Second Joint IEEE International Workshop on Video Surveillance and Performance Evaluation of Tracking and Surveillance,
In conjunction with the Tenth IEEE International Conference on Computer Vision, Beijing, China, October 15-21, 2005.

Modeling Background from Compressed Video

Weiqiang Wang[1,2] Datong Chen[1] Wen Gao[2] Jie Yang[1]

School of Computer Science, Carnegie Mellon University, Pittsburgh, 15213, USA[1]
Institute of Computing Technology &Graduate School, Chinese Academy of Sciences, Beijing 100080, China [2]
{wqwang, wgao}@jdl.ac.cn, {datong, jie.yang}@cs.cmu.edu

Abstract

Background models have been widely used for video surveillance and other applications. Methods for constructing background models and associated application algorithms are mainly studied in the spatial domain (pixel level). Many video sources, however, are in a compressed format before processing. In this paper, we propose an approach to construct background models directly from compressed video. The proposed approach utilizes the information from DCT coefficients at block level to construct accurate background models at pixel level. We implemented three representative algorithms of background models in the compressed domain, and theoretically explored their properties and the relationship with their counterparts in the spatial domain. We also present some general technical improvements to make them more capable for a wide range of applications. The proposed method can achieve the same accuracy as the methods that construct background models from the spatial domain with much lower computational cost (50% on average) and more compact storages.

1. Introduction

The explosive growth of video sources has created new challenges for data transmission, storage, and analysis. Various data compression technologies have been widely used to solve problems of video transmission and storage. We are now able to continuously record multiple video streams from video cameras onto a computer hard disk using hardware compression devices. However, how to process these video data is still an open problem. Traditional computer vision algorithms may not be efficient enough to process huge video data, because most computer vision algorithms are designed for uncompressed images in the spatial domain. Suppose we record video at 30 frames per second. For only one camera, we would have 2,592,000 frames per day. It is clear that we need more efficient ways to process these video data.

Video surveillance is a major source that can generate huge video data. Visual surveillance systems use video cameras to monitor the activities of targets in a scene, such as human activity in indoor environments and vehicles in parking lots. It is very difficult, however, for a human operator to remain alert for more than a few hours. Automatic tracking technologies have been studied for decades to replace or reduce human efforts. A fundamental technology used in the existing tracking systems is background subtraction, which segments moving regions in an image sequence captured from a static camera by comparing each new frame with a background model. A crucial step of this technique is to obtain a stable and accurate background model from a video sequence.

Much research has been directed to building background models. Background models have been estimated from pixel values at each location in a video sequence. Pixel values can be gray or color. Basic methods are the average method [1], the running average method, the median method [2], and the selective average method. Advanced methods are mostly based on statistical modeling techniques, such as the single Gaussian estimator (pfinder) [3], the mixture of Gaussian estimator [4], the kernel density estimator [5], the sequential kernel density estimator [6], the mean-shift estimator [6], the eigen background [7], and the robust PCA background [8]. Some methods also take the correlations of pixels with their neighborhood into account [9]. All the above methods require a video sequence in uncompressed format. On the other hand, researchers in multimedia processing areas have also proposed some methods for building background models in the compressed domain [10][11][12][13]. They were developed for segmenting moving objects or for encoding purposes. Most algorithms use only Discrete Cosine Transformation (DCT) DC coefficients and work on block level. For example, the algorithm in [11] extracts objects at a size of 8 by 8 blocks and can not obtain accurate object contour. Furthermore, these algorithms are disconnected from the spatial domain, where many computer vision and image processing technologies have been developed.

In this paper, we propose an approach to model background directly from a compressed video using DCT coefficients. The proposed approach can not only efficiently construct background models from compressed video but also achieve accuracy as good as that of algorithms in the spatial domain. This can lead to a more efficient framework to process compressed video data from both compressed domains and spatial domains. Furthermore, the proposed approach can take advantage of the structure and available information in the compressed video in implementing state-of-the-art background modeling algorithms. For example, when we use a mixture of Gaussian (MoG) to model the background from compressed video, the model has less non-zero parameters, because DCT coefficients are orthogonal.

The rest of the paper is organized as follows. In Section 2, we introduce modeling a background in the spatial domain and some basics for compressed video. We describe three

common background models: running average, median, and MoG, used in the spatial domain. In section 3, we present the proposed method. We discuss implementations of running average, median, and MoG in the compressed domain. The proposed three algorithms can achieve the same or comparable accuracy in the compressed domain as in the spatial domain but with much less computational cost. This means that those models obtained from the compressed domain can be directly used in the spatial domain, which bridges two domains together. In Section 4, we show the experimental results. In Section 5, we conclude the paper.

2. Problem Description

The goal of background modeling is to automatically obtain a static image that contains only background from a sequence of video captured by a fixed camera. Intuitively, we consider the main challenge of the background modeling is the occlusions of foreground objects. In practice, there are many other challenges from the motions of background objects and illumination changes of the environment, for example, the high-frequency background object motion (water waves, tree branches, and CRT display), camera oscillations, long-term static foreground object (e.g., a parked car), gradual lighting changes from sunshine, sudden lighting changes from clouds and regular lighting changes from indoor lights, etc.

2.1. Background Modeling in the Spatial Domain

Many background modeling methods have been proposed in the spatial domain. Here, we overview three typical algorithms in details. Lo et al. [1] proposed a fast algorithm that constructs the background image as the average of the previous *n* frames. The algorithm requires plenty of memory to store the previous *n* frames. A alternative method is called running average, which estimates the background B_{t+1} from only the current frame F_t and the previous background B_t:

$$B_{t+1} = \alpha F_t + (1-\alpha) B_t, \tag{1}$$

where the learning rate α is typically set as 0.05.

The drawback of these average methods is that the foreground objects can leave some "ghosts" in the background images. Cucchiara et al. [2] proposed to use a median function to obtain the background. In this algorithm, each location (x, y) in the background image $B_t(x, y)$ at time t is computed by the following equation:

$$B_t(x,y) = \arg\min_{i=0,\ldots,n} \sum_{j=0}^{n} D_t^{i,j}(x,y), \tag{2}$$

where $D_t^{i,j}(x,y) = \max_{c=\{r,g,b\}} \left| F_{t-i}(x,y).c - F_{t-j}(x,y).c \right|$, (3)

and the $F_t(x,y).c$ are the *R, G, B* values of the pixel at (*x, y*) in the frame for time t.

We can also use selective algorithm to remove residues from foreground objects. Each pixel in the current frame is first classified as either foreground or background. Those foreground pixels are not used in constructing the background model. The difficulty of the selective method is how to choose the classification threshold. Wren et al. [3] proposed to fit one Gaussian distribution to the histogram of the pixel values in previous *n* frames. This gives the background PDF with variances rather than single means (average values). Stauffer et al. [4] extended this idea into MoG $B_t(x,y) \sim N\left(\mu_t^i, \sigma_t^i, w_t^i\right)$. Each pixel has a MoG, which is firstly initialized by k-means and updated by every new frame. The algorithm first computes the matching models for each pixel (*x, y*):

$$M_t^i = \begin{cases} 1 & \left| F_t(x,y) - \mu_t^i \right| < 2.5\sigma_t^i \\ 0 & otherwise \end{cases}. \tag{4}$$

The weights are then updated as:

$$w_t^i = (1-\beta) w_{t-1}^i + \beta M_t^i, \tag{5}$$

where β is a constant related to the speed of the distribution change. The unmatched models remain the same and the matched model is updated as:

$$\mu_t^i = \left(1-\rho_t^i\right)\mu_{t-1}^i + \rho_t^i F_t(x,y) \quad \text{and}$$

$$\sigma_t^i = \left(1-\rho_t^i\right)\sigma_{t-1}^i + \rho_t^i (F_t(x,y) - \mu_t^i)^T (F_t(x,y) - \mu_t^i), \tag{6}$$

where $\rho_t^i = \frac{\beta}{\sqrt{2\pi}\left\|\sigma_t^i\right\|} \exp\left(-\frac{\left\|\left(F_t(x,y) - \mu_t^i\right)\right\|^2}{2\left\|\sigma_t^i\right\|^2}\right)$.

We will implement the counterparts of these three algorithms in the compressed domain in Section 3.

2.2 Compressed Video and DCT

To transmit and store video data efficiently, video compression techniques are employed to reduce the size of an image sequence by removing spatial and temporal redundancy. According to the popular international standards of video compression, such as MPEG-1, 2, 4, and H.26X, a compressed video consists of *I*, *P*, *B* frames where *P* and *B* frames can only be reconstructed by using adjacent *I* frames. Each *I* frame is first partitioned into 8 by 8 pixel blocks in the spatial domain, and then each pixel block is encoded as a set of Discrete Cosine Transformation (DCT) coefficients. The Discrete Cosine Transformation is defined as follows:

$$C(u,v) = \sum_{i=0}^{7}\sum_{j=0}^{7} I(i,j) b_{i,j}(u,v) \qquad u,v = 0,1,\ldots,7 \tag{7}$$

where $I(i,j)$ is a pixel value at the location (i,j) in pixel blocks, $C(u,v)$ is a DCT coefficient matrix, which characterizes the power distribution of signals with different frequencies (u,v). The basis matrix is defined by

$$b_{i,j}(u,v) = \alpha(u)\alpha(v)\cos\left(\frac{\pi(2i+1)u}{16}\right)\cos\left(\frac{\pi(2j+1)v}{16}\right), \tag{8}$$

where $\alpha(s) = \frac{1}{2\sqrt{2}}$ if $s = 0$, $\alpha(s) = \frac{1}{2}$, otherwise .

If we concatenate every column of the matrix $C(u,v)$ together into a 64-dimensional column vector c, and form another column vector p using all the pixels of $I(i,j)$ in the same way, DCT can be rewritten into a compact matrix multiplication, as

$$c = Kp, \tag{9}$$

where K is a 64 by 64 matrix and its m^{th} column is just the vector form of the matrix $b_{i,j}(u,v)$, $m = i + 8j$. Because DCT is an orthogonal transformation, $K^{-1} = K^T$. Thus, inverse DCT (IDCT) can be defined by Equation (10),

$$p = K^T c \tag{10}$$

where K^T denotes the transpose of the matrix K.

The IDCT is the most expensive part of video decoding. An algorithm operating in the compression domain generally means IDCT computation is not involved in the algorithm.

3. Background Construction in DCT Domain

In our framework, we use a set of DCT coefficients as the data structure to represent a background, i.e., $D = \{d^k, k = 1,2,...L\}$, where d^k is a 64-dimensional vector characterizing an 8 by 8 region corresponding to the k^{th} pixel block, L is the number of blocks in a frame of the analyzed video sequence. Through reviewing those state-of-the-art background subtraction techniques, we found many popular algorithms exploit a sequence of linear evaluations to construct their background models. Here, we will mathematically prove that, if a background construction algorithm only involves a sequence of linear evaluations, the proposed background representation will have a counterpart in the DCT domain, which has much lower time complexity but does not lose any accuracy for a generated background.

We use the matrix $F_i = [f_i^1 f_i^2 ... f_i^k ... f_i^L]$ to denote the frame i in a video sequence, and use a 64-dimensional column vector f_i^k to denote the k^{th} block. Thus, the DCT transformation of the matrix F_i can be computed by

$$D_i = KF_i, \tag{11}$$

where $D_i = \left[d_i^1 ... d_i^L \right]$ and the DCT coefficients of each block $d_i^k = Kf_i^k$. So the IDCT can be computed through:

$$F_i = K^T D_i. \tag{12}$$

Suppose a spatial domain background B_t is modeled using linear combination of recent frames,

$$B_t = \sum_{i=0}^{N} \omega_i F_{t-i}, \tag{13}$$

where $\omega_i (i = 1,2,...)$ are weights specified by background modeling algorithms. Let the matrix B_t and F_i have the same structure as those defined in Equation (11), i.e., each column vector corresponds to a block in a frame, and let both sides of Equation (13) be multiplied by the DCT kernel matrix K, we obtain $KB_t = \sum_{i=0}^{N} \omega_i (KF_{t-i})$

Let $D_t^B = KB_t$, $D_i = KF_i$, we have

$$D_t^B = \sum_{i=0}^{N} \omega_i D_{t-i}. \tag{14}$$

Apparently D_t^B is a matrix made up of DCT coefficients. Thus, Equation (14) gives an equivalent background computation model in the DCT domain, through which background can be constructed with the same accuracy as its counterpart does in the spatial domain, but fully decomposing a video sequence is not required.

3.1 Running Average Algorithm in DCT domain

In the running average algorithm only linear evaluations are involved, so we can obtain its equivalent version in DCT domain. If we initialize the background by $B_1 = F_1$ and let Equation (1) be iteratively extended, we will have

$$B_{t+1} = \alpha F_t + (1-\alpha) B_t = \left(\sum_{i=0}^{t-1} \alpha (1-\alpha)^i F_{t-i}\right) + (1-\alpha)^t F_1. \tag{15}$$

Following the same procedure from Equation (13) to (14), we derive the implementation of the running average algorithm in the compression domain as:

$$D_{t+1}^B = \alpha D_t + (1-\alpha) D_t^B \tag{16}$$

where $D_t^B = KB_t$ and $D_t = KF_t$. Each entry in the matrix D_t can be directly obtained with very small decoding cost for an MPEG video decoder. Apparently an algorithm operating on F_t will generally be far less efficient than the counterpart operating on D_t, because the former needs to frequently use Equation (12) to obtain F_t, while the latter does not. Moreover, the latter can obtain an estimation of background as accurate as that generated by the former algorithm by applying IDCT to D_{t+1}^B using Equation (12), when required.

3.2 Median Algorithm in DCT domain

The median algorithm in the pixel domain has been accepted as a simple and effective method through experimental evaluations. In this subsection, we first analyze mathematically and explain the rationality of the algorithm; then a theoretical principle will be derived for the proposed median algorithm in DCT domain through further analysis;

finally we present the details of our median algorithm and discuss some advantages compared with the median algorithm in pixel domain.

In a history window at the location (x, y), all the pixels can be partitioned into two sets and we let $FO(x, y)$ and $BO(x, y)$ denote the set of pixel values corresponding to a foreground object and the set for background respectively. It is easy to mathematically prove that, if

$$\max\{BO(x,y)\} < \min\{FO(x,y)\} \tag{17}$$

or $$\max\{FO(x,y)\} < \min\{BO(x,y)\} \tag{18}$$

holds, the median of the set $FO(x, y) \cup BO(x, y)$ will belong to the set that contains more elements. Generally Equation (17) or Equation (18) holds, and $BO(x, y)$ contains more elements in a history window, so the median will come from $BO(x, y)$ and corresponds to the background at the location (x, y). That is why the median algorithm in spatial domain can work well.

Now we consider a pixel block M_i^t that lies at the location i in the frame at time t in a history window. The average of all the pixel values in M_i^t is

$$a^t(i) = \frac{1}{64}\Big(\sum_{(x,y)\in FO^t(i)} p^t(x,y) + \sum_{(x,y)\in BO^t(i)} p^t(x,y)\Big),$$

where $FO^t(i)$ and $BO^t(i)$ respectively denote the set of foreground pixels and background pixels in M_i^t, $p^t(x, y)$ is the value of a pixel that lies at the location (x, y) in the frame t. A binary value function is defined: if $FO^t(i)$ is a empty set, $\Upsilon(\Phi_i^t) = 1$, else $\Upsilon(\Phi_i^t) = 0$, to check if the block Φ_i^t is completely covered by background. The function $\Upsilon(\Phi_i^t)$ partitioned all the blocks Φ_i^t, $t = s, s-1, ..., s-L$, in the history window into two sets. Correspondingly we use $A_B(i)$ and $A_F(i)$ to denote the set of averages of those blocks that belong the two sets respectively, i.e.:

$$A_B(i) = \{v \mid v = a^t(i),\ \Upsilon(\Phi_i^t) = 1,\ t = s, s-1, ..., s-L\}$$

$$A_F(i) = \{v \mid v = a^t(i),\ \Upsilon(\Phi_i^t) = 0,\ t = s, s-1, ..., s-L\}$$

If a pixel at (x, y) belongs to background, $p^t(x, y)$ generally changes in a very small range in a short history window, i.e. $\min\{BO(x,y)\} \cong \max\{BO(x,y)\}$. Here we assume that $p^t(x, y)$ takes the same value in the window when the pixel at time t is covered by background, and that for each location in a block, either Equation (17) always holds or Equation (18) always holds if both $FO(x, y)$ and $BO(x, y)$ are not empty. Then we can easily derive that for any member $b_i \in A_B(i)$, $f_i \in A_F(i)$, either $b_i < f_i$ always holds in the history window or $f_i < b_i$ always holds. Moreover, Since $A_B(i)$ generally contains more elements in a history window, the median of $A_B(i) \cup A_F(i)$ will belong to $A_B(i)$. Based on the above analysis, we know that for the pixel blocks Φ_i^t ($t = s, s-1, ..., s-L$) in a history window, if the average of a block $\mathrm{M}_i^{\tilde{t}}$ is just the median of the averages of all the pixel blocks in that history window and our assumptions hold, then all the pixels in $\mathrm{M}_i^{\tilde{t}}$ are covered by background. Thus we can use the median of $\{ a^t(i),\ t = s, s-1, ..., s-L \}$, to identify the block that can represent the background at the corresponding location.

If we let $u = 0, v = 0$ in Equation (7), we obtain

$$C(0,0) = \frac{1}{8}\sum_{i=0}^{7}\sum_{j=0}^{7} I(i,j). \tag{19}$$

Thus, the $C(0,0)$ reflects the average of the pixels in a block, and it is called DC coefficient in the literatures. We use the following notation to describe our algorithm:

- The subscript c is used in the following definitions to distinguish different components in color space *YCbCr*.
- $mdl_c^t(i)$: a 64-dimensional DCT coefficient vector for the pixel block i in the background image at time *t*.
- $BkG_c^t = \{mdl_c^t(i), i = 1,2,\ldots,\}$ is a representation of the background image using DCT coefficients.
- $dct_c^t(i)$: a 64-dimensional DCT coefficient vector of the pixel block i in the frame at time t.
- $W_c^t(i) = \{dc_c^t(i), dc_c^{t-\Delta t}(i), \ldots, dc_c^{t-n\Delta t}(i)\}$ denotes a recent history window, where $dc_c^t(i)$ is a DC coefficient for the pixel block i at time t.
- $mid(v)$: the function to evaluate the median of the set v

Our background model update procedure is shown as follows:

For each input frame at time t

$\{dct_c^t(i), c = Y, Cb, Cr, i = 1,2,...\}$

{

For each component $c \in$ {Y, Cb, Cr}

For each block i with the component c

{

$id(t) = \arg\underset{T}{mid}(\{dc_c^T(i), T = t, t-\Delta t, ...\})$

$mdl_c^t(i) = dct_c^{id(t)}(i)$

}

If needed, output $\{ BkG_c^t \mid c = Y, Cb, Cr \}$

}

Since median evaluation is a kind of nonlinear evaluation, the proposed algorithm is not equivalent to the median algorithm in spatial domain in theory. In comparison with the latter, the proposed algorithm in DCT domain has two advantages. First, the algorithm has much lower computational cost, since IDCT is not required. Furthermore, median evaluation is performed only one time for each pixel

block while the counterpart in pixel domain performs 64 times median evaluation. Second, the proposed block-based algorithm will not neglect the correlation of pixels in the same block. Our algorithm can guarantee each block in a background image completely comes from the same frame, while the median algorithm in pixel domain operates each pixel independently and cannot provide such guarantee.

3.3 MoG algorithm in DCT domain

The MoG algorithm in spatial domain models each pixel as a mixture of Gaussians. In this subsection, we propose an algorithm that models DCT coefficients of each block in DCT domain as a mixture of Gaussians.

We first present a block-based Gaussian background model in DCT domain. We assume DCT coefficients in a block satisfy a multivariable Gaussian distribution, i.e.,

$$p(d_k \mid \Lambda_k, \Sigma_k) = \frac{1}{M}\exp(-\frac{1}{2}(d_k - \Lambda_k)^T \Sigma_k^{-1}(d_k - \Lambda_k)),$$
$$M = (2\pi)^{32}\left|\Sigma_k\right|^{1/2}$$

where d_k is a 64-dimensional DCT coefficients vector for the block k, (Λ_k, Σ_k) are the mean vector and covariance matrix. Here the different components of d_k are independent of each other, since they are evaluated through projecting a pixel vector onto a set of orthogonal basis using Equation (9). MPEG Video compression standards have exploited this orthogonal property of the DCT to remove spatial correlation of pixels in a block. Therefore we can use a variance vector $\Delta_k = \left(\sigma_k^i \mid i = 1,...64\right)$ to represent the covariance matrix $\Sigma_k = \Delta_k I$. We can represent $p(d_k \mid \Lambda_k, \Sigma_k)$ as a product of 64 univariable Gaussians, i.e., $p(d_k \mid \Lambda_k, \Sigma_k) = \prod_{i=1}^{64} p(d_k^i \mid \Lambda_k^i, \sigma_k^{i2})$. Thus we can equivalently assume each DCT coefficient is a Gaussian random variable and estimate its parameters independently.

Equation (10) shows that each pixel in a block can be evaluated through a linear combination of 64 DCT coefficients. Probability theory in mathematics tells us that a linear combination of a sequence of independent Gaussian random variables is also a Gaussian random variable, and its parameters can be derived from the parameters of those independent Gaussians. In other words, Gaussians modeling pixels in spatial domain can be directly derived from 64 Gaussians for DCT coefficients. Therefore modeling backgrounds using Gaussians in DCT domain is consistent with doing it in pixel domain. So we can reasonably implement Gaussian based background construction algorithms in DCT domain. For example, we can use Equation (20) to estimate the parameters of a single Gaussian background model.

$$\begin{aligned} \Lambda_{k,t}^i &= (1-\rho)\Lambda_{k,t-1}^i + \rho d_{k,t}^i \\ \sigma_{k,t}^{i2} &= (1-\rho)\sigma_{k,t-1}^{i2} + \rho(d_{k,t}^i - \Lambda_{k,t}^i)^T(d_{k,t}^i - \Lambda_{k,t}^i) \end{aligned} \quad (20)$$

The MoG algorithm in DCT domain can be further implemented. We model each pixel block as a mixture of 64-demensional Gaussians, and assume each dimension of a Gaussian has the same standard variance. The MoG algorithm processes each pixel block just as the algorithm in [4] does on each pixel. The Euclidean distance is chosen as the matching function. A threshold is associated with it to determine if the current block matches a Gaussian, and the threshold will be updated in the similar way that its counterpart deals with variances.

A good property of the proposed algorithms is that a pixel block is represented with a very compact form. Generally DCT can lead to many zero DCT coefficients, which reflects signals with some specific frequencies are not contained by the pixel block. So a Gaussian in DCT domain can be discarded, if its parameter $|\Lambda_k|$ is very small. That means we can use fewer parameters to represent a background than the Gaussian model in pixel domain. Since the estimation of variances is not linear evaluation, the proposed algorithms are not equivalent to their counterparts in pixel domain in theory. The proposed MoG algorithm models a pixel block as a mixture of Gaussian, instead of 64 mixtures, so much less model parameters are estimated in the model adaptation, which results in high efficiency in computation.

3.4. Identifying and Segmenting Foreground Objects

To identify moving objects, the background subtraction technique subtracts an observed image from the estimated background image, and those pixels in the difference image that have a larger value than a predefined threshold will be included in a foreground object. In our framework, we first identify those blocks from an observed image that have a large difference from those at the same location in the background image. Let f^k, $k = 1, 2,..., L$ denote a column vector made up of pixel values in a block of the observed image, and $d^k = Kf^k$ is the corresponding DCT coefficients vector. For the background image, f_B^k and d_B^k are used to denote a corresponding pixel vector and its DCT coefficients vector respectively. We use the Euclidean distance between d^k and d_B^k in Equation (21) to measure the extent to which the background in the block k is covered by a foreground object.

$$\Omega^k = \left\| d^k - d_B^k \right\|_2 \quad (21)$$

The Sum of Squared Differences (SSD) is commonly exploited by video encoders to measure the extent to which a block matches another block, so a large value for $\left\| f^k - f_B^k \right\|_2$ represents the block k is being covered by a moving object. It is noted that

$$\begin{aligned}\left\|f^k - f_B^k\right\|_2 &= (f^k - f_B^k)^T (f^k - f_B^k) \\ &= (K^T d^k - K^T d_B^k)^T (K^T d^k - K^T d_B^k) \\ &= (d^k - d_B^k)^T KK^T (d^k - d_B^k) \\ &= (d^k - d_B^k)^T (d^k - d_B^k) \\ &= \left\|d^k - d_B^k\right\|_2\end{aligned},$$

thus Ω^k is a reasonable measure. If $\Omega^k > \tau$, where τ is a threshold, the block k will be labeled as a foreground block. Apparently based on the measure we can select those that are not labeled as foreground blocks to estimate a background, which can make the estimation of the background model more accurate. For example, the running average algorithm in the compression domain can be improved as

$$d(k)_{t+1}^B = \begin{cases} d(k)_t^B & \text{if } \Omega^k > \tau \\ \alpha d(k)_t + (1-\alpha) d(k)_t^B & \text{otherwise} \end{cases},$$

where $d(k)_t^B, d(k)_t$ denote the k^{th} column vector of the matrix D_t^B and D_t respectively at time t.

In some applications, the accurate shape information of a moving object is preferred. In our framework, given a background $D_t^B = \{d(k)_t^B, k = 1,2,...\}$ and an observed image $D_t = \{d(k)_t, k = 1,2...\}$, a function $S: D_t \to B_t$ is defined to accurately segment a foreground object, where $B_t = \{b(k)_t, k = 1,2...\}$ denotes a binary image. A background pixel is represented by the value 0 while a foreground pixel by 1. If the block is not a foreground block, $b(k)_t =$ **0**, **0** is a zero vector; otherwise, let $\overline{b(k)_t} = K^T(d_t(k) - d(k)_t^B)$, and $\overline{b(i,k)_t}$ is the i^{th} component of $\overline{b(k)_t}$, if the absolute value $\left|\overline{b(i,k)_t}\right| > \eta$, η is a threshold, $b(i,k)_t = 1$; else $b(i,k)_t = 0$. If we desire the texture and color of moving objects to be kept in the resultant image B_t and 0 represents a background pixel, just let $b(i,k)_t = K^T(d(k)_t - d(k)_t^B) + K^T d(k)_t^B$, when $\left|\overline{b(i,k)_t}\right| > \eta$.

4. Further Discussion of DCT-based Background Construction

In this section we will discuss some implementation issues of the proposed approaches for different applications.

4.1 Usage of *P* Frames and *B* Frames

P frames and *B* frames (see in Section 2.2) can also be exploited for updating a background model or extracting moving objects. A block in *P* frames or *B* frames is called as an intra-coded block, if it can be encoded independently. DCT coefficients of an intra-coded block can be easily obtained. An inter-coded block requires information from other blocks in encoding. It is usually encoded through motion compensation, i.e., a motion vector marking a reference block in reference images plus compensation errors, i.e.,

$$d(k) = d_r + d(k)_e, \tag{22}$$

where $d(k)$, d_r, $d(k)_e$ denote DCT coefficients vectors for the current inter-coded block, its reference block, and prediction errors. If a block in *P* frames is a part of background and is inter-coded, its reference block should intuitively be at the same location in a reference frame. Due to high-frequency background objects such as tree branches, and noises, if the motion vector $mv(k)$ is smaller than a very small threshold μ, i.e., $|mv(k)| < \mu$, we can approximate it as a zero vector (0,0), which means d_r can be estimated by the block at the same location in the reference frame, thus $d_r = d(k)_r$. $d(k)_e$ can be directly obtained. If $d(k)_r$ is available, $d(k)$ can be evaluated through $d(k) = d(k)_r + d(k)_e$ and then $d(k)$ can be used for background construction or foreground extraction. If $d(k)_r$ is not available, i.e., $|mv(k)| > \mu$, the block k will be ignored. For *B* frames, the similar process can be used to find those blocks whose DCT coefficients can be directly or indirectly obtained precisely, so that they can be used by the framework in the same way as blocks in *I* frames.

4.2 Filters and Preprocessing

In a generic background subtraction algorithm operating in the spatial domain, filters, especially linear filters are usually used in the preprocessing step. For example, simple temporal and/or spatial smoothing filters can be used as a preprocessing step to reduce noises. Here we will present a scheme to show how linear filters can be implemented and applied in DCT domain.

Let a 64-dimensional vector $f(k)$ denote a block k in the spatial domain, and the corresponding vector in the DCT domain is $d(k)$, so $f(k) = K^T d(k)$. A linear filter operating on a block in the spatial domain can be characterized by a 64 by 64 matrix F. Because the kernel matrix for DCT is an orthogonal matrix, we can derive

$$\begin{aligned}Ff(k) &= FK^T d(k) \\ &= (K^T K) FK^T d(k) \\ &= K^T [(KFK^T) d(k)]\end{aligned}. \tag{23}$$

Equation (23) tells us that if a linear filter F is applied onto a block in the spatial domain, a corresponding filter KFK^T can be found in the DCT domain, and the results obtained after applying two filters to the DCT pair of a single block also form a DCT pair. Thus, for a spatial linear filter F, we can construct a filter KFK^T to implement the

same function in the compressed domain for a single block. Compared with the spatial filters operating in pixel domain, the proposed scheme in the DCT domain operates in a special way. It filters a frame through filtering each block in the frame, so block effects may exist due to the same reason as margin effects may exist for a filter operating in pixel domain. The limitation is inherent due to the techniques adopted by the popular video compression standards

For a temporal linear filter T, T can be represented by a vector $[w_0, w_1, ..., w_n]$. Let $f_i(k)$, $i = t, t+1, ..., t+n$ denote pixel values vector in the block k at time i, and $d_i(k) = Kf_i(k)$, $i = t, t+1, ..., t+n$. Applying the filter T to the block k along the temporal axis, we have

$$\sum_{i=0}^{n} w_i f_i(k) = \sum_{i=0}^{n} w_i [K^T d_i(k)] = K^T \sum_{i=0}^{n} w_i d_i(k) \quad (24)$$

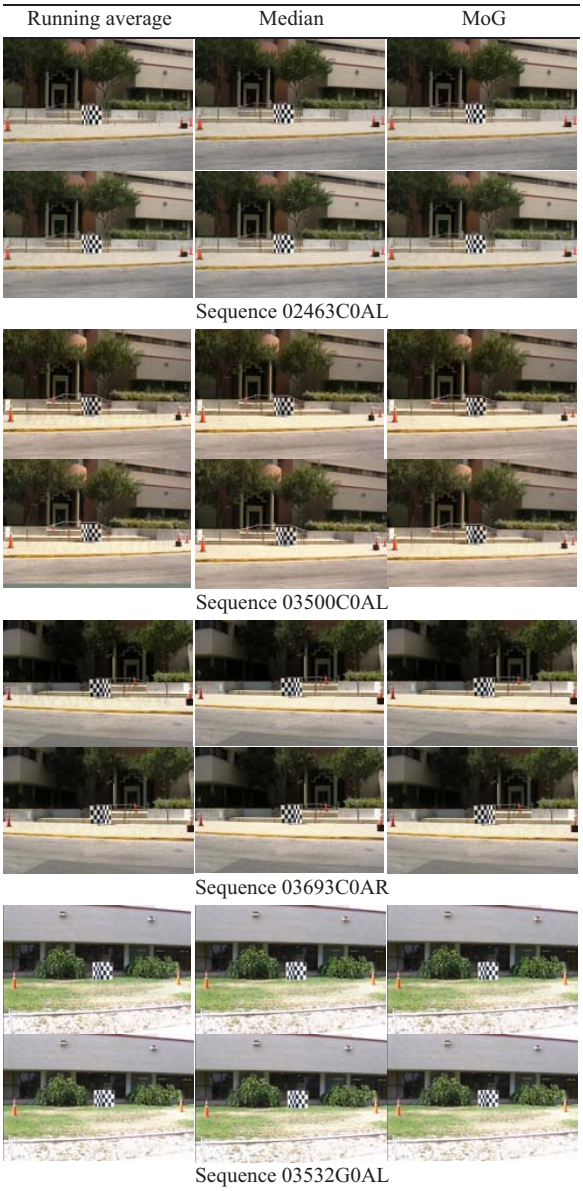
Equation (24) shows that in our framework we can directly apply a temporal linear filter T to a block's DCT coefficients to implement the same function. Different from the proposed spatial linear filter, the proposed temporal linear filter is completely equivalent to its counterpart in pixel domain.

5. Experiments

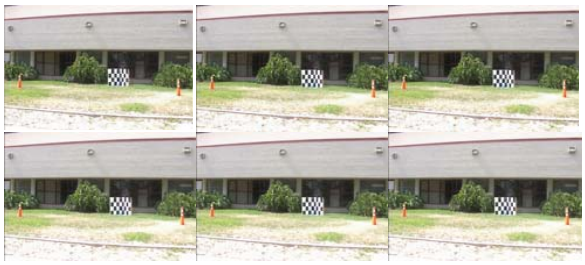
We have evaluated the proposed algorithms using the USF/NIST image sequences, which is publicly available for background subtraction and gait analysis [14]. We chose 6 outdoor sequences, and some representative frames are shown in Figure 1, captured at two locations with walking humans from the USF/NIST database. One location has concrete floor and the other has meadow. Three sequences for the same location are captured under different lighting conditions. There are many background variations in these sequences, such as sudden lighting changes, small motion of tree branches, small motion of bushes and shadow distortions by the walking people.

Figure 1 The Frames extracted from the testing image sequences

To simulate the video compression effects, we compressed each sequence into the MPEG-2 format. In our experiments, the background models are constructed only using *I* frames, which exist every other 9 frames in our compressed video streams. Figure 2 shows the background images generated by the proposed running average, median, and the MoG algorithms in the DCT domain, in comparison with those generated by the corresponding algorithms in the spatial domain. The learning rate α in the running average algorithms and β in the MoG algorithms are 0.05. The length of the history window is 9 in the median algorithm. The variance in the MoG algorithm is initialized as 10. All these configurations keep the same for the algorithms in both spatial and DCT domains.



Sequence 03653G0AL



Sequence 03678G0BR

Figure 2 Background generated by our algorithms and their counterparts in spatial domain

Figure 2 gives all the background images generated for six video sequences. For each sequence, the images in the first row are generated by our algorithms, and the second row by their counterparts in the spatial domain. Our eyes can not perceive evident difference in visual quality between them. The computation speed of the proposed methods is averagely 1.02 times faster than their counterparts in the spatial domain plus decoding cost. The detailed speed ratio of each algorithm is shown in the following Table.

Table. 1 Computation speed ratios between the proposed algorithms and it counterparts in pixel domain

	Average	Median	Gaussian	MoG
Speed	1.37	2.50	1.21	3.03

These experimental results are consistent with our theoretical analysis.

6. Conclusions

We have proposed some algorithms to construct background models directly from compressed video data. In the proposed methods, a background model is represented through a set of DCT coefficients representing the power of different frequencies, and computed based on each 8 by 8 pixel block, instead of per pixel. We have mathematically proved that if a background construction algorithm in the spatial domain only involves a sequence of linear evaluations, there must be a counterpart in the DCT domain, which has much lower computational complexity but the same accuracy. To demonstrate the validity of the framework, we have proposed three representative algorithms with different styles within the framework, i.e., running average, median, Gaussian, and further presented some general possible technical improvements to make them more capable for a wide range of applications. For each proposed algorithm we all give some theoretical derivation and analysis to explore their properties and the relationship with the counterparts in the spatial domain. The experimental results on standard evaluation video sequences are consistent with our theoretical discussion. Since our approach has the attractive visual accuracy for generated background images, much lower computational cost, compact model storage, as well as reasonable theoretical explanation, it has many potential applications in processing compressed video.

References

[1] B.P.L. Lo and S.A. Velastin, Automatic congestion detection system for underground platforms, Proc. of 2001 Int. Symp. on Intell. Multimedia, Video and Speech Processing, pp. 158-161, 2000.

[2] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting moving objects, ghosts and shadows in video streams. IEEE Trans. on PAMI, 25(10):1337-1342, 2003.

[3] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder:Real-time Tracking of the Human Body," IEEE Trans. on PAMI, 19(7):780-785, 1997.

[4] C. Stauffer, W.E.L. Grimson, Adaptive background mixture models for real-time tracking, Proceedings of CVPR, Vol. 2, pp. 246-252, 1999.

[5] M. Elgammal, D. Harwood, L. S. Davis, Non-parametric Model for Background Subtraction, Proceedings of the 6th ECCV, pp. 751 – 767, 2000.

[6] B. Han, D. Comaniciu, and L. Davis, Sequential kernel density approximation through mode propagation: applications to background modeling, Proc. ACCV, 2004.

[7] N. M. Oliver, B. Rosario, and A. P. Pentland, "A Bayesian Computer Vision System for Modeling Human Interactions," IEEE Trans. on PAMI, 22(8): 831-843, 2000.

[8] F. De la Torre, M. J. Black, A framework for robust subspace learning, International Journal of Computer Vision, 54(1-3): 117-142, 2003.

[9] M. Seki, T. Wada, H. Fujiwara, K. Sumi, "Background detection based on the cooccurrence of image variations", Proc. of CVPR 2003, vol. 2, pp. 65-72

[10] R.S. Aygun, A. Zhang, Stationary background generation in mpeg compressed video sequences, IEEE ICME, pp. 701- 704, 2001.

[11] X. Yu, L. Duan, Q. Tian, Robust moving video object segmentation in the MPEG compressed domain, Proc. ICIP vol.2, pp. 933-936, 2003.

[12] W. Zeng, W. Gao, D. Zhao, Automatic moving object extraction in mpeg video, Proc. of ISCAS'03, vol.2, pp. 524-527, 2003.

[13] B. Ugur Töreyin, A. Enis Çetin, Anil Aksay, M. Bilgay Akhan. Moving Region Detection in Compressed Video. ISCIS, pp.381-390, 2004.

[14] J. Phillips, S. Sarkar, I. Robledo, P. Grother, and K. Bowyer. Baseline results for the challenge problem of human id using gait analysis. Proceedings of Face and Gesture Recognition, 2002