An Efficient Method for Detection of Key Objects in Video Shots with Camera Motions

JungHwan Oh JeongKyu Lee Sae Hwang [*]

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 extract *key objects* instead of key frames. In order to facilitate this process, 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 key objects 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 very promising.

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

1 Introduction

With the rapid advances in networking and coding technologies, video has become an essential part of many important applications such as digital libraries, distance learning, public information systems, electronic commerce, and entertainment. However, due to the enormous size of video files and their semantically rich contents, organizing and managing video as data are much more complex than manipulating text.

Video obtained from various types of sources is called a *video clip* with varying lengths of time lasting 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. There is a need for a new basic unit to handle video data. For the purposes of this scheme, 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.

Several research projects [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] have been performed to find these shot boundaries automatically, in other words, to parse a long video into a number of shots which are the basic units for video processing. In our previous works [11, 12, 13, 14, 15], we proposed the shot boundary detection (SBD) technique, which processes substantially less data since it uses only backgrounds instead of whole area of images. Also it is much less sensitive to the threshold values, and very effective in reducing incorrect detection. After a video is decomposed into shots, usually, the next step is to extract a *key frame* from each shot for content based summarizing and understanding. In this paper, we extract *key objects*

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

instead of key frame, which can be one of the basic materials for the content-based video analysis. In order to do the extraction of key objects, we extend our previous technique for SBD to select two candidate frames out of each shot, and propose a new framework to segment key objects from those frames (see Figure 1).

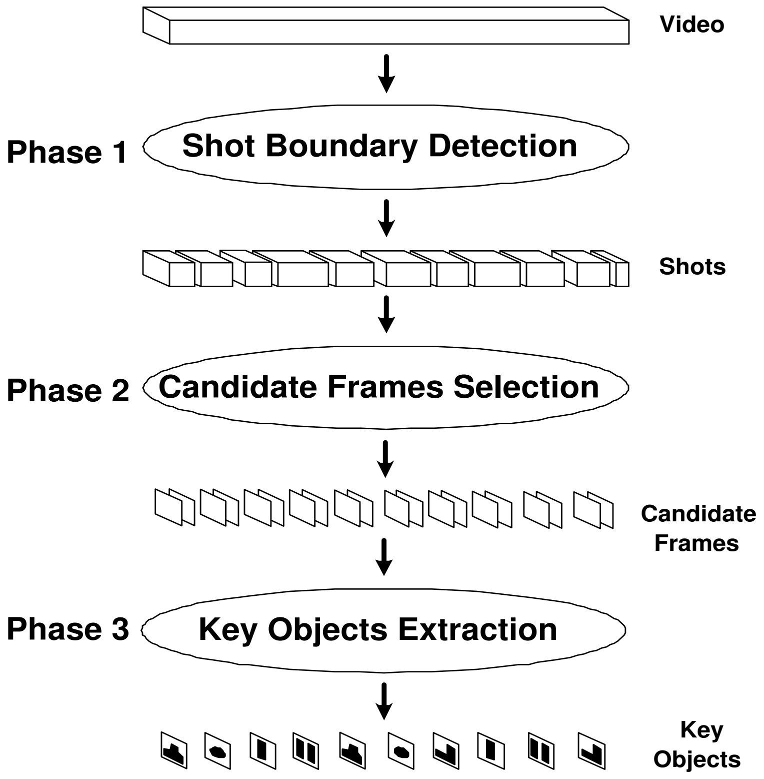


Figure 1: Procedure for Key Objects Extraction

This key object segmentation provides an abstraction of a long video and supports one of MPEG-4/MPEG-7 standards, which is *Video Object Layer* (VOL). The early video coding standards (i.e., H.261, H.263, MPEG-1, or MPEG-2) fall short for high-level interpretation and understanding of actual video contents. To support these content-based functionalities, the MPEG-4 [16] introduces the concept of VOL. The main purpose of VOL is to provide encoding of video sequences which have been segmented based on video contents, and to allow flexible and separate reconstruction and manipulation of video contents at the decoding. Therefore, the segmentation of objects from video sequences, which is described as a partition of semantically meaningful objects from background, plays an important role for the successful use of MPEG-4/MPEG-7.

A large number of researches concerning video objects have been done [17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28]. A traditional technique to object segmentation is to employ *optical flow* to select the segmented regions [29, 30], which is very inaccurate at the detection of object motion and its corresponding boundaries. In order to improve the motion estimation, which can ameliorate the accuracy, *Change Detection* algorithms have been developed [31, 32, 33, 34]. In these algorithms, the difference image of gray value between two corresponding frames is considered. The local gray difference value is computed for a small window which slides over the difference image. At each location, this value is compared with a threshold to decide whether the center pixel of the current window is marked as changed or not. The critical point is, obviously, how to decide optimal threshold since it often suffers from noise due to global thresholding. Several methods based on motion models are presented in [20, 21, 19, 23, 27]. A few papers [35, 25, 28] have studied a model based on contour. Recently, an algorithm based on edge detection using a traditional edge detector has been published [36, 22, 26]. This edge detection algorithm has the capabilities to reduce some noise, that is, the pixels not belonging to target objects can be eliminated since two edge maps generated from two corresponding gray images are compared. However, for most general video sequences, this algorithm requires manual processing in order to get a background edge map in the initial stage. Also, it is not well suited for the video sequence with some camera motions, in which the background is not fixed.

To address these issues, we first propose a technique to segment objects from a video sequence using *color quantization*, and apply the proposed technique to find key objects in a shot by using the extension of our previous SBD technique. The entire procedure is shown in Figure 1, and the advantages of the proposed framework can be summarized as follows:

- Segmentation algorithm is efficient and is capable of producing quick results 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 the background is not fixed, because it has a function to adjust the background position.

The remainder of this paper is organized as follows. In Section 2, we discuss the basic idea of object segmentation in video sequences using color quantization. The extension of our previous SBD work and its application to find key objects 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 Segmentation of Key Objects

In this Section, Phase 3 (Key Objects 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 algorithm to extract object edge map starts with color quantization of each original color frame. For example, if the color depth of original frame is $16M(2^{24})$, that of quantized frame can be $2M(2^{21})$, $256K(2^{18})$, $32K(2^{15})$, $4K(2^{12})$, $512(2^9)$, $64(2^6)$, and $8(2^3)$ since a pixel needs three values for red, green and blue. The quantized frame can be computed using the following formula.

$$F_n = \Lambda(f_n) \tag{1}$$

where F_n and f_n are quantized and original frames respectively, and the Λ is the quantization function that takes the pixel values from original frame as input values, and the outputs are recalculated pixel values. Thus, the above equation can be rewritten as

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

where the color depths of original and quantized frames are 2^k and 2^m respectively, and $d = \frac{k-m}{3}$. r, g and b are the RGB pixel values from original frame and R, G and B are the RGB pixel values from quantized frame. Figure 2 shows the examples of this color quantization. Figure 2 (a) is an original frame from a TV drama, and the other (b), (c), and (d) are the quantized frames with different color depths. As seen in this figure, the quantized frame with the minimum color depth (8) can still capture the semantics of the original. Therefore, either the color depths 8, or 64 is sufficient for the purposes of object segmentation.

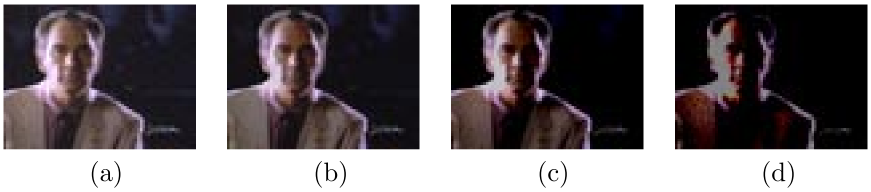


Figure 2: (a) Original Frame #1584(Color Depth=16M) (b) Quantized Frame(Color Depth=512) (c) Quantized Frame(Color Depth=64) (d) Quantized Frame(Color Depth=8)

2.2 Object Edge Extraction

Figure 3 (a) and (b) show two consecutive frames in a video clip (TV drama), which are quantized to the color depth of 64 using the above equation (2) where $d = 6$ since $k = 24$ and $m = 6$. From those two frames, we can find the difference image as shown in Figure 3 (c). The computation is very simple. If two pixels in the same position for 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. For the purpose of comparison, we give Figure 3 (d), which is the same difference image from the two original frames without any quantization, and has a great deal of noise. The existing techniques discussed in Section 1 ultimately attempted to obtain the edge map without noise such as Figure 3 (c) by using 'optical flow', 'motion or contour modeling' or 'edge detection' [36, 22, 26]. For example, the edge detection (Canny edge detector [37]) was applied to Figure 3 (d) to obtain Figure 3 (c). In other words, by using a simple 'color quantization', we can get the same result, and save a great deal of computational cost.

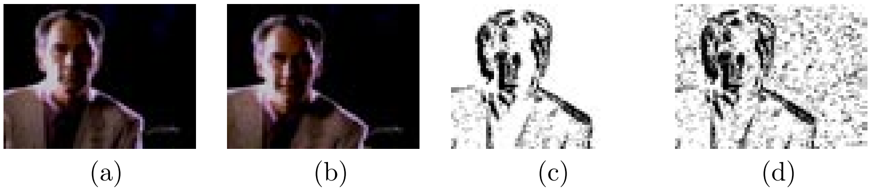


Figure 3: (a) Quantized Frame #1584 (b) Quantized Frame #1585 (c) Object Edge Map from quantized frames (d) Object Edge Map from original frames

2.3 Object Region Selection

After we obtain the object edge map such as in Figure 3 (c), the next task is to select an object region. The existing methods [36, 22, 26] scan the edge map horizontally and vertically, and try to find the first and the last edge points(pixels) for each row and column of pixels. They call the first, the last and their inside points for each row as 'horizontal candidates', and those for each column as 'vertical candidates'. Some methods use the logical-OR operation on horizontal and vertical candidates, while the others use the logical-AND. Figure 4 shows the two different results about selecting object region with logical-OR and logical-AND from the object edge map in Figure 3 (c). As seen in this figure, neither 'OR' nor 'AND' produces a satisfactory outcome.

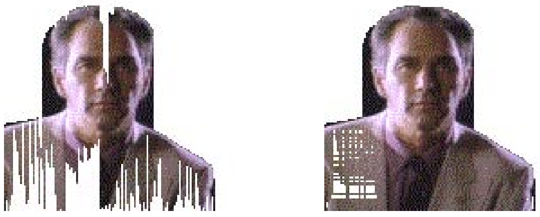

(a) (b)

Figure 4: (a) Selected Object Region with AND (b) Selected Object Region with OR

The problem arises from the connectivity of edge points. If some points in the edge are missing (which is a common occurrence in edge extraction), it cannot provide optimal results. Since we cannot guarantee that points are not missing in all cases of edge extraction, we approach this problem from a different angle 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 from the previous subsection is divided into a number of equivalent sized blocks. The usual block size used in JPEG or MPEG is 8×8 pixels, but to add more accuracy, 4×4 or 2×2 pixel blocks can be used in our scheme.

Step 2 : Each block is examined to determine the number of edge points. If a block has more than a certain number of points, it is considered to be in an object region. The threshold value depends on the block size. In our experiments, we used 4×4 pixels as the 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 an object region. These blocks are usually located inside an object region. To select these blocks as an object region, we use a simple algorithm which is illustrated in Figure 5 (a). As seen in the figure, to decide whether a center block (which has not enough points) is included in an object region, the following steps should be taken. First, we need to check its 8 neighboring blocks. If any 5 out of these 8 blocks belong to an object region, that is, they (the 5 blocks) are already selected as the object region in the previous step, the center block can be marked as an object region even if it does not have enough edge points. When a center block is positioned in the boundary of a frame, if any 3 out of 5 neighbors belong to the object region, the center block can be marked as an object region. Step 3 is performed recursively until no more blocks are added to an object region.

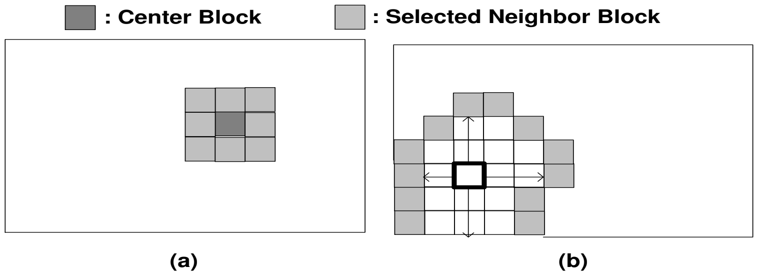


Figure 5: (a) Step. 3 (b) Step. 4

Step 4 : Step 3 can select most of the blocks inside an object region. However, any block which is not neighboring any selected block cannot be selected as an object region using Step 3 even if it is in an object region. Therefore, for each block that is not selected as the object region, we check all four directions as illustrated in Figure 5 (b). If a selected block is met at all four directions, the block is selected as an object region.

The complete process of object segmentation is illustrated in Figure 6.

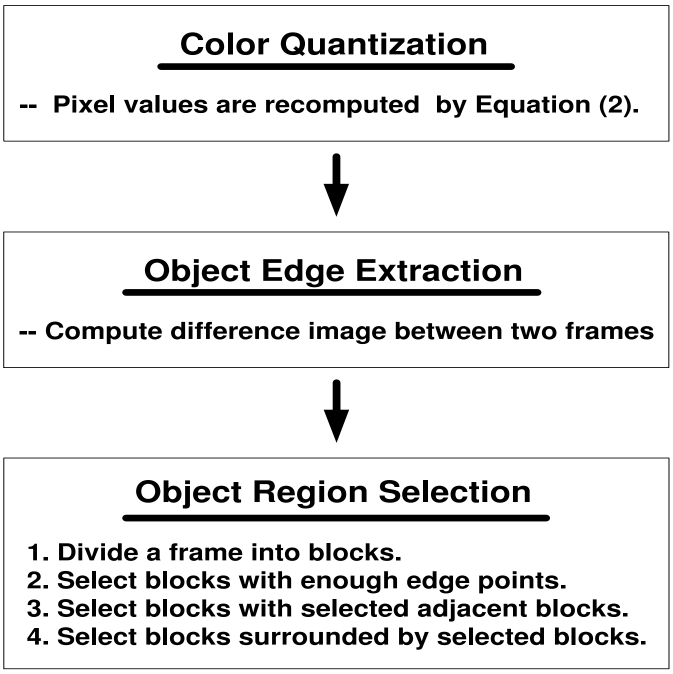

Color Quantization
-- Pixel values are recomputed by Equation (2).
Object Edge Extraction
-- Compute difference image between two frames
Object Region Selection
1. Divide a frame into blocks.
2. Select blocks with enough edge points.
3. Select blocks with selected adjacent blocks.
4. Select blocks surrounded by selected blocks.

Figure 6: Procedure for Object Segmentation

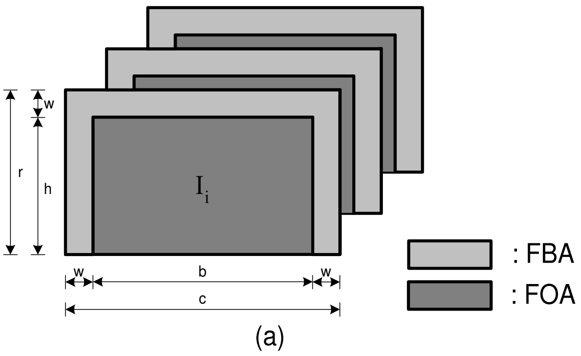
3 Shot Boundary Detection and Candidate Frames Selection

Phase 1 and 2 in Figure 1 are explained in this Section. We first present our previous works [11, 12, 13] about SBD to make this paper self-contained. We then discuss the extension of our SBD work and the process of selecting two candidate frames for each shot.

3.1 A Camera Tracking Approach to SBD and Its Extension

Since a shot is made from a single camera operation, tracking the camera motion is the most direct way to identify shot boundaries. This can be achieved by tracking the background areas of video frames. By comparing the background areas of two consecutive frames, we can tell if they belong to a same shot. This new approach has been referred to as the *Background Tracking* (BGT) technique in our previous papers [11, 12, 13].

In BGT, we define a *fixed background area* (FBA) for all frames as illustrated by the lightly shaded areas in Figure 7 (a). This ⊓-shaped FBA has three parts - two vertical columns and one horizontal bar. We can track the camera motion in different directions by comparing these three components as follows:



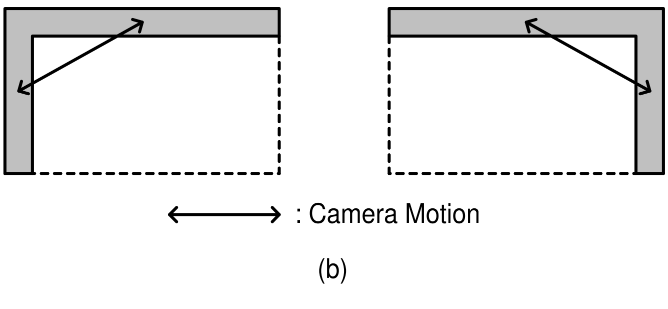


Figure 7: Background and Foreground Areas

- We can detect horizontal camera motions by searching for similar parts in the horizontal bars of two consecutive FBAs.
- We can detect vertical camera motions by searching for similar parts in the corresponding columns of two consecutive FBAs.

- By looking for similar parts in the left column of one FBA and the horizontal bar of the next FBA, we can track camera motions in one diagonal direction. Similarly, camera motions along the other diagonal direction can be detected by comparing the horizontal bar of one FBA against the right column of the next FBA. These two strategies are illustrated in Figure 7 (b).

We note that the bottom part of a frame is not considered as part of the background area since the bottom of an image is typically part of some objects in the foreground.

To make the background comparison more efficient, we rotate the two vertical columns of each ⊓-shaped FBA outward to form a *transformed background area* (TBA) as illustrated in Figure 8. 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* [38]. The idea of 'Gaussian Pyramid' was originally introduced to reduce an image to a smaller size. We use this technique to reduce a two-dimensional TBA into a single line of pixels (called a *signature*) and eventually a single pixel (called a *sign*). This process is illustrated in Figure 9. It shows a 13×5 pixel TBA being reduced in multiple steps. We note that this rather small TBA is only illustrative. We will discuss how to determine the area of a TBA shortly. In Figure 9, the five pixels in each column are first reduced to one pixel to make a single line of 13 pixels, which is used as the signature. This signature is further reduced to a *sign* (one-pixel) denoted by $sign_i^{BA}$. The superscript and subscript indicate that this is the sign of the *background area* of a $frame_i$. The term 'sign' is used to indicate that it is an abstracted form of a signature. The complexity of this reduction procedure is $O(2^{\log(m+1)})$ or $O(m)$, where m is the number of pixels involved.

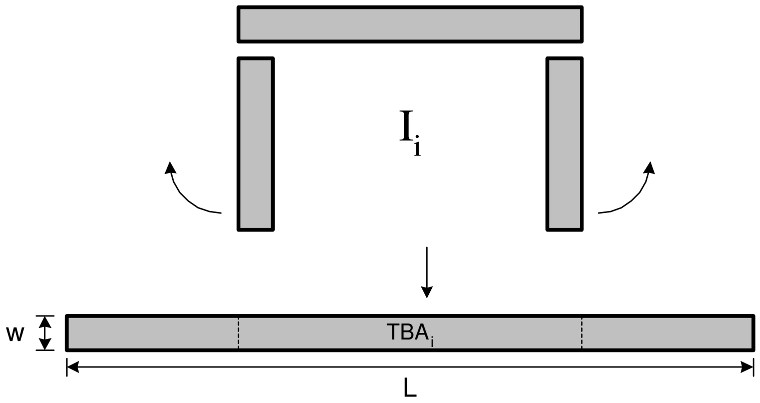


Figure 8: Construction of TBA

We use the signs and the signatures to detect shot boundaries as illustrated in Figure 10. The first two stages are quick-and-dirty tests used to quickly eliminate the easy cases. Stage 1 does not require the computation of signature and sign. In this stage, we compare each pixel of one FBA against the corresponding pixel and its neighboring pixels in the other FBA. If the difference is less than 10%, the two frames are in the same shot. If this test fails (i.e., we cannot determine if they are in the same shot), we proceed to Stage 2 to compute and compare the two signs. This test is helpful since the pixel-based comparison in Stage 1 could not recognize the similarity of the two frames if things are moving rapidly in the background (e.g., the camera is following a racing car). If the two signs differ by less than 10%, the two frames are considered to be in the same shot.

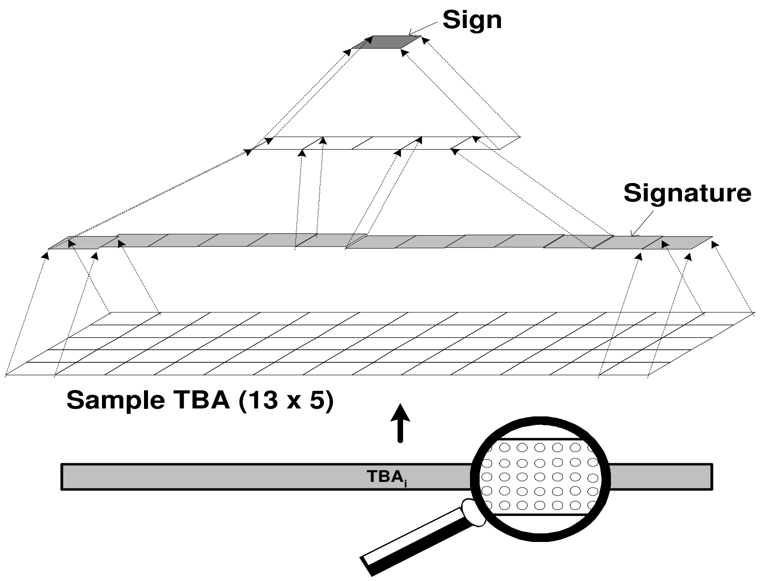


Figure 9: Computation of Signature and Sign

Only when the first two tests fail, we need to track the background in Stage 3 by shifting the two signatures of the two frames under test toward each other one pixel at a time (see Figure 11). For each shift, we compare the pixels in the overlapping region to determine the longest run of matching pixels. A running maximum is maintained for these matching scores. In the end, this maximum value indicates how much the two images share the common background. If the score is larger than a certain threshold, the two video frames are determined to be in a same shot.

The advantages of the camera-tracking approach are as follows:

- The frames in the same shot are generally very similar, so we usually need to perform only the test in Stage 1. Since the test is applied to only about 20% of the whole area of image, the proposed technique is about four times faster than existing SBD methods. We note that any existing SBD scheme can be used for our Stage 1.
- Although we adopt an existing scheme for Stage 1, it is less sensitive to threshold values compared to the existing techniques. This is due to the fact that we focus on the background areas, which are much less dynamic compared to the entire area of image. This characteristic makes it much easier for us to determine threshold values that are good for a wide range of background situations.
- If Stage 1 fails to recognize the similarity between two consecutive frames, we still have two more tests. In particular, since Stage 3 is based on comparing runs of pixels, not corresponding pixels, this strategy is much less sensitive to object motion, and therefore thresholds.

We define a *fixed object area* (FOA) as the foreground area of a video frame, where most primary objects appear. This area is illustrated in Figure 7 as the darkly shaded region of a video frame. 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 by using the Gaussian Pyramid as in computing $sign_i^{BA}$.

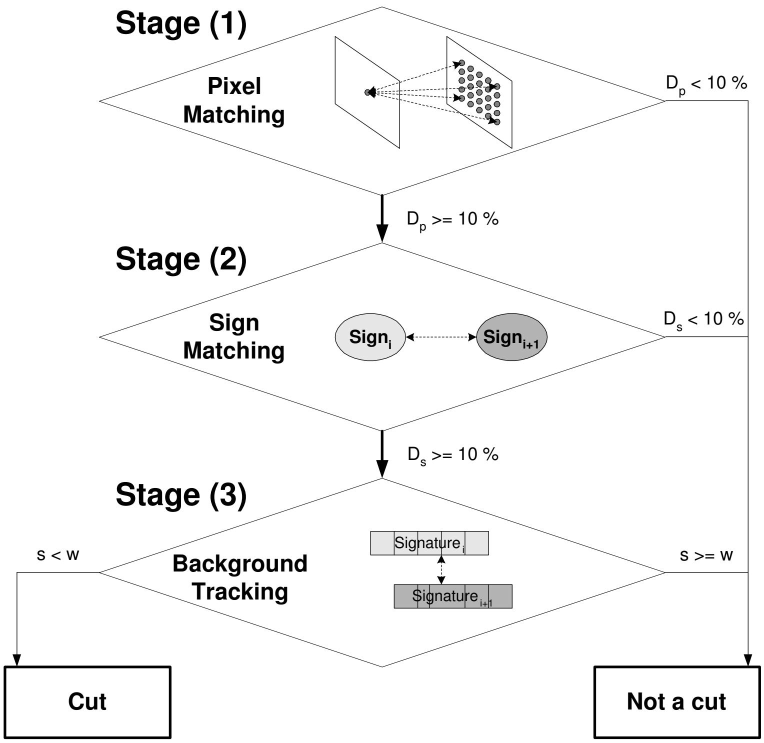


Figure 10: Shot Boundary Detection Procedure

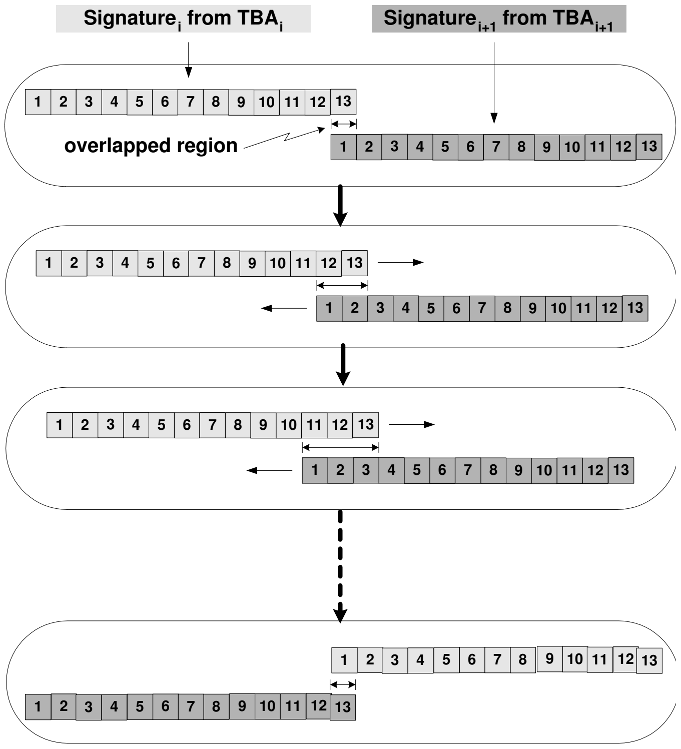

Signature$_i$ from TBA$_i$
Signature$_{i+1}$ from TBA$_{i+1}$
overlapped region
1 2 3 4 5 6 7 8 9 10 11 12 13

Figure 11: Background Tracking Technique

Given r and c as the dimensions of the video frame (see Figure 7), we discuss the procedure for determining the dimensions of TBA and FOA as follows. Let the dimensions of FOA be h and b, and those of TBA be w and L as illustrated in Figure 7. We first estimate these parameters as h', b', w', and L', respectively. We choose w' to be 10% of the width of the video frame, i.e., $w' = \lfloor \frac{c}{10} \rfloor$. This value was determined empirically using our video clips. Our experiments show that this value of w' results in TBAs and FOAs which cover the background and the foreground areas, respectively, very well. Knowing w', we can compute the other estimates as follows: $b' = c - 2 \cdot w'$, $h' = r - w'$, and $L' = c + 2 \cdot h'$.

In order to apply the Gaussian Pyramid technique, the dimensions of TBA and FOA must be in the *size set* $\{1, 5, 13, 29, 61, 125, ...\}$. This is due to the fact that this technique reduces five pixels to one pixel, 13 pixels to five, 29 pixels to 13, and so on. In general, the jth element (s_j) in this size set is computed as follows:

$$s_j = 1 + \sum_{i=2}^{j} 2^i \quad for \quad j = 1, 2, 3, ... \tag{3}$$

Using this size set, the proper value for w is the value in the size set, which is nearest to w'. This nearest number can be determined as follows. We first compute $j = 2 + \left\lfloor \log_2 (\frac{w'+3}{6}) \right\rfloor$. Substituting this value of j into Equation (3) gives us the desired value for w. We can similarly compute L, h, and b. This approximation scheme is illustrated in Table 1. As an example, let $c = 160$. We have $w' = \lfloor \frac{160}{10} \rfloor = 16$. The corresponding j value is 3. Substituting j into Equation (3) gives us 13 as the proper value for w.

h', b', w' or L'	Nearest value	h, b, w or L
1, 2	1	1
3, 4, ..., 8	5	5
9, 10, ..., 20	13	13
21, 22, ..., 44	29	29
45, 46, ..., 92	61	61
...	...	...

Table 1: The dimensions using the nearest value from the size set.

3.2 Two Candidate Frames Selection and Key Objects Extraction

Let us assume that a given video clip is decomposed into a number of shots using our BGT technique discussed above. To extract key objects from these shots using the object segmentation technique described in Section 2 optimally, we need two consecutive frames in which there is no change in the background, and small changes in the object area. These are necessary requirements since there would be a lot of noise in the background region if the camera is not still. Consequently, it is possible to segment false objects. The same scenario is feasible in the case that the object motion is very large. Using our SBD technique and its extension discussed above, we can verify these changes in the background and object regions.

The values of $sign^{BA}$ and $sign^{OA}$ indicate the average color values of the background and object areas for each frame. We can compute the differences of $sign^{BA}$s and $sign^{OA}$s between two frames ($frame_i$ and $frame_{i+1}$). They are called $DBA_{i,i+1}$ and $DOA_{i,i+1}$ for $sign^{BA}$ and $sign^{OA}$ respectively, and formulated as follows.

$$\begin{aligned} &DBA_{i,i+1} \\ &\quad = sign_{i+1}^{BA} - sign_i^{BA} \\ &\quad = \sqrt{(R_{i+1}^{BA} - R_i^{BA})^2 + (G_{i+1}^{BA} - G_i^{BA})^2 + (B_{i+1}^{BA} - B_i^{BA})^2} \end{aligned} \tag{4}$$

$$\begin{aligned} &DOA_{i,i+1} \\ &\quad = sign_{i+1}^{OA} - sign_i^{OA} \\ &\quad = \sqrt{(R_{i+1}^{OA} - R_i^{OA})^2 + (G_{i+1}^{OA} - G_i^{OA})^2 + (B_{i+1}^{OA} - B_i^{OA})^2} \end{aligned} \tag{5}$$

where R_i^{BA}, G_i^{BA} and B_i^{BA} are RGB values for $sign_i^{BA}$, and R_i^{OA}, G_i^{OA} and B_i^{OA} are for $sign_i^{OA}$. If $DBA_{i,i+1}$ is zero, we can verify that no change occurs in background because the average of the two background colors are the 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 $n-1$ of DBAs using Equation (4) and find a frame pair in which the value of DBA is zero. If there is more than one candidate, then compute DOAs by Equation (5) and find the smallest (but not zero) 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 a tiebreaking rule for all cases in which there are more than one candidate). If we cannot find a frame pair with zero value of DBA, we go to Step 2.

Step 2 : Since we cannot find a frame pair with zero value of DBA, we select a frame pair with the smallest value of DBA. Assume that the pair consists of $frame_i$ and $frame_{i+1}$. Here, we extend our background tracking method in Figure 11 so as to find not just how large the maximum matching score is but also where the maximum matching occurs. We compare $Signature_i$ with $Signature_{i+1}$ by the background tracking method discussed above. At this time, we focus on the location of maximum matching as well as the maximum matching score. Three cases are possible for the matching locations as seen in Figure 12. For convenience, we divide an FBA into 18 blocks in which blocks #1 through #6 are in the left horizontal bar, blocks #7 through #12 are in the vertical bar, and blocks #13 through #18 are in the right horizontal bar.

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 in three cases as follows. For simplicity, we assume that there is only one matching block.

- The first case is Figure 12 (a) in which the matching locations are only in the vertical bars. This case happens by tilting the camera (including vertical movement of camera itself). In the left horizontal bar, the block #3 of $Signature_i$ matches the block #1 of $Signature_{i+1}$ in which $1 - 3 = -2$, that is two blocks down, which means that the camera is tilting two blocks up. In the right horizontal bar, the block #16 of $Signature_i$ matches the block #18 of $Signature_{i+1}$ in which

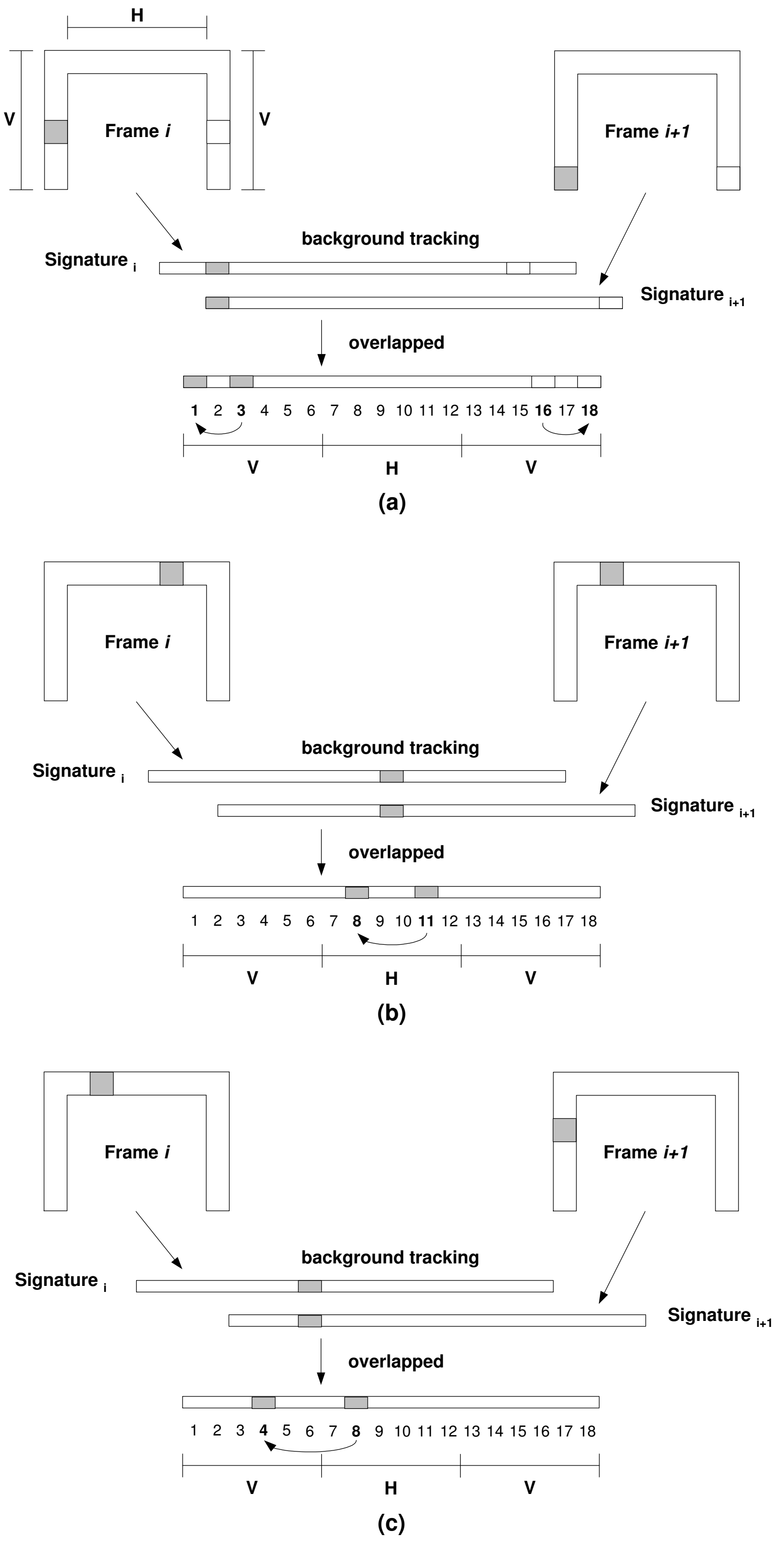


Figure 12: (a) Matching in Vertical bars (b) Matching in Horizontal bar (c) Matching in Horizontal and Vertical bars

$18 - 16 = 2$, that is also two blocks down, which means that the camera is tilting two blocks up. $MBN_i = 3$ and $MBN_{i+1} = 1$

- The second case is Figure 12 (b) in which the matching location is only in the horizontal bar. This case happens by panning the camera (including horizontal movement of camera itself). The block #8 of $Signature_i$ matches the block #11 of $Signature_{i+1}$ in which $11 - 8 = 3$, that is three blocks left, which means that the camera is panning three blocks right. $MBN_i = 8$ and $MBN_{i+1} = 11$
- The third case is Figure 12 (c) in which the matching locations are both in horizontal and vertical bars. This case happens by the combination of panning and tilting the camera (including horizontal and vertical movements of camera itself). The block #8 of $Signature_i$ matches the block #4 of $Signature_{i+1}$ in which $4 - 6 = -2$ for the vertical and $6 - 8 = -2$ for the horizontal, that is two blocks left and two block down, which means that the camera is panning two blocks right and tilting two blocks up. To distinguish between vertical and horizontal movements, we use the block #6 because it is the last block in the left vertical bar. $MBN_i = 8$ and $MBN_{i+1} = 4$

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 13. Assume that r is the number of pixels in a row, c is the number of pixels in a column, LB is the last block in the left vertical bar, and FB is the first block in the right vertical bar. In our case, LB is 6, and FB is 13.

Step 4 : Now, we apply the object segmentation technique discussed in Section 2 to not the entire area of frames ($frame_i$ and $frame_{i+1}$) but the part of them returned in the previous step as shown in Figure 14.

4 Experimental Results

We have applied the proposed scheme to the color videos of various types (i.e, TV commercials, movies, documentaries, TV dramas, and animations) in our experiments. The video clips used for this study were originally digitized in AVI format at 30 frames per second with a resolution of 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 per second. The details of this test set are summarized in Table 2.

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 an object region is segmented as discussed in Section 2. Some examples are shown in Figure 15, 16 and 17.

Figure 15 and 16 are the examples which do not need the frame adjustment discussed in Step 2, 3, and 4 of Section 3.2. As seen in these figures, our scheme can detect and segment the object of each shot very accurately. Figure 17 is an example of frame adjusting where the camera is panning (moving horizontally) 5 pixels left. As seen in 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 [34]. Their study suggested two ways which measure 'Spatial' and 'Temporal' accuracy. Spatial accuracy can be determined by

```
Algorithm to find Comparable Regions

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 13: Algorithm to find Comparable Regions in Step 3.

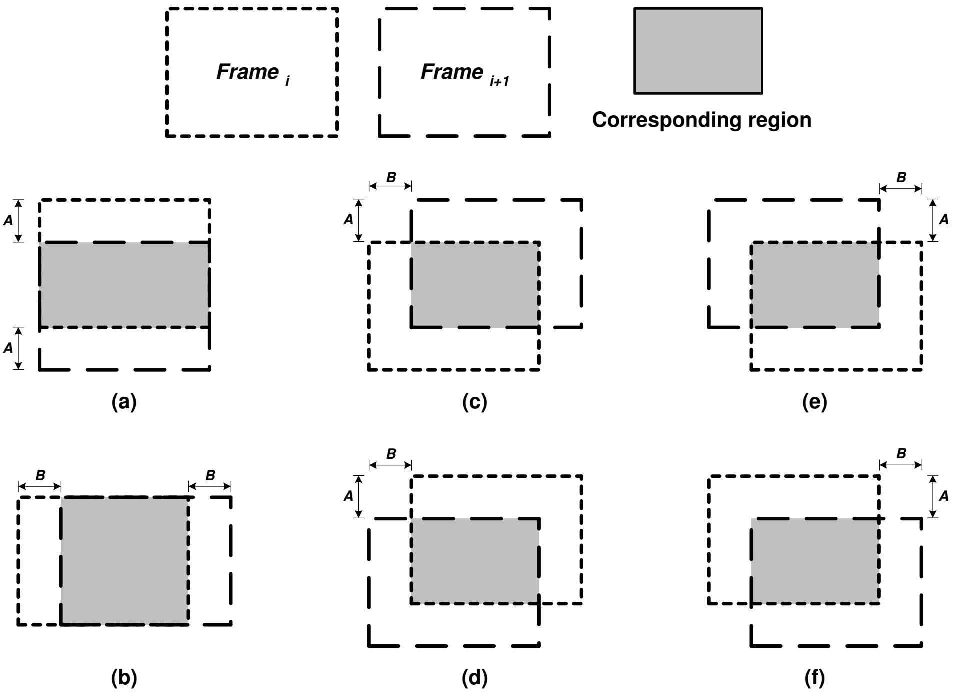


Figure 14: (a) Matching in Vertical bars (b) Matching in Horizontal bar (c)-(f) Matching in Horizontal and Vertical bars

Name	Duration (min:sec)	# of shots
Silk Stalkings (TV drama)	5 : 24	47
ATF (action movie)	3 : 36	74
TV Commercials	2 : 25	81
Bug's Life (animation)	3 : 09	24
For All Mankind (documentary)	3 : 16	38
Macgyver (TV drama)	6 : 19	54
Total	24 : 09	318

Table 2: Test Video Clips

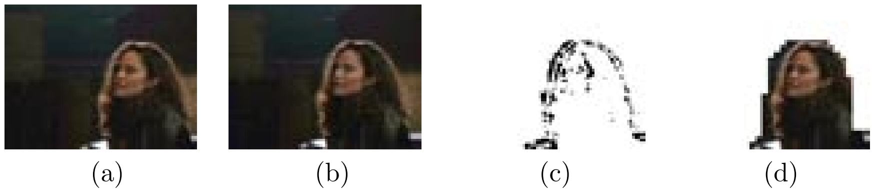


Figure 15: Key Object Segmentation Result in a shot of 'ATF (action movie)' (a) Frame #1948 (b) Frame #1949 (c) Difference Image (d) Segmentation Result

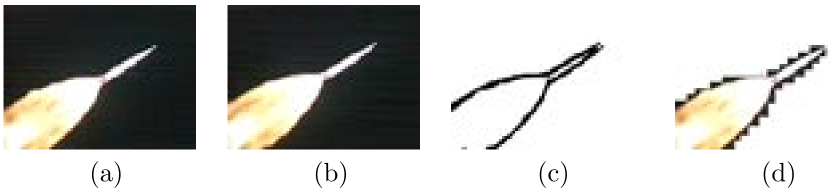


Figure 16: Key Object Segmentation Result in a shot of 'For All Mankind (documentary)' (a) Frame #1404 (b) Frame #1405 (c) Difference Image (d) Segmentation Result

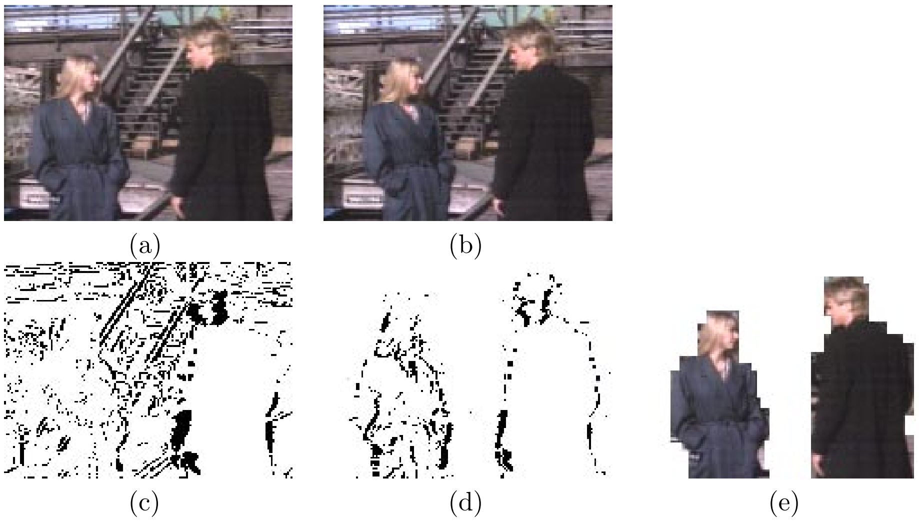


Figure 17: Key Objects Segmentation Result in a shot of 'TV Drama (Macgyver)' (a) Frame #27 (b) Frame #28 (c) Difference Image without background adjusting (d) Difference Image with background adjusting (e) Segmentation Result

the spatial distortion of an estimated binary video object mask at frame t, and defined as

$$d(O_t^{est}, O_t^{ref}) = \frac{\sum_{(x,y)} O_t^{est}(x,y) \oplus O_t^{ref}(x,y)}{\sum_{(x,y)} O_t^{ref}(x,y)} \tag{6}$$

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 the actual result of the proposed scheme. We have measured the spatial accuracy by Equation (6) using two candidate frames per shot considered to find main objects, and compared our scheme with the algorithm proposed in [36, 22]. In Figure 18, 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 [36, 22] while the middle one is the result from our proposed scheme. The top line in Figure 18 is the result without the manual background edge detection, the bottom line is the result with the manual background edge detection. We see that the spatial accuracy is the same for our scheme as for the method with the manual processing discussed in the literature [36, 22]. In other words, our scheme can achieve excellent accuracy without manual processing.

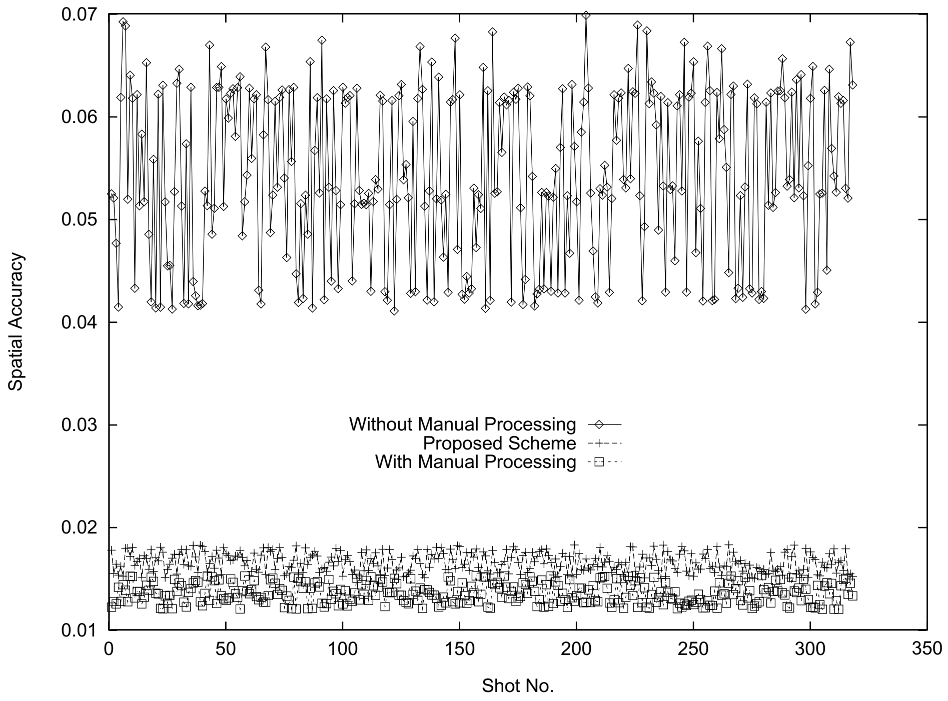


Figure 18: Spatial Accuracy Comparison

5 Concluding Remarks

The first contribution of this paper is that it introduces *key objects* extraction from shots instead of *key frames.* The existing schemes try to extract a key frame from a decomposed shot, but we try to extract key objects, which is one step ahead. The other contribution is to propose a simple and efficient technique for object segmentation in a video sequence using the color quantization. It is easy to implement and fast to compute since 'edge detection' or 'optical flow' is not used. In addition, manual processing for initialization, such as background edge detection is not necessary because two frames are repositioned for maximum matching, which results in a minimal amount of noise. Because of the frame repositioning, our scheme can be applied to the video sequences with a moving background. To assess the performance of the proposed technique, we evaluated it subjectively by showing the extracting results and objectively by comparing our results with the other technique using 'edge detection' throughout the experiments. The overall results indicate that our method can provide excellent accuracy with much less cost.

To obtain an edge map of object from two consecutive frames, the proposed technique needs an assumption that there must be certain small movements (differences) of objects between two consecutive frames. However, we cannot guarantee the assumption when we select two consecutive frames without background changes as we discussed in the paper. We will further study 'how much is enough movement?' and 'how can we get them?'.

References

[1] B. Truong, C. Dorai, and S. Venkatesh. New enhancements to cut, fade and dissolve detection processes in video segmentation. In *Proc. of ACM Multimedia 2000*, pages 219–227, LA, CA, Oct. 2000.

[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] R. Lienhart. Reliable dissolve detection. In *Proc. of SPIE conf. on Storage and Retrieval for Media Databases 2001*, pages 219–230, San Jose, CA, Jan. 2001.

[4] J. Nam and A. Tewfik. Wipe transition detection using polynomial interpolation. In *Proc. of SPIE conf. on Storage and Retrieval for Media Databases 2001*, pages 231–241, San Jose, CA, Jan. 2001.

[5] S. Han and I. Kweon. Shot detection combining bayesian and structural information. In *Proc. of SPIE conf. on Storage and Retrieval for Media Databases 2001*, pages 509–516, San Jose, CA, Jan. 2001.

[6] L. Zhao, W. Qi, Y. Wang, S. Yang, and H. Zhang. Video shot grouping using best-first model merging. In *Proc. of SPIE conf. on Storage and Retrieval for Media Databases 2001*, pages 262–269, San Jose, CA, Jan. 2001.

[7] 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.

[8] Y. Kwon, C. Song, and I. Kim. A new approach for high level video structuring. In *Proc. of 2000 IEEE Int'l Conf. on Multimedia and Expo(ICME)*, pages 773–776, New York, NY, July 2000.

[9] J. Huang, Z. Liu, and Y. Wang. Joint video scene segmentation and classification based on hidden markov model. In *Proc. of 2000 IEEE Int'l Conf. on Multimedia and Expo(ICME)*, pages 1551–1554, New York, NY, July 2000.

[10] Z. Lei, W. Chou, J. Zhong, and C. Lee. Video segmentation using spatial and temporal statistical analysis method. In *Proc. of 2000 IEEE Int'l Conf. on Multimedia and Expo(ICME)*, pages 1527–1530, New York, NY, July 2000.

[11] 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.

[12] 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.

[13] JungHwan Oh and Kien A. Hua. An efficient technique for summarizing videos using visual contents. In *Proc. IEEE International Conference on Multimedia and Expo.*, pages 1167–1179, New York, NY, August 2000.

[14] Kien A. Hua and JungHwan Oh. Detecting video shot boundaries up to 16 times faster. In *The 8th ACM International Multimedia Conference (ACM Multimedia 2000)*, pages 385–387, LA, CA, Oct. 2000.

[15] Kien A. Hua and JungHwan Oh. Non-linear approach to shot boundary detection. In *SPIE Conf. on Multimedia Computing and Networking 2001*, pages 1–12, San Jose, CA, Jan. 2001.

[16] T. Sikora. The mpeg-4 video standard verification model. *IEEE Transaction Circuits System Video Technology*, 7:19–31, Feb. 1997.

[17] C. Toklu, T. Fischer, and Shih-Ping Liou. Dynamic marker for collaborative discussion on video content. In *Proc. of SPIE conf. on Storage and Retrieval for Media Databases 2000*, pages 369–377, San Jose, CA, Jan. 2000.

[18] T. Yatabe, H. Kawasaki, H. Mo, and M. Sakauchi. Multi layer video object database based on interactive annotation and its application. In *Proc. of 2000 IEEE Int'l Conf. on Multimedia and Expo(ICME)*, pages 911–914, New York, NY, July 2000.

[19] K. S. Ntalianis, N. D. Doulamis, A. D. Doulamis, and S. D. Kollias. A feature point based scheme for unsupervised video object segmentation in stereoscopic video sequences. In *Proc. of 2000 IEEE Int'l Conf. on Multimedia and Expo(ICME)*, pages 1543–1546, New York, NY, July 2000.

[20] J. Y. Zhou, E. P. Ong, and C. C. Ko. Video object segmentation and tracking for content-based video coding. In *Proc. of 2000 IEEE Int'l Conf. on Multimedia and Expo(ICME)*, pages 1555–1558, New York, NY, July 2000.

[21] W. Chen and S. Chang. Motion trajectory matching of video objects. In *Proc. of SPIE conf. on Storage and Retrieval for Media Databases 2000*, pages 544–553, San Jose, CA, Jan. 2000.

[22] 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.

[23] H. Eng and K. Ma. Spatiotemporal segmentation of moving video objects over mpeg compressed domain. In *Proc. of 2000 IEEE Int'l Conf. on Multimedia and Expo(ICME)*, pages 1531–1534, New York, NY, July 2000.

[24] C. Toklu, T. Fischer, and Shih-Ping Liou. Dynamic object markers in a collaborative environment for video content discussion. In *Proc. of 2000 IEEE Int'l Conf. on Multimedia and Expo(ICME)*, pages 57–60, New York, NY, July 2000.

[25] Z. Lu and W. A. Pearlman. Semi-automatic semantic video object extraction by active contour model. In *Proc. of 2000 IEEE Int'l Conf. on Multimedia and Expo(ICME)*, pages 645–648, New York, NY, July 2000.

[26] J. Fan, D. Yau, M. Hacid, and A. Elmagarmid. Model-based semantic object extraction for content-based video representation and indexing. In *Proc. of SPIE conf. on Storage and Retrieval for Media Databases 2001*, pages 523–535, San Jose, CA, Jan. 2001.

[27] 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.

[28] S. Richter, G. Kuhne, and O. Schuster. Contour-based classification of video objects. In *Proc. of SPIE conf. on Storage and Retrieval for Media Databases 2001*, pages 608–618, San Jose, CA, Jan. 2001.

[29] G. Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-7:384–401, July 1985.

[30] 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.

[31] M. Hotter and R. Thoma. Image segmentation based on object oriented mapping parameter estimation. *Signal Processing*, 15(3):315–334, Oct. 1988.

[32] T. Aach, A. Kaup, and R. Mester. Statistical model-based change detection in moving video. *Signal Processing*, 31(2):165–180, March 1993.

[33] R. Thoma and M. Bierling. Motion compensating interpolation considering covered and uncovered background. *Signal Processing*, 1(2):191–212, Oct. 1989.

[34] 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.

[35] H. Luo and A. Eleftheriadis. Desiging an interactive tool for video object segmentation and annotation. In *Proc. of ACM Int'l Conf. on Multimedia*, pages 265–269, Orlando, FL, Oct. 1999.

[36] 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.

[37] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):34–43, Nov 1986.

[38] 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.