A Quick Algorithm to Search and Detect Video Shot Changes

Ehsan Amini
Department of Computer Science,
Dolatabad Branch, Islamic Azad University,
Isfahan, Iran

Somayyeh Jafarali Jassbi
Department of Computer Science,
Science and Research Branch, Islamic Azad University,
Tehran, Iran

ABSTRACT

The cost of video processing computation is very high and shot boundary detection is one of the reasons for this cost. Shot?s scope determination is an essential part in most of the video processing applications, especially for video indexing and content based video retrieval. This paper introduces a fast algorithm for searching the shots region and detecting their location, based on the divide and conquers technique. The main advantage of this method is due to skipping large parts of the video in the search step, which results in computational load reduction, leading to detection speed increase.

Keywords:

Content based video retrieval, Video shot change detection, Color histogram, Divide and conquer algorithm

1. INTRODUCTION

As a result of pervasive digital cameras, video production rate grew dramatically in recent years. This huge size of video needs various kinds of processes. A very essential preprocess is camera change boundary detection, that is defined as shot. Each shot determines a border between a series of related frames. Shots locations are widely used in many video applications such as indexing, retrieval, and editing, browsing and temporal segmentation. Shot types divided in two main groups, hard cut and gradual changes [1]. A hard cut happens when the scene suddenly changes. In this situation two successive frames are completely different and there are usually captured in two different times. Hard cut is the simplest shot type to detect. Gradual shots are happened when the scene changes slowly. In most of the time it is generated by adding video effects. The most popular scene change methods are fade, dissolve and wipe. In the fade effect a scene gradually transmits to a static image (fade out) or a scene will start from a constant image (fade in). For instance, the scene transmits to a black frame. A gradual transition from one scene to another is called dissolve effect. Here is a fade out in the first scene and a fade in in the second scene. Another gradual shot change is known as wipe. It happens when a line moves across the screen, with the new scene appearing behind the line. Generally, gradual shot change detections are more challenging than hard cut because several frames are in charge of scene changes and there is not an obvious boundary to specify. As mentioned, shot change detection is a very first and fundamental step in video processing purpose and this issue has been an attention in several studies [2–4]. To make this process faster, researchers have purposed various ideas. Jharna Majumdar et al. address a hardware based on field programmable gate arrays to video shot detection [5]. Bescos proposes a real time method for shot change detection and this method works over online MPEG-2 video [6]. The rest of the paper is organized as follows: Section 2 describes methods to measure the dissimilarities between frames. It also describes the global and local dissimilarity measurement, separately. The new algorithm for quick shot detection is proposed in section 3. Section 4 shows the experimental result. The algorithm accuracy is evaluated in section 5 and section 6 summarizes the paper.

2. DISSIMILARITY FUNCTIONS

Frames comparison is a very important issue in shot detection. It is necessary to know if two frames are in a scene or not. A dissimilarity function shows how much two frames are different. The dissimilarity functions introduced in this paper do not only focus on frame features but also they mention the region that frame is in. Base on frames distance, dissimilarity functions are divided into two types, Global Dissimilarity Functions (GDF); to compare frames which are far from each other and Local Dissimilarity Functions (LDF) which are used to make decisions about successive frames or frames that are near to each other.

2.1 Global Dissimilarity Function (GDF)

Shot search algorithm compares the first and the last representative frame of a specified part of the video. If they look the same, the algorithm skips searching inside that part as there is a low chance of finding a shot change. As a result the computation cost would decrease dramatically. Considering the constant change of frame details, where frames are not near each other; the comparison should be based on the public features such as background and color histogram. As shown in figure(4), this paper uses a set of frames and extracts representative frame to evaluate global dissimilarity. The process of extraction will be discussed later in section 3 but it would be useful to know that representative frame is same as background that subtracted from set of frames. The GDF returns the distance between two color histograms of representative frames which is computed based on color histogram euclidean distance as shows in (1).

Fig. 1. Four frames from a 30-frame-set: (a) first frame, (b) frame in offset 10, (c) frame in offset 20, (d) last frame

$$d^2(h_i, h_j) = \sum_A \sum_B \sum_C (h_i(a,b,c) - h_j(a,b,c))^2 \quad (1)$$

Where A, B and C represent color levels, in RGB case they are R, G and B and in HSV color space which is used here, they are H, S and V color levels. h_i and h_j are i^{th} and j^{th} and frame color histogram which are obtained from the following equation (2)

$$h_{A,B,C}(a,b,c) = (M \times N) Prob(A = a, B = b, C = c) \quad (2)$$

Let M and N be frame width and frame length respectively, therefore $N \times M$ is pixel number in a frame. Color histogram euclidean distance is wildly being used in content based image and video retrieval [7] and in shot change detection in some researches [8] and [9].

2.2 Local Dissimilarity Function (LDF)

In this paper LDF looks for a shot change in a local region and it returns frame index if finds it and zero otherwise. The strategy is same as traditional shot detection techniques, therefore, there are several alternatives to implement. Here, a combination of regional color histogram distance and edge density in frames is used to detect shot change. Regional histogram distance is able to detect sudden changes in frames to find hard cut shot changes. Observing region histogram change in the successive frames could help to detect gradual changes. Edge density measurement is also used to detect gradual shot changes such as fade in, fade out or dissolve.

3. REPRESENTATIVE FRAME

In order to reduce noise effect and achieve a better estimation, this algorithm uses representative frame from a set of successive frames. This frame is obtained by applying a time domain median filter on a set of successive frames. Figure(1), shows four frames from a 30-frame-set. The video in this example is from VISAL dataset [10] and [11].
This filter sorts pixels with the same location in all frames, selects a pixel with the median value from the list and place it in the exact position of the representative frame. Figure(2) shows the entire process step by step.
This filter is used as a background extraction technique [12] to assure that representative frame is enough stable to do the comparison. The complexity of computing depends on the sort method, which is $O(M \times N \times L \times \log(L))$, where L is the number of frames chosen in each set , (mentioned as $MinPartLen$ in the pseudo code), and M and N are the frame dimensions. Since L is a small number and it is possible to sort line of pixels in parallel, the computational cost is not too much. On the other hand Ryan and Tibshirani have introduced a method to find the median value which has $O(n)$ average complexity [13]. Figure(3), shows the representative frame of the 30-frame-set which is shown in figure(1).

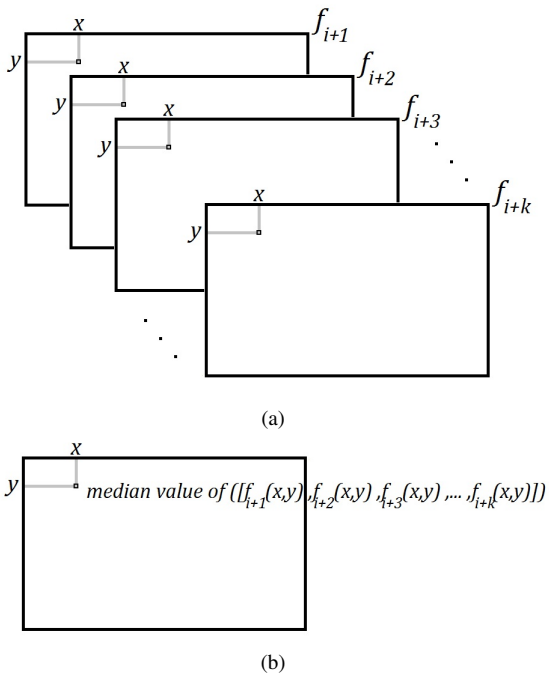

Fig. 2. (a),A set of frames and (b),their representative frame?s computation steps

Fig. 3. The representative frame of the 30-frame-set

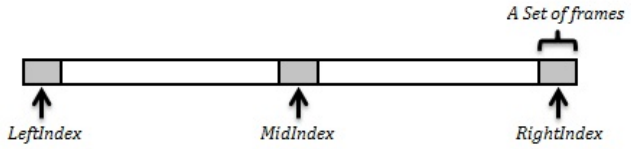

Fig. 4. Video division

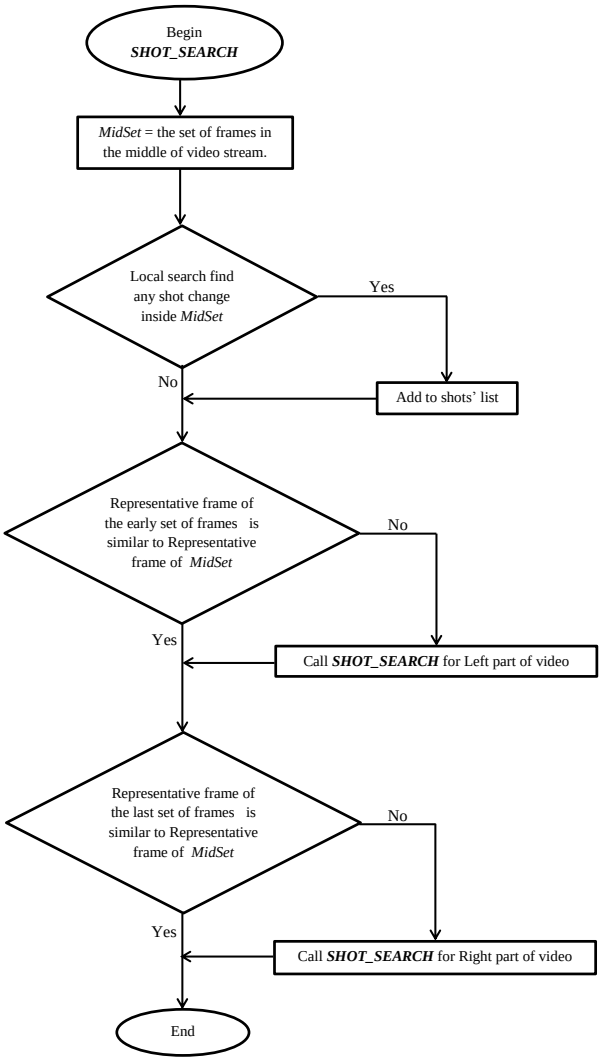

Fig. 5. Quick shot search flowchart.

4. FAST SHOT REGION SEARCH ALGORITHM

Figure(4), shows a video part structure which is divided into two parts and defines three sets of frames to specify each part boundary. Each set includes $MinPartLen$ frames and representative frames are determined in each section with the method described earlier.
The flowchart of the basic idea is presented in figure(5). As shown in pseudo code in algorithm (1), this algorithm accepts five

Algorithm 1 Quick Search and Detect

1: **procedure** SHOT SEARCH($LeftRep, LeftIndex, RightRep, RightIndex, MinPartLen$)
2: $PartLen \leftarrow LeftIndex - RightIndexb$
3: $MidIndex \leftarrow LeftIndex + (\frac{PartLen}{2})$
4: $[ShotLoc, LeftMidRep, RightMidRep] \leftarrow$ SHOT DETECTOR($MidIndex, MinPartLen$)
5: **if** $ShotLoc > 0$ **then**
6: Add ShotLoc to the list of shots
7: **end if**
8: **if** $PartLen > MinPartLen$ **then**
9: **if** GDF($LeftRep, LeftMidRep$) $<$ $FarSimilarityThreshold$ **then**
10: QUICK SHOT SEARCH($LeftRep, LeftIndex, RightMidRep, MidIndex, MinPartLen$)
11: **end if**
12: **if** GDF($RightMidRep, RightRep$) $<$ $FarSimilarityThreshold$ **then**
13: QUICK SHOT SEARCH($RightMidRep, MidIndex, RightRep, RightIndex, MinPartLen$)
14: **end if**
15: **end if**
16: **end procedure**

parameters which specify the following: representative frame of the first frame set,$LeftIndex$: the middle value of the first set of frames index ($\frac{MinPartLen}{2}$ in the first function call), $RightRep$: representative frame of last frame set, $RightIndex$: the middle value of the last set of frames index (video frame number - $\frac{MinPartLen}{2}$ in the first function call) and $MinPartLen$: the length of frames set that is the minimum size of a video part, respectively.
At first, the algorithm finds the input video part length and the index of middle frame, then it calls $ShotDetector$ function, which returns shot change location if find any and representative frames for middle set. This function steps will be described later. If $ShotDetector$ function returns none zero number it means a shot change has detected and it should be added to the shot locations list. After that, the algorithm checks if there is a chance of shot change in each part by calling GDF function. $ShotDetector$ function will be called recessively for each left or right parts, in case of possible shot change detection.

4.1 Shot detector algorithm

As can be seen in the algorithm (2), $ShotDetector$ function does a local search in the frame set by calling LDF function. If LDF function finds any shot change it will return its location. Afterwards, $ShotDetector$ function will be repeated to compute left representative frame in remained frames in the left part of shot change location and right representative frame in remained frames in the right part of shot change location. Otherwise left representative frame and right representative frame are the same. In this function $VideoFilter$ function is responsible for computing representative frame.

5. EXPERIMENTS

The algorithm described in this article, has implemented in C++ using Opencv library. To show the result, the algorithm applied to the Friends TV show, Season 2 Episode 18. In the figure (6), four shot changes and their frame numbers which detected by the algorithm are shown.

Algorithm 2 Shot Detector

procedure SHOT DETECTOR($MidIndex, MinPartLen$)
  $ShotLoc \leftarrow$ LDF($MidIndex - \frac{MinPartLen}{2}$, $MidIndex + \frac{MinPartLen}{2}$)
  if ShotLoc ¿ 0 **then**
    $LeftMidRep \leftarrow$ VIDEOFILTER($MidIndex - \frac{MinPartLen}{2}, ShotLoc - 1$)
    $RightMidRep \leftarrow$ VIDEOFILTER($ShotLoc, MidIndex + \frac{MinPartLen}{2}$)
  else
    $LeftMidRep \leftarrow RightMidRep \leftarrow$ VIDEOFILTER($MidIndex - \frac{MinPartLen}{2}, MidIndex + \frac{MinPartLen}{2}$)
  end if
  return $ShotLoc, LeftMidRep, RightMidRep$
end procedure

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

Fig. 6. Four simple results and their location. (a) Frame number is 129, (b) frame number is 761, (c) frame number is 806 and (d) frame number is 1818.

6. CONCLUSION

The method has presented in this paper is a divide and conquer algorithm which is able to do quick search for shot change probability and detect them. Algorithm divides the video into two parts and continues the search in each part. Worst case scenario, is when there are lot of shot changes in video. In this situation the algorithm complexity as it is shown in (3) is $O(n)$ because the algorithm should look for a new shot change in each part.

$$t(n) = 2t(\frac{n}{2}) + c \therefore t(n) \in O(n) \quad (3)$$

In general and in the ordinary cases, when shot change does not happen a lot, algorithm complexity belongs to $O(\log n)$ which is shown in (4).

$$t(n) = t(\frac{n}{2}) + c \therefore t(n) \in O(\log n) \quad (4)$$

This algorithm is widely applicable in video search engines and it is also usable in standalone devices such as smart phones and personal computers. A weakness of this algorithm is inability of detecting the shot boundaries located inside a long scene. It only happens when the video divisions are chosen in two sides of the shot. In future it is possible to develop this algorithm to achieve better result to detect short video parts shot change which is happened inside a long shot.

7. REFERENCES

[1] Jun Yu and M.D. Srinath. 2001. An effective method for scene cut detection. *Pattern Recognition Letters ELSEVIER*, 22:1379 – 1391.

[2] Ravi Mishra, S. K. Singhai, and Monisha Sharma. 2013. Comparative based study of different video-shot boundary detection algorithms. *International Journal of Advanced Research in Computer Engineering and Technology (IJARCET)*, 2(1).

[3] Salim A. Chavan and Sudhir G. Akojwar. 2013. Gradual transition detection algorithms in video segmentation: A survey. *International Journal of Scientific and Engineering Research (IJSER)*, 4(2).

[4] Nikita Sao and Ravi Mishra. 2014. A survey based on video shot boundary detection techniques. *nternational Journal of Advanced Research in Computer and Communication Engineering (IJARCCE)*, 3(4).

[5] Jharna Majumdar, Darshan K M, and Abhijith Vijayendra. 2013. Design and implementation of video shot detection on field programmable gate arrays. *International Journal of Robotics and Automation (IJRA)*, 2(1).

[6] J. Bescos. 2004. Real-time shot change detection over online mpeg-2 video. *IEEE Trans. on Circuits and Systems for Video Technology*, 14(4):475484.

[7] Rishav Chakravarti and Xiannong Meng. 2009. A study of color histogram based image retrieval.

[8] Vrutant Hem Thakore. 2013. Video shot cut boundary detection using histogram. *International Journal Of Engineering Sciences and Research Technology (IJESRT)*, 2.

[9] P. Swati Sowjanya and Ravi Mishra. 2012. Video shot boundary detection comparison of color histogram and gist method. *International Journal of Research in Engineering and Applied Sciences (IJREAS)*, 2(2).

[10] A. B. Chan and N. Vasconcelos. 2008. Modeling, clustering, and segmenting video with mixtures of dynamic textures. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 30(5):909–926.

[11] A. B. Chan and N. Vasconcelos. 2005. Probabilistic kernels for the classification of auto-regressive visual processes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[12] Shireen Y. Elhabian, Khaled M. El-Sayed, and Sumaya H. Ahmed. 2008. Moving object detection in spatial domain using background removal techniques. *Recent Patents on Computer Scienc*, 1:32–54.

[13] Ryan J. Tibshirani. 2008. Fast computation of the median by successive binning. *cornell university, CoRR abs / arXiv:0806.3301*.