Real-Time Foreground Segmentation for the Moving Camera Based on H.264 Video Coding Information

[1]Wei-Di Hong, [2]Tien-Hsu Lee, and [1]Pao-Chi Chang
[1]*Dept. of Communication Engineering, National Central University*
[2]*Dept. of Electrical Engineering, National Chi Nan University*
wdhung@vaplab.ee.ncu.edu.tw
thlee@vaplab.ee.ncu.edu.tw
pcchang@ce.ncu.edu.tw

Abstract

Foreground segmentation for video frames has played an important role in many video applications, such as video surveillance, video indexing, etc. Due to most videos are compressed, foreground segmentation can benefit from utilizing such coding information and save much processing time. In this paper, we propose a real-time foreground segmentation algorithm for the moving camera based on the H.264 video coding information. In the proposed algorithm, we first utilize the relative global motion model to calculate the approximate global motion vector and get the motion vector difference of each block. Then, according to the block partition modes, we assign different weightings and apply spatio-temporal refinement to these motion vector differences for further improving the accuracy of segmentation results. Finally, we segment out the foreground blocks by an adaptive threshold. With the aid of H.264 video coding information, the proposed segmentation algorithm is more practical than many other methods based on spatial domain information in computational complexity.

1. Introduction

Video data usually need to perform compression before storage or transmission. The H.264/AVC video coding standard [1], [2] has been developed to achieve significant improvements over the existing previous standards in the performance of compression efficiency. With the advance of semiconductor technology, various video applications, such as the video conferencing, video indexing, video surveillance, object tracking, and object-based video coding, also become much more feasible to be implemented in real-time. In these applications, the foreground segmentation plays an important role among the complicated processing tasks. For example, in a multi-video surveillance system, the camera with the large foreground area of video contents is supposed to contain significantly noticeable objects and deserve to obtain much more attention. The foreground segmentation results directly influence the entire system performance. Consequently, a fast and accurate foreground segmentation method is required.

Foreground segmentation is usually achieved by segmenting the moving part (relative to the background) of video contents. The motion of video contents can be extracted by means of motion estimation, optical flow, or differences between frames, etc. Note that a moving part of a non-rigid object is not necessary to be the whole object. This is one of the problems need to be resolved. To segment the entire object out, the spatial information such as the contour should be employed additionally.

Various foreground segmentation algorithms have been proposed for different purposes. According to [3], these algorithms can be grouped into three categories: the motion-based, change-based, and spatio-temporal segmentation.

Motion-based segmentation techniques extract regions with homogeneous motion vectors (MVs) and achieve fast segmentation [4], [5]. These methods treat the video compression as a kind of pre-processing, as the video coding is inevitable in most of video applications. In this way, motion-based segmentation techniques can reduce the computational costs on obtaining the motion information. Furthermore, for the video captured from the moving camera, the motion of background can be approximated by exploring motion vectors, under the assumption that most blocks belong to the background. By subtracting the background motion from all motion vectors, we can treat the results as from a fixed camera. A block-based Markov random field (MRF) model for the motion vector field was proposed in [3].

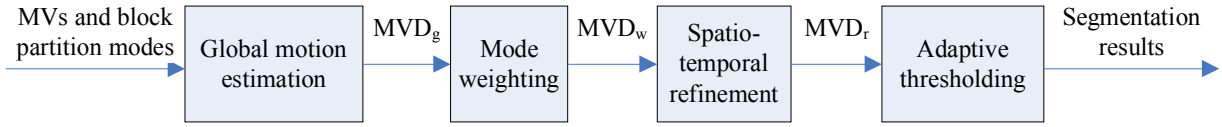

Figure 1. Block diagram of the proposed algorithm

Change-based segmentation techniques extract moving objects by comparing successive frames or comparing frames with a background model [6], [7]. For video contents from a fixed camera with a static background, change-based segmentation methods are very efficient since their efforts are spent on the changed parts of frames only. On the other hand, for a moving camera, the inter-frame changes are plenty and lead to unacceptable results, unless the background model is established with the entire video sequence. In this case, the process needs to be done offline [8].

Spatio-temporal segmentation techniques utilize motion vectors to obtain a relatively coarser intermediate result, while the spatial information guarantees that segmentation boundaries coincide with the object boundaries [9]-[11]. These properties benefit the applications such as the pattern recognition but require extra efforts compared to the motion-based algorithms.

In H.264 video coding standard, in addition to motion vectors, there are seven block partition modes for the inter-frame coding. It is observed that small-sized blocks tend to be the foreground, and vice versa. Therefore, our goal is to exploit such H.264 video coding information to achieve real-time foreground segmentation for the video captured from moving cameras. We choose the motion-based method, which achieves fast segmentation and favors real-time systems. In the next section, we will describe the proposed algorithm in detail. The simulation results are presented in Section 3. Section 4 concludes our works.

2. The proposed foreground segmentation scheme

Figure 1 is the block diagram of our proposed foreground segmentation algorithm for the moving camera based on the H.264 coding information. The coding information we exploit includes block partition modes and motion vectors.

The processing blocks are composed of global motion estimation, mode weighting, spatio-temporal refinement, and adaptive thresholding, respectively. First, we choose proper MVs according to partition modes to perform the global motion estimation and subtract the obtained global motion from all MVs to offset the MVs caused by the camera movement. Then, we obtain intermediate results for moving foreground. After that, the intermediate results are enhanced by different weighting coefficients for various partition modes. We also perform spatial and temporal filtering to refine the results. Finally, an adaptive threshold is employed to segment out the resultant foreground.

2.1. Global motion estimation

There exist two prerequisites to the global motion estimation for the H.264 coded video here:

1. For inter-frame global motion estimation, the number of reference frame is restricted to one in our algorithm.

2. We take 4x4 block as the basic processing unit. For example, a motion vector for a 16x16 partition is treated as 16 motion vectors with the same value for all 16 4x4 partitions. Thus, the positions of MVs become regular and dense.

In this paper, we utilize the least-squares estimator to approximate the global motion, as presented in [4]. It presents an iterative least-squares method for the compressed video with a six-parameter global motion affine model. The six parameters satisfy

$$\begin{pmatrix} p_1 & p_2 & p_3 \\ -p_2 & p_1 & p_4 \\ p_5 & p_6 & 1 \end{pmatrix} = \begin{pmatrix} \frac{F_2}{F_1} & \frac{F_2}{F_1}\theta_z & -F_2\theta_x \\ \frac{F_2}{F_1}\theta_z & \frac{F_2}{F_1} & F_2\theta_y \\ \frac{\theta_x}{F_1} & \frac{\theta_y}{F_1} & 1 \end{pmatrix} \quad (1)$$

where F_1 and F_2 are the focal lengths before and after zooming, respectively. θ_x, θ_y, and θ_z are the rotation angles along x-axis, y-axis, and z-axis, respectively.

With the minimal lens distortion, the global motion model formulated in (2) is a very good approximation if the camera motion is small and contains only the combination of panning, rotating, and zooming. x_t and y_t are the coordinates of motion vectors in the current frame, while x_{t-1} and y_{t-1} represent the motion compensated positions in the previous frame.

$$\begin{pmatrix} x_{t-1} \\ y_{t-1} \end{pmatrix} = \begin{pmatrix} \frac{p_1 x_t + p_2 y_t + p_3}{p_5 x_t + p_6 y_t + 1} \\ \frac{-p_2 x_t + p_1 y_t + p_4}{p_5 x_t + p_6 y_t + 1} \end{pmatrix} \quad (2)$$

Our modified iterative procedure for the global motion estimation of the H.264 compressed video is shown below:

1. Find the least-square solution to the six parameters using the motion vectors for all 16x16 partitions excluding the frame boundary blocks.

2. Compute the MVD_g, defined as the absolute differences between all MVs and the estimated global motion, and the standard deviation of MVD_g.

3. Reject the MVs of the blocks whose MVD_g are larger than a pre-defined threshold T_g (a reasonable choice here is 1.5 times of the standard deviation value) in the next iteration.

4. Perform three iterations to obtain the final six parameters of the global motion model and MVD_g. Note that in the 2nd and 3rd iterations, the MVs are no longer restricted to 16x16 partitions.

The reason for only choosing MVs of 16x16 partitions is the observed results of partition modes distribution, as shown in Figure 2. The partition modes are represented by an 8-bit gray scale image, where the pixel value of 0 stands for mode 0 (skip mode), 32 for mode 1, and so on. We can find that the large block partitions are likely to be the background. Note that the MVs of macroblocks on frame boundaries become irregular for moving cameras. Therefore they are removed from all intermediate calculation results. Note that applying only the MVs of 16x16 partitions in the 1st iteration for global motion estimation reduces the computational cost overall by 28% on average without degradation of segmentation results.

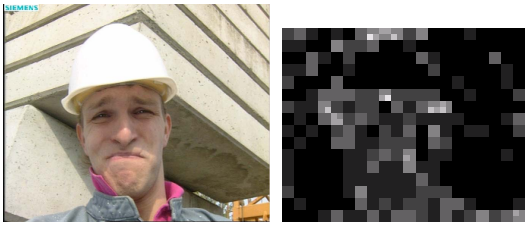

Figure 2. Partition modes distribution for the 137th frame of Foreman

2.2. Mode weighting

We utilize the observations described above and give different weightings to the MVD_g in accordance with their block partition modes. The principle is that the weighting factors are larger for the smaller block partitions according to the following equation:

$$MVD_w = W_n \times MVD_g \quad (3)$$

where MVD_g are the absolute differences between the estimated global motion and MVs; W_n is the weighting factor for the block partition mode n; MVD_w represent the resultant weighted differences. In our experiments, we set $W_0 = 3$, $W_1 = 4$, $W_2 = W_3 = 5$, $W_4 = 8$, and 10 for the rest partition modes.

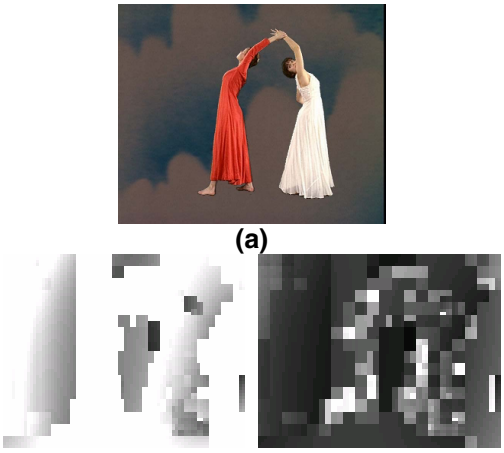

Figure 3. The 2nd frame of Dancer: (a) original image, (b) MVD_g, and (c) MVD_w

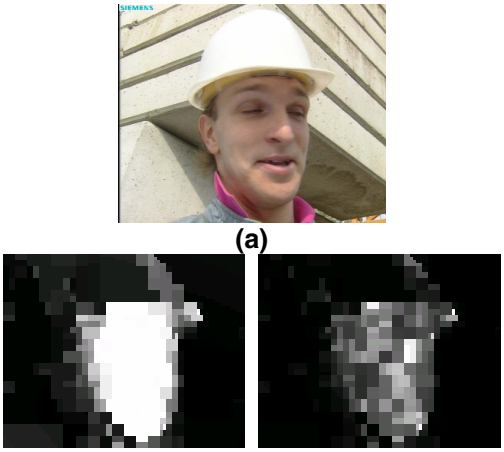

Figure 4. The 102nd frame of Foreman: (a) original image, (b) MVD_g, and (c) MVD_w

Figure 3 and Figure 4 show the MVD_g and MVD_w results for the 2nd frame of Dancer and the 102nd frame of Foreman. For Dancer, the MVD_g provides little information for the foreground segmentation owing to the homogeneous background blocks with noisy MVs. Through the mode weighting for different partitions, the foreground objects are successfully highlighted. However, part of foreground blocks are given light weightings in Foreman, which result in unsatisfied fragmented segmentations. To keep the intactness and further enhance the segmentation results, we employ spatial and temporal filters for refinements.

2.3. Spatio-temporal refinement

We have to perform spatio-temporal refinements due to some defects that result in MV noises. The reasons are explained as follows. First, MVs do not accurately represent the true motions of objects. They are generated by the minimum cost consideration only. Besides, there are revealing blocks and slightly moving background objects. Of course, our mode weighting for different partition modes may be inadequate in a few cases. Thus, we propose low-pass filters for partitions of mode 0 to 3, and mode 4 to 7, respectively. The proposed spatial filter for mode 0 to 3 is expressed by the following equation:

$$MVD_s = \frac{(5-l_u)\times MVD_{w,u} + (5-l_d)\times MVD_{w,d} + (5-l_l)\times MVD_{w,l} + (5-l_r)\times MVD_{w,r} + N\times MVD_{w,i}}{(5-l_u)+(5-l_d)+(5-l_l)+(5-l_r)+N} \quad (4)$$

where MVD_s is the filtered value for a 4x4 block; l_u, l_d, l_l, and l_r are the distances from the 4x4 block to the nearest vertical and horizontal neighboring partitions in the unit of blocks; $MVD_{w,u}$, $MVD_{w,d}$, $MVD_{w,l}$, and $MVD_{w,r}$ are the MVD_w values of the neighbors; $MVD_{w,i}$ is that of the processing block.

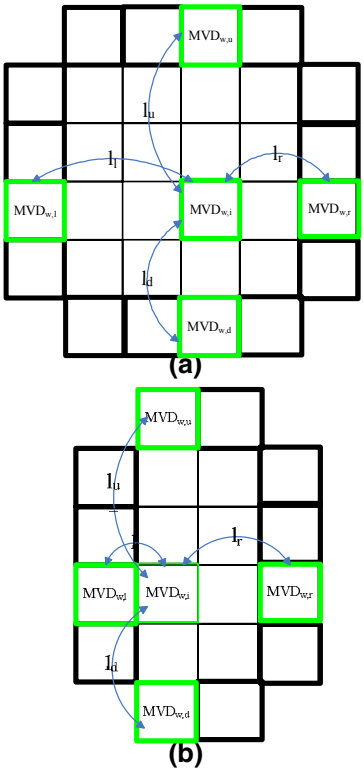

Figure 5. Examples of the spatial filer for mode 0~3: (a) 16x16 partition and (b) 8x16 partition

0	1	0
1	2	1
0	1	0

Figure 6. Spatial filter for mode 4~7

We set $N = 10$ for mode 0 and 1, $N = 8$ for mode 2 and 3. Figure 5(a) and 5(b) illustrate two examples of our spatial filter.For mode 4 to 7, we use the filter shown in Figure 6 to eliminate MV noises. In H.264, there could be intra coded MBs in P and B frames. We apply a filter similar to that for mode 0~3 but without the central coefficient.

For temporal refinement, we take the average of the MVD_s value of the processing block in the current frame and the spatio-temporal filtered result, MVD_r, of the corresponding motion-compensated block in the previous frame as follows.

$$MVD_r^t = \left(MVD_s^t + MVD_r^{t-1}\right)/2 \quad (5)$$

where t represents the current frame and t-1 stands for the previous frame. The positions of the corresponding blocks in the previous frame are obtained by adding up the MVs and positions of the processing blocks in the current frame. The temporal refinement can help keep the true foreground and remove the burst MV noises.

2.4. Adaptive thresholding

The global motion estimation errors result in noises, especially for blocks near frame boundaries. It is because motion vectors caused by zooming and rotating are proportional to the distance from the original point, i.e., the center of a frame. Consequently, it is inadequate to apply the same threshold over the entire frame. The threshold should increase with this distance. In addition, to avoid the dramatic variations in segmentation results, we define the mean value of MVD_r multiplied by a coefficient as the adaptive threshold, which is formulated as follows.

$$Th(x,y) = \max\left(C\left(1+\frac{x^2+y^2}{x_{\max}^2+y_{\max}^2}\right)MVD_{r,mean}, T\right) \quad (6)$$

where $Th(x, y)$ is the adaptive threshold for a block in position (x, y). x_{max} and y_{max} are the maximum x and y values. $MVD_{r,mean}$ is the mean value of MVD_r. C is a constant. A constant threshold T is also applied to remove noises in case the mean value is too small. In our experiments, C = 1.5 and T = 16. A comparative example is exhibited in Figure 7. The segmentation for fixed thresholding uses threshold = 7 here.

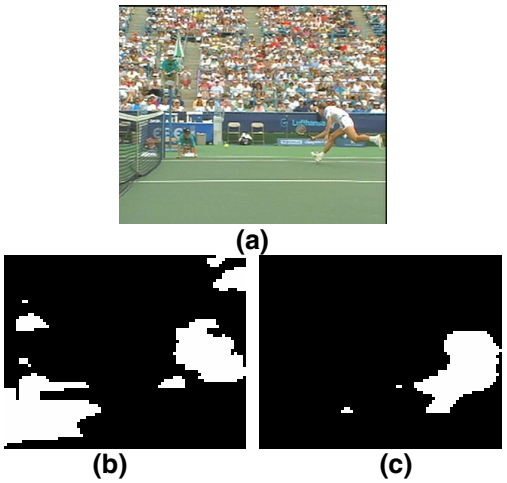

Figure 7. The segmentation results for the 274th frame of Stefan: (a) original image, (b) fixed threshold, and (c) adaptive threshold

3. Simulation results

The segmentation results for the CIF-size test sequences Bus, Coastguard, Dancer, and Foreman are shown in Figures 8-11, respectively, while the encoding parameters are listed in Table 1. All test sequences were captured from a moving camera or with the moving background. The frame boundaries are not processed so that there would not exist segmented foreground blocks.

From the segmentation results shown here, we can observe the fact that the motion-based methods are more suitable for rigid foreground objects than for non-rigid ones. This is the result of that motion-based methods do not segment the regions without motion vectors. While the foreground objects are rigid, such as Figure 8 and 9, the uniform motion vectors make the segmentation results more accurate than those of sequences with non-rigid foreground objects, as shown in Figure 10 and 11. In addition, the revealing background blocks in Figure 10 result in false segments. The similar colors of different objects could also affect segmentation results, such as the hat and the background in Figure 11.

Our proposed refinement method can eliminate motion vector noises effectively. For example, the ripples in Figure 9 are almost removed from the segmentation result. For further accurate segmentation, the additional contour or other spatial information may be much helpful. However, the computational costs will increase and may be not suitable for real-time applications.

Figure 8. Segmentation result for the 72nd frame of Bus

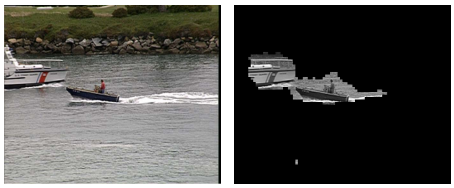

Figure 9. Segmentation result for the 24th frame of Coastguard

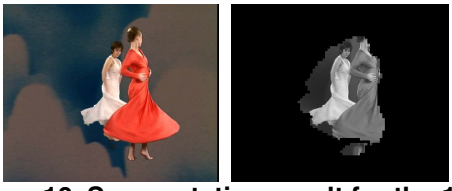

Figure 10. Segmentation result for the 108th frame of Dancer

Figure 11. Segmentation result for the 172nd frame of Foreman

Table 1. Encoding parameters

Profile	JM 10.2 Baseline
Frame Rate	30
Intra period	30
QP	28
Search Range	16
FME Used	UMHexagonS
Number of Reference Frames	1
Partition Mode Used	All on
Rate Control	Off

Table 2. The processing speed (frame per second; FPS) on test sequences

Video source	Bus	Coast-guard	Dancer	Foreman	Stefan
FPS	**40.3**	**39.9**	**39.8**	**39.2**	**39.5**

The positions and areas of foreground objects in the video captured from moving cameras can be rapidly acquired by our proposed method. Near 40 fps for CIF-size video sequences is achieved without using the assembly language on a PC with P4 2.8GHz CPU and 1GB RAM, as shown in Table 2.

4. Conclusions

In this paper, we present a novel motion-based foreground segmentation algorithm for the H.264-encoded video captured by moving cameras. The global motion, including panning, tilting, and zooming, is estimated by the least-squares method using the motion vectors of large partitions. In addition to motion vectors, inter-coding partition modes are considered in the proposed method to enhance the segmentation results by means of different weightings and spatio-temporal refinements. We also propose an adaptive threshold method for the final foreground segmentation. The proposed segmentation algorithm requires low computational costs and suits for real-time applications such as the video surveillance system and real-time object tracking.

5. References

[1] *ITU-T Recommendation H.264 & ISO/IEC 14496-10 AVC: Advanced video coding for generic audiovisual services*, March 2005.

[2] T. Wiegand, G. J. Sullivan, G. Bjontegaard and A. Luthra, "Overview of the H.264/AVC video coding standard," IEEE *Transactions on Circuits and Systems for Video Technology,* vol. 13, 2003, pp. 560-576.

[3] W. Zeng, J. Du, W. Gao, and Q. M. Huang, "Robust moving object segmentation on H.264/AVC compressed video using the block-based MRF model," *Real-Time Imaging*, 2005, pp. 290-299.

[4] J. Wang, N. Patel, and W. Grosky, "Moving camera moving object segmentation in an MPEG-2 compressed video sequence," *Proceedings of the Conference on Multimedia Content Analysis, Management, and Retrieval (IS&T/SPIE Symposium on Electronic Imagery),* 2006, pp. 372-379.

[5] F. Arnell and L. Petersson, "Fast object segmentation from a moving camera," *Proceedings of IEEE, Intelligent Vehicles Symposium,* 2005, pp. 136-141.

[6] K. Toyama, J. Krumm, B. Brumitt and B. Meyers, "Wallflower: Principles and practice of background maintenance," *Proceedings of the 1999 7th IEEE International Conference on Computer Vision (ICCV'99),* 1999, pp. 255-261.

[7] A. Makarov, "Comparison of background extraction based intrusion detection algorithms," *Proceedings of the 1996 IEEE International Conference on Image Processing, ICIP'96. Part 1 (of 3),* 1996, pp. 521-524.

[8] Y. Sugaya and K. Kanatani, "Extracting moving objects from a moving camera video sequence," *Proceedings of the 10th Symposium on Sensing via Imaging Information,* June 2004, pp. 279-284.

[9] J. Y. Wang, A. E. Adelson, "Spatio-temporal segmentation of video data," *Proceedings of SPIE on Image and Video Processing II,* Vol. 2182, San Jose, February 1994, pp. 120-131.

[10] R. Wang, H. Zhang, and Y. Zhang, "A confidence measure based moving object extraction system built for compressed domain," *Proceedings of IEEE Int. Symp. Circuits and Systems,* vol. 5, 2000, pp. 21-24.

[11] R. Cucchiara, A. Prati, and R. Vezzani, "Object segmentation in videos from moving camera with MRFs on color and motion features," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* vol. 1, 2003, pp. 405-410.