

**A Multi-Modal System for Soccer Video Summarization**

**Graduation Project** **Progress Report 5**

Team Members

* Ahmed Maher
* Ahmed Salama
* Moamen Hassan
* Mohamed Talaat

Supervised by

* Dr. Magda Fayek

**Audio Version of la5sly**

In this version the video summarization is based only on the audio feature i.e. when the audio level is high that gives an indication of an important event in the football match

**Reason to make this version**: the initial results of the main version showed that theaudio feature is contributing significantly in the summarized final video so making a version with audio is worth the effort and would be much faster than the original version, with that said it’s not without its drawbacks

**Advantages**:

* Very less computations compared to the main version.
* Depends only on the Audio module and the shot boundary module but the main version has more modules to take into considerations.
* Much faster runtime

**Disadvantages**:

* The summarized output of this version is relatively longer in minutes than the main version as in many matches the audiences sometimes have loud voice while cheering even if it’s not really an important event in the game.
* The summarized video can get longer depending on audience and commentator attitude in every match.
* In rare situation, some important events are not included because the audio level is not larger than the audio threshold

**Used Modules Details**

**Used modules:** Audio, Shot Boundary

1. **Audio module Input:** video clip

**Output:** times in video having volume level > 90% of the video volumesAlgorithm:

* Read video clip:
* Extract audio from the video clip:
* Get average volume of each 10 seconds
* Get the difference between every two averages then detect the increases and decreases in volume:
* Determine peaks indices of volumes:
* Get peaks volumes:
* Get Times of peaks having volume level > 90% [largest 10%]

1. **Shot Boundary module Input:** two frames **Output:** cur or no cut

**Algorithm:**

* Get color histograms of the two frames in RGB space
* Calculate histogram intersection and correlation between the two histograms
* If intersection >6 and correlation >5 then no cut
* Divide the two frames into blocks, each block 150px x 150px and calculate intersection and correlation between individual blocks.
* If intersection <4 and correlation <4 then block is 100% changed
* If intersection >4 and correlation <4 then block is 75% changed
* If intersection <4 and correlation >4 then block is 25% changed
* If none of the above conditions are met then the block is not changed
* Count changed block and calculate percentage of changed blocks. 10- If percentage of changed blocks > 30% then Cut else no cut

|  |  |
| --- | --- |
| Histogram intersection | Histogram correlation |
|  |  |
| This method simply compares, for each bin, the two values in each histogram, and keeps the minimum one. The similarity measure is then simply the sum of these minimum values. Consequently, two images having histograms with no colors in common would get an intersection value of 0, while two identical histograms would get a value equal to the total number of pixels. | is based on the normalized cross-correlation operator used in signal processing to measure the similarity between two signals,  cross-correlation is a [measure of similarity](https://en.wikipedia.org/wiki/Similarity_measure) of two series as a function of the displacement of one relative to the other. This is also known as a sliding [dot product](https://en.wikipedia.org/wiki/Dot_product) or sliding inner-product. |

For both Correlation and Intersection methods, the higher the metric, the more accurate the match.

There are other metrics that we have tried but discarded, they will be discussed in the final project document

The two frames below are obviously consecutive frames from the same shot i.e. there is no cut.

that should be reflected in the values of histogram intersection and correlation



*Histogram Intersection = 9.04120311407678*

*Histogram correlation = 9.971235345958373*

The values are higher than the thresholds mentioned in the above algorithm (the thresholds are based on trial and error) so no cut

Here is another example to fully explain how the shot boundary algorithm works.

These are also consecutive frames but there is a transition (cut) between them



*Histogram intersection = 0.24404063194742776*

*Histogram correlation = 0.002219072222348816*

Both values are smaller than the thresholds so we perform step 4 of the algorithm in which dividing the two frames into blocks, each block 150px x 150px and calculate intersection and correlation between individual blocks then, according to the values of intersection and correlation of the blocks to consider whether that block is considered full changed or half changed or not changed at all.

Then, step 9 in which we count how many blocks have changed and calculate the percentage with respect to all block

In the above two frames the percentage of changed blocks is 100% i.e. all frame blocks have has changed so, there is a cut.

Another example to full demonstrate the importance of having two methods.

The following two frames are consecutive but in the midst of a logo transition (gradual transition)



*Histogram intersection = 2.1293090697099615*

*Histogram correlation = 9.844120057485238*

Here the two methods disagree, the intersection indicates that there are minor similarities in the two frames but the correlation indicates that that’s they are variations of each other i.e. the histogram is a shifted version of the other histogram for example.

So, we resort to the blocks process once again to determine if there is a cut or not. The percentage of changed block is 37% which makes sense because there are areas that didn’t change but according to the threshold this is still considered a cut and that is logical because it’s a logo transition.

**Audio Version Algorithm**

**Input:** Video Clip

**Output:** Summarized video clip

**Algorithm:**

1- Read video clip

2- Get Peaks times using Audio module, peaks [i…n]

3- For each peak time peaks[i], get its frame number

4- For each frame get its shot using the shot boundary module.

1- Starting from this frame explore frames descending tell finding a shot cut

= shot start

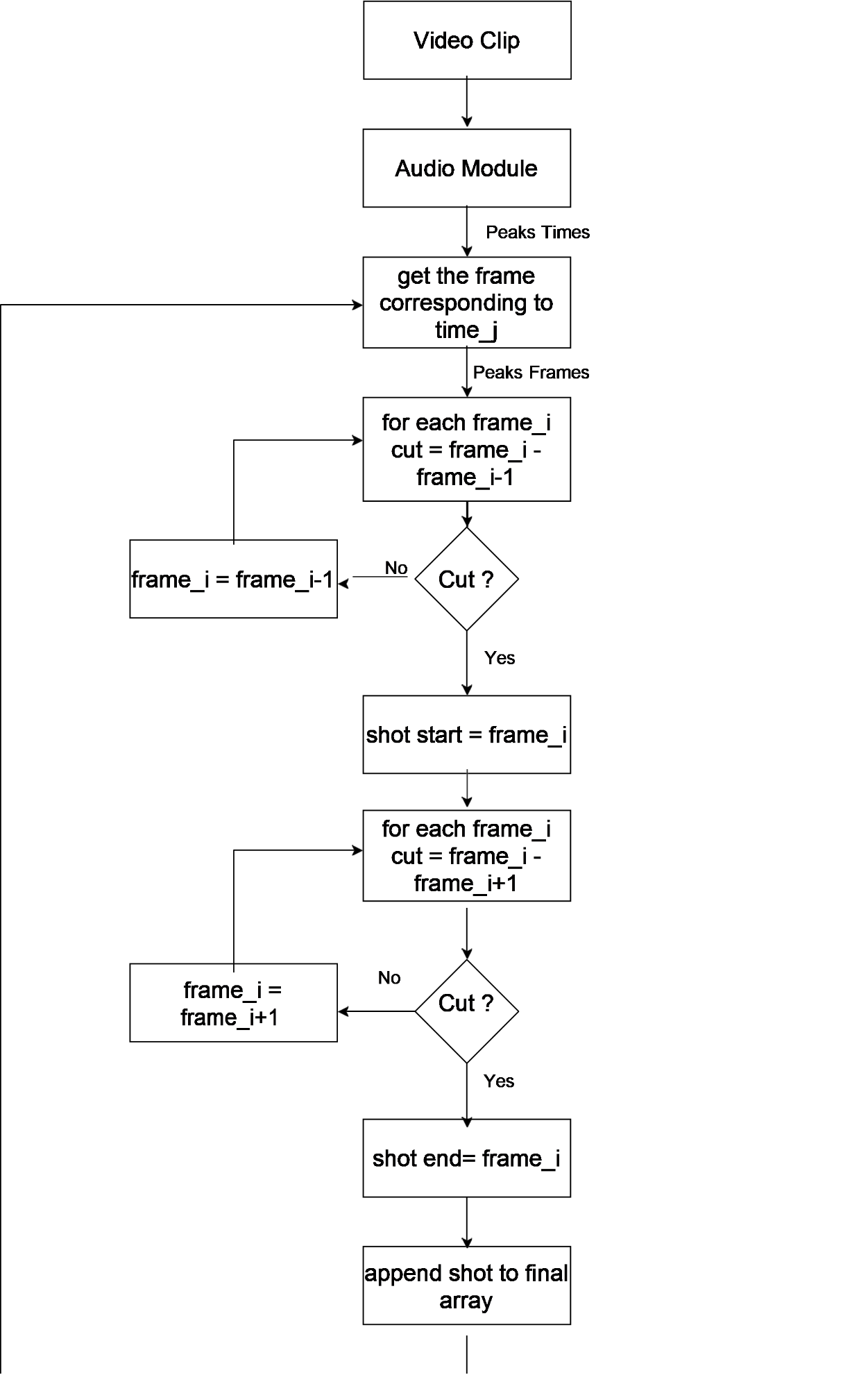
2- Starting from this frame explore frames ascendingly tell finding a shot cut

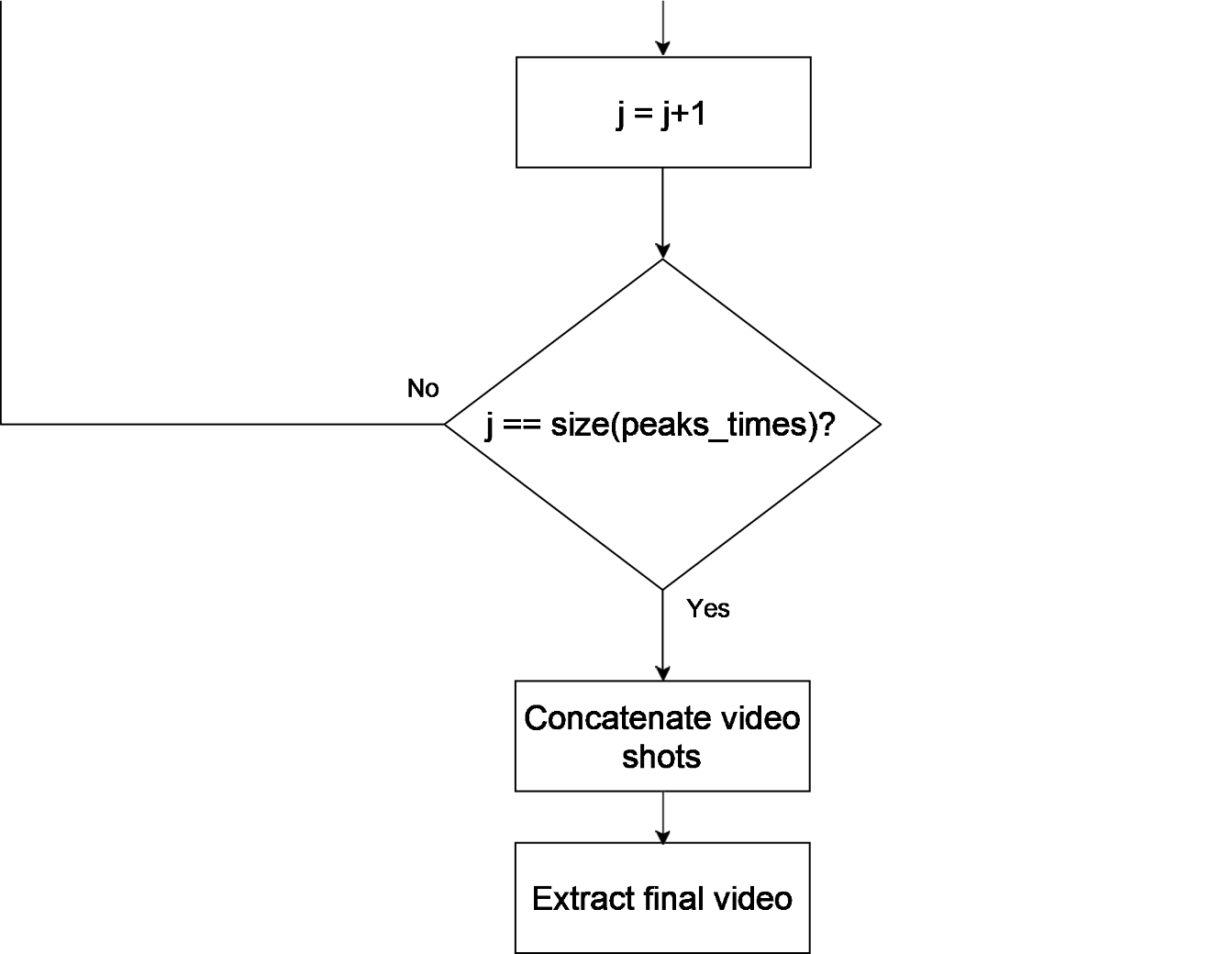
= shot end

3- Having shot start and shot end, append this shot to the final array

5- Concatenate shots into one array

6- Extract the summarized video

****

****

**Results**

To evaluate each audio threshold level in different leagues we calculate a score based on the goals detected, the interesting events detected and also the length of the output summarized video, goals is the most interesting event so we multiply the ratio of the detected goals and total number of goals in the match by 0.6, for other interesting events we multiply it’s ratio by 0.3 and we take the difference of 1 and ratio of length of output video and total length of video and multiply it by 0.1 to give a higher score for smaller length output video, then we add results to obtain the score.

**Equation:**

**Score** = 0.6\***E1** + 0.3\***E2** + 0.1\*(1 - **E3**)

E1: no. Goals detected / total goals

E2: no. Of other interesting events detected / total no. Of events

E3: Length of output video / total length of video (=90)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| League | Match | Audio Level Threshold | Output video Length in minutes | Goals Detected accuracy | Other events detection accuracy | Score |
| BUNDESLIGA | Fc Augsburg  VS  Dortmund | 85 | 28:06 | 4/8 | 7/13 | 0.53 |
| 90 | 10:03 | 3/8 | 3/13 | 0.38 |
| 95 | 8:51 | 2/8 | 0/13 | 0.24 |
| Bayern Munich  VS   1. Bremen | 85 | 16:06 | 6/7 | 16/16 | 0.9 |
| 90 | 12:07 | 5/7 | 14/16 | 0.78 |
| 95 | 8:28 | 4/7 | 11/16 | 0.64 |
| Bayer Leverkusen  VS   1. Dusseldorf | 85 | 20:35 | 3/3 | 16/16 | 0.98 |
| 90 | 15:50 | 3/3 | 10/16 | 0.87 |
| 95 | 6:47 | 2/3 | 6/16 | 0.6 |
| LIGUE 1 | PSG  VS  Monaco | 85 | 31:14 | 6/6 | 9/12 | 0.89 |
| 90 | 23:46 | 6/6 | 10/12 | 0.92 |
| 95 | 14:48 | 5/6 | 7/12 | 0.76 |
| Rennes  VS  Nantes | 85 | 17:28 | 5/5 | 11/14 | 0.92 |
| 90 | 12:38 | 4/5 | 8/14 | 0.74 |
| 95 | 5:01 | 4/5 | 4/14 | 0.66 |
| PSG  VS  Marseille | 85 | 26:27 | 4/4 | 8/10 | 0.91 |
| 90 | 19:44 | 4/4 | 7/10 | 0.89 |
| 95 | 8:35 | 4/4 | 6/10 | 0.87 |
| PREMIER LEAGUE | Man. United  VS  Man. City | 85 | 29:38 | 2/2 | 5/8 | 0.85 |
| 90 | 21:42 | 2/2 | 4/8 | 0.83 |
| 95 | 10:33 | 2/2 | 2/8 | 0.76 |
| Aston Villa  VS  Man. City | 85 | 25:16 | 7/7 | 8/8 | 0.97 |
| 90 | 17:01 | 7/7 | 7/8 | 0.94 |
| 95 | 13:07 | 7/7 | 4/8 | 0.83 |
| Chelsea  VS  Arsenal | 85 | 24:29 | 4/4 | 7/10 | 0.88 |
| 90 | 18:04 | 4/4 | 6/10 | 0.86 |
| 95 | 6:51 | 3/4 | 3/10 | 0.63 |

In the above chart, for each league an average score is calculated for each audio threshold level by adding their scores and divide it by total number of matches (=3).

**Conclusion:**

For most leagues we have tested level 85 threshold has the best results.

In premier league level 90 threshold has an average score very close to level 85 threshold so we can use it instead to reduce the length of the output summarized video.