Automatic Highlights Generation for Cricket Matches

Alina Sarwar

Summary

Cricket is the second most watched sports game in the world. It has a complex set of rules and is played for a longer duration than most other sports. Even the shortest format of cricket, T20, has a 3-3.5 hours long match, on average. It is no surprise that a lot of people, despite being cricket lovers, are unable to keep up with the live matches because of their busy work schedules. Most people are only able to catch fleeting finishes to the matches. Also, once the result of the game is out and people already know who eventually won the game, there is much less motivation left to watch the whole package. Rather, people prefer watching highlights of the game, which includes just the most interesting events of the game. Producing sports highlights manually, especially those of cricket, is a time-consuming and cumbersome task. An efficient method to generate these highlights automatically would not only save manpower but also make sure that the highlights become available for the viewers to watch soon after the match ends.

In this project, we aim to implement an approach introduced in a recent paper by Agarwal et al [1] for automatically generating cricket highlights. The proposed method reformulates the problem of generating cricket highlights as simply annotating each and every ball of the game in terms of whether an important event (eg. a four, a six, a fall of wicket etc) occurred during that ball. It works by segmenting a full match video into individual ball clips using a self-trained Convolutional Neural Network (CNN) to detect the start of the ball and Optical Character Recognition (OCR) for detecting the changes in the scorecard and signaling the end of that ball. Once every ball has been tagged as "interesting" enough or not, the process of generating highlights is then as simple as stitching all such relevant ball clips together into a highlights package.

Proposed Methodology

The proposed approach can be divided into the following steps:

1. Match Video Segmentation.

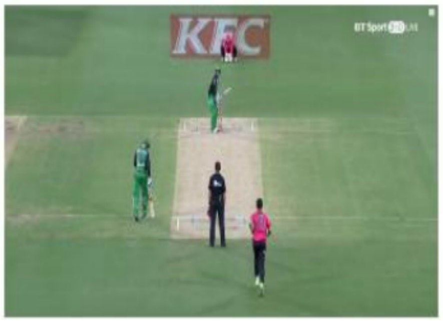
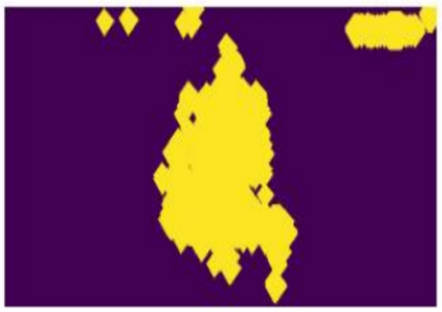
The first step involves breaking down a full match video into separate scenes using an open source command line application called PySceneDetect[1]. The *detect-content* module is used to compare the difference in content between adjacent frames against a set threshold, which if exceeded, triggers a scene cut.

[1] http://scenedetect.com/projects/Manual/en/latest/cli/commands.html

Each scene is represented by three key frames [2], marking the start, middle and end of the scene. This helps us cut down the processing cost as all further steps will only be making use of these key frames so that there is no need to analyze each and every frame of the whole video.

2. Ball Start Frame Detection

In this step, we will train a CNN for a binary classification problem: detecting a ball-start frame Vs a non ball-start frame. The challenge here will be to generate training samples for the CNN since we will have to dig out ball start frame samples from the match videos ourselves. To make our lives a little easier, we will make use of image dilations in a semi-supervised approach to get a pool of potential positive and negative samples for ball-start frames, which we will then manually review to prepare our final training set.

a) Ball-start frame b) Dilated version

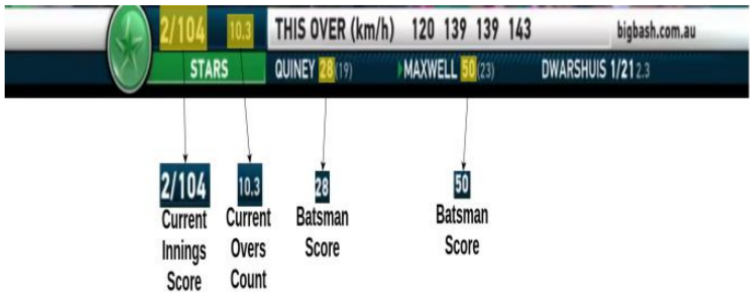
The figure above shows a typical ball start frame where the batsman, non-striker, bowler, wicket-keeper and pitch are all clearly visible. Next to it is shown a dilated version of the same image. This sample dilation will be compared to the dilations of other frames of the same match video to shortlist a set of images that are likely or not likely to be ball start frames. Dilation[2] is a morphological operator that we will apply to an image in order to enhance its primary features and join disparate elements to get a sense of the distribution of foreground objects in the image.

3. Ball End Frame Detection using OCR

To fetch the complete clip of any given ball, we need to find its start and end frames. Start frame will be detected by the CNN classifier. In order to determine the end frame of the ball clip, we will use the Optical Character Recognition (OCR) technique to detect a change in the score count or over count for the current innings. As soon as one of these changes, we can say the ball has finished.

[2] https://docs.opencv.org/3.4/db/df6/tutorial_erosion_dilatation.html

4. Ball Clip Annotation using OCR



Once we have demarcated a full ball clip, we will annotate it using OCR over the scorebox region of the video. All regions of interest including the score count, over count, batsmen's individual scores and wickets lost will be marked once manually before any video processing begins and will stay constant for the whole match.

5. Relevant Clips Filtration and Highlights Generation

Once the whole match video has been annotated, we will generate a highlights clip of the cricket match using the following criteria for a ball to be included in the package:

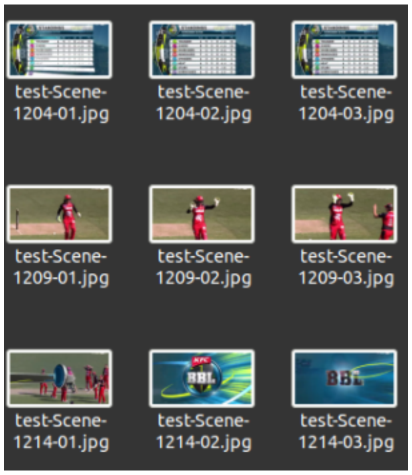
a) Four or more runs scored on that ball
b) A wicket falls on that ball
c) Ball clip is marked as milestone i.e. a batsman crosses 50,100,150… score
d) Ball clip is the last of the innings

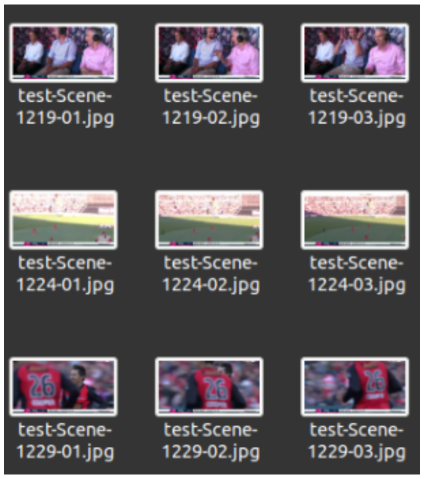
All such ball clips are then stitched together to generate the final highlights clip using the Python library MoviePy[3].

Preliminary Results

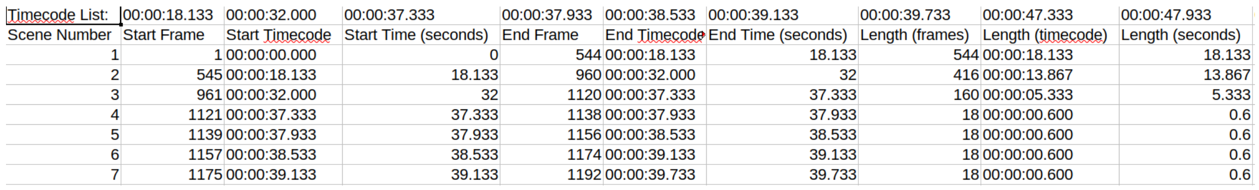
We played around with the different parameters of the PySceneDetect package as part of exploring some of our match videos. Using a hit and trial method, we believe a threshold of 10-12 will be appropriate for triggering fast scene cuts for our matches. Some sample results of generated scenes are shown below:

[3] https://zulko.github.io/moviepy/





We were also able to use *list-scenes* command of the same package to help us write out the results to a CSV file in the form of a table containing all scenes, their start/end timecodes, and frame numbers.

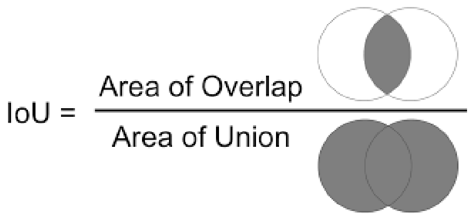


Timecode List:	00:00:18.133	00:00:32.000	00:00:37.333	00:00:37.933	00:00:38.533	00:00:39.133	00:00:39.733	00:00:47.333	00:00:47.933
Scene Number	Start Frame	Start Timecode	Start Time (seconds)	End Frame	End Timecode	End Time (seconds)	Length (frames)	Length (timecode)	Length (seconds)
1	1	00:00:00.000	0	544	00:00:18.133	18.133	544	00:00:18.133	18.133
2	545	00:00:18.133	18.133	960	00:00:32.000	32	416	00:00:13.867	13.867
3	961	00:00:32.000	32	1120	00:00:37.333	37.333	160	00:00:05.333	5.333
4	1121	00:00:37.333	37.333	1138	00:00:37.933	37.933	18	00:00:00.600	0.6
5	1139	00:00:37.933	37.933	1156	00:00:38.533	38.533	18	00:00:00.600	0.6
6	1157	00:00:38.533	38.533	1174	00:00:39.133	39.133	18	00:00:00.600	0.6
7	1175	00:00:39.133	39.133	1192	00:00:39.733	39.733	18	00:00:00.600	0.6

Dataset description and Results Evaluation

We will collect a dataset of ~12 complete T20 match videos, both international and league games. Out of these, 8 will be used to train the CNN and the remaining 4 will be used to evaluate our model. Since the model here has more of a qualitative output, the evaluation of results is certainly going to be a bit tricky. The key point to note here is that the usefulness of a highlights package is largely a subjective measure. Some people may prefer to watch each and every boundary of the game as part of the highlights while others might just want to have the most defining moments of the match and wicket falls included in the highlights.

Nevertheless, we plan to make use of the Intersection over Union (IoU) metric to evaluate our highlights with respect to the highlights provided by the official broadcaster.



An IoU score of 1 denotes that the sets being compared are the same, whereas a score of 0 implies that the two sets are mutually exclusive. IoU will be calculated as the ratio of the number of events common to both our and the broadcaster's highlights and the total number of events in both the packages. It is important to note here that we are only going to limit the scope of our evaluation to the four types of events we described before in step 5 of the proposed methodology. These are the events that are usually deemed to be important enough to be included in a standard highlights package of a cricket game. There will certainly be a degree of manual intervention involved in assessing the quality of the highlights produced, the details of which will be more clear once we progress to the later stages of the project.

Project Milestones

Phase I progress update (25th October)

- Dataset acquisition
- Generation of scenes represented by key frames
- Generation of positive and negative training samples for CNN

End of Phase I

- Training of CNN for detecting ball-start frames
- Start implementing OCR

Phase II progress update

- Complete OCR implementation for ball end frame detection
- Ball clip annotation

End of Phase II:

- Linking all modules together for final highlights generation
- Debugging and code clean ups
- Evaluation of results and reporting

As of now, this project is majorly focused on implementing the approach proposed in the paper and evaluating how well it performs. As the development progresses, we may choose to replace some methods with more efficient ones if we think that could improve our results. If time allows, we might also try implementing additional features such as generating player-specific highlights or customized highlights based on user preferences.

Github repository

References:

[1] Agarwal et al, "Automatic Annotation of Events and Highlights Generation of Cricket Match Videos", in International Journal of Innovative Technology and Exploring Engineering (IJITEE), 2018.
[2] W. Wolf, “Key frame selection by motion analysis,” in Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on, vol. 2. IEEE, 1996, pp. 1228–1231.