FACULTY OF SCIENCE
UNIVERSITY OF COPENHAGEN

DEPARTMENT OF COMPUTER SCIENCE

Project outside course scope

Shot boundary detection from TV newscast

Aksel Krog Samuelsen <gwc553@alumni.ku.dk>

Supervisor
Kim Steenstrup Pedersen <kimstp@di.ku.dk>

Abstract

In this report, we present a method for shot boundary detection, specially developed to the *DR1 18:30 newscast*. The method use a combination of computing the MSE for color images, as well as a k-means model fitted with histograms from the RGB and HSV channels. Our resulting method yielded an F1 score of 0.9662, when predictions were accepted up to 1 second from the ground truth.

Contents

1 Introduction

The Danish Royal Library is responsible for collecting and storing both Danish radio and television. Broadcasts from 2005 and later is available to watch online, and copies of the data can be obtained for research purposes, for journalists, students, and researchers. The Danish Royal Library want to store meaningful metadata along with the recordings, as it would be of great value for researching the broadcasts. Examples of the metadata could be labels describing content in some categories, e.g. politics, society, crime, or the name of a specific person that appears.

We will limit the scope of the project, only to focus on the video data from the newscast "18:30 nyhederne" from the television channel DR1. In order to make the labels more meaningful and accurate, the labels should *not* be assigned to the entire video, but rather, the labels should be assigned to individual *sections* in the in the newscast. A newscast typically contains 10 to 15 sections, where each section is a self-contained news story, i.e. a stand-alone story that has no direct connection to each other section. It often show different scenes and may include several interviews from different persons, however, it all revolves around the same story.

In order to assign labels to each section of the newscast, we will need to know the start and end of each such news-section. In this project, we will attempt to perform shot boundary detection on the video data with the intention that it can be used to divide the newscasts into sections. A shot boundary is the transitions from one shot to another. A sections typically contains many shot boundaries. Furthermore, we will attempt to distinguish regular shot boundaries from longer graphical cut-scene, as they might carry more information about the program. I section 3.3, we will elaborate what a graphical cut-scene is, and how we detect them.

2 Data

The data used in the project is provided by The Danish Royal Library, who is responsible for collecting and storing all television programs from Danish TV. We will work with footage from the "18:30 nyhederne" newscast from the TV-station DR1. The duration of each newscast is approximately 25 minutes long, with 25 frames per second, which makes each video a total of about 37500 frames. There are typically 250 shot boundaries for an entire newscast, which corresponds to a new shot every 6 seconds. The data is in the `.mp4` file format with a 640×360 px resolution.

We will only make use of the image data, however, both a audio and multiple subtitle tracks are available. Furthermore, we will only look at data from the year 2018, as we have only been granted access to this portion of the data. This will ensure that the video data is consistent, as it happens that the newscasts from DR1 change appearance from time to time. This means, that the methods and results presented throughout this report, may *not* represent a general solution for different years, nor does it reflect a general approach for newscast from other TV-stations, as they may use completely different video content.

3 Shot boundary detection

Throughout this section, we will look at what Shot Boundary Detection (SBD) is, and some common methods to detect such shot boundaries.

3.1 A brief introduction to shot boundary detection

A shot is an continuous sequence of frames from the same camera. A shot boundary is any transitions from one shot to another with different visual content, and can be to a different camera or point in time. There are multiple categories of shot boundaries with different structure.

Among the most common ones are *hard cuts* or *abrupt transitions*, where the shot is suddenly replaced by a different shot. This is the most simple shot boundary, as the two shots never appear simultaneously, i.e. the frame number i belongs to the first shot and $i + 1$ to the second.

A *soft transitions* is a gradual transitions between two different shots, where frames from both shots are partially visible at the same time. Frames from both scenes are typically dissolved into one another, or fades from one to the next shot. Figure 1 shows an example of a soft transitions using the dissolve effect. From frame 2635 to 2638, both shots are clearly visible at the same time. A *wipe* is a kind of soft transitions where the old shot is replaced with the new, by the shot sliding across the screen, "pushing" the old shot off screen. However, the wipe cut is rarely used in our data.

Figure 1: *Example of a soft transitions with dissolved effect. The number under each image indicates the frame number. From top left to lower right, we see how the we gradually change from one to another shot.*
The image is from [3]

3.2 Analysis of cuts in the data

In this project, we will perform shot boundary detection on video data from the television newscast "18:30 nyhederne".

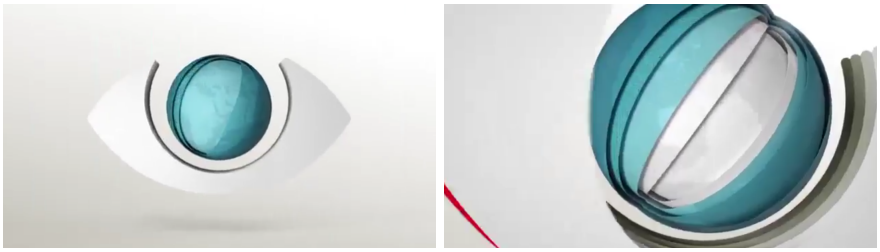
In Section 3.1, we saw different versions of cuts and how they differ. In this section, we will look at the DR1 18:30 broadcasts and investigate which kind of cuts used as transitions between shots.

Figure 2: *Both images are from newscast 26-09-2018. The left have frame number 3899, and the right has 3900.*

Figure 2 shows two frames from one of the newscast. The frame on the left is frame # 3899, while the frame on the right is # 3900, i.e. the frames are back to back in the video sequence. Hence, this is a hard-cut, as the change is sudden and no frames are in between the first and the second scene are both fully visible. This is the case for all cuts throughout the broadcast, except for certain graphical cut-scene which we will discuss further in section 3.3.

3.3 Graphical cut-scene

The newscast contains some special cut-scenes, where the transition between shots are not divided with hard cuts, but rather an animation of the tv-stations logo. Figure 3 is an example of two different frames from the cut-scene. The length of the cut-scene varies from just under one second (23 frames), to almost 7 seconds (170 frames), as well as the visual appearance of the animation changes also throughout the broadcast.



(a) (b)

Figure 3: *Examples of how the the cut scene looks.*

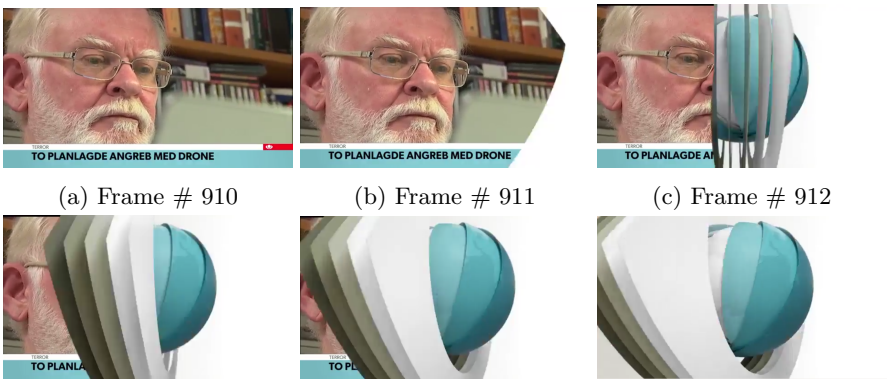
The animations on the cut-scene use a colour scheme, that are used throughout

the entire newscast. The four colours used are, white, grey, cyan, and red, as well as many shades and variations of the four. Figure 4 shows two examples of the colours used in other situations. However, unlike animations used in other parts of the broadcast, the cut-scene is very fast and violent. The speed of the animation also differs from the general tempo of the remaining of shots. The video contains 25 frames per second, which means that if we look at each frame in a sequence after another, they typically look very similar. Whereas the animation changes a lot, making each frame look different from the next.

(a) (b)

Figure 4: *Examples of the colour scheme used throughout the newscast. In the image to the left, we see how the same colours from the cut-scene are used in the background of the studio.*
In the image to the right, the text bar at the bottom of the screen has the same cyan colour as the cut-scene.

At the beginning and at the end of the cut-scene, a soft transitions somewhat similar to a wipe cut is used to change shots. Figure 5 shows an example of how the animation gradually takes over and "pushes" the previous shot of the screen.



(a) Frame # 910 (b) Frame # 911 (c) Frame # 912

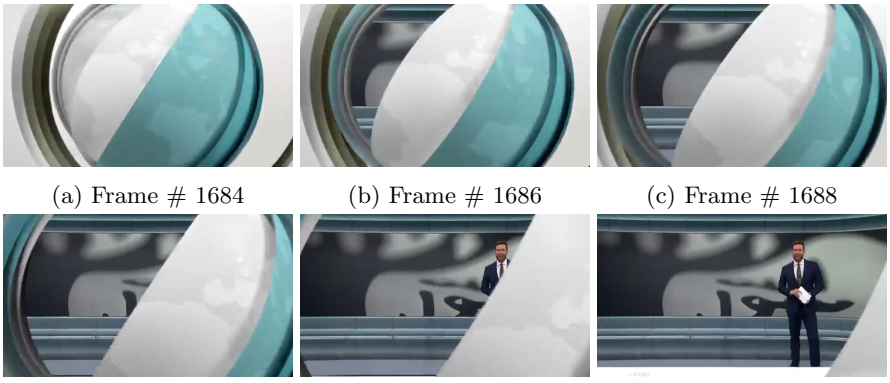
(d) Frame # 913 (e) Frame # 914 (f) Frame # 915

Figure 5: *Example of how the beginning of a cut-scene could look like. In the first image (top left), the first shot is fully visible. In the next five frames, the first shot becomes gradually less visible, as the animation from the cut-scene appears.*

Figure 6 shows an example of how the end of the graphical cut-scene could look

like. The animation is a bit different compared to the beginning of the cut-scene, as it uses a different kind of wipe cut. As we will later see, because of the gradual opening and closing, it is hard to determine which frames we should consider the true start and end of the cut-scene.

In this project, we will only consider the use of video footage for shot boundary detection. However, for future work it could to interesting to extract and process the signal from the audio track, as a specific jingle is played every time the animation is shown. This could help detect the graphical cut-scene, as the cut-scenes is the only time throughout the broadcast where the sound is played.



(a) Frame # 1684 (b) Frame # 1686 (c) Frame # 1688

(d) Frame # 1690 (e) Frame # 1692 (f) Frame # 1694

Figure 6: *Example of how the end of a cut-scene look like. The exact transition may vary from scene to scene. Similar to beginning, the end makes a gradual transition, but in a slightly different way. Parts of the animation "opens up" in the center, revealing the next shot.*
Note that there are two frames between each images.

4 Implementation

As the task of this project is twofold, we will solve each part individually. In section 4.1, we will discuss how to detect and report the majority of the shot boundaries, namely the regular hard-cuts. In section 4.2, we focus on detecting and handling the graphical cut-scene. We will end this section by describing how we have gathered the two individual solutions into a single method for solving the overall problem.

4.1 Detecting regular cuts

As we saw in section 3.2, the majority of the cuts in the data, consist of hard-cuts. The hard-cuts are often fairly simple to detect, as the sudden change between frames are very drastic compared to a soft transitions.

We will use openCV [1] to read and work with each frame in the video data, while making use of NumPy [2] for handling data structures as well as vector and matrix computations.

Two different approaches has been tested to detect shot boundaries. The first consist of comparing intensity histograms of neighboring frames, and the second compute the Mean Squared Error (MSE) between neighboring frames.

4.1.1 Comparing intensity histograms

As suggested in [5], shot boundary detection using intensity histograms in different color spaces is a widely used approach for shot boundary detection. We have experimented with different color spaces, and found no real difference between them. Therefor, when comparing histograms we us RGB as it requires no transformation, as the video data consist of RGB images.

For each frame, we produce a 3D intensity histogram, i.e. a 2D histogram for each channel in the RGB image, and compare it to the histogram of next frame in the video.

4.1.2 Mean Squared Error based similarity measure

As suggested in [4], MSE performes well for detecting hard-cuts, or *abrupt scene change* as they phrase it. We have used a MSE-based approach by pair wise comparing pixel values between adjacent frames. This gives a measure of how similar the frames are to each other. Equation (1) gives the formular for computing the MSE of two images. If the mean squared error exceeds a pre-determined threshold, we report a cut.

$$MSE(x, y) = \frac{1}{nm} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} ||x_{ij} - y_{ij}||^2 \tag{1}$$

where x and y is the two input images, and n and m is the shape of the images. x and y are expected to have the same dimensions. As both x and y is RBG images, we measure the error over the distance of the two RBG vectors.

In order to accommodate situations in the newscast where the MSE are general high, e.g. scenes from concerts with high level of motion due to lights and other effects, we will look at the MSE from an interval of frames, around the current frame. We compute the standard deviation over the MSE of adjacent frames in a certain range. We have implemented a queue to maintain the range list, so we only have to compute the MSE for each frame once. Figure 7 shows a illustration of how the range-window "slides" over each frame, and compute the standard deviation from the range. The result will be the determining value for that particular frame. In order to avoid detecting multiple shot from the same true cut, we discard any detections made within the same range of another detection.

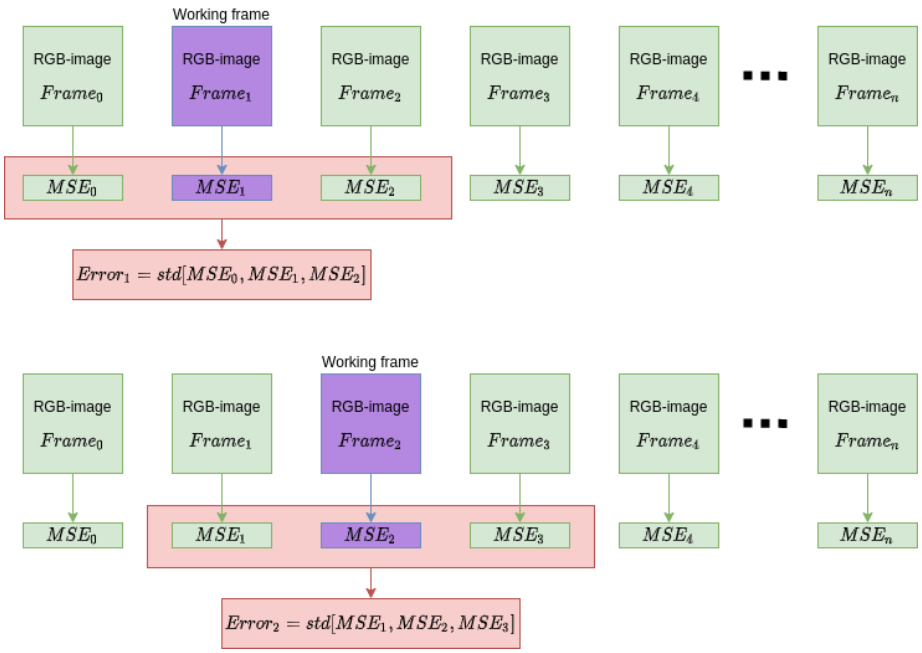


Figure 7: *Illustration of two iterations. The standard deviation is calculated from a range of MSE's computed from adjacent frames. In this case, a range of 3 is used. The first row shows hos the error value for* $frame_1$ *is found. Similar, the second row shows the next iteration, and how the error value for* $frame_2$ *is found.*

We have let the range be a parameter and experimented with different values. Table 1 and Figure 8 shows the performance for different values of the MSE range parameter. Both accuracy, precision, specificity, and F1 increase as we increase the range length. Recall tops at the range of 3 and 4, and declines thereafter. This could indicate that a large acceptance range will decrease the number of false positives. We will use range value of 4, as it maximize the recall and gives a high score for the remaining measurements.

Range	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Accuracy	0,8902	0,8970	0,9021	0,9067	0,9114	0,9170	0,9191	**0,9208**
Precision	0,8207	0,8243	0,8328	0,8432	0,8541	0,8643	0,8683	**0,8714**
Recall	0,9754	**0,9879**	**0,9879**	0,9837	0,9796	0,9798	0,9799	0,9799
Specificity	0,8169	0,8188	0,8275	0,8387	0,8504	0,8598	0,8630	**0,8662**
F1 score	0,8914	0,8987	0,9037	0,9081	0,9125	0,9184	0,9208	**0,9225**

Table 1: Performance metric for different values of the MSE range parameter

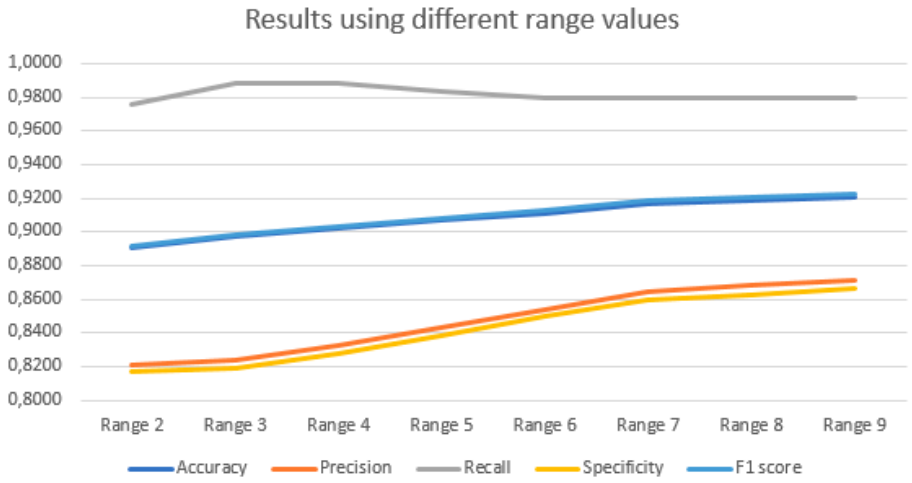


Figure 8: *Experiment of using different values for the MSE range parameter*

4.2 Detecting the graphical cut-scene

In this section, we will show how we have implemented a solution for detecting and handling the graphical cut-scene. The method for shot boundary detection presented in section 4.1 is performing poorly on the graphical cut-scenes. We correctly find the first and last frames as cuts in the animation, however many intermediate frames are classified as shot boundaries. As we have chosen only to define the first and last frame of the animation as true shot boundaries, the method is not a sufficient way to find these cut-scenes. The falsely classified frames is due to the violent movement in the graphics. Therefore, we want a separate routine for handling that particular part.

As we described in section 3.3, the cut-scenes throughout the video all consist of the same color scheme. Furthermore, special structures are used in the animation, e.g. see Figure 3, 5, and 6. However, as the scenes differs each time, we want to detect frames similar to the animation.

We have used a k-means model to recognise the graphical cut-scene. For each frame in the newscast, we compute a 1D intensity histogram for each channel in the HSV color space. The three histograms make up the features for a single frame, and

is used to fit the model. We have used a k-means implementation from *scikit-learn* library [6].

We can use the fitted k-means model to predict the labels for each frame, i.e. if the frame belongs to cut-scene or not. As it can happen that the model misclassify some of the frames in the cut-scene, we patch the predictions with a range of 10, i.e. if two frames f_k and f_{k+10} is classified as a cut-scene, all frames between f_k and f_{k+10} will also get the cut-scene label. Figure 9 illustrates a cut-scene prediction before and after patching.

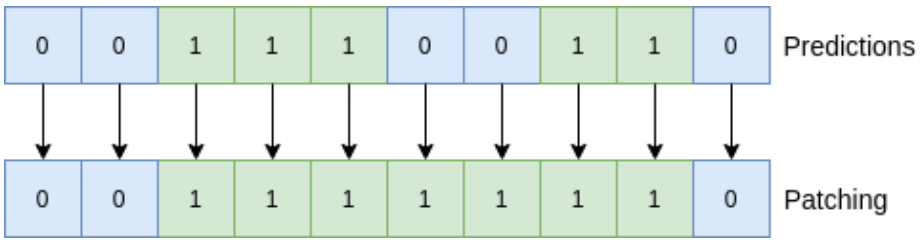


Figure 9: *Illustration of how the cut-scene prediction is patched. The label 1 belong to the cut-scene, 0 otherwise.*

With the coherent cluster classifications, we overwrite any shot boundaries detected by the MSE approach described in section 4.1, by the first and last frame from our clusters and remove any intermediate detections.

Now, we have combined our models for detecting regular hard cuts, and for classifying the graphical cut-scene.

5 Results

As there are no existing ground truth labels for shot boundaries, we have made the ground truth by hand. For each shot-change, we have calculated the frame number from: $frame\ number = (minutes * 60 + seconds) * fps + extra\ frames$, where fps are 25 in our case, and $extra\ frames$ is the number of frames since the last known second. This is of course error prone, as we might have made mistakes or made calculation errors.

Furthermore, as seen in Figure 5 and 6, the graphic cut-scenes begins and ends with an soft transitions where parts of both shots are visible at the same time. This means, that we have to determine which frames we should treat as part of the cut-scene. As we see in Figure 5, there are many frames where both the animation graphics and the previous is visible. We choose to define the frame where the animation is first visible, as the beginning of the cut-scene.

As we have just described, we can not expect the ground truth to be exact, and it may contain errors. Therefore, we will measure the performance of our model, with varying accepting ranges of precision. As the results of the model will be used to divide a video into sub-stories of a newscast, it is reasonable to allow for some slack when measuring accuracy, i.e. the range between a predicted cut to a true cut. Table 2 show the result of the final combined model described in section 4.2. We

Acceptance range	3	5	10	15	25
Accuracy	0.9986	0.9987	0.9989	0.9991	0.9994
Precision	0.8482	0.8581	0.8812	0.9109	0.9439
Recall	0.9847	0.9848	0.9852	0.9857	0.9896
Specificity	0.9987	0.9988	0.9990	0.9992	0.9995
F1	0.9113	0.9171	0.9303	0.9468	0.9662

Table 2: Result from the combined model using different settings for acceptance range in number of frames. Recall that 25 frames correspond to to 1 second.

measure the performance in accuracy, precision, recall, specificity, and f1 score. We have 5 different settings for the acceptance range, i.e. how many frames between a prediction and a true cut we accept as a correctly classified cut. Recall that 25 frames correspond to to 1 second.

From Table 2 we see that precision is relative low compared with the remaining metrics. This indicates that our model has a tendency to predict shot boundaries that are true cuts. If we look at which frames that have been falsely classified, i.e. false positives, it correspondence with how we defined a shot boundary. Figure 10 shows 6 frames from a scene which shows a police car with the light siren on. The light from the car makes a bright flash which illuminates the surrounding objects. The changes between each frame are drastic, as large portions of the images completely change colors. This is detected as a scene change, as the MSE between each frame is greater then the predetermined threshold.

(a) Frame # 10850 (b) Frame # 10851 (c) Frame # 10852

(d) Frame # 10853 (e) Frame # 10854 (f) Frame # 10855

Figure 10: *Example of six continues frames. The scene shows a police car with the sirens on, which makes the entire images flash rapidly.*

Furthermore, the k-means method for detecting the graphical cut-scenes also made false positive predictions. Figure 11 shows a frame from a scene showing a map of Copenhagen. Both the structural properties and the colors looks very similar to parts of the animation in the cut-scenes. Therefore, it gets classified as a graphical cut-scenes.

Figure 11: *A example of a scene showing a map of Copenhagen.*

6 Discussion and Future work

The goal of the report, has been to produce a method for shot boundary detection as well as detecting the special graphical cut-scenes. The results should be reasonable in the sense that, they can be used for future study in a M.Sc. thesis. However, as the final results will be used to index and label the data with metadata, it is not vital that the predicted shot boundaries is exact, as small variations in the order of a few seconds is acceptable.

We have seen how we have combined two methods that together was able to produce reliable and adequate results for the domain, as it correctly finds almost all shot boundaries, however, it has a tendency to falsely predict to many cuts. We saw examples of this in section 5, where a police light siren triggered many false positive cuts. In order to solve this issue, we might make a rule set that are able to accommodate this. E.g. if many cuts are detected very close to each other, we can simply pick the first and last cut in the sequence, and discard the remaining intermediate predictions. This should not remove any true cuts, as we could choose a relative short range, e.g. 1 second. However, this will only work for scenes like the one we saw in Figure 10, but will properly not work as intended for scenes with a single flash or if they are far temporal apart.

Another issue we found, was due to wrong prediction by the k-means model. As we discussed in section 3.3, a very specific sound is playing when the graphical cut-scenes are shown. If we could use the signal from the audio track, we may be able to analyse the the sound, and use it to strengthen the model.

As this report was aimed towards 2018 newscasts from DR1, the methods we have presented was developed to solve data from that period. Therefore, we can *not* expect to see similar results for data from other TV-stations or different years,

as it may contain very different content from what we have worked with.

7 Conclusion

Throughout this report, we have presented a method for shot boundary detection from TV newscast. We have build a technique for handling special cases with the graphical cut-scenes. The model is specifically developed for the DR1 18:30 newscast from the year 2018, however, the shot boundary detector should work for other videos as well, perhaps with modifications to some threshold parameter. The same might be true for the k-means model, however, it should be fitted to any new data.

Overall, the model presented performs acceptable, especially if we allow for predictions to be within a second of the true cuts.

References

[1] openCV

[2] NumPy

[3] Methods and Challenges in Shot Boundary Detection: A Review
Sadiq H. Abdulhussain, Abd Rahman Ramli, M. Iqbal Saripan, Et al.
Published in Entropy 2018 page 6

[4] A unified approach to scene change detection in uncompressed and compressed video. - IEEE Transactions on Consumer Electronics, 2000, 769 - 779.
W. A. C. Fernando, C. N. Canagarajah and D. R. Bull

[5] Performance Characterization of Video-Shot-Change Detection Methods - ieee transactions on circuits and systems for video technology, Feb 2000, 1 - 13
Ullas Gargi, Rangachar Kasturi, and Susan H. Strayer

[6] scikit-learn
sklearn.cluster.KMeans