2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance

Spatio-Temporal LBP based Moving Object Segmentation in Compressed Domain

Jianwei Yang[1], Shizheng Wang[2], Zhen Lei[2], Yanyun Zhao[1], Stan Z. Li[2]
[1]School of Information and Communication Engineering,
Beijing University of Posts and Telecommunications, China
[2]CBSR & NLPR, Institute of Automation, Chinese Academy of Sciences, China
{yjwterry, zyy}@bupt.edu.cn; {szwang, zlei, szli}@cbsr.ia.ac.cn

Abstract

With the increasing amount of surveillance data, moving object segmentation in the compressed domain has drawn broad attention from both academy and industry. In this paper, we propose a novel moving object segmentation method towards H.264 compressed surveillance videos. First, the motion vectors (MV) are accumulated and filtered to achieve reliable motion information. Second, considering the spatial and temporal correlations among adjacent blocks, spatio-temporal Local Binary Pattern (LBP) features of MVs are extracted to obtain coarse and initial object regions. Finally, a coarse-to-fine segmentation algorithm of boundary modification is conducted based on the DCT coefficients. The experimental results validate that the proposed method not only can extract fairly accurate objects in compressed video, but also has a relatively low computational complexity.

1. Introduction

Nowadays, large amounts of visual data, especially the surveillance videos, contain many activities. To further analyze the activities in the surveillance videos, accurate and automatical moving object segmentation is needed. The recent object segmentation methods can be conducted in both the pixel domain and the compressed domain. In the pixel domain, many efficient algorithms [11] are proposed to achieve accurate pixel-level moving object regions, which can widely facilitate the high-level applications, such as video retrieval, video abstraction and video fast browsing. However, most of surveillance videos are transmitted from surveillance cameras to the data processors in stream format, resulting in a relatively higher computational complexity of stream decoding for pixel-level algorithms, thus more and more object segmentation algorithms are introduced to the compressed domain to exploit the information from compressed video stream directly, which decreases the time cost efficiently. In the compressed domain, MV and DCT coefficients are the main information used to segment the moving objects. Babu et al. [1] used the MV as the cue to segment moving objects in MPEG. They obtained a dense MV field through the accumulation and interpolation of MVs, followed by expectation maximization (EM) algorithm to extract final objects. Zeng et al. [10] used the sparse MV field to extract the moving object in compressed video. First, MVs are classified into four types. A Markov Random Field (MRF) model is then constructed for the coarse-to-fine segmentation of moving object from background. To accelerate the processing, a MV quantization method was proposed and combined with MRF model in [2][3]. In this method, the MVs are quantized to decide the parameters of MRF model, including the number of motion segmentation and the corresponding statistics. After the coarse segmentation by MRF model, boundary refinement based on the Y-component is implemented to obtain the pixel-level result. In the compressed domain, the DCT coefficients contain the residual information derived from motion estimation and motion compensation. Therefore, many approaches exploited the DCT coefficients to extract the object [5][9]. In addition, Poppe et al.[8] introduced a method only based on the coding size of macroblock (MB) to segment moving objects in H.264. In this paper, we present a novel segmentation method to obtain accurate moving object regions in 4×4 block precision. The proposed method makes use of MVs and DCT coefficients in the segmentation process. First, the spatio-temporal correlations among MVs are measured using LBP feature to conduct a fast and reliable initial segmentation. Second, the distribution and value of DCT coefficients are exploited to refine the object regions. In our method, the spatial and temporal correlation among MVs is extracted for every 4×4 block locally, which is quantified to LBP value directly. Without heavily computational burden of MRF optimization [10] [2] and

978-0-7695-4797-8/12 $26.00 © 2012 IEEE
DOI 10.1109/AVSS.2012.68

the over-segmentation or under-segmentation from cluster algorithms [1] [2], accurate moving object segmentation results can be achieved with a lower computational complexity in this paper.

The rest of the paper is organized as follows: the flow chart of our framework is shown in section 2. In section 3, we introduce the initial segmentation method based on spatio-temporal LBP feature. The details of our moving object region refinement procedure based on the DCT coefficients are presented in section 4. Afterward, the experimental results are provided in section 5. At last, the conclusions and future research issues are discussed in section 6.

2. Overview of The Proposed Method

The flow chart of the proposed object segmentation method is shown in Fig.1. The proposed method consists of four stages: a) MV field preprocessing; b) Initial object segmentation based on LBP features; c) Region refinement based on the distribution of DCT coefficients; d) Object region projection to I-Frames. Considering that only one I-frame is generated in our scheme, the projection procedure will not be introduced in detail in this paper.

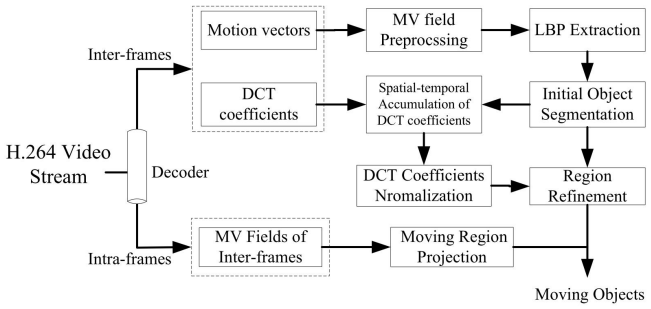

Figure 1. Flow chart of proposed object segmentation method.

3. Initial Object Segmentation

3.1. MV Field Preprocessing

In H.264, each macroblock (MB) can be partitioned into various sizes of blocks, and each block corresponds to a MV and a reference frame generally. However, since the coding-oriented criterion, macroblock (MB) in the inter-frames may be predicted as intra mode (=0) for the homogeny of adjacent blocks. In this case, MV field may be sparse and noisy, which decrease the accuracy of object segmentation. Therefore, the MVs of intra blocks in the inter-frames should be estimated contextually. In our algorithm, all the 4-adjacent regions of intra blocks are located and denoted by R . Then the MBs in region $r(r \in R)$ are assigned with new MVs according to:

$$MV_i = \begin{cases} MV_\mu & \text{if } size(r) < I_s \\ MV_i & \text{otherwise} \end{cases} \quad (1)$$

where MV_μ is the average MV of MBs around the region r. If the size of region r in MB precision is smaller than a predefined threshold I_s, then assign MV_μ to all blocks in region r. After the processing in (1), intra blocks in the inter frames are endowed with new MVs, which can eliminate the local sparsity in MV field efficiently. Afterward, the MV field is sampled in 4×4 block precision to obtain an uniform MV field for object segmentation. To obtain a more reliable and dense MV field, the uniform MV field are normalized, followed by a temporal accumulation and a 3×3 median filtering [6].

3.2. Initial Object Segmentation

Initial object segmentation is based on the spatio-temporal LBP feature. In the pixel domain, LBP [7] is a powerful texture descriptor extracted by comparing neighboring pixel value with the central pixel value. The LBP at the location (x_p, y_p) can be derived from the following formulation:

$$LBP(x_c, y_c) = \sum_{n=0}^{N-1} s(v_n - v_c)2^n \quad (2)$$

where v_c is the value of central pixel, v_n represents the value of N neighborhood pixels. $s(\cdot)$ is a sign function defined as follows:

$$s(x) = \begin{cases} 1 & \text{if } x \geq \tau \\ 0 & \text{if } x < \tau \end{cases} \quad (3)$$

where τ is a threshold which is often set to 0. Motivated by the method proposed in [4], which uses LBP histograms to detect and extract the moving objects, we introduce a spatio-temporal LBP in the MV field to obtain a coarse object region. Concretely, we combine MV field and block modes to extract the spatio-temporal LBP feature for each 4×4 block, and then locate the object region by thresholding the value of LBP feature. In our algorithm, LBP feature is extracted according to the similarity among MVs for the reason that the MV indicates the moving trend of blocks in objects, and the MVs are similar in one object region, whereas are different at the boundary of object regions. The more similar the MVs are to each other, the more likely the blocks should be merged to one moving object. To measure the similarity between two MVs, we introduce a similarity metric function as following:

$$S(MV_i, MV_j) = \frac{MV_i \cdot MV_j}{max\{||MV_i||, ||MV_j||\}^2} \quad (4)$$

Obviously, in (4), the similarity value between two vectors in (4) ranges from -1 to 1, and -1 represents the completely difference between two vectors, whereas 1 indicates

the vectors are completely the same to each other. Specifically, given the central block and neighboring blocks in the LBP extraction, the similarity is manually set to be 1 if the central motion vector MV_c is a zero vector, and set to be 0 if MV_c is a non-zero vector yet its neighboring motion vector MV_n is a zero vector. It is formulated as follows:

$$S(MV_c, MV_n) = \begin{cases} 1 & \text{if } MV_c = 0 \\ 0 & \text{if } MV_c \neq 0 \; and \; MV_n = 0 \end{cases} \quad (5)$$

Given the similarity metric function, the LBP bit value can be derived by thresholding the similarity with a threshold τ defined in function (3). In our algorithm, the neighboring blocks of each 4×4 block are selected according to the block modes. Concretely, given a 4×4 block with corresponding mode, the blocks neighboring the raw block where it locates are selected, and the block scale is amplified according the block modes until all the eight neighboring blocks are located. In our algorithm, the neighboring blocks are traversed in clockwise order with the top-left block as the start. Because of different sizes of blocks, the traversing order is arranged according to the direction pointing from the center of central block to the centers of neighboring blocks. As shown in Fig.2, the LBP of the purple 4×4 blocks in the raw 8×4 block is extracted by calculating the similarity of MVs to their surrounding green blocks. Having set up the extraction criterion for 4×4 block, the

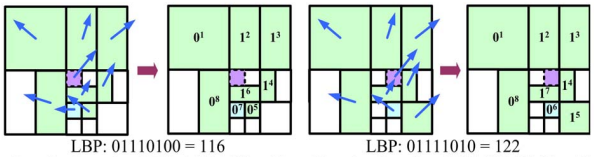

Figure 2. Comparison of LBP extraction for two neighboring 4×4 blocks: (a): extraction for left-side purple 4×4 block; (b) extraction for right-side purple 4×4 block. In these examples, τ is set to be 0.

spatio-temporal LBP feature can be obtained via the combination of the LBP in current frame and that referring to previous frames. The expression terms in our algorithm are defined as follows:

Assume $b_t(x, y)$ represents the block at the location (x, y) in the frame t; its MV is denoted by $MV_t(x, y)$, and its LBP feature in current frame t referring to the frame t_p is denoted by $LBP_t^{t_p}(x, y)$; here, the frame distance from current frame to the previous frame is no larger than a positive const T, that is,$(t - t_p) \in [0, T]$. To achieve the spatio-temporal LBP of block $b_t(x, y)$, the derivation of LBP of $b_t(x, y)$ referring to the frame t_p, $(t - t_p) \in [0, T]$, can be partitioned to three steps: a) project the MV of $b_t(x, y)$ to its corresponding location (x, y) in the frame t_p, denoted by $MV'_{t_p}(x, y)$; b) modify the motion vector $MV_{t_p}(x, y)$ using the function:

$$\hat{MV}_{t_p}(x, y) = \omega MV'_{t_p}(x, y) + (1 - \omega) MV_{t_p}(x, y) \quad (6)$$

where ω is a weight ranging from 0 to 1; afterward, c) calculate $LBP_t^{t_p}(x, y)$ referring to MV field of frame t_p with the central MV is $\hat{MV}_{t_p}(x, y)$.

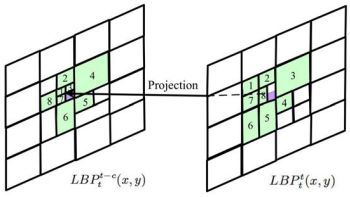

Figure 3. Spatio-temporal LBP feature extraction based on block modes. The number in the blue blocks represents the traversing order.

As shown in Fig.3, the purple 4×4 block has different neighboring blocks between the frame t and t_p because the block modes distribution is different. When the three steps are accomplished, the LBP feature set of $b_t(x, y)$ referring to the current frame and all the previous T frames can be achieved, and denoted by:

$$\{LBP_t^t(x, y), LBP_t^{t-1}(x, y), ..., LBP_t^{t-T}(x, y)\} \quad (7)$$

After extracting LBP feature set for one block by (7), the feature fusion is necessary to obtain the spatio-temporal LBP feature to decide whether the block is a foreground block or not. In this paper, we propose a simple yet efficient method to fuse the LBP feature. It is defined as follows:

$$STLBP_t(x, y) = \cup_{c=0}^{T} LBP_t^{t-c}(x, y) \quad (8)$$

where $\cup$ is an operator on each bit of LBP features. It outputs 1 when the number of '1' in corresponding bit of LBP features is more than that of '0', otherwise, outputs 0. Afterward, we convert the preliminary LBP from (8) to rotation invariant LBP by cyclic shifting of the 8-bits until get the minimum value. In this case, the LBP values in each previous frame vote for the decision on the LBP values of the current frame, which improves the robustness of LBP extraction. Without explicit illustration, all mentioned LBP features in the following part are rotation invariant LBP features. After obtaining the fused LBP feature for one block, the next work is to classify the block into a foreground block or a background block.

For a static scene, many blocks have a zero MV, resulting in a very sparse MV field. In such a field, the higher values of LBP features are more likely to appear in the background region and the regions inside the moving objects, while the lower values are supposed to locate in the boundary of moving objects according to (3)-(5). Likewise, as for a dynamic

scene, since the background blocks have a similar global motion, the distribution of LBP features in the dynamic situation is similar to that in the static scene. After extracting the LBP feature of all blocks, LBP image is constructed for object segmentation.

Figure 4. LBP image of sequence *Hall Monitor*. The first column is input frame; the second is the 4×4 block-level ground truth; the third is the corresponding object mask; the last column is the LBP features of image.

The extracted spatio-temporal LBP features of image is shown in Fig.4. As we can see, the LBP value of blocks inside the moving object and in the background is higher, whereas the LBP value of boundary blocks is much lower because of the dissimilarity among the blocks. Therefore, we first conduct a simple Gaussian smooth to the LBP image, then the foreground blocks can be labeled according to the following function:

$$M_t(x,y) = \begin{cases} 1 & \text{if } LBP_t(x,y) \leq Th_s \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

In (9), the block $b_t(x,y)$ is labeled as a foreground block if the LBP is no larger than Th_s, otherwise, it is labeled as a background block. In the LBP image, since the blocks inside object regions may be classified as background blocks by (9), the misclassified blocks inside object regions are converted to be foreground, and whose LBP values are assigned with zero.

4. Coarse-to-Fine Object Segmentation

After all the blocks in one frame are labeled according to the LBP image, the DCT coefficients are introduced to refine the coarse object region. In H.264, the DCT coefficients are used to code the residuals of the motion compensation, and larger DCT coefficients are more likely to appear at the boundary of object region. Therefore, the DCT coefficients can provide reliable cues for the refinement. In our algorithm, DCT coefficients in one frame are first temporally accumulated for several frames, then, the sum of DCT coefficients (SDCT) in each 4×4 block is calculated, followed by normalization to a value ranging from 0 to 255. Afterward, a modification algorithm is conducted to re-label the blocks in the contour of object regions based on the LBP image and SDCTs.

In the refining algorithm, we first locate the contour blocks of object region r in t-th frame, all the blocks in object region are scanned, and the blocks whose neighboring blocks include one or more background blocks are pushed into the block set, which is denoted by $C(t,r)$. Then the posterior probability of being a foreground block or background block for each contour block in $C(t,r)$ is calculated according to its LBP value and the location with respect to the distribution of SDCTs in one frame, defined as follows:

$$P(C|F) = P(C|LBP, L_0) \tag{10}$$

Assuming the independence between LBP image and SDCTs, the formulation (10) can be transformed to:

$$P(C|F) \propto P(C)P(LBP|C)P(L_0|C) \tag{11}$$

Given the function (11), the re-labeling of contour blocks can be regarded as Maximum a Posterior (MAP) estimation. To reduce the computation cost, the posterior probability of being a background block for contour block is set to be a const value in our algorithm, denoted by P_b. Therefore, MAP estimation can be simplified to be a thresholding problem. If the posterior probability $P(Fg|F)$ is less than P_b, the block is re-labeled as background block. To simplify the procedure, the prior probability $P(Fg)$ is regarded as a const, which is set to be 0.8 empirically in our algorithm. In (11),$P(LBP|Fg)$ is calculated for each contour block by considering the 4-adjacent neighboring blocks. It is expressed as:

$$P(LBP_t(x,y)|FG) = \begin{cases} 1 & \text{if } LBP_t(x,y) < \alpha\mu \\ \Phi(LBP_t(x,y)) & \text{otherwise} \end{cases} \tag{12}$$

where $LBP_t(x,y)$ is the spatio-temporal LBP value of contour block $b_t(x,y)$. $\Phi(\cdot)$ is the Gaussian probability density function, whose mean is the average LBP value of contour blocks in $C(t,r)$, denoted by μ, and the corresponding variance of LBP features is denoted by σ^2. In this function, if the LBP value is smaller than a threshold $\alpha\mu$ (in our algorithm, α is set to 1.2), the probability is set to 1; otherwise, calculate the probability using $\Phi(\cdot)$. Meanwhile, the probability $P(L_0|Fg)$ for one contour block given the distribution of non-zero SDCTs is calculated by:

$$P(L_0|Fg) = \frac{\sum_{i=1}^{M} S_t(x_i,y_i)K(||L_0 - L_i||^2)}{\sum_{i=1}^{M} S_t(x_i,y_i)} \tag{13}$$

where $S_t(x_i,y_i)$ is non-zero SDCT of block $b_t(x_i,y_i)$, whose 4×4 block-level coordinate is denoted by L_i. The Gaussian kernel function $K(\cdot)$ is utilized to estimate the likehood of being a foreground block for $b_t(x_0,y_0)$. In addition, a window whose size is the same as the bounding box of current object region is implemented to restrict the bandwidth of $K(\cdot)$. After all the contour blocks of object region r are re-labeled, the contour set $C(t,r)$ is updated and the parameters of $\Phi(\cdot)$, including μ and σ^2 are updated as well. Started with the initial object region, the refinement procedure repeats until the contour set $C(t,r)$ is unchanging. Upon completion, a more accurate object region is obtained. The coarse-to fine segmentation results of LBP image shown in Fig.4 are presented in Fig.5.

Figure 5. Object region refinement. The first column is input frame from *Hall Monitor*; the second is the 4×4 block-level ground truth; the third is the distribution of SDCTs; the last column is the final segmentation result.

5. Experimental Results

In this section, the performance of our method is evaluated on three standard test sequences: *Hall Monitor* (300 frames), *Daytime* (first 1000 frames) and *Table tennis* (112 frames). In our experiment, all the coding and decoding tasks are conducted according to the H.264 standard, and the edition number of the core encoder and decoder is JM12.4. All the test sequences are IPPP coded with the quantization parameter (QP) equal to 30, and only the first frame are code as I-frame. The MV search range is [-32, 32]. All the experiments are performed on a desktop computer with Intel Core i5, CPU 2.67 GHz, 2G RAM, and Microsoft Windows XP Professional. The parameters in our algorithm are set as: $I_s = 10$, $\tau = 0.5$, $T = 3$, $\omega = 1$, $Th_s = 80$, $P_b = 0.3$.

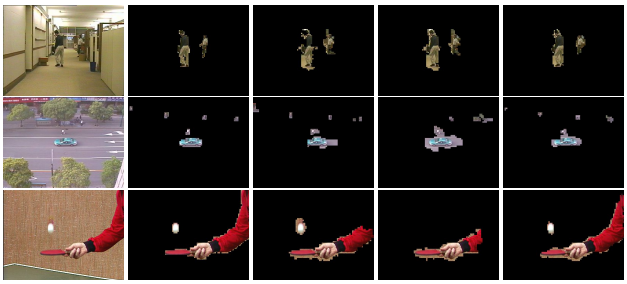

Figure 6. Comparison of segmentation performance, row one *Hall Monitor* (frame 87); row two: *Daytime* (frame 221); row three: *Table tennis* (frame 10).

In the experiment, the segmentation results of our algorithm is 4×4 block-level,and the 4×4 block-level ground truth in this paper is labeled manually. Concretely, the block that covers one or more foreground pixels in the pixel-level ground truth is labelled as a foreground block. To give a fair comparison, we conduct the algorithm in [2] based on the 4×4 block precision without further process of boundary refinement. The segmentation results of three methods are compared in Fig.6. In Fig.6, the first column shows original frames from the test sequences, and the second column presents the 4×4 block-level ground truth for these frames. The third column shows the segmentation results from [10]. The segmentation results of [2] are presented in the fourth column. And the segmentation results by the proposed method are shown in the last column.

To make a persuasive comparison, two metrics, precision (**P**) and recall (**R**) are used to quantify the performance of the proposed method and the other two methods on the test sequences with the 4×4 block-level ground truth. In addition, we use F-measure to evaluate the equilibrium between precision and recall as [2]. At this point, we select many frames from three sequences, that is, one frame from every five frames for the sequence *Hall Monitor* (interval: 20-300); one frame from every ten frames for *Daytime* (interval: frames with car); first 21 frames and last 29 frames for *Table tennis*.

Table 1. Average performance for three sequences

Sequence	Method	**P**	**R**	F-measure
Hall Monitor	Proposed	0.72	0.84	0.78
	Ref.[10]	0.60	0.84	0.70
	Ref.[2]	0.57	0.73	0.64
Daytime	Proposed	0.62	0.68	0.65
	Ref.[10]	0.45	0.60	0.51
	Ref.[2]	0.40	0.86	0.63
Table tennis	Proposed	0.86	0.85	0.85
	Ref.[10]	0.85	0.69	0.76
	Ref.[2]	0.77	0.75	0.76

The quantitative curves are shown in Fig.7 (blue: Precision, red: Recall, green: F-measure). In the first row, the algorithm in [10] obtains a temporally stable but slightly poor segmentation results. In the second row, though the recall of segmentation results from the method [2] is higher than the others, a much lower precision and F-measure occur to all the three sequences, which is not applicable to the requirement of accurate segmentation of surveillance videos. In contrast, the proposed method not only achieves higher precision and recall, but also keeps the balance between them towards all the test sequences. In Table 1, the comparison of average quantitative also proves that the segmentation results from this paper are more accurate than the other two methods. Since the coarse classification of MV is merely based on rigid thresholding, the segmentation result in [10] is not robust to the variety of MV field. In [2], without considering the spatio-temporal correlations among blocks, the segmentation results are scattered and fragmented, meanwhile, false estimation of the number of MRF classes is often occurred in their work, resulting in over-segmentation or under-segmentation in many cases. Since the exploiting of LBP, the proposed algorithm enjoys a lower computational complexity. Moreover, to further accelerate the speed of the algorithm, we introduced a pre-selection method to locate the regions where the LBP features should be extracted. Concretely, a larger scale (16×16 or 32×32) accumulating of MVs is conducted to decide the candidate regions of moving objects. The average processing time per frame is 68 ms for the three sequences.

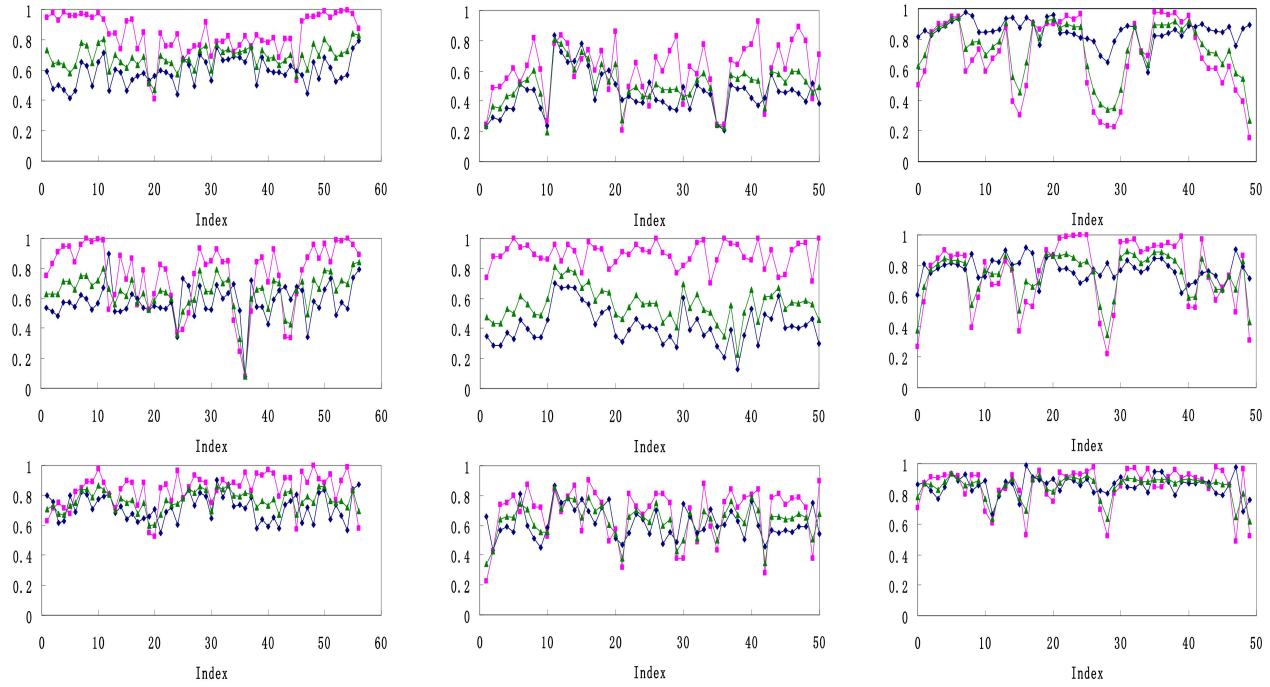

Figure 7. Performance evaluation of three sequences: *Hall Monitor*, *Daytime*, *Table tennis* (from left to right). The first row is the quantitative curve of method in [10]; the middle one is from the method in [2]; the last row is the segmentation results of the proposed method.

6. Conclusions

In this paper, a novel object segmentation method is presented and discussed. The spatio-temporal LBP is introduced to extract the coarse moving objects in the H.264 compression domain, followed by a region refinement according to the distribution of DCT coefficients. As the experiments show, the algorithm provides a reliable and accurate segmentation results compared with the previous methods.

7. Acknowledgement

This work is supported by National Natural Science Foundation of China under Projects 90920001 and 61101212.

References

[1] R. V. Babu, K. R. Ramakrishnan, and S. H. Srinivasan. Video object segmentation: A compressed domain approach. *IEEE Trans. Circuits Syst. Video Technol.*, 14(4):462–474, 2004.

[2] Y.-M. Chen, I. V. Bajic, and P. Saeedi. Motion segmentation in compressed video using markov random felds. *Proc. IEEE ICME*, pages 760–765, July 2010.

[3] Y.-M. Chen, I. V. Bajic, and P. Saeedi. Moving region segmentation from compressed video using global motion estimation and markov random fields. *IEEE TRANSACTIONS ON MULTIMEDIA*, 13(3), June 2011.

[4] M. Heikkila, M. Pietikainen, and J. Heikkila. A texture-based method for detecting moving objects. *British Machine Vision Conference*, pages 187–196, 2004.

[5] J. S. P. HW. Moving object segmentation in dct-based compressed video. *Electronics Letters*, 36:1769–70, 2000.

[6] Z. Liu, Y. Lu, and Z. Zhang. Real-time spatiotemporal segmentation of video objects in the h.264 compressed domain. *Visual Communication and Image Representation*, June 2007.

[7] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[8] C. Poppe, S. de Bruyne, T. Paridaens, P. Lambert, and R. van de Walle. Moving object detection in the h.264/avc compressed domain for video surveillance applications. *Visual Communication and Image Representation*, 20:428–437, 2009.

[9] W. Wang, J. Yang, and W. Gao. Modeling background and segmenting moving objects from compressed video. 18(5):670–681, 2008.

[10] W. Zeng, J. Du, W. Gao, and Q. Huang. Robust moving object segmentation on h.264/avc compressed video using the block-based mrf model. *Real-Time Imaging*, 11:290–299, 2005.

[11] D. S. Zhang and G. Lu. Segmentation of moving objects in image sequence: a review. *Circuits Syst. Signal Process*, 20:143–183, 2001.