A fast video shot segmentation tool

Tiago Henrique Trojahn
University of São Paulo
Federal Institute of São Paulo
São Carlos - SP - Brazil
ttrojahn@icmc.usp.br

Rodrigo Mitsuo Kishi
University of São Paulo
Federal University of Mato Grosso do Sul
São Carlos - SP - Brazil
rodrigokishi@usp.br

Rudinei Goularte
University of São Paulo
São Carlos - SP - Brazil
rudinei@icmc.usp.br

ABSTRACT

The video shot segmentation is a useful step on multiple video-related tasks, including research ones like automatic video summarization or high-level segmentation, along with video production like editing and/or post-production.

Although it can be useful, there is not an user-friendly automatic video shot segmentation application: those whose can be used requires either paid subscriptions, advanced user knowledge or has an complex installation process.

That way, this paper presents an efficient video shot segmentation with an user-friendly web-interface. The application, along with its OS independent interface, features a fast segmentation process with a number of customizable parameters to achieve a high-quality shot segmentation.

Keywords

Shot segmentation, multimedia

1. INTRODUCTION

Video shot segmentation is often regarded as an initial step for various videos analysis, including indexing, summarization and high-level scene segmentation [1].

Although automatic shot segmentation is often regarded as "essentially solved" [2], with a plethora of techniques available in literature, there seems to be a lack of an easily usable application which performs shot segmentation. Some of the available applications seems to be usable only by researchers or experts, as Video Shot and Scene Segmentation[1], ImageLab Shot Detector[2], FFmpeg[3] or the Automatic Video Segmentation Application (AVSA) [3] tools. Most of these do not provide user-friendly interface, require comprehension of the implemented method to specify adequate thresholds or programming skills to properly compile and install it.

That way, this paper focuses on describing an user-friendly application with a web interface which enables the average user to easily perform automatic video shot segmentation. More than that, the application also enable advanced users to customize the segmentation itself, which can improve the obtained results.

This paper are organized as follows: Section 2 presents the proposed shot segmentation application along with its performance evaluation; Section 3 describes the web-interface and its features; Section 4 presents some of the application discussions and envisioned future works.

[1]http://mklab.iti.gr/project/video-shot-segm
[2]http://imagelab.ing.unimore.it/
[3]https://ffmpeg.org/

In: Sessão de Pôsteres do Workshop de Ferramentas e Aplicações, 15., 2016, Teresina. Anais do XXII Simpósio Brasileiro de Sistemas Multimídia e Web. Porto Alegre: Sociedade Brasileira de Computação, 2016. v. 2.

ISBN: 978-85-7669-332-1

2. SHOT SEGMENTATION APPLICATION

The proposed application aims to identify shot transitions, both gradual and abrupt ones, for a given input video. The approach consist on analysis of HSV 8:4:4 color histograms extracted from the input video. By using two similarity measures, euclidean distance and histograms' intersection, along with two independent sliding-windows with variable size (obtained by means of high adjacent frames dissimilarity), the technique can identify most of abrupt shot transitions.

As some transitions doesn't have high dissimilarity between then, local thresholds are automatically calculated within each independent sliding window, based on average dissimilarity distance value. These thresholds are used to identify more subtle transitions within each sliding window, including most of the gradual shot transitions. An example of 3 histogram intersection and 2 euclidean distance sliding-windows' with its local thresholds is presented in Figure 1.

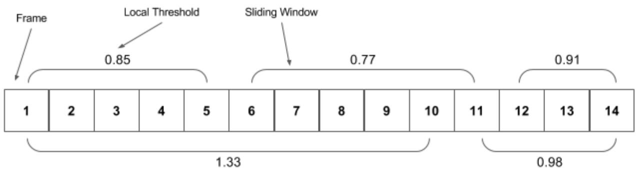

Figure 1: An example of independent sliding-windows and local thresholds over 14 frames.

To detect gradual transitions a heuristic is applied: consecutive transitions which are distant by a predefined amount of frames are merged into a single gradual transition.

2.1 Performance evaluation

The shot segmentation were implemented using a multi-thread approach in the C++ language with the OpenCV 3.0 library. The library provide most of required low-level data for the developed technique like video decoding, color conversions and histogram' extraction and comparisons. The implementation, along with its simple CLI interface is also public available at `https://github.com/Trojahn/FAST`.

To evaluate the segmentation performance, it was used the 24fps 10m Big Buck Bunny open movie[4] to measure the obtained processing fps rate and time with different H.264/AVC video resolutions. The results, described on Table 1, were obtained using an Intel Core i7 x990 (12 logical threads) with 16GB of RAM running a x64 Ubuntu 16.04.

Table 1: Average obtained FPS and processing time from the proposed implementation when shot segmenting the 10m Big Buck Bunny video with different video resolutions.

Video configuration	FPS	Processing Time
480p (SD)	752.8	00:19
720p (HD)	340.5	00:42
1080p (FullHD)	152.1	01:34
2160p (4K)	38.9	06:10

The obtained processing time shows that the implementation could achieve real-time even on most demanding video configuration at 2160p (also know as "4K"). In other words, the application can process more frames than the video fps rate, making it adequate even for live video content.

Another interesting result is that processing time gets a huge boost on smaller resolution videos: the 480p video could be processed more than 30 times faster than the video duration itself. This indicates that the application can segment large video datasets within a reasonable time, which could benefit video providers like YouTube, Facebook and TV channels, for example.

3. WEB INTERFACE

In order to facilitate the access of users which are interested in the proposed tool, a web based user interface was developed. This interface was programmed using Meteor[5], an open source platform for web, mobile and desktop applications. Meteor platform follows the client/server architecture and its applications are usually written in Javascript with HTML and CSS.

As Javascript is originally a client side language, Meteor uses Node.js[6] to run its server side code. Meteor projects come with embedded MongoDB[7] databases at the time of their creation. MongoDB is an open source document-oriented database system, which is interesting for some kinds of applications because of its flexibility.

The web based user interface allows multiple file uploads, where each uploaded video can be segmented one time with standard or custom parameters, generating an `.csv` output file containing the frames of the shot boundaries which can be downloaded from the server.

The shot segmentation process, described on the previous section, occurs asynchronously on the server. When the segmentation process is finished, the interface updates itself, showing the resulting csv file, without any user interaction. A screenshot of the web interface with three already uploaded files is shown on Figure 2.

The web-interface implementation is public available at `https://github.com/rodrigokishi/FASTweb`.

[4]http://bbb3d.renderfarming.net/download.html
[5]https://www.meteor.com/
[6]https://nodejs.org
[7]https://www.mongodb.com/

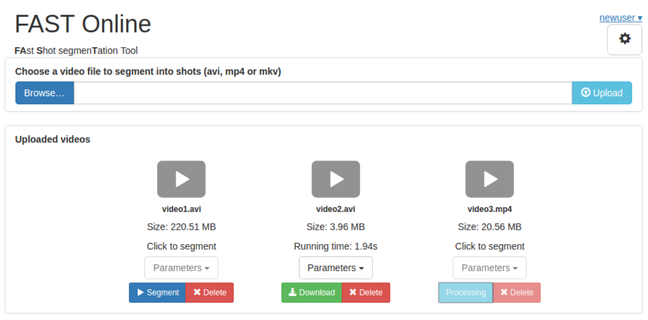

Figure 2: A screen capture of the web based user interface.

4. DISCUSSIONS AND FUTURE WORKS

This paper proposed an application, along with its web-based interface, which can perform a fast video shot segmentation. The adopted shot segmentation process, based on frame-to-frame color histogram dissimilarity, obtained high performance values in our evaluation being able to even segment 2160p videos (often regarded as "4K") in real-time.

Some of the main advantages of the proposed application is that it can ease shot segmentation tasks by providing the users with a friendly interface and a cost-effective video shot segmentation. Also, the shot segmentation parameters can be customized by the user to obtain superior results given particular users needs.

As future works, some of the devised improvements over the application and its web-interface are listed below:

- A set of selectable output formats like XML and JSON;
- An notification system which could notify the user when the video segmentation finished;
- Web interface back-end optimizations which may improve the overall user experience, like queueing to support multiple fast segmentation tasks;
- Couple a video player on the web interface, allowing the user to watch the uploaded videos. This player should also have support to show the predicted shot boundaries of each video;
- Adapt the web version of the user interface and produce mobile and desktop versions, using the `meteor` capabilities;

5. REFERENCES

[1] L. Baraldi, C. Grana, and R. Cucchiara. Shot and scene detection via hierarchical clustering for re-using broadcast video. In *CAIP*, 2015.

[2] M. Del Fabro and L. Böszörmenyi. State-of-the-art and future challenges in video scene detection: a survey. *Multimedia Systems*, 19(5):427–454, 2013.

[3] T. H. Trojahn and R. Goularte. AVSA: An automatic video segmentation application. In *Anais dos Workshops e Mini Cursos do WebMedia*, volume 1, pages 71–74, 2012.