Available online at www.sciencedirect.com
ScienceDirect
Computer Standards & Interfaces 29 (2007) 335 – 342
COMPUTER STANDARDS & INTERFACES
www.elsevier.com/locate/csi

A real-time shot cut detector: Hardware implementation

Lotfi Boussaid [a,c,*], Abdellatif Mtibaa [b], Mohamed Abid [a], Michel Paindavoine [c]

[a] *Ecole Nationale d'Ingénieurs de Sfax: Unité de recherche CES, France*
[b] *Ecole Nationale d'Ingénieurs de Monastir: LaboratoireEμE, France*
[c] *Laboratoire LE2i, France*

Accepted 24 May 2006
Available online 24 July 2006

Abstract

With the enormous growth in digital audiovisual (**AV**) information in our life, there is an important need for tools which enable describing the **AV** content information. In this context, the **MPEG-7** standard was developed in order to provide a set of standardized description tools which generate metadata about **AV** content. However, before any content-based manipulations, the hierarchical structure of video must be determined. This process is known as shot boundary detection or in other case scene change detection. In this paper, an old and reliable method based on local histogram has been used to implement shot cut detector for real-time applications. Since software implementation on **PC** is not suitable for this algorithm due to the sequential treatments of the processor, we have used an **FPGA**-based platform.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Shot boundary detection; Video segmentation; Local histogram; **MPEG-7**; **FPGA**; Virtex XCV800; Hardware implementation

1. Introduction

Video data is becoming very important in many application domains such as digital broadcasting, interactive-**TV**, video-on-demand, computer-based training, and multi-media processing tools. Furthermore the development of the hardware technology and communications infrastructure has made automatic analyzing of video content very challenging.

In this work, which is part of a project thesis, we present the different steps of the hardware implementation of shot cut detector based on local histogram algorithm. This old and reliable approach is a descriptor in the **MPEG-7** standard. The objective of the **MPEG-7** ("Multimedia Content Description Interface") standard is to specify a standard set of descriptors and description schemes for describing the content of **AV** information. It specially standardizes a number of description tools which describe **AV** content ranging from low-level features to high level semantic information. In other words, it provides a set of standardized description tools which generate metadata (data about data) about **AV** content by extracting information of interest from it, to facilitate a variety of applications including image and scene retrieval [1]. In this context, the local histogram approach constitutes a low-level feature which is utilized for video segmentation and image and scene retrieval. In order to develop any content-based manipulations on video information, hierarchical structure must be determined. In this way, a standard hierarchical video model was defined as shown in Fig. 1. This model is composed of some elementary units as scenes, shots, and frames. In this structure a shot is defined as an unbroken sequence of frames from a single camera, where a scene is a set of shots with semantic link, location unit and action unit [2].

In produced video such as television or movies, shots are separated by different types of transitions, or boundaries. Although well known video editing programs such as Adobe Premiere or Ulead Media Studio provide more than 100 different types of edits, we classify in general transition effects into two categories [3]. The simplest transition is a cut, an abrupt shot change that occurs between two consecutive frames. Gradual transition such as fades and dissolves are more complex. Shot boundaries are fades when the frames of the shot gradually change

* Corresponding author. Ecole Nationale d'Ingénieurs de Sfax: Unité de recherche CES, France.
E-mail addresses: lotfiboussaid@yahoo.fr (L. Boussaid), Abdellatif.mtibaa@enim.rnu.tn (A. Mtibaa), mohamed.abid@enis.rnu.tn (M. Abid), paindav@u-bourgone.fr (M. Paindavoine).

0920-5489/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.csi.2006.05.009

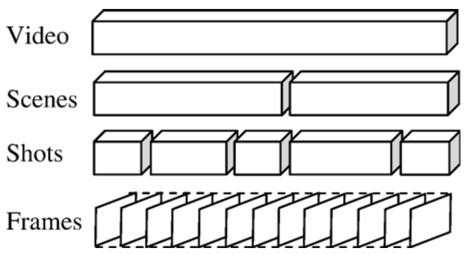

Fig. 1. Standard hierarchical video model.

from or to black, and can be dissolved when the frames of the first shot are gradually morphed into the frames of the second [4]. Fig. 2 shows an example of transition effects.

Most of the existing methods of video segmentation have to challenge the difficulty of finding shot boundaries in the presence of a camera or object motion and illumination variations which can lead to false detection. In other cases, frames that have different structures but similar color distributions can give a missed detection [5]. The study of the state of the art shows that several methods for **SBD** were proposed. These methods can operate in different environments such as temporal, frequency, uncompressed and compressed domains.

On the other hand, Dailianas and Lefèvre [6,7] have distinguished two classes of methods: Those which could be done off-line and have high complexity, and others which are dedicated for real-time applications. In this case, some constraints have to be taken into account. In this paper we have used an old and reliable method based on local histogram and proposed by Nagasaka and Tanaka [8]. They have divided each frame into 16 blocks and computed local histogram before evaluating a difference metric. Histogram-based methods have shown a good performance for shot cut detection.

To operate in real-time condition, computational time of the difference metric mustn't exceed the blanking time which is about 2 ms. Since software implementation on **PC** is not suitable for the local histogram algorithm due to the serial architecture of the microprocessor, we have designed our system on a hardware platform based on Virtex xcv800 **FPGA**.

This paper is organized as follows. In Section 2 we present the different methods proposed for the detection of the abrupt shot changes. Section 3 describes the specifications of the local histogram method which we tested on a set of video sequences, in different color spaces, different types of quantization and different formats of sub-sampling. The concept of the hardware design and the interpretation of the hardware implementation results are presented in Section 4. Finally Section 5 brings the conclusions and the future works.

2. Related work

An important variety of shot boundary detection algorithms was proposed in the last decade. The study of the current state of the art shows that we can classify these algorithms into three generations. The first generation concerns methods which measure distance of similarity between adjacent frames by using elementary features extraction such as pixel differences, global and local histogram differences, motion compensated pixel differences and **DCT** coefficient differences [9–12]. In the second generation, hybrids of the above methods have been investigated [13–15]. In this way J. S. Boreczky [16] has combined audio and video features in a hidden Markov model to increase the shot change detection. Although these techniques have brought improvement to the quality of shot change detection, they increased complexity and computational time. The most recent algorithms have introduced intelligent algorithms such as fuzzy approaches and those based on neural network [17,18].

To compare the ability of real-time or close to real-time implementation of the different methods, Dailianas [6] has evaluated the complexity of many approaches by estimating the number of operations when measuring dissimilarity between two consecutive frames. In this way, he has used an assumption that addition, subtraction and multiplication require time equivalent to one operation, whereas divisions take approximately four times

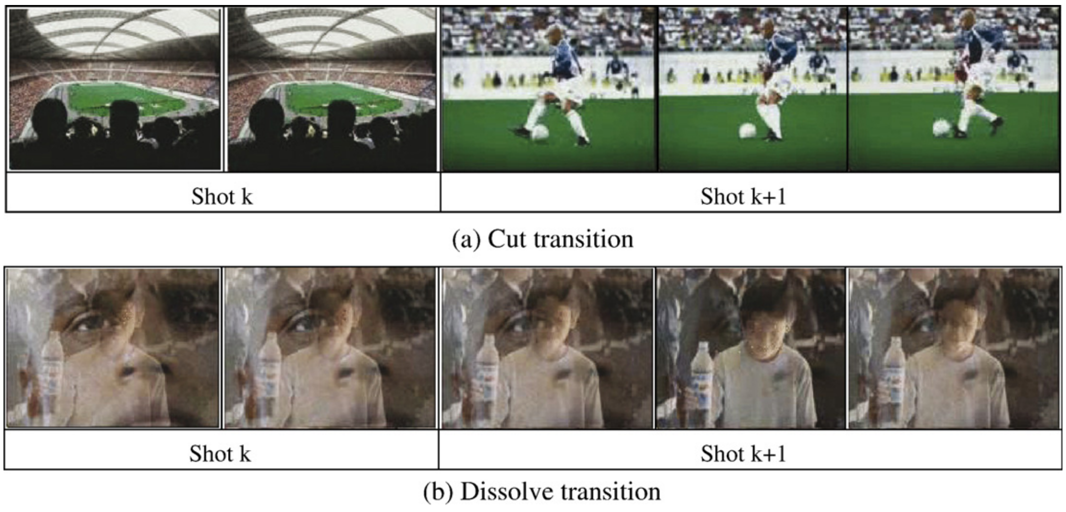

Fig. 2. Example of transition effects.

Table 1
The test video set

Video	Duration	Dim.	Cuts	Dissolves
Soccer	30 s:10	216×192	20	–
Clip	25 s:01	320×288	11	–
Doc.	58 s:10	352×288	7	6
Movie	58 s:09	352×288	15	13
Com.	45 s:80	160×120	31	3
News	1′:01	384×288	7	3
Cartoons	1′:10	640×480	23	1

more. Implementation issue such as assignment of variables to registers, use of pointers and arrays, memory access time and others, are ignored.

In the following sections we denote N as the number of levels (bins) of pixel value and P the number of pixels per frame.

3. Video segmentation: case studies

3.1. Local histogram specifications

The color histogram for an image is constructed by counting the number of pixels of each color. More formally, the color histogram is defined as the probability mass function of the image intensities.

To increase the quality of shot change detection block-based methods were proposed [8,19–21]. The main advantage of these methods is their relative insensitivity to noise and camera or object motion.

In this work we have used the approach proposed by Nagasaka [8] who divided each frame into 16 blocks and computed the distance of dissimilarity between consecutive frames f_n and f_{n+1} as follows:

$$D(f_n,f_{n+1}) = \sum_{c\in\mathrm{RGB}} \sum_{b=1}^{16} \sum_{j=0}^{N-1} |H(f_{n+1},c,b,j) - H(f_n,c,b,j)|$$

Where c, b and N are respectively the luminance of color components of the picture (red, green and blue), the number of blocks and the number of bins of the pixel value.

$H(f_n,c,b,j)$ is the histogram value for frame f_n and for the discrete value of the intensity k. The value of k is in the range $[0, N-1]$, where N is the number of discrete values a pixel can have.

To detect break shots, the metric $D(f_n,f_{n+1})$ is compared to a global threshold. When this metric exceeds the value of the threshold, it indicates that a shot transition has occurred.

Table 2
Results of cut detection for different color spaces

Video	RGB (%)	YC_bC_r (%)	C_bC_r (%)	Gray (%)
Soccer	100	84	57	100
Clip	100	100	92	100
Doc.	63	88	71	86
Movie	71	59	50	96
Com.	71	77	84	88
News	18	36	40	25
Cartoons	56	41	38	18
Average	68	69	62	73

Table 3
Results of dissolve detection for different color spaces

Video	RGB (%)	YC_bC_r (%)	C_bC_r (%)	Gray (%)
Doc.	50	33	33	50
Movie	46	46	30	38
Com.	100	100	100	100
News	33	33	33	66
Average	57	53	51	62

Along the experiments, we have tried to reduce complexity and speed up the algorithm while preserving the same accuracy. To do so, we have performed the local histogram approach on a set of video sequences, in different color spaces, different types of quantization and different formats of sub-sampling.

3.2. Video corpus

The test set of video sequences consisted of various sources, partially digitized from different **TV** channels as **MPEG1** of the size 352×288 and partially copied from the **CD**s as **MPEG1** of the size of 640×480 and 352×288, and downloaded from internet as **AVI** of the size of 216×192 and 160×120. It includes sequences from “le Roi Scorpion” film, “Pepsi” **TV** commercial, “Mulan of Walt Disney” cartoons, “Arabia” **TV** news, “E=M6” documentary, “Rotana” clip and “Soccer” match.

The content of this video database is listed in Table 1 which presents the duration, the frame size and the number of cuts and dissolves.

This set of video sequences contains a variety of transition effects, such as abrupt and gradual transitions. Moreover, it involves special effects that lead to false and missed detections (appearance of text in frames, illuminations, rapid object motion, zooming, panning etc.).

3.3. Color spaces evaluation

A color space is defined as a model for representing color in terms of intensity values [22]. Typically, a color space defines a one-to-four dimensional space. A color component, or a color channel, is one of the dimensions. A color dimensional space (i.e. one dimension per pixel) represents the gray scale space.

For the experiments, RGB, YC_bC_r and gray scale spaces have been used. To reduce the sensitivity to flash and illumination

Table 4
Results of cut detection for different quantization

Video	Gray U4 (%)	Gray U8 (%)	Gray Ln8 (%)
Soccer	100	100	80
Clip	100	100	100
Doc.	86	86	86
Movie	96	96	85
Com.	88	88	87
News	25	19	35
Cartoons	18	53	60
Average	73	77	76

effects, we have also used C_bC_r without the luminance component *Y*.

The metric which we defined in *RGB* space, was determined also in YC_bC_r, C_bC_r and gray levels color spaces.

A color in *RGB* space is converted to YC_bC_r color space by using the following equations:

$$Y = 0{,}299R + 0{,}587G + 0{,}11B$$

$$C_b = (B-Y)/1{,}772 + 0{,}5$$

$$C_r = (R-Y)/1{,}402 + 0{,}5$$

The gray scale can be represented only with *Y* component.

The results of cut detection obtained by performing local histogram across the above color spaces are shown in Table 2.

For the dissolve detection, results are shown in Table 3.

In Table 2, results show that cuts are reliably detected except for news and cartoons video. In the case of news, the performance was not satisfactory because of special effects presence during a topic change. For cartoons, large object motion, rapid object motion, zooming, panning, rapid lighting changes are the principal causes of the great number of false detections.

In general, for a typical video, histogram-based algorithms can identify over 90% of the real shot transitions [6]. However, for *RGB* color space experiments have shown that this technique is sensitive to illuminations. The use of YC_bC_r and C_bC_r color spaces has reduced false detection caused by rapid light changes (e.g. better performance for commercial sequence), however they introduced missed detections when consecutive frames of different shots have similar chrominance.

Finally, performing the local histogram method in the gray levels space has presented a better quality of detection comparing to the other color spaces. Though, illumination changes have given several false detections.

Results in Table 3 describe the performance of the local histogram algorithm for gradual transition (dissolve). They show that this technique gives poor quality of detection especially when slow transition occurs. As a consequence, the approach of local histogram is not a suitable technique for gradual transition detection.

3.4. Quantization evaluation

In this section, we have tested the quantization effect on cut detection. Thus, we have firstly performed uniform quantization into 8 and 4 levels (bins). After, logarithmic quantization has been used.

Table 5
Results of cut detection with sub-sampled frames

Video	Gr 4:1 (%)	Gr 4:2 (%)	YC_bC_r 4:1:1 (%)	YC_bC_r 4:2:0 (%)
Soccer	100	100	95	95
Clip	100	100	100	100
Doc.	86	86	57	57
Movie	96	96	82	84
Com.	88	88	86	86
News	23	15	15	12
Cartoons	10	10	23	27
Average	72	71	65	66

For uniform quantization into 4 levels, we define 4 ranges of discrete values. These ranges are represented by one centre value for each one.

$0 \rightarrow 63$	First range	(0)
$64 \rightarrow 127$	Second range	(1)
$128 \rightarrow 191$	Third range	(2)
$192 \rightarrow 255$	Fourth range	(3)

The same principle for quantization into 8 bins has been applied.

For the logarithmic quantization we have used the European “A-law”:

$$F(x) = \begin{cases} \dfrac{A|x|}{1+\ln A}\,\mathrm{sign}(x) \rightarrow 0 \le \dfrac{|x|}{x_{\max}} \le \dfrac{1}{A} \\ x_{\max}\dfrac{1+\ln\left[\dfrac{A|x|}{x_{\max}}\right]}{1+\ln A}\,\mathrm{sign}(x) \rightarrow \dfrac{1}{A} \le \dfrac{|x|}{x_{\max}} \le 1 \end{cases}$$

The main idea of this non-uniform quantization is to make compression of amplitude followed by a uniform quantization. This technique was proposed in signal treatments to improve SNR especially when handling low amplitude signals. As a result, we obtained a new range of discrete values as follows:

$0 \rightarrow 1$	First range	(0)
$2 \rightarrow 4$	Second range	(1)
$5 \rightarrow 8$	Third range	(2)
$9 \rightarrow 16$	Fourth range	(3)
$17 \rightarrow 33$	Fifth range	(4)
$34 \rightarrow 65$	Sixth range	(5)
$66 \rightarrow 131$	Seventh range	(6)
$132 \rightarrow 255$	Eighth range	(7)

Table 4 shows the results of cut detection with different types of quantization.

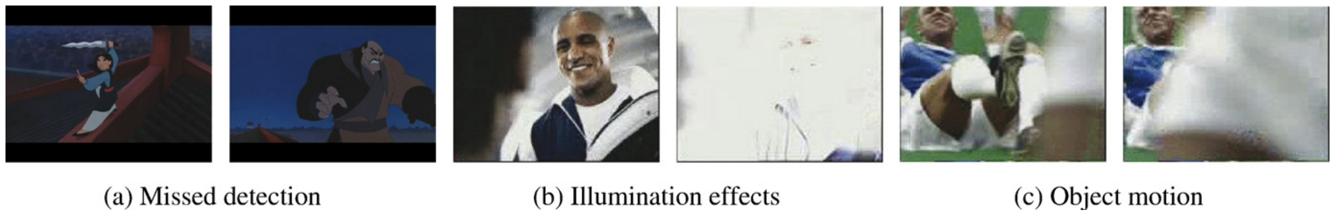

Fig. 3. Examples of missed detection and false alarms.

Table 6
Computational requirements for various methods

Method	Number of operations
Pixel pair difference	$9P$
Red histogram difference	$P+2N$
X^2 red histogram difference	$P+7N$
Edge change fraction	$26P$
Local histogram (*RGB*)	$3P+6Nb$
Local histogram (YC_bC_r)	$3P+6Nb$
Local histogram (C_bC_r)	$2P+4Nb$
Local histogram (gray levels)	$P+2Nb$

We denote "b" the number of blocks, "N" the number of levels and "P" the number of pixels. In practice we have chosen $b=16$ and $N=4$.

The use of logarithmic quantization has reduced missed detections caused by similarity between different frames. However the sensitivity of this method has led to several false alarms. Table 4 shows that the average of cut detection is almost identical for each type of quantization. Examples of false alarms and missed detected cuts are given in Fig. 3.

3.5. Sub-sampling evaluation

In this section we have worked into gray levels and YC_bC_r spaces with sub-sampled frames 4:1:1 and 4:2:0. Table 5 summarizes the obtained results which are slightly less effective.

3.6. Computation time estimation

According to the assumption of Dailianas and Lefèvre [3,6], we present in Table 6 the computational requirements for the main known algorithms and local histogram across 4 color spaces.

4. Hardware implementation

4.1. Introduction

If some processing can be executed in differed time as in the field of the data analysis, others require data processing in real-time. In this case, it is a question of respecting the blanking time constraint (time between two consecutive frames). In fact, image difference must be computed before the coming of the next image. Since the use of software implementation is not suitable for our application due to its sequential treatments, an **FPGA**-based solution has been chosen.

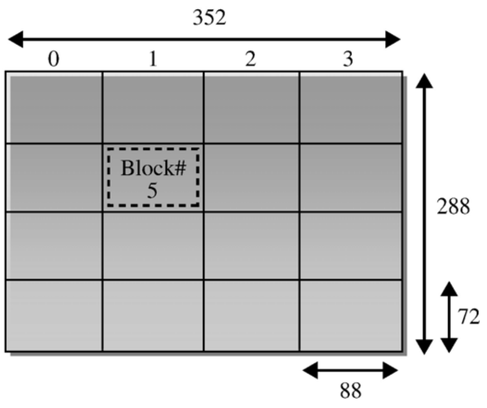

Fig. 5. Block number calculation.

Recently the use of the **FPGA**-based platform is becoming very popular and attractive, because **FPGA** offers a compromise between **ASIC** (Application Specific Integrated Circuit) hardware and general purpose processors. **ASIC**s offer high performance, but take longer to design and lack programmability and adequate flexibility. Once deployed, systems built using **ASIC**s suffer long time-to-market and high costs for even minor changes.

Thus, compared with **ASIC** and software implementation, **FPGA** has lower costs, respectively higher speeds and especially more flexibility because it can instantaneously be configured.

Our previous work [23,24] consists in studying simultaneously algorithmic and architectural aspects in order to optimize hardware implementation and reduce the time of the design. Thus, the main objective is to find the best adequacy between algorithm and architecture while taking into account real-time constraints and minimization of logic resources (Table 6).

The hardware development system is based on the XSV800 board. The core of the board is a Virtex XCV800 **FPGA** with 800 k gates, 9408 slices, 18816 Flip Flops, 18816 **LUT**s, 320 **IOB**s, 4 **GClk**s and 28 configurable internal Block **RAM** of 4 kb.

The architecture is described with the **VHDL** language in Register Level Transfer (**RTL**) with the Integrated Software Environment (**ISE**) of Xilinx.

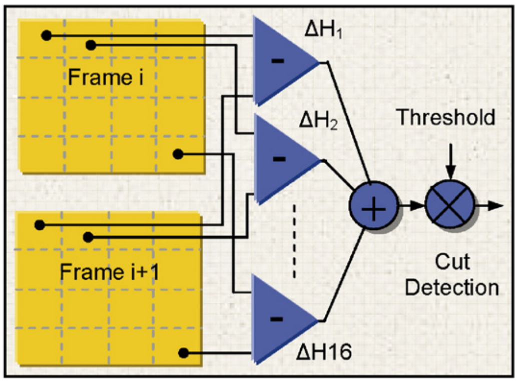

Fig. 4. Cut detection principle.

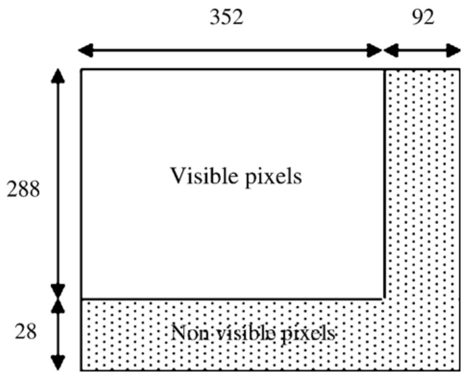

Fig. 6. Entire image video.

4.2. Temporal constraints

For analog **TV**, there are two important synchronization signals:

– The horizontal synchronization which gives periodically a negative pulse during the dead time between two consecutive lines belonging to the same image.
– The vertical synchronization which generates a negative pulse between two consecutive images. This pulse width is equal to the inter-image time.

In digital **TV**, the streaming video at 25 Mb/s is composed by **DCT** based compression video, two or four tracks of audio, a time code (**TC**) track and **CRC** redundancy check track. Audio, **TC** and **CRC** tracks represent the inter-image time presented for the analog **TV**.

For digital video, we digitalize only 720 pixels instead of 856 per line. The rest of the entire line is replaced by sound, **TC** and **CRC**. For audio we have to choose between:

– 2 tracks at 48 kHz 16 bits = 1536 kb/s
– 4 tracks at 32 kHz 12 bits = 1536 kb/s

In digital, the inter-frame time is about 1.9 ms. This dead time must be sufficient to compute our approach in real-time condition.

In experiments, we have chosen a **CIF** format (a quarter of screen) which has the size of 352 × 288 pixels. Since block histogram (local histogram) consists in dividing image into 16 blocks, so a block has a size of 88 × 72 pixels.

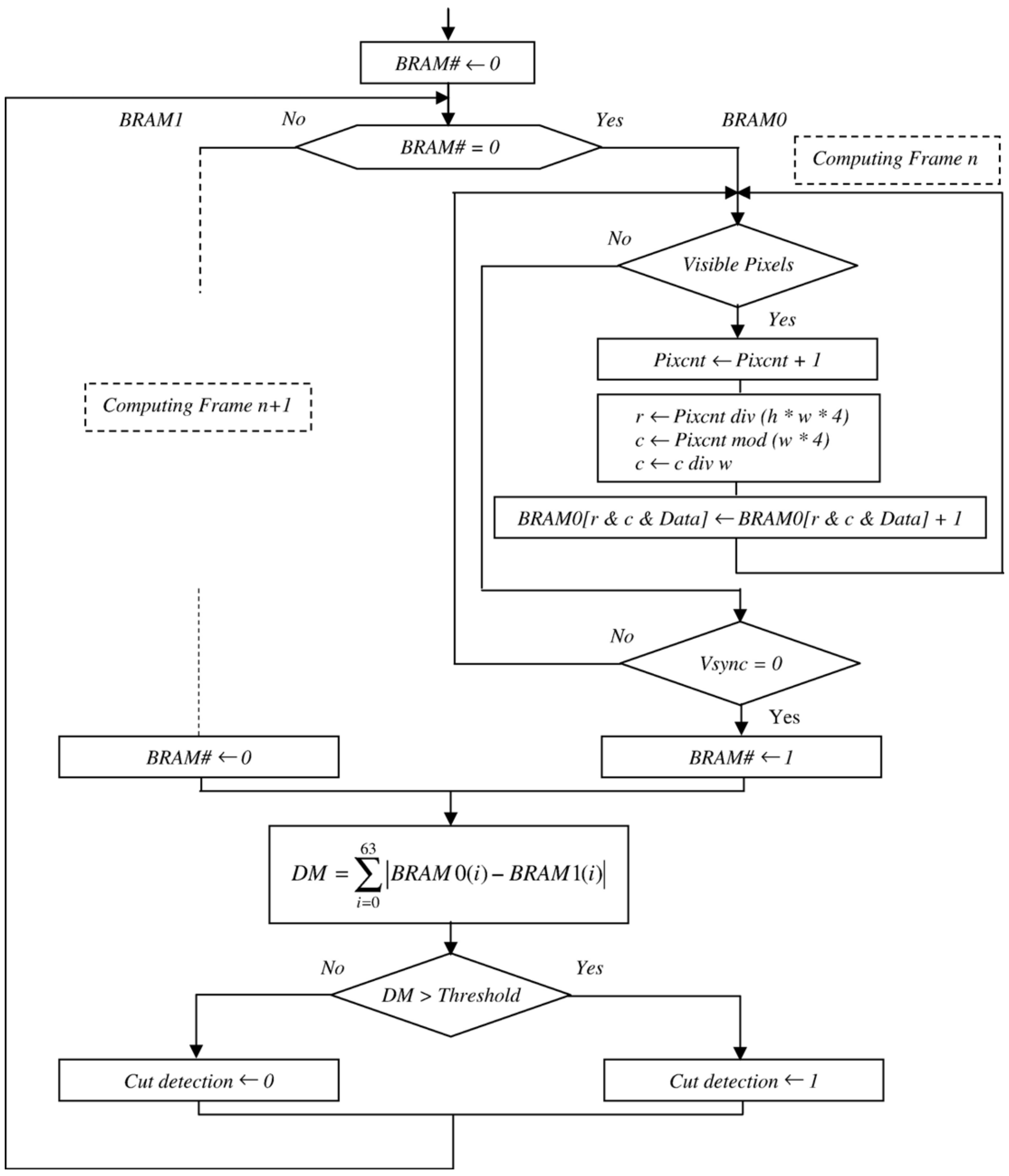

Fig. 7. Flowchart for dissimilarity metric processing.

4.3. Implementation schemes

The process of cut detection based on local histogram is described in Fig. 4.

It includes three important steps:

– histograms calculation;
– sum of histograms differences of consecutive frames and;
– comparison to a predetermined threshold value.

Following the experiments presented above, we have chosen to work in gray levels space and uniform quantization at 4 levels.

The image size used for test and simulation is of 352×288 pixels.

Let W, H, w and h represent respectively the width and height of a frame, and the width and height of one block in the frame.

$$W = w * 4 \quad \text{and} \quad H = h * 4.$$

We denote "r" a row composed by 4 blocks, and "c" a column which is also composed by 4 blocks. A block number is determined as shown in Fig. 5 by the expression:

$$\text{Block\#} = 4r + c$$

With $r, c \in [0 \cdot\cdot 3]$

$$\text{Block5} = 4(r) + c = 4(1) + 1 = 5$$

To compute correctly the dissimilarity metric of consecutive frames, we must take into account the non-visible pixels when Hsync and Vsync (synchronization signals) send negative pulses. Fig. 6 shows the regions of visible and non-visible pixels.

The flowchart described in Fig. 7, presents the whole concept of the design.

The variables used in the design are defined as follows:

BRAM# Is Boolean that can take 0 or 1 and represents the memory number in which frame will be saved.

BRAM0 and **BRAM1** Are two block **RAM** of 4 kb. In fact, XCV800 **FPGA** provides 28 internal **RAM** memories of 4 kb called Block **RAM**. Each one of these memories can be configured in width and depth according to the needs of the application. In our case, only two **BRAM**s configured as 256 words of 16 bits have been used. Though only 64 words are useful to save 16 histograms of dimension 4 (quantization into 4 levels).

Pixelcnt Represents the visible pixel counter.

r Gives the row number in a frame.

c Gives the column number in a frame.

BRAM0[r & c & data] We concatenate in this case (r), (c) and (data) to firstly locate the memory cells reserved for a block and after the specific cell according to the data value $[0 \cdot\cdot 3]$.

Vsync and Hsync Represent the vertical and horizontal synchronization.

Threshold Is predetermined value deduced in experiments.

DM Represents the dissimilarity metric which should be compared to threshold in order to make decision.

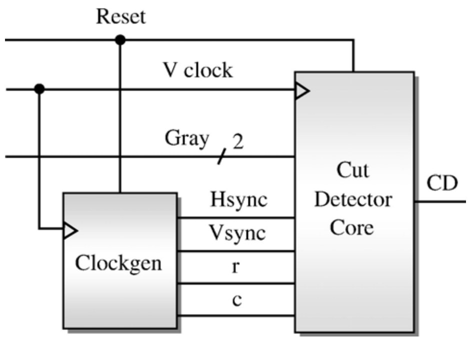

Fig. 8. Synoptic of the design in the **RTL** level.

Table 7
Table of results

Slices	Flip Flops	**LUT**s	**IOB**s	Min. period
152 (1%)	112 (0.6%)	268 (1%)	4 (1%)	8.488 ns

In the **RTL** level, design has been divided into parts as described in Fig. 8.

4.4. Implementation results

After validation by simulation using Modelsim, compilation and synthesis by Xilinx Integrated Software Environment (**ISE**) gave the results presented in Table 7. We notice that the occupied area of the design is almost 1% of total resources plus two internal blockRAMs of 4 kb. The minimum period is 8.48 ns.

5. Conclusion

In this study, we have tested and evaluated the local histogram approach across several types of video, in different color spaces, different types of quantization and sub-sampling.

The experiments have shown that the gray space at four levels has presented reliable results and relatively low computation time.

On the other hand, the hardware implementation on a Virtex **FPGA**-based board has used almost 1% of logical resources plus two Block **RAM**s. By using a clock system of 50 MHz, the computation time obtained is 0.3 ms which less than the inter-frame dead time estimated at 1.91 ms.

Our future work consists in developing a video analysis and content description template for real-time applications. This template is based on extraction of audio and video features.

References

[1] P. Browne, et al., Evaluating and Combining Digital Video Shot Boundary Detection Algorithms, Proceedings of the Fourth Irish Machine Vision and Information Processing Conference, Queens University, Belfast, September 1999.

[2] R. Lienhart, et al., A systematic method to compare and retrieve video sequences, Proceedings of storage and retrieval for image and video databases VI, vol. 3312, SPIE, January 1998, pp. 271–282.

[3] R. Lienhart, Comparison of automatic shot boundary detection algorithms, Proc. SPIE Conference on Storage and Retrieval for Image and Video Databases VII, Jan. 1999, pp. 290–301, San Jose, CA.

[4] S.V. Porter, et al., Detection and classification of shot transitions, in: T. Cootes, C. Taylor (Eds.), Proceedings of the 12th British Machine Vision Conference, BMVA Press, September 2001, pp. 73–82.

[5] J. Mas, et al., Video shot boundary detection using color histogram, TRECVID2003 Conference, Gaithersburg, Maryland, USA, October 2003.

[6] A. Dailianas, et al., Comparison of Automatic Video Segmentation Algorithms, International Issues in Large Commercial Media Delivery Systems, Proc. SPIE, vol. 2615, October 1995, pp. 2–16.

[7] S. Lefèvre, et al., A review of real-time segmentation of uncompressed video sequences for content-based search and retrieval, Real Time Imaging 9 (1) (2003) 73–98.

[8] A. Nagasaka, Y. Tanaka, Automatic Video Indexing and Full-Video Search for Object Appearances, Video Database Systems II, 1992, pp. 113–127.

[9] E. Ardizzone, M. Cascia, Automatic video database indexing and retrieval, Multimedia Tools and Applications 4 (1997) 29–56.

[10] M. Abdel-Mottaleb, et al., Content-based image and video access system, Proc. of ACM International Conference on Multimedia, ACM Press, New York, NY, USA, November 1996, pp. 427–428.

[11] H.-H. Yu, W. Wolf, A Visual Search System for video and Image Databases, Proc. IEEE International Conference on Multimedia Computing and systems, IEEE Computer Society, Washington, DC, USA, June 1997, pp. 517–524.

[12] F. Arman, et al., Image Processing on Compressed Data for Large Video Dtabase, Proc. of ACM International Conference on Multimedia, 1993.

[13] C. Taskiran, E.J. Delp, Video Scene Change Detection using the generalized sequence trace, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal processing, May 1998, pp. 2961–2964, Seattle, WA.

[14] JungHwan Oh, Kien A. Hua, Efficient and Cost-Effective techniques for Browsing and Indexing Large Video Databases, Proc. of ACM SIGMOD International Conference on Management of Data, ACM Press, New York, NY, USA, May 2000, pp. 415–426.

[15] H.D. Wactlar, et al., Lessons learned from building terabyte digital video library, Computer (February 1999) 66–73.

[16] J.S. Boreczky, L.D. Wilcox, A hidden Markov Model Framework for Video Segmentation Using Audio and Image Features, Proc. ICASSP'98, vol. 6, May 1998, pp. 3741–3744.

[17] R.S. Jadon, et al., A Fuzzy Theoretic Approach for Video Segmentation using Syntactic Features, Pattern Recognition Letters, vol. 22 issue 13, Elsevier Science Inc, November 2001, pp. 1359–1369.

[18] Xiang Cao, Ponnuthurai N. Suganthan, Neural network based temporal video segmentation, International Journal of Neural Systems 12 (3–4) (2002) 263–269.

[19] R. Kasturi, R.C. Jain, Dynamic Vision, in: R. Kasturi, R.C. Jain (Eds.), Computer Vision: Principles, IEEE Computer Society Press, Washington, 1991, pp. 469–480.

[20] W. Ren, et al., Automated Video Segmentation, International Conference in Information, Communication, and Signal processing, Octobre 2001, Singapore.

[21] D. Swanberg, et al., Knowledge Guided Parsing and Retrieval in Video Databases Systems, SPIE Proceedings, vol. 1908, 31 January–5 February 1993, San Jose, CA, USA. 244 pp., ISBN: 0-8194-1141-8.

[22] James Z. Wang, Integrated Region-Based Image Retrieval, Kluwer Academic Publishers, Boston, 2001.

[23] L. Boussaid et. al., *"A Real-Time Shot Boundary Detection Algorithm Based on Local Histogram",* SSD'05, IEEE International Conference on Signals Systems Decision and Information Technology, Mars 2005, Sousse, Tunisia.

[24] L. Boussaid et. al., *"Conception d'un descripteur de Contenu Vidéo sur une Architecture à base de **FPGA** Virtex",* Journées Francophones sur l'Adéquation Algorithme Architecture (JFAAA'05), du 18–21 Janvier 2005, Dijon-France.

L. Boussaid received his Master in **NTSID** (Nouvelles Technologies des Systèmes Informatiques Dédiés) from the National School of Engineering of Sfax, Tunisia (2003) and his Diploma in Electrical Engineering from the National School of Engineering of Monastir, Tunisia (1989).
Since 1990, he was a computer systems engineer in **ENIM** School. Currently, he is a PhD student. His research interests include hardware design of multimedia video content descriptors for real-time applications.

A. Mtibaa received his PhD degree in Electrical Engineering at the National School of Engineering of Tunis. Since 1990 he has been an assistant professor in Micro-Electronics and Hardware Design with Electrical Department at the National School of Engineering of Monastir.
His research interests include high level synthesis, rapid prototyping and reconfigurable architecture for real-time multimedia applications.

M. Abid is a professor of electrical engineering at Sfax University in Tunisia. He holds a Diploma in electrical engineering in 1986 from the University of Sfax in Tunisia and received his PhD degree in Computer Science in 1989 at the University of Toulouse in France. His current research interests include Hardware–Software codesign, design space exploration and prototyping strategies for real-time systems.

M. Paindavoine received the PhD in electronics and signal processing from the Montpellier University, France, in 1982. He was with Fairchild **CCD** Company for two years as an engineer specializing in **CCD** sensors. He joined the Burgundy University in 1985 as "Maitre de Conferences" and is currently a full professor and member of **LE2I**, the laboratory of Electronic, Computing and Imaging Sciences, Burgundy University, France. His main research interests are image acquisition and real-time image processing. He is also a member of **ISIS** (a research group in signal and image processing of the French National Scientific Research Committee).