of target frame per second). The control input is update interval \( M \). \( M \) refers to the jump of interval in updating of the frame.

![Floating-point and fixed-point exponential values](image)

![The diagram of the feedback frame rate control](image)
In this paper, we try to stabilize the processing frame rate by controlling the load of the update stage. Let the frame rate deviation $E$ be the absolute difference between the processing frame rate $X$ and the target frame rate $V$. The processing frame rate is determined by TIMER module in DSP [16]. The TIMER module consists of a 32-bit counter that can count the number of required cycle for each frame. Let $Cycle_t$ be the number of required cycle for the frame $t$. Then the present processing frame rate for frames within $t$ to $t + V - 1$ is determined by the average of required cycle per frame as

$$X(t) = \frac{\text{Frequency of DSP}}{V} \sum_{i=0}^{t+V-1} Cycle_i$$  \hspace{1cm} (13)

Since the frequency of DSP essentially represents how many cycles can be executed per second, dividing the frequency of DSP with the average of required cycle per frame is the number of processing frame per second.

Then, stabilizing the processing frame rate is turned to minimize the deviation $E$. An efficient way to minimize $E$ is to enlarge/diminish the MB update intervals defined in Table 2 by the previous processing frame rate. Specifically, the MB update intervals should be enlarged for the next $V$ frames (i.e. frames within $t$ to $t+V-1$) once the previous processing frame rate $X(t-V)$ is smaller than the target frame rate, or the MB update intervals should be diminished once $X(t-V)$ is larger than the target frame rate. The factor $M(t)$, which defines the increased jump interval in the MB update period for next frames within $t$ to $t+V-1$, is estimated as

$$M(t) = \left[ \frac{V}{X(t-V)} \cdot M(t-V) \right]$$ \hspace{1cm} (14)

where $M(0) = 1$ at initial. However, the increased update interval of MB may induce a huge amount of MB to be updated at a certain frame and lead to a long processing latency.

To overcome such problem, we partition the entire frame into $M(t)$ slices for each camera to alleviate the long latency occurred at certain frame. Figure 7 shows an example of frame rate control and associated data flow of background subtraction for a 4-camera surveillance system. In this case, the target frame rate $V$ is 30 and frames within $t$ to $t + 29$ of each camera are partitioned into three slices by (14), i.e. $M(t) = 3$. Let $F_i$ be denoted as the frame at time $t$ and $F_i^t$ be denoted as $i$-th slice in the frame. Then the update layer is performed in slice domain where only one certain slice $F_{t+n}^{\text{mod}(n,M(t))}$ in the frame at time $t+n$, $0 \leq n \leq V - 1$, is updated. $\text{mod}(n,M(t))$ is the modulo operation which finds the remainder of division of $n$ by $M(t)$. After slice $F_{t+n}^{\text{mod}(n,M(t))}$ is updated, the slice of background image $EB_{n}^{\text{mod}(n,M(t))}$ is reconstructed using the updated information. In contrast, the differentiation layer is performed in frame domain in order to extract foreground objects in the current frame. The Difference($F,EB$) represents thresholding the absolute difference value between current frame $F$ and background image $EB$, and applying the post-processing to the resulted binary image $O$.

4. Experimental Result

The proposed method has been evaluated on various surveillance video sequences including Pets2001, Pets2007, Shopping center (SC), and benchmark data set [20]. These sequences involve indoor and outdoor environments which can evaluate the performance against several canonical background modeling situations mentioned in [7], [20].

**Parameters and performance measures:** our method comprises the learning rate $\alpha$, the threshold $Th$, and the amount of Gaussian clusters $\tau$. Due to the consideration in computation complexity, $\tau = 3$, which is the minimum number of Gaussian to character the multimodal background distribution, is chosen. To see how the proposed method is sensitive to small changes of the rest parameter values, we calculated the error classifications for different parameter settings. As error classification measures, we used the $\text{recall} = N_c/N_{gt}$ and $\text{precision} = N_c/N_{det}$ measures, where $N_c$ denotes the number of foreground pixels correctly detected, $N_{gt}$ the number of foreground pixels in the ground truth, and $N_{det}$ the number of detected foreground pixels. The ground truth is achieved by manually labeling some frames. Also, we used the F-measure defined as $F = 2(\text{precision} \cdot \text{recall})/(\text{precision} + \text{recall})$. Figure 8 shows the different curves obtained by varying the threshold values $Th$. For all parameters, a good value can be chosen across a wide range of values. Overall, in the cases of Pets2001 sequence shown in Fig. 8(a), a value of $\alpha = 0.01$ with $Th = 16$ gives the best F-measure performance. In the cases of SC sequence shown in Fig. 8(b), a value of $\alpha = 0.05$ with $Th = 10$
gives the best F-measure performance. In the following experiments, these two parameter settings are applied to all indoor type of sequences and all outdoor type of sequences respectively.

Detection performance evaluations: the foreground detection results are compared with a FD based algorithm [19], the GMM algorithm [3], the MBM [5], and a LBP based algorithm [10]. To show the effect on the updating by LUT-Gaussian PDF, TDCP technique, and frame rate control scheme, both the foreground detection results without post-processing and with post-processing are obtained. Figure 9 shows the foreground detection results by sequences in [20]. The misclassified pixels caused by the truncation errors of LUT-Gaussian PDF and prediction errors of TDCP can be satisfactorily removed using the opening by reconstruction, and foreground aperture effect can also be filled using the closing by reconstruction. Similar results can be observed from Fig. 10, in which the foreground detection results by Pets2001, SC, and Pets2007 are shown respectively. Except the foreground shadows are misclassified as foreground, no other misclassified background pixels are observed even under background suffering weather change and light reflected material in background. The quantitative evaluation result is given in Table 3. The performance of proposed system much approaches to the performance of [5] after post-processing. As comparing with a more sophisticated algorithm [10], the results of proposed algorithm are generally lower than the results of [10] before post-processing. However, the results of proposed algorithm are closed to that of [10] after post-processing. Therefore, as consideration to the design of real-time system for foreground detection, we adopt a low complexity of background modeling procedure and combine it with a robust post-processing instead of adopting a high complexity of background modeling only.

The detection performance with frame rate control is evaluated by multiple input video sources. Indoor surveillance and outdoor surveillance are considered respectively as different control performance. Each input source in this evaluation is assumed to be the same. Figure 11 shows the results of frame rate control against different number of camera. In Fig. 11 (a), the frame rate for single camera is much unstable than that in Fig. 11 (b) without applying the control scheme. This is because the illumination change and the background variation will be often encountered in outdoor surveillance than in indoor surveillance. The appearances of such variations will cause a huge amount of MB to be updated in a short period by the TDCP technique, and so decrease the frame rate. This situation can be observed from the period within 75 s to 85 s in Fig. 11 (a), at which strong illumination change is encountered due to the cloud quickly occluding sun. In such situation, the frame rate will suffer...
Table 3  The quantitative evaluation and comparison result.

| Method | Without post-processing | Indoor environment | Outdoor environment | Average |
|--------|-------------------------|--------------------|---------------------|---------|
|        | Precision               | Recall             | Precision           | Recall  | |
| FD     | 0.90  0.58             | 0.72  0.52         | 0.68  0.59          | 0.79  0.54 | 0.61  0.52 |
|        | 0.62  0.52             | 0.68  0.59          | 0.61  0.43          | 0.61  0.43 | 0.61  0.23 |
|        | 0.72  0.68             | 0.79  0.61          | 0.61  0.43          | 0.61  0.43 | 0.61  0.23 |
|        | 0.52  0.43             | 0.59  0.34          | 0.61  0.43          | 0.61  0.43 | 0.61  0.23 |
| With post-processing | 0.72  0.68             | 0.79  0.61          | 0.61  0.43          | 0.61  0.43 | 0.61  0.23 |
| GMM    | 0.92  0.62             | 0.72  0.51          | 0.74  0.66          | 0.77  0.54 | 0.61  0.34 |
|        | 0.71  0.66             | 0.87  0.54          | 0.81  0.43          | 0.66  0.35 | 0.66  0.35 |
|        | 0.65  0.54             | 0.87  0.43          | 0.81  0.35          | 0.64  0.35 | 0.64  0.35 |
| With post-processing | 0.72  0.64             | 0.51  0.43          | 0.87  0.43          | 0.51  0.43 | 0.51  0.43 |
| MBM    | 0.98  0.65             | 0.73  0.66          | 0.88  0.51          | 0.74  0.51 | 0.67  0.34 |
|        | 0.73  0.51             | 0.86  0.66          | 0.91  0.51          | 0.74  0.51 | 0.74  0.51 |
|        | 0.66  0.51             | 0.88  0.66          | 0.72  0.51          | 0.66  0.51 | 0.66  0.51 |
|        | 0.66  0.51             | 0.73  0.51          | 0.68  0.51          | 0.66  0.51 | 0.66  0.51 |
| LBP    | 0.98  0.65             | 0.70  0.66          | 0.73  0.51          | 0.77  0.51 | 0.73  0.51 |
|        | 0.71  0.51             | 0.90  0.66          | 0.69  0.51          | 0.73  0.51 | 0.73  0.51 |
|        | 0.65  0.51             | 0.87  0.66          | 0.71  0.51          | 0.69  0.51 | 0.69  0.51 |
| With post-processing | 0.72  0.66             | 0.51  0.51          | 0.74  0.51          | 0.70  0.51 | 0.70  0.51 |
| This work | 0.97  0.65             | 0.70  0.66          | 0.74  0.51          | 0.81  0.51 | 0.81  0.51 |
|        | 0.72  0.66             | 0.86  0.51          | 0.74  0.51          | 0.81  0.51 | 0.81  0.51 |
|        | 0.66  0.51             | 0.86  0.51          | 0.74  0.51          | 0.69  0.51 | 0.69  0.51 |
|        | 0.69  0.51             | 0.87  0.51          | 0.74  0.51          | 0.69  0.51 | 0.69  0.51 |

much serious degradation once the number of camera is increasing. The proposed frame rate control scheme successfully maintains the real-time processing at a rate of 30 fps/s for the multi-camera surveillance. Once the frame rate by previous $M$ cannot achieve real-time processing, (14) adaptive increases the current $M$ value. On the other hand, (14) decreases the current $M$ value once the processing frame rate exceeds the target frame rate too much. Similar results are observed from the Fig. 11 (b) by the indoor sequence SC.

Figure 12 summarizes the averaged performances with frame rate control for indoor and outdoor sequences. In the indoor surveillance, both the precision rate and recall rate exhibit graceful degradation proportionally to the number of input source. In outdoor surveillance, the precision rate shown in Fig. 12 (a) exhibits sharp degradation when the number of input source exceeds 4. This implies that the performance can be more easily influenced in outdoor surveillance than that in indoor surveillance with the same update.
period. With the proposed frame rate control, no hardware resource is additionally required to stabilize the processing frame rate with an acceptable detection quality loss.

System performance evaluations: The IEK-C6416 platform is used to validate the proposed background subtraction framework. The IEK-C6416 is based on C6000 VLIW Harvard architecture run at 600 MHz. The video source is a standard surveillance CCD camera connected to a composite input on the platform. The video output is taken from the VGA output and displayed on a standard display. Figure 13 shows the environment for validation of the proposed method. This prototype evaluates the performance of background subtraction by capturing and processing visual data in real-time. The demonstration video can be found in http://dsp.ee.ncu.edu.tw/surveillance.

Figure 14 shows the average number of DSP cycle for foreground detection at a frame with CIF resolution, after each one of the seven optimization strategies. The overall system before optimization can only achieve 1.1 fps, in which update stage dominates over 95% of complexity. By applying the TDCP, the complexity of update stage has been reduced to 173 Mega cycles and the frame rate is increasing to 3.4 fps. We also evaluate the optimizations related to the DSP architecture. Without the data dependency procedure, the software pipeline can only reduce complexity to 114 Mega cycles. The data dependency procedure can further decrease the complexity to 46.9 Mega cycles by splitting data dependency and thus increase the software pipeline efficiency. After LUT-Gaussian PDF, double buffer scheme, and intrinsic functions, the complexity is reduced to 30.2 Mega cycles, 24.2 Mega cycles, and 20.08 Mega cycles respectively. The system performance of the proposed system is compared with other foreground detection systems realized on DSP based platform. In [19], a Markov random filed (MRF) based FD algorithm is implemented by a 33 MHz Motorola 96002 DSP. In [17], the MoG based algorithm is implemented by a 720 MHz DM642 DSP. Table 4 shows the performance evaluation in terms of the required cycles per pixel and averaged F-measure. Comparing with the system in [19], the foreground detection requires the fewest cycles/pixel since this method is based on FD. On the other word, [19] cannot adapt to the environment containing background variation. As for the proposed system with frame rate control scheme, the cycles/pixel outperforms [17] and [19]. This is because our system can detect foreground for multiple cameras concurrently by stabilized processing frame. To ensure that the detection quality can be satisfied with the successive content analysis tasks such as foreground tracking, the maximum numbers of supported cameras for indoor and outdoor surveillance are given as 16 and 4 respectively according to the evaluation result depicted in Fig. 12.

5. Conclusion

This paper proposes a DSP-based background subtraction for multi-camera surveillance system. The proposed method is based on temporal data correlation predictor which dis-
covers the attribution of MB. Each different attribution is associated with different update interval to save unnecessary pixel update. The software pipeline built in update stage is optimized by splitting data dependency. The computational extensive floating-point PDF is replaced with a fast LUT based PDF approach. Parallel intrinsic functions and double buffer arrangement on L2 internal memory are also utilized. The self-frame-rate-control is described to stabilize the processing frame rate without additionally computational resource when a mass deployment of end cameras is needed. The performance is evaluated by the subjective view and quantitative evaluation. Both these two results are satisfactory and show that the truncation errors caused by optimization strategies are alleviated. The evaluation on multi-camera surveillance shows that the processing frame rate can be stabilized without additional hardware cost with an acceptable detection quality loss.

References

[1] Valera S. and Velastin, “Intelligent distributed surveillance systems: A review,” IEEE Proc. Vis. Image, Signal Process., vol.152, no.2, pp.192–204, April 2005.
[2] S.C. Liu, C.W. Fu, and S. Chang, “Statistical change detection with moments under time-varying illumination,” IEEE Trans. Image Process., vol.7, no.12, pp.1258–1268, Sept. 1998.
[3] C. Stauffer and W.E.L. Grimson, “Adaptive background mixture models for real-time tracking,” Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp.246–252, 1999.
[4] O. Javed, K. Shafique, and M. Shah, “A hierarchical approach to robust background subtraction using color and gradient information,” Proc. IEEE Workshop Motion Video Computing, pp.22–27, Dec. 2002.
[5] T.-H. Tsai, W.-T. Sheu, and C.-Y. Lin, “Foreground object detection based on multi-model background maintenance,” IEEE Int. Symposium on Multimedia Workshop, 2007.
[6] D.-S. Lee, “Effective Gaussian mixture learning for video background subtraction,” IEEE Trans. Pattern Anal. Mach. Intell., vol.27, no.5, pp.827–832, May 2005.
[7] A. Elgammal, R. Duraiswami, D. Harwood, and L.S. Davis, “Background and foreground modeling using nonparametric kernel density estimation for visual surveillance,” Proc. IEEE, vol.90, no.7, pp.1151–1162, July 2002.
[8] J. Zhong and S. Sclaroff, “Segmenting foreground objects from a dynamic textured background via a robust Kalman filter,” Proc. ECCV, pp.44–50, 2003.
[9] S. Zhang, H. Yao, and S. Liu, “Dynamic background modeling and subtraction using spatio-temporal local binary patterns,” Proc. IEEE Int. Conf. Image Process, Sept. 2008.
[10] M. Heikkila and M. Pietikainen, “A texture-based method for modeling the background and detection moving object,” IEEE Trans. Pattern Anal. Mach. Intell., vol.28, no.4, pp.657–662, April 2006.
[11] K. Yokoi, “Probabilistic BPRC: Robust change detection against illumination change and background movements,” IEICE Trans. Inf. & Syst., vol.E93-D, no.7, pp.1700–1707, July 2010.
[12] P. Salembier and M. Pardas, “Hierarchical morphological segmentation for image sequence coding,” IEEE Trans. Image Process., vol.3, no.5, pp.639–651, Sept. 1994.
[13] TMS320C64x DSP CPU and Instruction Set Reference Guide, SPRU732GE, Feb. 2008, http://www.ti.com
[14] TMS320C64x DSP Image/Video Processing Library Programmer’s Guide, SPRUF30A, Oct. 2007, http://www.ti.com
[15] TMS320C64x DSP Two-Level Internal Memory Reference Guide, SPRU610B, Aug. 2004, http://www.ti.com

Tsung-Han Tsai received the B.S., M.S., and Ph.D. degrees in electrical engineering from the National Taiwan University, Taipei, Taiwan, in 1990, 1994, and 1998 respectively. From 1999 to 2000, he was a professor of electronic engineering at Fu Jen University. In 2000, he joined the Department of Electrical Engineering, National Central University, Taiwan, where he is currently a Professor. He has been an IEEE member for over 10 years, and is also a member of the Audio Engineering Society (AES) and the Institute of Electronics, Information and Communication Engineers (IEICE). Dr. Tsai has been awarded 14 patents and has published more than 120 referred papers in international journals and conferences. His research interests include VLSI signal processing, video/audio coding algorithms, DSP architecture design, wireless communication, and System-On-Chip design. Dr. Tsai received the Industrial Cooperation Award in 2003 from the Ministry of Education, Taiwan. He is a member of the Technical Committee of IEEE Circuits and Systems Society, and serves as Technical Program Committee member or Session Chair of several international conferences.

Chung-Yuan Lin received the M.S. degree from the Department of Electrical Engineering, National Central University (NCU), Taoyuan, Taiwan, R.O.C., in 2003. He is currently pursuing the Ph.D. degree at NCU. His research interests include computer vision problem on surveillance system, video processing algorithm design, algorithm optimization for DSP architecture, and embedded system design.