SHOT DETECTION ALGORITHM

Helena Cots, Paula Castillo

ABSTRACT

In this paper, we try several shot detection algorithms in order to learn and compare them by visualizing the results obtained with each one. We have proposed two discriminant features to work with and three similarity measures to compare those extracted features.

Keywords - shot change, detection, features, metrics, threshold.

1. INTRODUCTION

In a video, a shot is the longest continuous sequence from a single camera take, meaning sequence of images in an interrupted run. [1]

In video processing, there is the need of segmenting the video into smaller units in order to process each unit independently. Shot detection is used to segmentate a temporal video into shots. This type of segmentation is preferable because the video content within a shot tends to be continuous. Therefore, the detection between shots in two consecutives frames requires the computation of an appropriate continuity or similarity metric between those two frames. But there are three main complications.

The first, is defining the continuity metric that must be insensitive to gradual changes in the shot, like camera parameters, lighting, physical scene content, as others, but must be discriminant enough to detect the shot change. One way to do this is to extract one or more feature vectors from the frames of the video and compute distance functions on these feature vectors to compare the frames.

Regarding the first problem, the features to extract can be the luminance or the color of the frame, the histograms, the edges, the motion… Depending on the video one feature might work better than another. These features can be extracted from different parts of the frame: from single pixels, by blocks, arbitrary regions, or the whole frame. This last option is very robust with respect to motion within a shot but poor in performance if shots are similar, which we have seen through this lab.

The second complication is establishing the decision method, which has to decide if the values of the continuity metric correspond to a shot change or not. There are different methods to choose, such as establishing a static threshold, which has to be manually set for each video, an adaptive threshold, within a temporal window, probabilistic detection or a trained classifier.

The third one, and the most difficult one, is the fact that not all shot changes are abrupt. The easier shot changes to detect are the cut ones, because it is an abrupt shot transition. However, there could be gradual shot transitions in a video. The most common gradual changes are the dissolving, the fading, wiping and other transitions types such as special effects.

2. PROPOSED ALGORITHM

The shot detection algorithm must have three main steps: the extraction of the feature, the similarity measure computation and a decision step to decide whether there is a change of shot or not. In order to process our video, we did some decisions about these parameters.

The first step is to choose what features we want to extract from the frames of the video. In this project we choose to compute the histogram of the frames, both color and luminance histograms. In the videos we chose this decision worked well, except on a video where a shot change had the same histogram values (PV2.mp4).

The size of the region from which the features are extracted is important. In this case, we decided to extract the histogram values from the whole frames of the video, because the video we chose has a lot of motion within the shots. However, in order to prove our algorithm, we make the same performance but extracting the feature from a central block in the frames.

Moreover, we thought about which temporal domain of the continuity metric we should use. When using a video where changes between shots are abrupt, it is good to use a continuity metric between two frames.

The similarity measures that we implemented are distance functions. The first one is the Bhattacharyya distance [2], closely related to the Bhattacharyya coefficient. The coefficient can be used to determine how much similar the two frames are. Being $H_i(x)$ and $H_{i+1}(x)$ the histogram values from two frames, the Bhattacharyya coefficient is

$$BC(H_i(x), H_{i+1}(x)) = \sum_{x \varepsilon X} \sqrt{Hi(x) * Hi + 1(x)}$$

and the Bhattacharyya distance

$$D_B(H_i(x), H_{i+1}(x)) = -\ln(BC)$$

from which we can evaluate whether the frames are similar or not.

The second similarity measured that we chose is the Mean Absolute Difference [3], that computes the expected value of two variables, in this case the two frames. This difference is computed as the following:

$$MAD(H_i(x), H_{i+1}(x)) = \int_{-\infty}^{\infty} |Hi(x) - Hi + 1(x)|$$

This measure worked well in our algorithm. Moreover, to better see the difference between the two frames, we used the Mean Square Error [4] similarity measure, which punish the errors better than the MAD measure due to the quadratic factor.

$$MSE(H_i(x), H_{i+1}(x)) = \int_{-\infty}^{\infty} |Hi(x) - Hi + 1(x)|^2$$

For the third condition, we decided to use a static threshold, which we manually set for each video we use. With the most appropriate threshold, the shot detection algorithm works mostly perfect, despiste some shot changes where the histogram is almost the same. Apart from setting the threshold for each video, it must also change if we use MAD or Bhattacharyya distance due to the large difference between values extracted from these two similarity measures.

3. EVALUATION

To evaluate our code, we start from the *ground truth*, the fact that we already know the results we should obtain in case the detection was perfect. This way, we can know and quantify how well our code is doing its job.

We used four components for evaluation: recall, precision, accuracy and F1-score [5]. Before explaining how to achieve those values, due to all them depends on four parameters, we must introduce them with the following table:

		Predicted	
		shot = YES	*shot* = NO
Actual	*shot* = YES	True Positives (TP)	False Negatives (FN)
	shot = NO	False Positives (FP)	True Negatives (TN)

In that case, the *recall* is the portion of the real transitions that have been detected.

$$Recall = \frac{TP}{TP+FN}$$

The *precision* is the amount of detected transitions that are actually transitions.

$$Precision = \frac{TP}{TP+FP}$$

The *accuracy* is the proportion of both transitions and non-transitions that are correctly detected as transitions and non-transitions, respectively.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

The *F1-score* considers both *precision* and *recall* to compute the score.

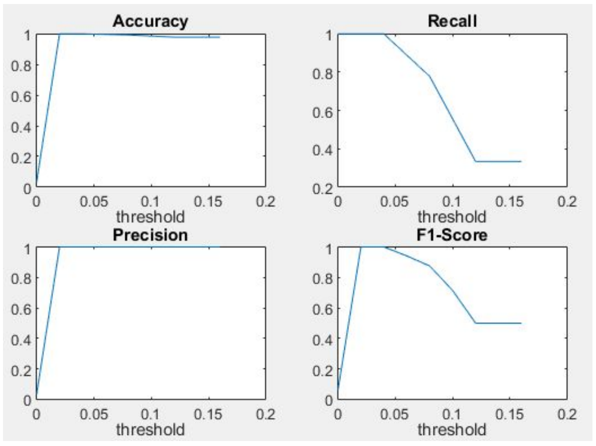
$$F1score = \frac{2 * recall * precision}{recall + precision}$$

The goal is to reach the threshold value that achieves the highest values of those four components that is 1.

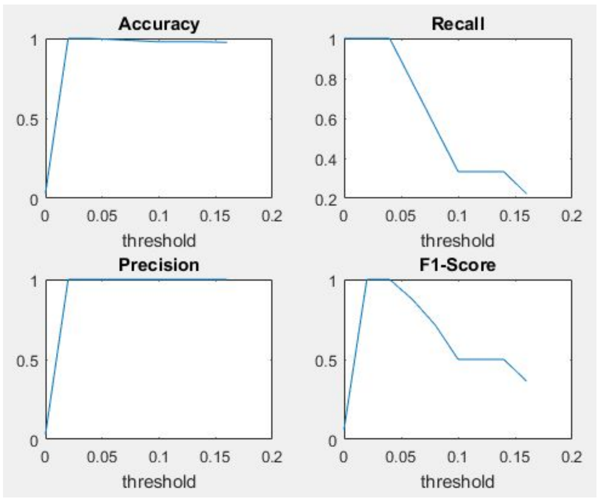
In the graphics below it is shown the recall, precision, accuracy and F1-score related with the threshold value in order to visualize the appropriate threshold we should establish for each of the similarity measures and features used.

1- Bhattacharyya

a) Luminance histogram



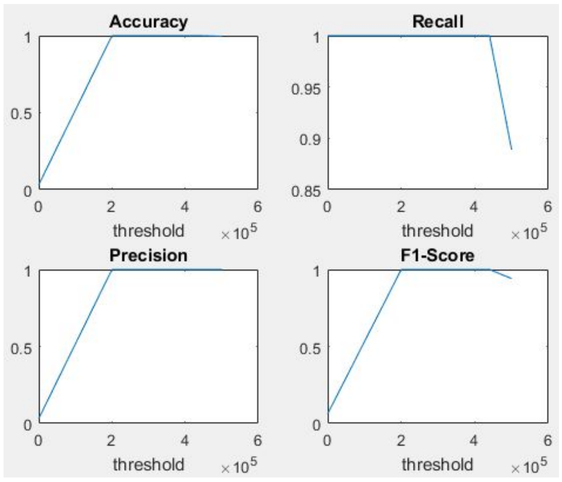
b) Colour histogram



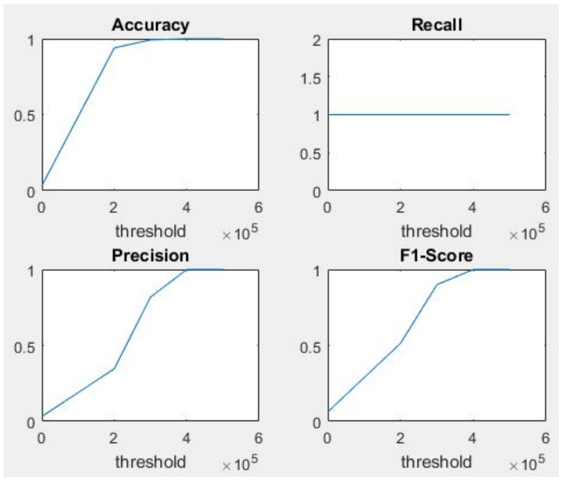
Using the Bhattacharyya coefficients, the threshold value we can extract as the most befitting is equal to 0.04.

2- MAD

a) Luminance histogram



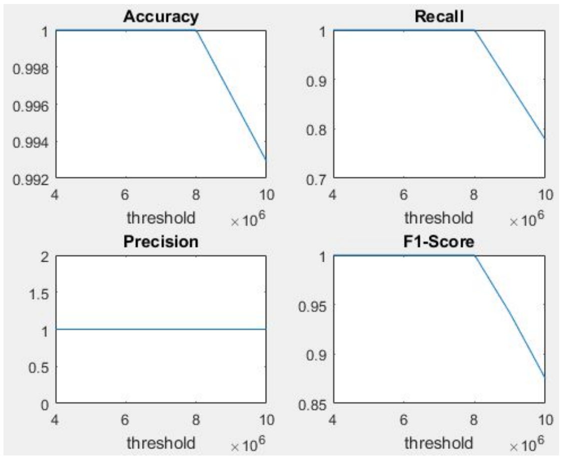
b) Colour histogram



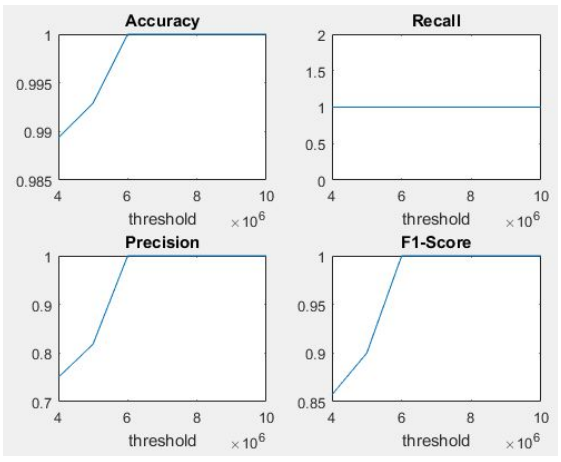
When we use MAD as a similarity measure, the threshold that allows the algorithm to detect most of the transitions is $4.1*10^5$

3- MSE

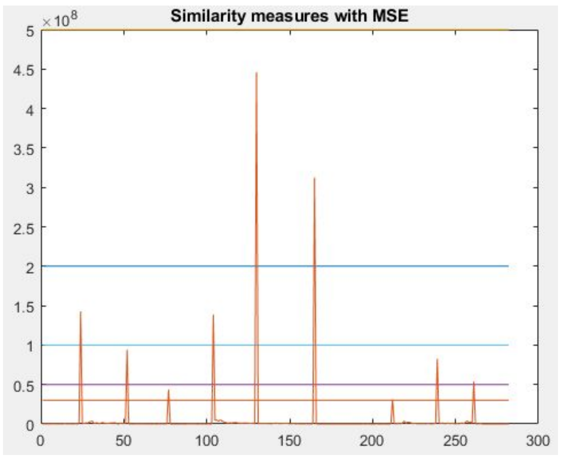
a) Luminance histogram



b) Colour histogram



Finally, the most appropriate threshold value taking as the similarity measure the Mean Squared Error is a value of 30000000, for example. We can see how thresholds decide how much shot changes are in the video:



4. CONCLUSIONS

After going through all the complications, it is certain that defining a perfect shot detection algorithm it is almost impossible due to all the changes that happen within the video such as wipe transitions, other transitions that still for us are difficult to detect, etc.

It seems to work well if the parameters are correct, but it is computational heavy to perform as the way we did. However, with the algorithm we create, it was impossible to detect one shot change where the histograms where exactly the same between the two consecutive frames.

Observing at this results, we arrive to the conclusion that it was very difficult to define a general algorithm for any video. Despite this difficulty, we think that it could be good when we have to segmentate one video into shots and we can focus on the characteristics of this video.

Moreover, defining a good algorithm, shot detection can detect gradual transitions which would be difficult for other type of algorithms.

5. REFERENCES

[1] Costas Cotsaces, Nikos N. and Ioannis P. Video Shot Detection and Condensed Representation. IEEE Signal Processing Magazine. March 2006 URL:file:///C:/Users/TRABAJO/Downloads/video_shot_detection.pdf
[2]Bhattacharyya distance. Wikipedia. URL https://en.wikipedia.org/wiki/Bhattacharyya_distance
[3]Mean Absolute Difference. Wikipedia. URL https://en.wikipedia.org/wiki/Mean_absolute_difference
[4]Mean Square Error. Wikipedia. URL https://en.wikipedia.org/wiki/Mean_squared_error
[5]Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures - Exsilio Blog.URL:http://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures