AVSA: An automatic video segmentation application

Tiago H. Trojahn
Institute of Computer Sciences and Computational Mathematics
Av. Trabalhador São-Carlense 400
São Carlos, São Paulo
ttrojahn@icmc.usp.br

Rudinei Goularte
Institute of Computer Sciences and Computational Mathematics
Av. Trabalhador São-Carlense 400
São Carlos, São Paulo
rudinei@icmc.usp.br

ABSTRACT

Segmentation is an important preprocessing step for a number of current multimedia applications using video, like recommendation, personalization or indexing. Since manual segmentation is prone to interpretation error and are time demanding, researchers concentrate efforts in developing automatic segmentation methods.

Automatic techniques need a number of technical input parameters, requiring specialists to be operated. Moreover, these techniques need the specialist to give a threshold, used to decide when a shot or scene transition occurs. Obtaining an adequate threshold is time consuming and mostly an empirical process. The precision is greatly affected by particularities of the input video, so, an inadequate threshold can lead to over or under-segmentation.

To addresses these problems, this paper presents a friendly user application developed in Java which has two main contributions: perform video segmentation using both an automatic method for calculate the needed threshold and a heuristic to overcome some gradual shot transitions issues. The application, named AVSA, uses the video histogram intersection or histogram absolute differences to perform the segmentation. Furthermore, performance tests are presented in order to testify the precision and the recall of the application when segmenting newscast videos.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Shot Detection

General Terms

Algorithms

Keywords

Shot detection, video segmentation, application

1. INTRODUCTION

In recent years there has been a huge development of video content authoring, due to lower prices for cameras, laptops and other devices which can create videos. This resulted in a large amount of video data, available in a wide range of storages, like video streams sites as YouTube[1] and vimeo[2].

That situation leads to a problem: How, in this vast amount of data, can we find a specific video segment of interest? One approach to answer that question is to segment the video in smaller pieces which can be addressed and indexed through some directives, like the presence of specific actors, the action being recorded and so on. Segmentation is a key step for important domains like multimedia information retrieval, content-based video retrieval, summarization and personalization [1].

A traditional way to segment is using experts to manually segment the video. They watch the video at least one time, possible dozens of times, and manually perform the indexing using the boundaries of the desired video segment. Unfortunately, the manual segmentation is labor intensive, making it a non-pratical solution [12].

To avoid these problems, an automatic video segmentation technique can be used. The literature contains a number of these techniques [2, 4, 8, 10]. In general, these methods focuses in segmenting the video in shots, defined as an "unbroken sequence of frames taken from one camera" [7], or "scenes", which do not have a consensual definition. Among these techniques, the most usual method to perform video segmentation is the frame-to-frame histogram comparison [3]. The histogram comparison stands out for being simple, invariant to camera movement and has low computational cost [11].

Unfortunately, little effort [6, 12] was done to develop tools which can be used by non-specialists users. Most of the implementations is not available for general use, have a confusing interface requiring the user to set dozens of parameters, or even need some adjusts in the source code and a posterior compilation process. Our tool, on the other hand, was developed to avoid these needs and thereby facilitating its use.

One of these input parameters is the threshold. The threshold value is of the utmost importance: an incorrect value can lead the technique to perform poorly, giving many false positives and negatives. Then, to free the user to input the threshold value, can be used a fixed threshold for all possible video inputs. That approach can lead to poor results

[1]http://www.youtube.com/
[2]http://vimeo.com/

because the appropriate threshold varies from domain to domain of the input video. Even videos of the same domain can require a specific threshold to achieve at least acceptable results. An adequate threshold is normally discovered through several manual empirical tests, which demands time and a prior ground truth. Our tool, however, can generate a good approximate threshold through an automatic adaptive threshold method for a given input video.

This work presents the AVSA (stands for *Automatic Video Segmentation Application*), which can be effectively used by users to perform an automatic video segmentation. The program can segment a large number of input video formats in terms of shots, returning to the user a set of possible outputs, like a XML description of the shot boundaries or the frames for each shot.

This paper is organized as follows: the **Section 2** presents the application and his features. The **Section 3** presents some details of the application architecture. Finally, in **Section 4**, are presented our conclusions and some future works.

2. THE APPLICATION FEATURES

The AVSA fundamental feature and main goal is to provide a simple method to segment a video file in shots using an automatic threshold.

To operate, the application requires only a video input file. Most of the video formats and video coding procedures are supported by the application, being limited to formats and CODECS which are installed in the users computer and which are supported by the OpenCV[3] library. Since the expected output may vary from user to user, AVSA offers:

- A XML description of the video input showing the transitions boundaries.
- All frames of the video, grouped in shots, using the lossy JPEG or the lossless PNG formats.
- A video sequence for each shot of the input video.

The AVSA most basic mode of operation consists in 1) select the desired output, 2) select the segmentation procedure (intersection and/or differences), 3) select the input video and 4) click on the "Process" button. To relieve the user of repeating the steps 1 and 2, the program comes with default settings loaded at run-time.

The user can also adjust some parameters of the AVSA segmentation procedure. The options relate to the threshold used in both segmentation methods: it can be used a specific threshold or a technique which automatically calculates the threshold. For default, both intersection and differences threshold method are set to automatic with default boundaries defined. Also, the gradual transitions heuristics is enabled by default. More details of the automatic threshold and the gradual transitions heuristics are presented at **Subsection 3.2.1** and **Subsection 3.2.2**, respectively.

Other feature of AVSA application is the Comparing tool, available at the "Tools" menu. The Comparing is a simple and useful tool for developers allowing to compare a segmentation file against a ground truth. The tool requires a ground truth input file and the file to be compared, the latter being provided by the AVSA itself, as a result of a video segmentation. The results are show in terms of True Positives, False Positives, False Negatives, Precision, Recall and F1-Score [5].

Another available tool at the "Tools" menu are the Manual Creating tool. With that tool, the user can manually insert the boundaries of the shots and export it to a valid XML, which can be used in the Comparing tool as a ground truth, for example. The aim of this tool is to ease the need to manually create a XML or develop a separate application only for that.

Both the AVSA and his tools uses the TRECVID[4] *shotBoundaryReferenceSegmentation* Document Type Definition (DTD). Additionally, the AVSA XML provides more informations like threshold used and heuristics strength value, useful for testing purposes, for example.

3. THE PROPOSED APPROACH

This section presents the AVSA implementation at **Subsection 3.1**. At **Subsection 3.2** is presented the segmentation technique as well as the tests performed in order to measure the tool efficiency.

3.1 Implementation details

The application is divided into six distinct classes which performs different functionalities.

The main class, named *GUI*, was developed using the well-know Java Swing API, being responsible to present the user interface responsible to manage users' interactions. The other classes are instantiated in appropriate moments to perform specific tasks.

The *ConfigsHandler* class is responsible to read and write the configuration files and adjust the graphic user interface according to them. This class is called when an user saves the actual configuration, load the default configuration or even when the application is launched.

The *Segment* class is responsible to, given an input video, creates a shot transitions index of the video. The class was developed with the JavaCV[5] library, which provides the OpenCV functionalities to Java language. After the processing is done, another class, named *OutputHandler*, is called to create and organize the desired output to the user (the XML description, shots frames and/or the shot video sequence).

The *Comparing* class represents the Comparing tool and is responsible to open two XML description files and to compare them: the test file against the ground truth file.

The Manual Creating tool, represented by the class *ManualCreating*, simply receives a set of pairs of integer values and creates a XML description file which can be read by the Comparing tool, for example.

The main processing of the application, the segmentation technique, is described in the **Subsection 3.2**.

3.2 Segmentation technique

The main processing of the application is segmenting videos in shots performed by the *Segment* class.

First, the video is opened and a frame grabber is created. For each frame decoded is calculated it's HSV histogram [6] using 8 bins for H, 4 for S and 4 for V [9]. The resulting histogram is stored and the frame is deallocated. Was used the HSV histogram because it's presents higher precision

[3]http://opencv.willowgarage.com/

[4]http://trecvid.nist.gov/

[5]http://code.google.com/p/javacv/

values at the same recall when comparing with the RGB color space [6] and the 8:4:4 quantization was adopted to follow the basis technique [9]. It's important to mention that the technique itself isn't limited by most video details like resolution, frame rate or video length.

After decoding all frames with corresponding histograms extraction, AVSA performs comparisons with adjacent histograms using histograms intersection and/or histograms absolute differences, generating a histogram differences array. This array is then normalized in order to contain values in the range [0,1], in case of histogram intersection, or [0,2] in case of histogram differences. In the first case, 1.0 means that the two histograms are equal to each other and in the second case 2.0 means the two histograms does not have any similarity.

The shot detection is then performed analyzing the differences values calculated and stored in an array with a desired threshold. If the k-value, a value in the array, is lower than the threshold, in the case of histogram intersection, or higher than the threshold, in case of histogram differences, the k and $k+1$ frames are detected as possessing a transition between then. If consecutive transitions are detected, the algorithm merge then, forming a gradual transition.

This shot segmentation technique is based on a technique know as Backward Shot Coherence (BSC) [9], where the authors use the shot segmentation as initial step towards the scene segmentation. Our implementation, however, presents two major contributions: the automatic threshold calculation, presented in the **Subsection 3.2.1**, and the gradual transitions heuristics, presented in **Subsection 3.2.2**. Furthermore, the segmentation based in the histogram absolute differences were not used in the BSC technique.

3.2.1 The automatic threshold calculation method

The automatic threshold calculation method is defined in **Equation 1**. We defined two values, called lower bound (LB) and upper bound (UP), which are used as references to calculate the average values between then. The V_i represents the i-value of the differences array and K the number of frames which have values between LB and UP.

$$Threshold = \frac{\sum_i^K V_i}{K} \qquad \forall i \mid LB \leq V_i \leq UP \qquad (1)$$

If the K value are 0, i.e. no value was been found between LB and UP, the threshold used is equal to $LB * 1.2$.

The **Equation 1**, in other words, calculates the average values needed to detect gradual transitions which values falls between the LB and UP. The LB value must be small enough to be considered as the minimum similarity between two adjacent frames of the same shot, in case of histogram intersection, or the minimum dissimilarity of two adjacent frames of different shots, in case of histogram absolute differences. Furthermore: if the i-value is lower than LB, the i and $i+1$ frames are automatically considered as being of different shots, in case of histogram intersection, or as being of the same shot, in case of histogram absolute differences.

The UP value, on the other hand, has different meanings to histogram intersection and to histogram absolute differences. In the first case, the value must be high enough to represent all possible transitions, i.e., no shot transition can have value above UP. In the second case, the UP value must be high enough to represent gradual transitions, but can not represent abrupt transitions. That arises from the fact of the abrupt transitions often have overly high values, which in turn would rise the threshold value, ignoring the gradual transitions which values are closer to the LB value.

Adequate values for LB and UP were determined through several performance tests in the newscast domain. The default LB and UP values for the intersection are 0.5 and 0.9 respectively. For the absolute differences, the default values are 0.3 and 0.9, respectively. The obtained precision and recall of the automatic threshold calculation method using these values is presented in **Subsection 3.2.3**.

The LB and UP are the only needed parameters for video segmentation in AVSA. These values can be defined only once and, even so, the application comes with default LB and UP values, making them effectively transparent to the users. The default LB and UP values are adequate to newscast domain: adequate values can change to other domains.

3.2.2 The gradual transitions heuristics

Techniques which compare adjacent frames can detect changes comparing the similarity/dissimilarity against a threshold. If the threshold is exceeded, a transition, classified as an abrupt transition, are flagged. However, modern digital videos often presents gradual transitions: in these transitions, the similarity decrease and the dissimilarity increases over time. These techniques, under such circumstances, can over-segment the video, detecting several abrupt transitions when there is actually a single gradual transition.

One way around this problem is merging consecutive abrupt transitions in a single and larger gradual transition. However, some complex gradual transitions can shows subtle changes spanning over several frames, resulting in a set of gradual and abrupt transitions when there is a single gradual transition. To avoid these outliers, we created a heuristics to detect such gradual transitions.

When a transition is detected, we search the N (N is the size of the search window) previous histogram absolute differences or histogram intersections to see if a shot transition was detected. If there is a shot transition in those frames, all frames from the last transition up to the actual frame are considered as belonging to the same gradual transition. The default number of searched frames, called "strength", is 3. Although a high "strength" value does not influence in the abrupt detection, if a shot length are smaller than the window size, the shot will not be indexed properly.

3.2.3 Evaluation tests

For the evaluation tests, we used a set of videos obtained through direct TV signal capture. The video domain were newscast transmitted at different channels. Some video details, like name, duration, number of frames and number of transitions, are presented in **Table 1**. The video set was digitalized using 720x480 progressive resolution at 30 fps. The commercial breaks were removed.

Table 1: Some details for each video used in the evaluation tests

Video Name	Duration	# frames	# transitions
NewsA	24:09	43470	317
NewsB	23:07	41610	278
NewsC	1:04:51	116730	808

In terms of shot transition type, the video set presents

a larger amount of abrupt transitions when compared with gradual transitions. The NewsC, however, presents about the same rate of abrupt and gradual shot transitions, reflecting a slight lower precision, recall and F1 score than with the NewsA and NewsB videos.

First, we built a ground truth base for the shot transitions. Then the results of the segmentations presented by our application was compared with the ground truth, resulting in the precision, recall and F1 values presented in **Table 2** and **Table 3**. The tests results were obtained through the Comparing tool, using plus and minus 3 frames as frame error tolerance for gradual transitions. In these tests, the gradual heuristics strength was set to 3.

The **Table 2** presents the results of the segmentation using the histograms intersection with the automatic threshold method presented in **Equation 1** with lower and upper bounds values at 0.5 and 0.9, respectively.

Table 2: Precision, recall and F1 score obtained with histogram intersection

	NewsA	NewsB	NewsC
Precision	89.57%	85.76%	71.28%
Recall	92.11%	86.69%	81.43%
F1	90.82%	86.22%	76.02%

The **Table 3** presents the results of the segmentation using the histograms absolute differences with the automatic threshold method presented in **Equation 1** with lower and upper bounds values at 0.3 and 0.9, respectively.

Table 3: Precision, recall and F1 score obtained with histogram absolute differences

	NewsA	NewsB	NewsC
Precision	97.30%	89.25%	84.43%
Recall	68.45%	80.93%	65.09%
F1	80.37%	84.90%	73.51%

The results show a higher precision when using the histogram absolute differences when comparing with the histogram intersection obtained results. The histogram intersection, on the other hand, presented a significantly higher recall value.

4. CONCLUSIONS AND FUTURE WORKS

In this paper we presented the AVSA application, which can be used both by casual users and also by developers or experts to easily segment videos in shots. The features and the architecture of the application was presented. Also, the techniques used by the application was described, presenting also a set of precision and recall tests.

As future works, we intent to add support to other comparison techniques, like Euclidean Distance, in order to measure similarities/dissimilarities. We also plan to apply AVSA to other video domains.

Is also planned add a help assistant in the application. Useful information like the explanation of the lower and upper bounds and the constraints of the application could ease the AVSA usage and improve the results.

Another possible future work is the development of tools which can simplify the burden of specific tasks such as XML editing or even to visualize the results.

5. ACKNOWLEDGMENTS

We would like to thanks to the FAPESP for the financial support of this work.

6. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17:734–749, June 2005.

[2] E. Bruno and D. Pellerin. Video shot detection based on linear prediction of motion. In *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on*, volume 1, pages 289 – 292 vol.1, 2002.

[3] P. Geetha and V. Narayanan. A Survey of Content-Based Video Retrieval. *Journal of Computer Science*, 4(6):474–486, June 2008.

[4] A. Hampapur, T. Weymouth, and R. Jain. Digital video segmentation. In *Proceedings of the second ACM international conference on Multimedia*, MULTIMEDIA '94, pages 357–364, New York, NY, USA, 1994. ACM.

[5] X.-S. Hua, D. Zhang, M. Li, and H.-J. Zhang. Performance evaluation protocol for video scene detection algorithms. In *Workshop on Multimedia Information Retrieval, in conjunction with 10th ACM Multimedia*, 2002.

[6] S. Jeong. Histogram-based color image retrieval. Technical report psych221/ee362, Stanford university, Mar. 2001.

[7] I. Koprinska and S. Carrato. Temporal video segmentation: A survey. *Signal Processing: Image Communication*, 16(5):477–500, 2001.

[8] S. Manjunath, D. S. Guru, M. G. Suraj, and B. S. Harish. A non parametric shot boundary detection: an eigen gap based approach. In *Proceedings of the Fourth Annual ACM Bangalore Conference*, volume 1 of *COMPUTE '11*, pages 14:1–14:7, New York, NY, USA, 2011. ACM.

[9] Z. Rasheed and M. Shah. Scene detection in hollywood movies and tv shows. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II – 343–8 vol.2, june 2003.

[10] K. Sawai, T. Takahashi, D. Deguchi, I. Ide, and H. Murase. Scene segmentation of wedding party videos by scenario-based matching with example videos. In *Proceedings of the 19th ACM international conference on Multimedia*, volume 1 of *MM '11*, pages 1545–1548, New York, NY, USA, 2011. ACM.

[11] M. Swain. Interactive indexing into image databases. In *In Storage and Retrieval for Image and Video Databases*, pages 95–103, 1993.

[12] H. Zhong, L. Wenyin, and S. Li. Interactive tracker - a semi-automatic video object tracking and segmentation system. In *Multimedia and Expo, 2001. ICME 2001. IEEE International Conference on*, pages 1167 –1170, aug. 2001.