Modeling Background from Compressed Video

Weiqiang Wang, Datong Chen and Jie Yang

School of Computer Science
Carnegie Mellon University
{datong, 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 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 data. The proposed approach utilizes the information exploited 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 the proposed approach 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 (26% 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 for monitoring the activities of targets in a scene, such as monitoring human activity in indoor environments and performing surveillance on 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 to a model of the scene background. 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. The pixel value can be gray or color values. Basic methods are the average method [4], the running average method, the median method [2], and selective average method. Advanced methods are mostly based on statistical modeling techniques, such as single Gaussian estimator (pfinder) [11], mixture of Gaussian estimator [9], kernel density estimator [3], sequential kernel density estimator [5], mean-shift estimator [5], eigen background [6], and robust PCA background [10]. There are also methods using the correlations of the pixels with the neighborhood into account [8]. All these methods model a background in the spatial domain, which requires 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 [1][12][13]. These methods have been 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 [12] extracts objects at a size of 8x8 blocks and cannot 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 data 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 domain and spatial domains. Furthermore, the proposed approach can take advantage of the structure and available information in the compressed video stream 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 data, 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 compressed domain. We theoretically prove that we can achieve the same accuracy in the compressed domain as in the spatial domain with much less computation cost. This means that the models obtained from the compressed domain can be directly used in the spatial domain, which bridges two domains together. In Section 4, we show 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. [4] 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:

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

where the learning rate α is typically set as 0.05.

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

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

where

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

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

We can also use selective algorithm to remove the pollution from the foreground objects. Each pixel in the current frame is first classified as either foreground or background. We then ignore the foreground pixel in the background model. The difficulty of the selective method is how to choose the classification threshold. Wren et al. [11] 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. [9] 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 at every new frame. The algorithm first computes the matching models for each pixel (*x, y*):

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

The weights are then updated as:

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

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:

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

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

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

These three algorithms are used as examples to be implemented in the compressed domain in the following sections.

2.2 Compressed Video and DCT

In order to transmit and store video data, video compression techniques are employed to reduce the size of an image sequence by removing spatial and temporal redundancy. Figure 1 illustrates a video data compression framework adopted by almost all the popular video compression techniques, especially the international standard 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.

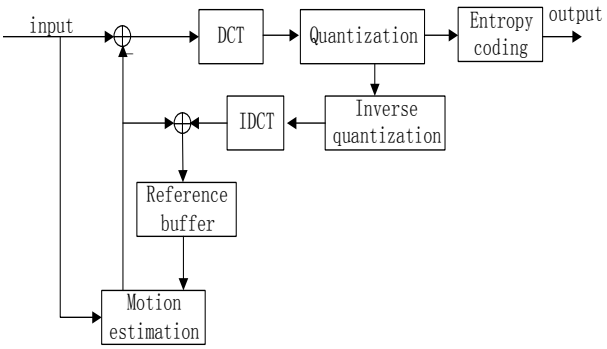


Figure 1 Encoding framework of MPEG standards

Each *I* frame is coded as a set of DCT coefficients that transformed from 8x8 block of pixels in the spatial domain. DCT is defined by the following equation

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

where $I(i,j)$ is a pixel value at the location (i, j), $C(u,v)$ is the corresponding DCT coefficient Matrix, which characterizes the power distribution of signals for different frequencies. The basis matrix and its element with the index (u,v) is defined as.

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

where $$\alpha(s)=\begin{cases}\frac{1}{2\sqrt{2}} & \text{if } s=0\\ \frac{1}{2} & \text{if } s=1,2,...,7\end{cases}. \tag{9}$$

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

$$c=Kp, \tag{10}$$

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 the following equation,

$$p=K^T c, \tag{11}$$

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

The IDCT is the most expensive part of video decoding. Our goal is to construct background models directly using the DCT coefficients in *I* frames of video. To this end, we first build a bridge between the spatial domain and the DCT domain on top of the three background models as discussed in Section 2.1.

3. Background Construction from DCT

In our framework, we use a set of DCT coefficients as the data structure to represent a background model, i.e., $M=\{d^k, k=1,2,...L\}$, 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. While reviewing the state-of-the-art background subtraction techniques, we found almost all the popular algorithms exploit a sequence of linear evaluations to construct their background models. In the following, we will mathematically prove that, if a background construction algorithm only involves a sequence of linear evaluations, the proposed background representation will has a counterpart in the DCT domain, which has much lower time complexity but does not lose any accuracy for a generated background.

Let us denote matrix $F_i=[f_i^1 f_i^2 ... f_i^k ... f_i^L]$ as frame i in a video sequence, and f_i^k as 64-dimensional column vectors of the k^{th} block. The DCT transform of the matrix F_i is therefore computed as:

$$D_i=KF_i, \tag{12}$$

where $D_i=\left[d_i^1 ... d_i^L\right]$ and the DCT coefficients of each block $d_i^k=Kf_i^k$. In the same way, the IDCT is computed as:

$$F_i=K^T D_i. \tag{13}$$

Spatial domain background B_t is modeled by computing pixel values or some parameters, such as Gaussian parameters, using linear combination of recent frames:

$$B_t=\sum_{i=0}^{s}\omega_i F_{t-i}, \tag{14}$$

where $\omega_i (i=1,2,...)$ are the weights specified by the background modeling algorithm. Let the matrix B_t and F_i have the same structure as those defined in Equation (14), i.e., each column vector corresponds to a block in a frame, and let both sides of Equation (14) be multiplied by the DCT kernel matrix K, we derive

$$KB_t=\sum_{i=0}^{s}\omega_i(KF_{t-i})\cdot \tag{15}$$

Let $D_t^B=KB_t$, $D_i=KF_i$, then D_t^B is a matrix made up of a set of DCT coefficients. Equation (15) describes how to construct a background model without fully decomposing a video sequence, referencing to an algorithm oriented to pixels that only involves linear evaluations.

3.1 Running Average Algorithm Using DCT

We first take the running average algorithm as an example to illustrate how to derive its version in DCT domain. Let us

initialize the background $B_1 = F_1$. The right hand side of the Equation (1) can then be extended as:

$$B_{t+1} = \alpha F_t + (1-\alpha)B_t = (\sum_{i=0}^{t-1} \alpha(1-\alpha)^i F_{t-i}) + (1-\alpha)^t F_1 . \quad (16)$$

Using Equation (15), we can directly derive the implementation of the running average algorithm in compression domain as:

$$D^B_{t+1} = \alpha D_t + (1-\alpha) D^B_t , \quad (17)$$

where $D^B_t = KB_t$ 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. It is easy to understand that 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 (13) to obtain F_t, while the latter does not. Apparently the latter can obtain a background estimation as accurate as which generated by the former algorithm through applying IDCT to D^B_{t+1} using Equation (13), if necessary.

3.2 Median Algorithm Using DCT

Median algorithms in pixel domain assume a median evaluated from a pixel's history representing the pixel value of a background at the same location. Let $FO(x, y)$ represent a set of pixel values from a history window of the pixel at the location (x, y), $FO(x, y)$ = { $I(x, y) \mid I(x, y)$ denote pixel values when the location (x, y) is covered by a foreground object}. Similarly we can define $BO(x, y)$ to denote the set for background. It is easy to mathematically prove that, if $FO(x, y)$ and $BO(x, y)$ satisfy

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

or

$$\max\{FO(x, y)\} < \min\{BO(x, y)\}, \quad (19)$$

the median of the set $FO(x, y) \cup BO(x, y)$ will belong to the set that contains more elements. Generally $BO(x, y)$ contains more elements in a history window, so the median will belong to $BO(x, y)$ and can represent the background at the location (x, y). The condition of the above proposition always holds in general cases, therefore median algorithms work well in spatial domain.

Now we consider the case for a block $BLK(i)$, let $FO(i)$ and $BO(i)$ denote the set of foreground pixels and background pixels, the average $a(i)$ of all the pixels in $BLK(i)$ is

$$a(i) = \frac{1}{64}\left(\sum_{(x,y)\in FO(i)} p(x, y) + \sum_{(x,y)\in BO(i)} p(x, y)\right).$$

We classify block i in a history window into two classes. One is made up of blocks completely covered by background, i.e. $FO(i)$ is an empty set, and use $A_B(i)$ to denote the set of averages of these blocks; the other class is made up of blocks at least partly covered by the foreground, and use $A_F(i)$ to denote the set of averages of the corresponding blocks. If a pixel at (x, y) is a background pixel, $p(x, y)$ will generally change 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 $p(x, y)$ keeps the same value and foreground pixels in a block, all satisfying equation (18) in the window or all satisfying equation (19) in the window. Correspondingly, we can easily derive that for any $a_B(i) \in A_B(i)$, $a_F(i) \in A_F(i)$, $a_B(i) < a_F(i)$ always holds in the history window or $a_B(i) > a_F(i)$ always holds. 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 in a history window for a block location, if the average of a block is just the median of all the averages in the window and our assumptions hold, then all the pixels in the block are covered by background. Thus we can use the median of the averages of blocks in an observation history window to identify the block that can represent the background at the corresponding location.

If we let $u = 0, v = 0$ in Equation (1), we obtain

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

Thus, the first DCT coefficient can reflect the average of 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 definition to distinguish different components *Y*, *Cb*, *Cr*.
- $mdl^t_c(i)$: 64-dimensional DCT coefficient vector of block i in the background image at time *t*.
- $BkG^t_c = \{mdl^t_c(i), i = 1,2,\ldots,\}$ is a representation of the background image using DCT coefficients.
- $dct^t_c(i)$: the 64-dimensional DCT coefficient vector of block i in the frame at time t.
- $W^t_c(i) = \{dc^t_c(i), dc^{t-\Delta t}_c(i), \ldots, dc^{t-n\Delta t}_c(i)\}$ denotes a recent history window, where $dc^t_c(i)$ is a DC coefficient for 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 follow:

For each input frame at time t

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

{
For each component $c \in \{Y, Cb, Cr\}$
For each block i with the component c
{
$id(t) = \arg\min_T mid(\{dc_c^T(i), T = t, t-\Delta t, ...\})$
$mdl_c^t(i) = dct_c^{id(t)}(i)$
}
If needed, output $\{ BkG_c^t \mid c = Y, Cb, Cr \}$
}

Compared with the median filter algorithm in the spatial domain, the median algorithm implemented in DCT domain has two advantages. First, the algorithm has much lower computational cost. Besides that IDCT is not required, median evaluation is involved only one time for each pixel block while the counterpart in pixel domain involve 64 times median evaluation. Second, the block-based algorithm will not pollute the correlation of pixels in the same block, since our algorithm can guarantee each block in a background image completely comes from the same frame. Those median algorithms in pixel domain operate each pixel independently and cannot provide such guarantee.

3.3 MoG Background Model Using DCT

The special MoG algorithm models each pixel as a mixture of Gaussians. Intuitively, we model the DCT coefficients of each block in the DCT domain as a mixture of Gaussians in a 64 dimensional space:

$$p(d_k \mid \Lambda_k, \Sigma_k) = \frac{1}{M} \exp\left(-\frac{1}{2}(d_k - \Lambda_k)^T \Sigma_k^{-1} (d_k - \Lambda_k)\right), \quad M = (2\pi)^{32} |\Sigma|^{1/2} \tag{20}$$

where d_k is a 64-dimensional DCT coefficients vector for the block k, (Λ_k, Σ_k) are the means and variances of the multi-dimensional Gaussian parameters. It should be noted that different components of d_k are independent of each other, since it is evaluated through projecting a pixel vector onto a set of orthogonal basis using Equation (13). MPEG Video compression standards have exploited this good property of the DCT to remove spatial correlation of pixels in a block. Therefore we can use a variance vector $\Delta_k = \left(\sigma_k^i \mid 1 = 1, ...64\right)$ to represent the covariance matrix $\Sigma_k = \Delta_k I$ in Equation (20). The independency among the components makes $p(d_k \mid \Lambda_k, \Sigma_k)$ be represented as a product of 64 uni-variable Gaussians $p(d_k \mid \Lambda_k, \Sigma_k) = \prod_{i=1}^{64} p(d_k^i \mid \Lambda_k^i, \sigma_k^{i2})$. We thus can estimate each component a 1-d mixture of Gaussian independently.

Equation (14) shows us that each pixel 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, whose parameters can be derived from the parameters of those independent Gaussians. That means 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. Many Gaussian based background construction algorithms can also be implemented in the proposed frame. The MoG algorithm in the DCT domain updates the model parameters using the following equation:

$$\Lambda_{k,t}^i = (1-\rho)\Lambda_{k,t-1}^i + \rho d_{k,t}^i$$
$$\sigma_{k,t}^{i2} = (1-\rho)\sigma_{k,t-1}^{i2} + \rho (d_{k,t}^i - \Lambda_{k,t}^i)^T (d_{k,t}^i - \Lambda_{k,t}^i)$$

An advantage of the block based Gaussian background model in the DCT domain is that a pixel block is represented with a very compact form, because each DCT coefficient represents the power of a specific frequency for signals contained by the block. Many small values can be omitted during the video compression, which leads to many zeros in the DCT coefficients. As we can discard a coefficient sequence that contains only zero values, we can save a lot of computations. We can also remove the dimensions that contain very small means by set these means to be zeros. This selection process provides a more robust background modeling in the DCT domain compared to the original MoG algorithm in the spatial domain.

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.

$$difblk^k = \left\| d^k - d_B^k \right\|_2 . \tag{21}$$

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

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

thus $difblk^k$ is a reasonable measure. If $difblk^k > \tau$, 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 background model more accurate. For example, the running average algorithm in the compression domain can be improved as

$$d(k)_{t+1}^B = \begin{cases} d(k)_t^B & \text{if } d(k)_t \text{ is a foreground block} \\ \alpha d(k)_t + (1-\alpha) d(k)_t^B & \text{else} \end{cases},$$

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

In some applications, the shape information of a moving object is preferred and is generally characterized by a binary image. A pixel with 0 represents it belongs to a background; otherwise it is a foreground pixel. In our framework, given a background D^B and an observed image D_t, with their column vector $d(k)^B$ and $d(k)_t$ made up of 64 DCT coefficients in the block k, a function $S: D \to B$ is defined to implement the process of accurately segmenting a foreground object. If the block is not a foreground block, $b(k) = 0$, 0 is a zero vector; otherwise, let $\overline{b(k)} = K^T (d(k)_t - d(k)^B)$, $\overline{b(i,k)}$ is the i^{th} component of $\overline{b(k)}$, if the absolute value $\left|\overline{b(i,k)}\right| > \eta$, η is a threshold, $b(i,k) = 1$; else $b(i,k) = 0$. If we desire the real pixel values of foreground objects to be kept in the resultant image B and 0 represents a background pixel, a small modification of the just proposed algorithm can be made for a foreground block to implement this, i.e., for pixels satisfying $\left|\overline{b(i,k)}\right| > \eta$, $b(i,k) = K^T (d(k)_t - d(k)^B) + K^T d(k)^B$.

4. Further Discussion of DCT-based Background Construction

This section discusses some implementation problems of the proposed approach for different applications.

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

P frames and *B* frames 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. An inter-coded block 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 a background from a stationary camera and is inter-coded, its reference block should intuitively be at the same location in 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 be exploited for background construction or foreground extraction. If $d(k)_r$ is not available or $|mv(k)| > \mu$, 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 noise. Here we will elaborate how linear filters can be applied in the compressed domain of the proposed framework to implement the same function.

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

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

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 a DCT transformation pair also form a DCT transformation 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 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

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

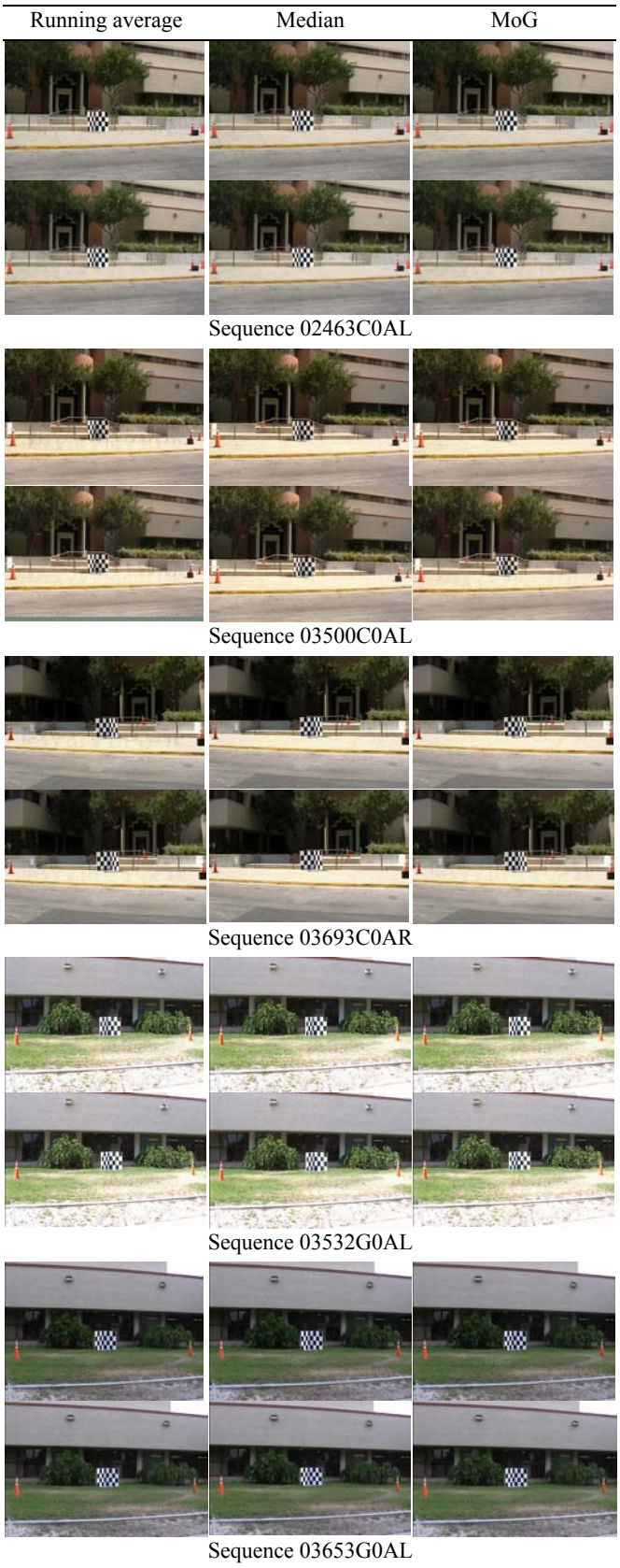
Equation (24) shows that in our framework we can directly apply a temporal linear filter T to a block's DCT coefficients to implement the same function.

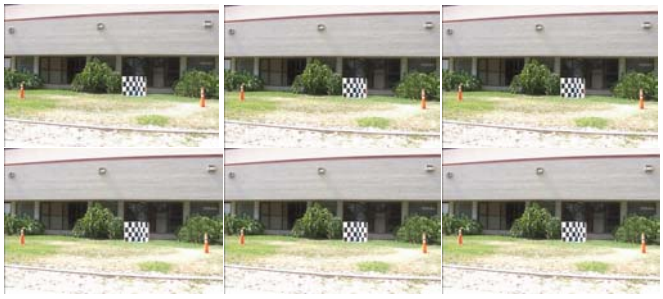
5. Experiments

We have evaluated the proposed algorithms using the USF/NIST image sequences, which is publicly available for background subtraction and gait analysis [7]. We chose 6 out door sequences, shown in Figure 2, captured at two locations with walking humans from the USF/NIST database. One location has concrete floor the other has meadow. Three sequences are taken 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 2 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 backgrounds are constructed only using *I* frames, which exist every other 9 frames in our compressed video streams. Figure 3 shows the background images generated by the running average, the median, and the MoG algorithms implemented using the proposed methods in the DCT domain in comparison with the implementations in the spatial domain. The learning rate α in the running average algorithms and β in the MoG algorithms are 0.05. Our 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.

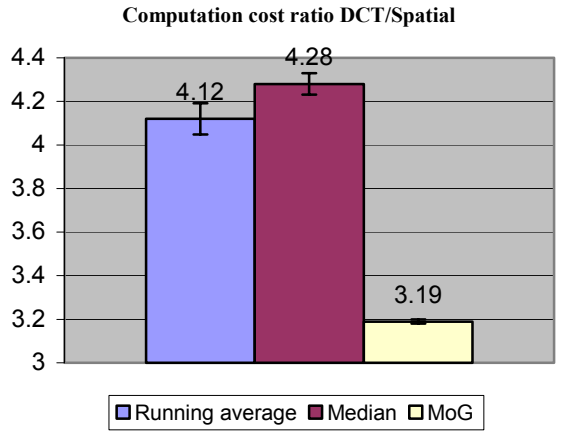




Sequence 03678G0BR

Figure 3 A comparison of the proposed background modeling methods between spatial domain and DCT domain. The first row of the each sequence shows the results of spatial domain algorithms.

The results in Figure 2 show that the background images generated by our algorithms in the DCT domain have as good visual quality as those generated by the counterparts in pixels. The computation cost of the proposed methods is 3.86±0.28 times faster then the corresponding algorithms in the spatial domain plus decoding cost. The details are shown in the following figure.



These experimental results are consistent with our theoretical analysis.

6. Conclusions

We have proposed a method to construct background models directly from compressed video data. In the proposed method, 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 our framework, which has much lower time complexity but the same accuracy. To demonstrate the validity of the framework, we have implemented three representative algorithms with different styles within the framework, i.e., average, median, Gaussian, and further presented some general possible technical improvements to make them more capable for a wide range of applications. For each algorithm implemented, 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 involved in processing compressed video.

References

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

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

[4] 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.

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

[6] 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.

[7] 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

[8] 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

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

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

[11] 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.

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

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