A Novel FPGA-based Implementation of Time Adaptive Clustering for Logical Story Unit Segmentation

Sutjipto Arifin
Imperial College London
Circuits and Systems Group
Department of EEE, London (UK)
sutjipto.arifin@imperial.ac.uk

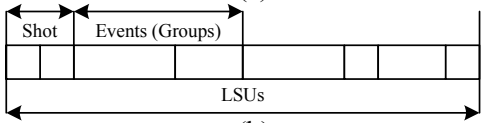
Peter Y. K. Cheung
Imperial College London
Circuits and Systems Group
Department of EEE, London (UK)
p.cheung@imperial.ac.uk

Abstract

Time Adaptive Clustering (TAC) is a cognitive Logical Story Unit (LSU) segmentation algorithm that is found to show good and consistent results. This paper presents an efficient hardware implementation for approximating the TAC algorithm. The design consists of three main blocks. The first block generates similarity values needed in the clustering process. To take full advantage of the parallelism of Field Programmable Gate Arrays (FPGA) devices, a video shot sequence is divided into subsets and processed in parallel by the second block. The third block combines all the output results of each subset. The design is implemented on a Xilinx Virtex-II xc2v3000 on board a RC203E board and it runs 27 times faster than a Pentium 4-based PC at 3.4 Ghz.

	Temporal Distance Function	
	Binary	Continuous
Sequential Clustering	Overlapping Links	Video Coherence
	Time Constraint Clustering	Time Adaptive Clustering

(a)



(b)

Figure 1. (a) Classification of LSU segmentation techniques [8] and (b) Hierarchical Model of Movie Structure.

1. Introduction

The advancement in VLSI technology, broadband networks and compression standards, is spurring the creation and handling of increasing high volume of digital video data. This leads to an explosive growth of visual information available in the form of multimedia archives. As a consequence, sophisticated technology for modeling of multimedia data is necessary. Although many algorithms and systems had been proposed, a significant impact is yet to be generated in the marketplace.

Generally, video content perception can be differentiated into low-level, cognitive and affective level. Most of the hardware work done so far has focused on low level features and little research effort has been invested in the higher levels. This paper describes a novel method of approximating the Time Adaptive Clustering (TAC) algorithm using a Xilinx Virtex-II xc2v3000 on board a RC203E board. The segmentation performance, speed and area performance have been evaluated for the design. This paper's contributions include a novel method of approximating the TAC algorithm (modTAC), a modified evaluation criteria based on [8] and the proposed architecture of modTAC.

The rest of the paper is organized as follows. A survey of related work is presented in section 2. Section 3 provides some background knowledge of the TAC algorithm. In section 4, the modified TAC (modTAC) algorithm is described. In section 5, the proposed architecture is introduced. Section 6 describes the method of evaluation and presents some performance evaluation results. Concluding remarks are presented in section 7.

2. Related Work

Segmenting videos into shots are considered as a low level video analysis. While shot boundary detection organizes video content at the syntactic level, high level parsing provides natural segmentation of video that viewers can associate with [2]. In high level video parsing, Logical Story Units (LSU) of videos are to be searched for. There are

3-9810801-0-6/DATE06 © 2006 EDAA

generally four types of LSU segmentation techniques (Figure 1(a)), Overlapping Links [3, 5], Video Coherence [4, 7], Time Constraint Clustering [9, 6] and Time Adaptive Clustering [4, 10].

Techniques that employ binary temporal distance functions are more sensitive to the choice of threshold. Techniques that employ continuous functions are less sensitive because of the inverse proportionality of visual distance and temporal distance. Sequential comparison techniques perform pair-wise shot distance comparisons, whereas clustering comparison techniques perform group-wise shot distance comparisons. The TAC algorithm is currently the best method to segment a large collection of video [8].

3. Background

3.1. Definitions

A movie structure has a hierarchical model which consists of three levels: Shots, Events (groups) and LSUs. A shot is an unbroken sequence of frames recorded in a single camera. Events are defined as series of shots unified by location or incident. LSUs are defined as a series of temporally contiguous shots characterized by links that connect shots with similar visual content elements [2, 3]. A movie can be understood as a concatenation of LSUs (Figure 1(b)).

3.2. Time Adaptive Clustering

The TAC algorithm can be summarized as follows. Given a shot set {shot 0,1,2,$\cdots$,N_{shots}}, shot 0 is initialized as a member of group 0 and LSU 0 in iteration 0. In iteration i, shot i is compared with the latest members of groups {group 0,1,2,$\cdots$,N_{gps}}. Shot i belongs to a group which has the highest similarity value that exceeds a threshold. If no such group exists, shot i is placed in a new group N_{gps}+1 which is then compared with existing LSUs {LSU 0,1,2,$\cdots$,N_{LSUs}}. The similarity comparison is done between shot i of group N_{gps}+1 and the average similarity of each LSU's group members. Group N_{gps}+1 belongs to the LSU that has the highest similarity value which exceeds a second threshold. Group N_{gps}+1 is placed in a new LSU N_{LSUs}+1 if no such LSU exists.

3.3. Problem Formulation

Shot are compared based on their visual and activity distances. The shot activity measure provides a means of measuring the tempo of the shot and can be computed by:

$$A_i = \frac{1}{N_i - 1} \sum_{k=1}^{N_i - 1} d(k, k+1) \quad (1)$$

where $d(k, k+1)$ is the visual distance between frames k and $k+1$ and N_i is total number frames in shot i. The Hue (H) and Saturation (S) histogram intersect distance is used as the distance measure, given by:

$$d(k, k+1) = 1 - \sum_H \sum_S \frac{h_k(H,S) - h_{k+1}(H,S)}{\sum_H \sum_S h_k(H,S)} \quad (2)$$

where $h_k(H,S)$ is the HS histogram of frame k. The activity distance between a shot pair is given by:

$$ASim_{ij} = max(0, \varphi_{ij}) \times |A_j - A_i| \quad (3)$$

$$\varphi_{ij} = 1 - \frac{0.5 \times (b_j + e_j) - 0.5 \times (b_i + e_i)}{LShot_{ave} \times LLSU_{ave}} \quad (4)$$

where $LShot_{ave}$ is the average shot length in frames, $LLSU_{ave}$ is the average LSU length in shots, b_i and e_i are the first and last frame of shot i. b_i and e_i are keyframes of shot i. Note that although there are other more sophisticated keyframes extraction methods, most of them have very high computational costs and using them might not provide tremendous improvements. The interested reader is referred to [1] for a good review on keyframes extraction techniques.

The visual distance of a shot pair is determined by first computing the raw visual differences between the keyframe pairs (b_j, b_i), (b_j, e_i), (e_j, b_i) and (e_j, e_i) using 1 - $d(k, k+1)$. The raw similarity values are adjusted by temporal attraction constants given by:

$$Attr_{xy} = max(0, 1 - \frac{d(x,y)}{LShot_{ave} \times LLSU_{ave}}) \quad (5)$$

where $d(x,y)$ is the frame index difference between the keyframes. The maximum of the adjusted shot color similarities are selected to be the final shot color similarity $CSim_{ij}$.

The overall shot similarity can be calculated as a weighted sum of the normalized $CSim_{ij}$ and $ASim_{ij}$. The weights can simply be obtained determining the standard deviations of $CSim_{ij}$ and $ASim_{ij}$. The weights are given by:

$$W_c = \frac{\sigma_c}{\sigma_c + \sigma_a}, W_a = \frac{\sigma_a}{\sigma_c + \sigma_a} \quad (6)$$

where σ_a and σ_c are the standard deviations of the activity and visual distances respectively. The overall shot similarity can be computed using the following equation:

$$SSim_{ij} = W_c \times \frac{CSim_{ij} - \mu_c}{\sigma_c} + W_a \times \frac{ASim_{ij} - \mu_a}{\sigma_a} \quad (7)$$

where μ_a and μ_c are the mean values of $CSim_{ij}$ and $ASim_{ij}$. $SSim_{ij}$ values are used by the algorithm to perform Time Adaptive Clustering.

4. The modTAC

Despite the good performance, the TAC algorithm has very high computational cost. In order to take advantage of the parallelism of FPGA devices, a video shot sequence can be partitioned into subsets and processed in parallel. However, since the clustering result of each iteration is very dependent on the existing clustered structure, a brute force partition and process method will greatly degrade the final segmentation output. The challenge is to design a suitable combiner that introduces minimum degradation.

The combining algorithm can be implemented by taking advantage of the TAC algorithm's nature. Since performance degradation only occurs near the partition boundaries, similarity comparisons are only necessary between the first and last LSU of a structure pair. Given a set of output structures $\{ST_0,ST_1,\cdots,ST_N\}$, compute the similarity distances between the first shot members of all the groups in LSU 0 of ST_{i+1} and the latest shot members of all the groups of the last LSU of ST_i. Since it is safe to assume that shots belonging to the same group also belong in the same LSU, the LSU structure of ST_{i+1} can be updated once a similar shot is located. Group structure updating is slightly more complicated than the LSU case because shots belonging to different groups may belong to the same LSU. The group updating process is described in section 5.3.

5. Proposed Design for modTAC

The proposed design consist of three fundamental blocks. The description of each block's architecture are presented in the following sections.

5.1. Block 1 Architecture

The architecture of block 1 is depicted in figure 2. During the visible scan period, each pixel is first subjected to a RGB to HSI color space conversion. $FIFO$ 0 and $FIFO$ 1 are used to store HSI histogram values used to compute the visual distance of each frame. These distances are computed by $CSim_{ij}/Asim_{ij}Calculator$ and accumulated in register act_acc. $FIFO$ 2 to $FIFO$ 5 consists of RAMs used to store histogram values of keyframes for the past $LLSU_{ave}$ shots. This process is controlled by $ControllerB0$, which consists of a set of counters that generate addresses and read/write status to the RAMs.

During the non-visible scan period, $ControllerB0$ checks whether the frame count is on a shot boundary. Shot boundary indices are stored in cut_index_1 and cut_index_2. If the test is true, the value stored in act_acc is divided by the difference between the values read from cut_index_1 and cut_index_2 (total frame count in the shot). The result reflects the activity measure of the shot and is stored in act_i_mat.

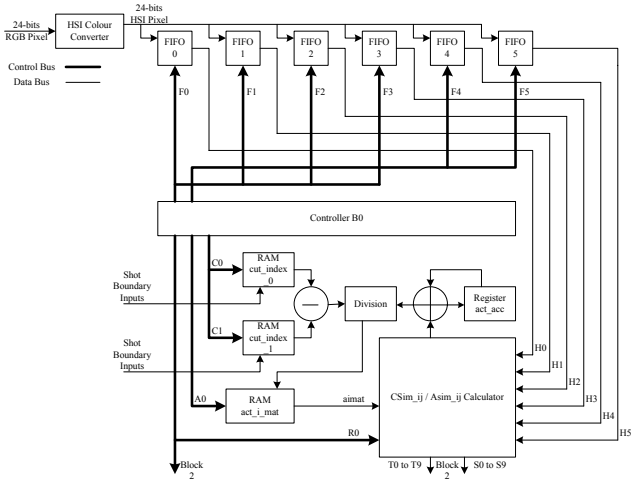


Figure 2. Architecture of Block 1.

The $CSim_{ij}/Asim_{ij}Calculator$ reads histogram and activity values from the FIFOs and act_i_mat to compute the similarities between the current shot and the past $LLSU_{ave}$ shots. The $CSim_{ij}/Asim_{ij}Calculator$ has $LLSU_{ave}$+1 calculator blocks that run in parallel, each capable of computing distances between a shot pair. $ControllerB0$ ensures that the computed values are written to correct storage locations in block 2.

5.2. Block 2 Architecture

Block 2 consists of parallel clustering blocks that are able to perform clustering to a video shot subset (figure 3). The similarity values stored in $shotActSim\_store$ and $shotColorSim\_store$ are used to estimate the mean and standard deviation of the similarity values. These values are also reused in $findGroupSim$, $findSceneSim$, $UpdateStructure$ and $findSCGpEnt$ to perform the clustering. $ControllerB1$ generates addresses and control signals to control the overall operation.

The general operation can be summarized as follows. For each shot, $ControllerB1$ instructs $findGroupSim$ to compute the similarities between the shot and the existing groups. The highest value is tested to see if it exceeds a preset value. If the test is true, $UpdateStructure$ will update the group and LSU structure stored in $Group\_mat$ and $Scene\_mat$. If the test is false, various counters in $ControllerB1$ are updated and $findSceneSim$ is instructed to compute the similarities between the shot and the existing LSUs. This value is compared with a sec-

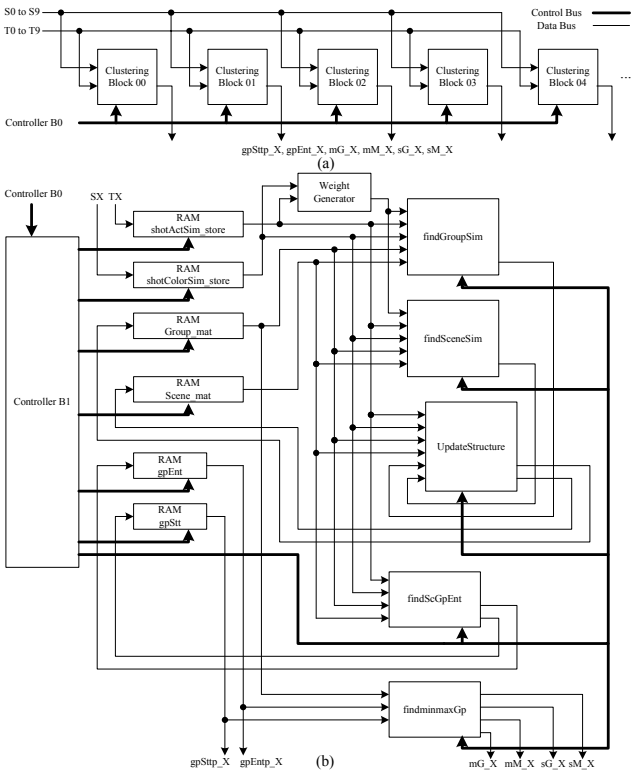


Figure 3. (a) Architecture of Block 2 and (b) Architecture of a Clustering Block within the modTAC Structure.

ond predefined value and $Group\_mat$ and $Scene\_mat$ are updated by $UpdateStructure$ accordingly depending on the test result. Once the clustering is completed, $findSCGpEnt$ and $findminmaxGp$ generate control signals (mG_X, mM_X, sG_X and sM_X) that are used in block 3 for addressing.

5.3. Block 3 Architecture

The architecture of block 3 is depicted in figure 4(a). Each combiner block is capable of combining a pair of subset structures in parallel. For example, given a set of structures $\{ST_0$,ST_1,ST_2,$ST_3\}$, $CombinerBlock$ 00 and $CombinerBlock$ 01 combine $\{ST_0$,$ST_1\}$ and $\{ST_2$,$ST_3\}$ respectively in parallel in the first time step. In the next time step, $CombinerBlock$ 00 will combine $\{ST_1$,$ST_2\}$.

The basic operation of each combiner block can be described as follows. Consider the case where the $CombinerBlock$ 00 is attempting to combine $\{ST_0$,$ST_1\}$. (mG_0, mM_0, sG_0, sM_0) and (mG_1, mM_1, sG_1, sM_1) from $ClusteringBlock$ 00 and $ClusteringBlock$ 01 are used by $ST0GpCt$ to generate control signals that ensure correct similarity values are read by $findGroupSim$ and $findSceneSim$ to compute the group and LSU distances. The distance values are sent to $ScUpp$ to generate the updated structure data. Control signals are also generated to ensure that the right addresses of $Group\_mat$ and $Scene\_mat$ of $ClusteringBlock$ 01 are updated with the new data.

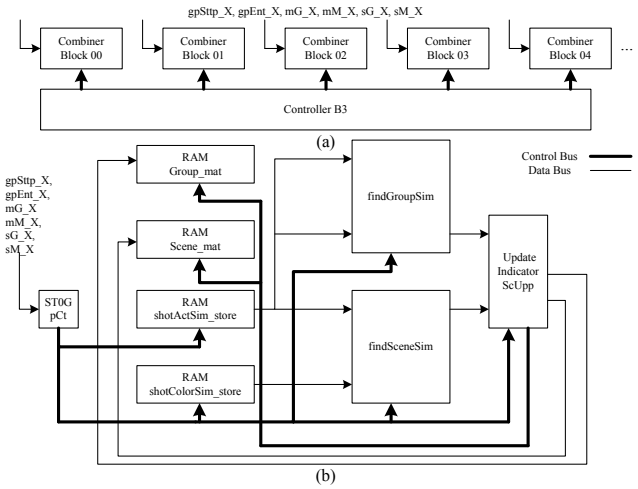


Figure 4. (a) Architecture of Block 3 and (b) Architecture of a Combiner Block.

The addressing process in the combining stage is illustrated in figure 5. Each address location of $Group\_mat$ and $Scene\_mat$ represents the shot number. The contents in $Group\_mat$ and $Scene\_mat$ corresponds to the group and LSU that the shot belongs to. For example, address 0 of ST_1 $Group\_mat$ and $Scene\_mat$ shows that shot 0 belongs to group 0 and LSU 0. The following paragraph is dedicated to the description of the slightly more complicated group combining process.

Shot 0 and shot 1 are the first shot entries of group 0 and 1 of the first LSU of ST_1. These shots are compared with last shot entries 1, 2, 8, 7 and 9 of groups 21 to 26 of the last LSU of ST_0 respectively. In this example, group 0 of ST_1 is found to be similar to group 24 of ST_0. The contents of address locations 0 and 2 are updated by adding the content difference (24 - 0). The contents of the remaining ST_1 address locations are updated by adding the the largest value in $Group\_mat$ of ST_0 (26).

6. Results

The described architecture has been implemented using the Handel-C language. The design is mapped on a Xilinx

Before Combining Process

Last LSU Group_mat Structure of ST_0	21	22	23	21	24	25	21	24	26
Last LSU Scene_mat Structure of ST_0	8	8	8	8	8	8	8	8	8
1st LSU Group_mat Structure of ST_1	0	1	0	1					
1st LSU Scene_mat Structure of ST_1	0	0	0	0					

After Combining Process

Last LSU Group_mat Structure of ST_0	21	22	23	21	24	25	21	24	26
Last LSU Scene_mat Structure of ST_0	8	8	8	8	8	8	8	8	8
1st LSU Group_mat Structure of ST_1	24	27	24	27					
1st LSU Scene_mat Structure of ST_1	8	8	8	8					

Figure 5. Illustration of Combining Process.

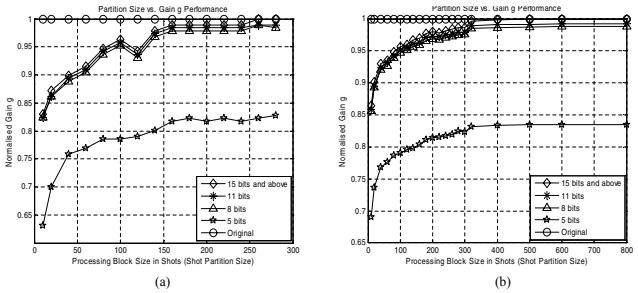


Figure 6. (a) Evaluation Curve for Test Movie 08 and (b) Average Curve of 13 Test Movies.

Virtex-II xc2v3000 on board a RC203E board. It's performance is evaluated based on its segmentation, speed and area performance.

6.1. modTAC Segmentation Performance

Users usually have doubts about the exact start and end of a LSU. The evaluation criterion should therefore measure "how incorrect the boundary is" instead of measuring "if the boundary is incorrect" [8]. The evaluation method presented in [8] is a powerful criterion which allows comparison of different LSU segmentation techniques. However, the number of false positives and negatives are not of great importance in their criteria. Since the proposed design performs clustering to smaller shot subsets, false positives and negatives might be introduced. The number of false positives and negatives therefore becomes an important issue and the criterion has to be modified.

Denote ε as the total number of shot boundaries, ζ as the total number of ground truth boundaries, η as the total number of segmentation result boundaries and θ as the number of boundaries in the segmentation result that is equal to the ground truth. The modified evaluation criterion can be formulated as follows:

$$g = \frac{(\zeta - \theta) + (\eta - \theta)}{(\varepsilon - \zeta)} \tag{8}$$

The first term of the numerator corresponds to the number of false positives, the second term of the numerator corresponds to the number of false negatives and the denominator corresponds to the worst case segmentation (every shot is a LSU).

The evaluation is performed by first evaluating the software version of the TAC algorithm using 13 full length movies as test data (3× cookery programs, 3× Star Trek Enterprise, 3× Frasier, 1× Cartoon, 1× Eight Simple Rules, 1× Constantine, 1× Star Wars Episode II, total 6518 shots, 370 LSUs). The evaluation is repeated for the proposed design for different number precisions.

It is observed that the degradation increases exponentially when the subset size is too small. This is because of two reasons. Firstly, a smaller subset size will result in a greater loss in coherence due to the partitioning of the shot sequence. Secondly, as the subset size approaches the average LSU length ($LLSU_{ave}$ = 10), more false positives are introduced during the combining process. This is because the probability of a single LSU being partitioned into two or more subsets is higher.

The evaluation curve shows that a reasonable performance (>90%) can still be achieved with clustering blocks that can process 20 shots (×2 of the average LSU length in shots) using 8-bits number representations.

6.2. Speed Comparisons

A speed comparisons of the current work with the software implementation has been presented in figure 7. The software implementation has been performed in MATLAB on a Pentium 4 work station (3.4 GHz, 1 GB RAM), and the computation time to process a 35000 frame movie with an average shot length of 100 frames is 118 minutes and 41 seconds. The clustering process takes about 50 seconds. The throughput rate is 0.05 shots per second.

The proposed design runs at a maximum frequency of 40 MHz, with a throughput of approximately 1.3 shots per second. The comparison is done by setting the number of clustering blocks to 10 and frame size of 640 × 480. The time taken for the design to process the same movie is approximately 4 minutes 30 seconds, which is about a 27 times speed up (Figure 7(d)). The clustering process has a speed up of as high as 1,000 times depending on the number of shots required to be processed in each clustering block (Fig-

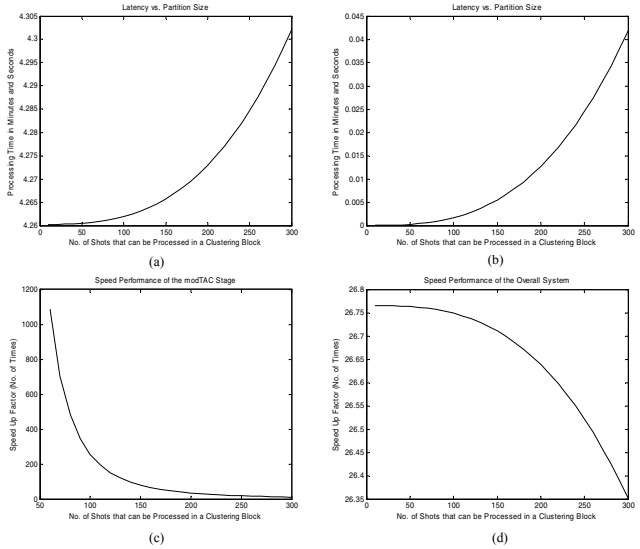


Figure 7. Time Taken to Process a 35000 Frames Movie for (a) the Whole Design and (b) Block 2. Speed up for (c) the Clustering Stage and (d) the Whole Design.

ure 7(c)). The bottleneck of the proposed design is in block 1 because it takes time to scan and process the pixels.

6.3. Area Analysis

Figure 8 depicts the area occupied by the proposed architecture that consists of 10 clustering blocks and 5 combiner blocks. It is observed that the area increase is gradual as the number of shots that can be processed by a clustering block increases. This is because most of the storage elements in the design are implemented as dual-port RAMs (DRAMs), which are more efficiently packed into Look-Up Tables (LUTs). For a design consisting of 10 clustering blocks that can process 20 shots each (200 shots in total) and using 8-bits wordlength distance values represention, the area occupied is approximately 18%. Note that the Xilinx Virtex-II xc2v3000 has 14,336 slices, 28,672 LUTs and 96 block RAMS (BRAMs).

7. Conclusions

Time Adaptive Clustering is a computationally intensive algorithm requiring many keyframe comparisons in every iteration. The proposed modTAC is able to approximate the TAC algorithm with minimum degradation. The proposed architecture presented outperforms the software implementation in terms of computation time and throughput rate. Future work includes incorporating affective content analysis capabilities to the design, and using this as a primary block for a video summarization system.

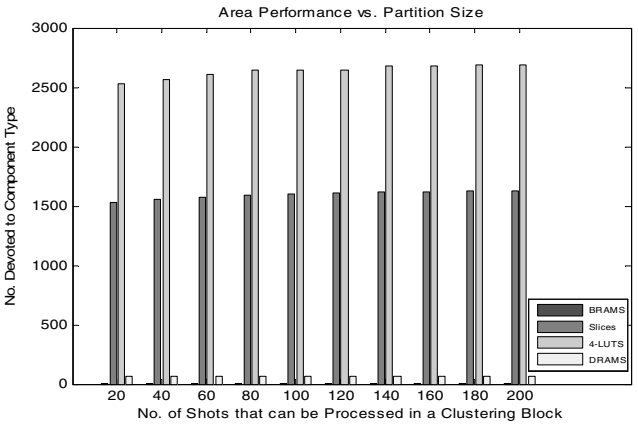


Figure 8. Area Utilization.

References

[1] A. Hanjalic. *Image and Video Databases: Restoration, Watermarking and Retrieval*. Elsevier Science B.V., Delft, Netherlands, 2000.

[2] A. Hanjalic. *Content Based Analysis of Digital Video*. Kluwer Academic, Delft, Netherlands, 2004.

[3] A. Hanjalic and R. Lagendijk. Automated high level segmentation for advanced video retrieval systems. *IEEE Trans. Circuits and Systems for Video Technology*, 9(4):580–588, June 1999.

[4] J. Kender and B. L. Yeo. Video scene segmentation via continuous video coherence. *Proc. IEEE Computer Vision and Pattern Recognition*, 1998.

[5] Y. M. Kwon, C. J. Song, and I. J. Kim. A new approach for high level video structuring. *IEEE Int. Conf. Multimedia and Expo*, 2:773–776, 2000.

[6] R. Lienhart, S. Pfeiffer, and W. Effelsberg. Scene determination based on video and audio features. *Proc. 6th IEEE Int. Conf. on Multimedia Systems*, 1:685–690, 1999.

[7] H. Sundaram and S. F. Chang. Determining computable scenes in films and their structures using audio visual memory models. *Proc. 8th ACM Multimedia Conf., Los Angeles, CA*, 2000.

[8] J. Vendrig and M. Worring. Systematic evaluation of logical story unit segmentation. *IEEE Trans. on Multimedia*, 4(4):492–499, December 2002.

[9] M. Yeung, B. L. Yeo, and B. Liu. Segmentation of video by clustering and graph analysis. *Computer Vision and Image Understanding*, 71(1):94–109, 1998.

[10] R. Yong, T. Huang, and S. Mehrotra. Constructing table of content for videos. *Multimedia Systems, Special Section on Video Libraries*, 7(5):359–368, 1999.