An Efficient Technique for Segmentation of Key Object(s) from Video Shots

JungHwan Oh *JeongKyu Lee* *Eswar Vemuri*

Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, TX 76019-0015 U. S. A.
e-mail: {*oh, jelee, evemuri*}*@cse.uta.edu*

Abstract

The most fundamental task in video processing is to partition long video sequences into a number of shots and find a *key frame* of each shot for indexing and browsing. In this paper, we want to extract *key object(s)* instead of key frame. In order to do that, we extend our previous technique for shot boundary detection (SBD) to select two candidate frames out of each shot, then propose a new framework to segment object(s) from those selected frames using *color quantization* and background adjustment. We make our scheme cost-effective and automatic by avoiding expensive computations, and removing manual processing. Experimental results show that the proposed scheme is promising.

KEYWORDS: Video object segmentation, shot boundary detection, color quantization, MPEG-4/MPEG-7.

1 Introduction

A video obtained from various sources is called a *video clip*. It can last from a few seconds to several hours. For most video applications a video clip is not a convenient unit for data entry since an entire video stream is too coarse as a level of abstraction. We need to find a new basic unit for handling video data. It has been agreed that *shot*, which is defined as a collection of frames recorded from a single camera operation, is the winner among the several candidates since shot boundaries can be decided objectively and mechanically.

There has been a large amount of researches [1, 2, 3] to find these shot boundaries automatically, in other words, to parse a long video into a number of shots. In our previous works [4, 5], we also proposed shot boundary detection (SBD) technique which needs to process substantially less data since it uses only backgrounds instead of whole images. After a video is decomposed into shots, the next step is usually to extract *key frame* from each shot for content-based summarizing and understanding. In this paper, we want to extract *key object(s)* instead of key frame. In order to do that, we, first, extend our previous technique for shot boundary detection (SBD) to select two candidate frames out of each shot, then propose a new framework to segment object(s) from those frames (see Figure 1).

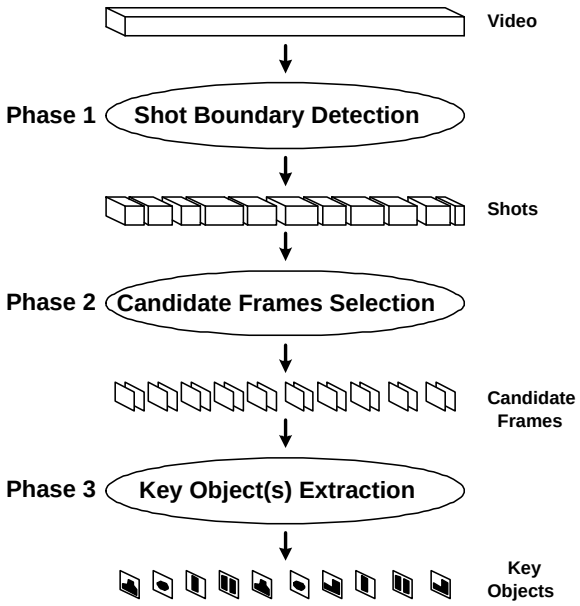

Figure 1: Key Object(s) Extraction Procedure

A large amount of researches about video object has been done recently [6, 7, 8, 9, 10]. However, for most video sequences, this algorithm needs manual processing to get a background edge map in the initial stage. In addition, it does not work very well for the video sequence with some camera motions, in which background is not fixed.

To address these issues, we first propose a technique to segment object(s) from a video sequence by *color quantization*, and apply the proposed technique to find key object(s) in a shot by using the extension of our previous SBD technique. The whole procedure is shown in Figure 1, and the advantages

of the proposed framework can be summarized as follows:

- Segmentation algorithm is efficient and achieve fast speed since it does not use 'edge detection' or 'optical flow'.
- Manual process is not necessary for the initialization.
- It can be applied to the video sequences with some camera motions, in which background is not fixed, because it has a function to adjust background position.

The remainder of this paper is organized as follows. In Section 2, we discuss the basic idea of object segmentation in video sequence using color quantization. The extension of our previous SBD works and its application to find key object(s) in a decomposed shot are presented in Section 3. In Section 4, we discuss our experimental results. Finally, we give our concluding remarks in Section 5.

2 Key Object(s) Segmentation

In this section, Phase 3 (Key Object(s) Extraction) in Figure 1 is discussed first for better understanding. We present the color quantization, the basic idea of object edge extraction, and how to obtain the object region from the extracted object edges.

2.1 Color Quantization

Our object edge map extraction algorithm starts with color quantization of each original color frame. The quantized frame can be computed using the following formulas. $F_n = \Lambda(f_n)$, where F_n and f_n are quantized and original frames respectively, and the Λ is quantization function in which the inputs are pixel values from original frame, and the outputs are recalculated pixel values. Thus, the above equation can be rewritten as

$$\begin{aligned}&(R, G, B) = \Lambda(r, g, b)\\&= ((r\ div\ 2^d) \cdot 2^d,\ (g\ div 2^d) \cdot 2^d,\ (b\ div\ 2^d) \cdot 2^d) \qquad (1)\end{aligned}$$

where $d = \frac{k-m}{3}$, and the color depth of original and quantized frames are 2^k, 2^m respectively. r, g, and b are the RGB pixel values from original frame and R, G, B are the RGB pixel values from quantized frame.

2.2 Object Edge Extraction

Figure 2 (a) and (b) show two consecutive frames in a video clip (TV drama), which are quantized to the color depth 64 using the above equation (1) in which $d = 6$ since $k = 24$ and $m = 6$. From those two frames, we can find the difference image as shown in Figure 2 (c). If two pixels in the same position from two frames have the same color (value), then the color of the pixel in the same position of the difference image becomes white, otherwise, it becomes black. As seen in the figure, there is no noise. By using a simple 'color quantization', we can get the same accuracy of the results. Consequently, we can save a great deal of computation cost.

(a) (b) (c) (d)

Figure 2: (a) Quantized Frame #1584 (b) Quantized Frame #1585 (c) Object Edge Map from quantized frames (d) Object Edge Map from original frames

2.3 Object Region Selection

After we obtain the object edge map such as Figure 2 (c), the next task is to select object region. The existing methods [9, 10] scan the edge map horizontally and vertically, and try to find the first and last edge points(pixels) for each row and column of pixels.

The problem comes from the connectivity of edge points. If some points in the edge are missing(which usually happens in the edge extraction), it cannot give optimal results. Since we cannot guarantee that no point is missing in all cases of edge extraction, we approach differently to obtain object regions. Here, we propose a scheme based on *block estimation*. This scheme can be described as follows.

Step 1 : First, the edge map obtained in the previous subsection is divided into a number of same size blocks. The usual block size used in JPEG or MPEG is 8×8 pixels, but to add more accuracy, $4 \times\ 4$ or $2 \times\ 2$ blocks can be used in our scheme.

Step 2 : Each block is examined to find how many edge points it has. If a block has more than a certain number of points, it can be considered in the object region. The threshold value depends on the block size. In our experiments, we used $4 \times\ 4$ as block size, and we picked blocks with more than four points (25%).

Step 3 : However, there are some blocks which do not have enough edge points, but should belong to object region. These blocks are usually located inside object region. To select these blocks as object region, we used a simple algorithm which is illustrated in Figure 3 (a). As seen in the figure, to decide whether the center block (which has not enough points) belongs to object region or not, first check its 8 neighbor blocks. If any 5 out of these 8 blocks belong to object region, in other words, they are already selected as object region in the previous step, the center block can

be marked as object region even if it has not enough edge points. When a center block is positioned in the boundary of frame, and if any 3 out of 5 neighbors belong to the object region, the center block can be marked as object region. This Step 3 is performed recursively until no more block is added to object region.

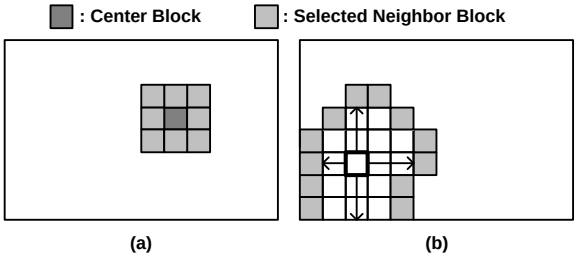

Figure 3: (a) Step. 3 (b) Step. 4

Step 4 : Step 3 can select most of the blocks inside the object region. However, any block, which is not neighboring any of the selected blocks, cannot be selected as object region using Step 3 even if it is in object region. Therefore, for each block which is not selected as object region we check all four directions as illustrated in Figure 3 (b). If a selected block is met in all four directions, the block is selected as object region.

3 Shot Boundary Detection and Candidate Frames Selection

Phase 1 and 2 in Figure 1 are explained in this section. We first present our previous works [4, 5] about Shot Boundary Detection (SBD) briefly to make this paper self-contained. We then discuss the extension of our SBD work and how to select two candidate frames for each shot.

3.1 A Camera Tracking Approach to SBD and Its Extension

To decompose a video to shots, we use our technique (*Background Tracking* (BGT)) developed in our previous works [4, 5]. To make the computation efficient, we rotate the two vertical columns of each $\sqcap$ shape a *fixed background area* (FBA) outward to form a *transformed background area* (TBA) From each TBA, which is a two-dimensional array of pixels, we compute its *signature* and *sign* by applying a modified version of the image reduction technique, called *Gaussian Pyramid* [11]. We use this technique to reduce a two-dimensional TBA into a single line of pixels (called $signature_i$) and eventually a single point of pixel (called $sign_i^{BA}$). The superscript and subscript indicate that this is the sign of the *background area* of some $frame_i$. The term 'sign' is used to indicate that it is an abstracted form of the signature.

We define the *fixed object area* (FOA) as the foreground area of a video frame, where most primary objects appear. We reduce the FOA of each $frame_i$ to one pixel. That is, we want to compute its sign, $sign_i^{OA}$, where the superscript indicates that this sign is for an FOA. This parameter can be obtained using the Gaussian Pyramid as in computing $sign_i^{BA}$. As mentioned above, the details of the other computation (i.e., how to decide the dimensions of FBA and FOA) can be found in our previous works [4, 5].

We have used $sign^{OA}$ and $sign^{BA}$ as extracted features from each shot for shot indexing purpose in our previous work [5]. In this paper, we add more sophisticated features extracted from each shot. More details will be discussed in the next section.

3.2 Two Candidate Frames Selection and Key Object(s) Extraction

To extract key object(s) from these shots, in other words, to use the object segmentation technique described in Section 2 optimally, we need two consecutive frames in which there is no change in background, and as little as possible changes in object area. Using our SBD technique and its extension discussed above, we can verify these changes in background and object region. The values of $sign^{BA}$ and $sign^{OA}$ indicate the average color values of background and object areas for each frame. We can compute the differences of $sign^{BA}$s and $sign^{OA}$s between two frames i and $i+1$. They are called $DBA_{i,i+1}$ and $DOA_{i,i+1}$ for $sign^{BA}$ and $sign^{OA}$ respectively. If $DBA_{i,i+1}$ is zero, we can verify that no change occurs in background because two background average colors are same. We can apply the same concept for $DOA_{i,i+1}$. Therefore, the selection algorithm of candidate frames for a shot with n frames can be summarized as follows.

Step 1 : First, we compute $n-1$ number of DBAs using Equation (2) and find a frame pair in which the value of DBA is zero. If there are more than one candidate, then compute DOAs by Equation (3) and find the smallest among the selected candidates. If there are still more than one candidate even after considering DOAs, we can just select the temporally earliest one (This can be used as tiebreaking rule for all cases in which there are more than one candidate). If we cannot find any frame pair with zero value of DBA, we go to Step 2.

Step 2 : Since we cannot find any frame pair with zero value of DBA, we select a frame pair with the smallest value of DBA. We compare $Signature_i$ with $Signature_{i+1}$ by the background tracking method discussed above. In this time, we focus on the location as well as the amount of maximum matching. Three

cases are possible for the matching locations In this step, we compute and return the matching block numbers (MBN_i) of $Signature_i$, and the matching block numbers (MBN_{i+1}) of $Signature_{i+1}$. The positions of matching blocks can be categorized into three cases as follows. For simplicity, we assume that there is only one matching block.

Step 3 : Using the values of MBN_i and MBN_{i+1} returned in the previous step, we compute the comparable regions for $frame_i$ and $frame_{i+1}$ by the algorithm in Figure 4. Assume that r is the number of pixels in a row, c is the number of pixels in a column, LB is a last block in the left vertical bar, and FB is a first block in the right vertical bar. In our case, LB is 6, and FB is 13.

```
If ( ( MBN i+1 <= LB ) AND ( MBN i <= LB ) ) OR ( ( MBN i+1 >= FB ) AND ( MBN i >= FB ) )
    A = MBN i+1 - MBN i ,
    If ( MBN i+1 <= LB ) AND ( MBN i <= LB )
        if ( A < 0 )
            return  the ranges ( 0 <= x <= c ) and ( 0 <= y <= r - A ) for frame i, and
                    the ranges ( 0 <= x <= c ) and ( A <= y <= r ) for frame i+1
        else
            return  the ranges ( 0 <= x <= c ) and ( A <= y <= r ) for frame i, and
                    the ranges ( 0 <= x <= c ) and ( 0 <= y <= r - A ) for frame i+1
    If ( MBN i+1 >= FB ) AND ( MBN i >= FB )
        if ( A < 0 )
            return  the ranges ( 0 <= x <= c ) and ( A <= y <= r ) for frame i, and
                    the ranges ( 0 <= x <= c ) and ( 0 <= y <= r - A ) for frame i+1
        else
            return  the ranges ( 0 <= x <= c ) and ( 0 <= y <= r - A ) for frame i, and
                    the ranges ( 0 <= x <= c ) and ( A <= y <= r ) for frame i+1
Else If ( LB <= MBN i+1 <= FB ) AND ( LB <= MBN i <= FB )
    B = MBN i+1 - MBN i,
    if ( B < 0 )
        return  the ranges ( 0 <= y <= r ) and ( B <= x <= c ) for frame i, and
                the ranges ( 0 <= y <= r ) and ( 0 <= x <= c - B ) for frame i+1
    else
        return  the ranges ( 0 <= y <= c ) and ( 0 <= x <= r - B ) for frame i, and
                the ranges ( 0 <= y <= c ) and ( B <= x <= r ) for frame i+1
Else
    If ( LB < MBN i < FB )
        If ( MBN i+1 < LB )
            A = LB - MBN i+1,
            B = MBN i - LB,
            return  the ranges ( B <= x <= c ) and ( 0 <= y <= r - A ) for frame i, and
                    the ranges ( 0 <= x <= c - B ) and ( A <= y <= r ) for frame i+1
        If ( MBN i+1 > FB )
            A = MBN i+1 - FB,
            B = FB - MBN i,
            return  the ranges ( 0 <= x <= c - B ) and ( A <= y <= r ) for frame i, and
                    the ranges ( B <= x <= c ) and ( 0 <= y <= r - A ) for frame i+1
    If ( LB < MBN i+1 < FB )
        If ( MBN i < LB )
            A = LB - MBN i,
            B = MBN i+1 - LB,
            return  the ranges ( 0 <= x <= c - B ) and ( 0 <= y <= r - A ) for frame i, and
                    the ranges ( B <= x <= c ) and ( A <= y <= r ) for frame i+1
        If ( MBN i > FB )
            A = FB - MBN i+1,
            B = MBN i - FB,
            return  the ranges ( B <= x <= c ) and ( A <= y <= r ) for frame i, and
                    the ranges ( 0 <= x <= c - B ) and ( 0 <= y <= r - A ) for frame i+1
```

Figure 4: Algorithm for finding Comparable Regions

Step 4 : Now, we apply the object segmentation technique discussed in Section 2 to not the whole $frame_i$ and $frame_{i+1}$ but the part of them returned in the previous step

4 Experimental Results

We have applied the proposed scheme to the color videos of various kinds (i.e., TV commercials, movie, documentary, TV dramas, and animation) in our experiments. The video clips used for this study were originally digitized in AVI format at 30 frames/second. Their resolution is 160×120 pixels. To reduce computation time, we made our test video clips by extracting color frames from these originals at the rate of 3 frames/second.

Each video is decomposed into a number of shots using our SBD technique, and two consecutive candidate frames are selected accordingly for each shot as discussed in Section 3. These two frames are quantized from the original color depth of 16M to 64. From these two quantized frames, a difference image is computed, and object region is segmented as discussed in Section 2. Some examples are shown in 5.

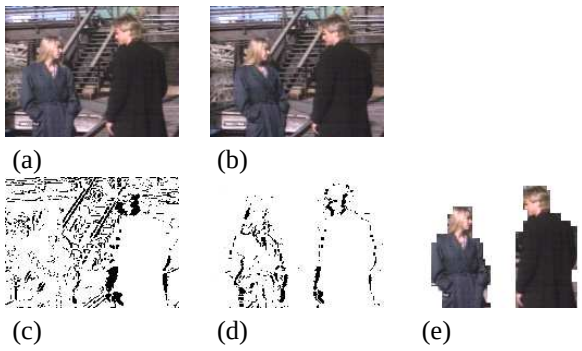

Figure 5: Key Objects Segmentation Result in a shot of 'TV Drama (Macgyver)' (a) Frame #27 (b) Frame #28 (c) Difference Image without background adjusting (d) Difference Image with background adjusting (e) Segmentation Result

Figure 5 is showing the frame adjusting example in which the came is panning (moving horizontally) 5 pixels left. As seen those figures, the results of the proposed scheme applied to the various kinds of videos are quite acceptable.

In fact, for the objective evaluation of the proposed scheme, we tried the simple pixel-based quality measure studied by Wollborn and Mech [7]. Their study suggested two ways which measure 'Spatial' and 'Temporal' accuracy. Spatial accuracy can be determined by the spatial distortion of an estimated binary video object mask at frame t, and defined as

$$d(O_t^{est}, O_t^{ref}) = \frac{\sum_{(x,y)} O_t^{est}(x,y) \oplus O_t^{ref}(x,y)}{\sum_{(x,y)} O_t^{ref}(x,y)} \quad (2)$$

where O_t^{est}, and O_t^{ref} are the reference and the estimated binary object masks at frame t, respectively, and $\oplus$ is the binary 'XOR' operation. Usually, the reference mask is manually segmented from the original sequence and the estimated mask is

actual result of the proposed scheme. We have measured the spatial accuracy by Equation (4) using two candidate frames per shot that are considered to find main object(s), and compared our scheme with the algorithm proposed in [9, 10]. In Figure 6, the top and the bottom curves which are represented as Without Manual Processing and With Manual Processing respectively are obtained by using the algorithm proposed in [9, 10] while the middle one is the result from our proposed scheme. The top line in Figure 6 is the result without the manual background edge deletion, the bottom line is the result with the manual background edge deletion. We see that the spatial accuracy of our scheme is very similar to the algorithm [9, 10] with manual processing. In other words, our scheme can achieve excellent accuracy without manual processing.

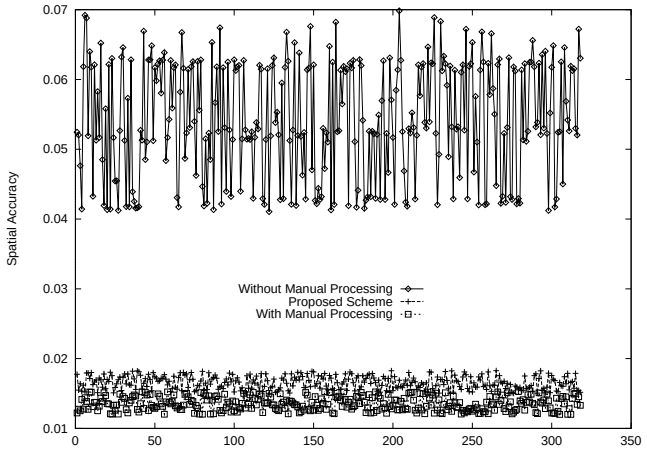

Figure 6: Spatial Accuracy Comparison

5 Concluding Remarks

The first contribution of the paper is that it introduces *key object(s)* extraction instead of *key frame(s)*. The existing schemes try to extract a key frame from a decomposed shot, but we attempt to extract key object(s) which is one step ahead. The other contribution is to propose a simple and efficient technique for object segmentation in video sequence using color quantization. It is easy to implement and fast to compute since there is no use of 'edge detection' or 'optical flow'. In addition, manual processing for initialization such as background edge deletion is not necessary because two frames are repositioned for maximum matching, which can cause minimum noise. Because of this frame repositioning, our scheme can be applied to the video sequence with moving background.

To assess the performance of the proposed technique, we evaluated it subjectively by showing the extracted results and objectively by comparing with the other technique using 'edge detection' throughout the experiments. The over all results indicate that our method can give better accuracy with substantially less cost.

References

[1] W. Xiong and J. C. Lee. Efficient scene change detection and camera motion. *Computer Vision and Image Understanding*, 71(2):166–181, August 1998.

[2] T. Zhang and C. Kuo. An integrated approach to multimodal media content analysis. In *Proc. of SPIE conf. on Storage and Retrieval for Media Databases 2000*, pages 506–517, San Jose, CA, Jan. 2000.

[3] K. Kupeev and Z. Sivan. An algorithm for efficient segmentation and selection of representative frames in video sequences. In *Proc. of SPIE conf. on Storage and Retrieval for Media Databases 2001*, pages 253–261, San Jose, CA, Jan. 2001.

[4] JungHwan Oh, Kien A. Hua, and Ning Liang. A content-based scene change detection and classification technique using background tracking. In *SPIE Conf. on Multimedia Computing and Networking 2000*, pages 254–265, San Jose, CA, Jan. 2000.

[5] JungHwan Oh and Kien A. Hua. An efficient and cost-effective technique for browsing and indexing large video databases. In *Proc. of 2000 ACM SIGMOD Intl. Conf. on Management of Data*, pages 415–426, Dallas, TX, May 2000.

[6] P Bouthemy and E. Francois. Motion segmentation and qualitative dynamic scene analysis from a image sequence. *International Journal of Computer Vision*, 10(2):157–182, 1993.

[7] R. Mech and M. Wollborn. A noise robust method for segmentation of moving objects in video sequence. In *Proc. of ICASSP'97, Vol.4*, pages 2657–2660, April 1997.

[8] A. Ekin, A. M. Tekalp, and R. Mehrotra. Object-based video description: From low level features to semantics. In *Proc. of SPIE conf. on Storage and Retrieval for Media Databases 2001*, pages 362–372, San Jose, CA, Jan. 2001.

[9] C. Kim and J. Hwang. Fast and robust moving object segmentation in video sequences. In *Proc. of IEEE Int'l Conf. on Image Processing(ICIP'99)*, pages 131–134, Kobe, Japan, Oct. 1999.

[10] C. Kim and J. Hwang. An integrated scheme for object-based video abstraction. In *Proc. of ACM Multimedia 2000)*, pages 303–311, LA, CA, Oct. 2000.

[11] P. J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. In *IEEE Transactions on Communications V COM-31*, pages 532–540, April 1983.