Video Content and Structure Description Based on Keyframes, Clusters and Storyboards

Marc Junyent*, Pablo Beltran*, Miquel Farré*, Jordi Pont-Tuset*, Alexandre Chapiro*†, Aljoscha Smolic*
*Disney Research Zürich, †ETH Zürich

***Abstract*—In this paper we present a novel system to extract keyframes, shot clusters and structural storyboards for video content description, which can be used for a variety of summarization, visualization, classification, indexing and retrieval applications. The system automatically selects an appealing set of keyframes and creates meaningful clusters of shots. It further identifies sections that appear recurrently, which are called anchors, and typically divide television shows into different parts. This information about anchors can then be used to browse video content in a new fashion. Finally, our system creates a new type of interactive storyboard suitable to visualize and analyze the structure of the video in a novel way.**

I. Introduction

The offer of audiovisual media has increased significantly over the last years, presenting a plethora of new content as well as numerous ways to consume it. The relation between shows and viewers is stronger thanks to the interaction through new channels such as social networks, which motivates professionals in the media industry to post supplementary content on these networks to engage their audiences.

The growing amount of online content and the immediateness required by viewers, however, poses a challenge for professionals providing *good* and *complete* supplementary content. As attention spans are getting shorter and users are flooded with a variety of content, the produced material needs to be *brief* and *appealing* to succeed.

Video metadata plays a key role in meeting these criteria, as it helps professionals and computer systems in finding, creating, and promoting the right content as fast as possible. As a consequence, having video content annotated by as much and as good metadata as possible is becoming a major competitive advantage in the media industry. In particular, improvements in identifying specific clips and sections of a show and highlighting content e.g. through selection of visually pleasant keyframes, can make the difference between the content being extensively shared and consumed, or simply ignored and lost.

In this paper we present a novel system that contributes to the mentioned challenges in three different ways: (i) providing an appealing set of keyframes, (ii) providing meaningful clusters of shots called video clips, and (iii) detecting the anchoring blocks that divide the video under analysis into different sections, such as the jury discussion in a talent show, the anchorman in news, or the conversations in a morning television show before changing topics.

978-1-4673-7478-1/15/$31.00 ©2015 IEEE

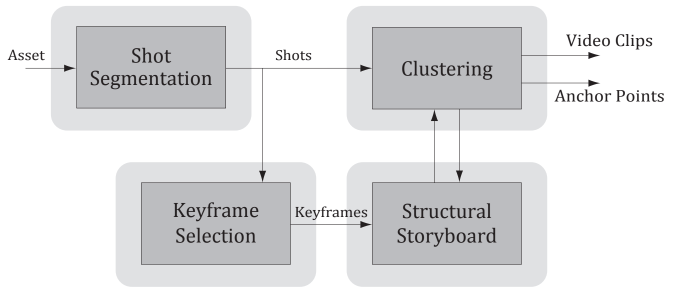

Fig. 1. This figure shows the pipeline of our method: after an initial shot segmentation, we perform keyframe selection and cluster the segmented shots to obtain video clips and anchor blocks. Finally all the information is organized in a structural storyboard where the user can visualize keyframes, clips and anchor blocks, and also refine the initial results.

The mentioned anchoring blocks can be used to segment a show into sections and distribute it as a set of clips, which also enables new playback opportunities, e.g. jumping from block to block. Furthermore, our structural storyboard tool organizes keyframes in a novel visualization format that also allows human interaction to refine initial automatic results. The full pipeline is shown in Figure 1 and each step is explained in more detail in the following sections.

II. State of the Art

Video summarization has been a very active field of research over recent years. According to Truong et al. [1] there are two fundamental types of video summaries: static video summaries and dynamic video skimming. Static video summaries are composed of a set of keyframes extracted from the original video, while dynamic video summaries are composed of a set of shots and are produced taking into account the similarity or relationships among video shots. Comprehensive surveys of past video summarization results can be found in [1] and [2].

A number of keyframe extraction methods exist, with varying degrees of complexity. Some method select the first or last frame of a shot as the keyframe [3], [4]. Qu et al. [5] transform content into the HSV colorspace where shot segmentation can be performed at a lessened cost. Keyframe selection can also be treated as a clustering problem where the frames are points in a space of features and the representative points are selected as keyframes [6]. Peker et al. [7] use spectral clustering with face detection: the keyframes are selected according to the detected faces' location and size while Chasanis et al. [8] use a novel improved spectral clustering method. Some works take a semantic approach to keyframe extraction. Liu et al. [9] segment video shots following consecutive motion patterns. The turning points from accelerating to decelerating motions are selected as keyframes. Kang and Hua [10] try to learn which frames produce more representative matchings with a set of descriptors previously selected by humans. Evangelopoulos et al. [11] suggest that audiovisual saliency can be used to select keyframes for a video summarization task. Recently, Luo et al. [12] use several F-divergences to calculate frame by frame distances and segment the video to obtain the keyframes afterwards. For a general overview on keyframe extraction we refer the reader to [13].

Several techniques have been studied to represent multiple keyframes on screen while maximizing the compactness of the representation, making a good coverage of the different sections of the video and presenting them in an intuitive way such as in comic-like layouts [14]. Space limitation can also be overcome presenting the keyframes in an interactive environment [15], [16] that allow the user to navigate across a large set of keyframes and refine the ones to be shown on screen through interactions. Barnes et al. [17] suggest a summarization design inspired by medieval tapestries where keyframes are melded into a continuous timeline that represents the content. Another approach is to match the number of keyframes on screen to the available representation area [18] by means of scalable storyboards. This approach has been studied in detail in [19] for the particular case of handheld devices.

Video decomposition in scenes can be classified in rule-based approaches - which study the way a scene is structured in a professional production to decompose the video in scenes [20] - and graph-based methods where shots are arranged in a graph representation and then clustered by partitioning the graph. The Shot Transition Graph [21] is one of the most used models in this category. In [22] normalized cuts is employed to optimally obtain clusters and a shortest path algorithm is used to detect the scenes. Sidiropoulos et al. [23] extended the Shot Transition Graph using multimodal low-level and high-level features. Brandi et al. [24] presents a scene detection algorithm that combines local image descriptors and temporal clustering techniques.

Our paper extends previous work as follows:

- We extract suitable keyframes based on face detection, text detection and frame quality. Available text and face detectors are extended and optimized, and a new notion of key faces is introduced.
- We provide a method for meaningful clustering of shots (could also be used to automatize the backbone detection in [16]). It further identifies anchoring sections in shows. An anchoring section or block can be understood as a section of the show that recursively appears to introduce or discuss about the next or previous section.
- Finally, we provide a new storyboard representation which allows better visualization of the structure of the video compared to other approaches. The storyboard can also be used to provide feedback to the anchoring section detection and clip generation algorithm.

In the following sections we will outline each step of our method's pipeline and finally present our experimental results.

III. Shot segmentation

Video is segmented in shots using rank-tracing [3], calculating the histogram of the frames in the HSV color space. Results are enhanced with [25] to detect dissolve transitions between shots, as well as a flash detector to avoid false positives. Frames belonging to a dissolve transition are not considered as candidates for keyframe selection and are not used in any other part of the algorithm.

We denote S as the set of shots, $S = \{S_1, S_2, \cdots, S_n\}$, of the video asset. Each shot S_i contains a set of m_i valid frames.

IV. Keyframe selection

In order to select nice-looking keyframes we rely on a per-frame score which takes into account text on screen, several face features, and quality of the image. Details related to each feature, its temporal filtering, and final combination are described below.

A. Text detection and tracking

We detect text blobs calculating the Stroke Width Transform (SWT), as described by Epshtein et al. [26] over all frames. For each shot S_i we obtain a set of text blobs $B_i = \{b_{jk} : 1 \leq j \leq m_i, k \geq 0\}$.

In order to track the text blobs across the shot, we begin by improving the temporal coherence of the detections obtained from SWT. To do this, we group text blobs in families. Each family T contains the same text blob across the frames of the shot. By definition, a blob cannot be part of two families and a family cannot have two blobs on the same frame. The rule to cluster text blobs in families is the following: if b_{jk} is part of a family T, any blob $b_{j'k'}$ such that $|j - j'| \leq d$ and having a similar spatial position and size as b_{jk} is considered to belong to the family T. With $d = 1$ we would be considering contiguous blobs in time but to overcome possible misdetections in isolated frames we experimentally set $d = 6$ for videos with 24 frames per second (fps).

We define the length of a family $\mathrm{length}(T)$ as the distance between the first and the last blobs' frames plus one. The families with length less than a second and the families where the actual detected blobs to length ratio is small are considered false positives and are discarded. Once the false positives have been discarded, families that are close in time and spatial position are merged.

Finally, we ensure the temporal continuity of the text blobs within a family. For this, proxy blobs are generated in intermediate frames where a text blob for the given family was not detected. To generate these proxies, we first determine if the text represented by the family is static or moving. If the text is static, the proxy blobs generated are the union of the previous text blobs of the family and if the text is moving, the proxy blobs are obtained by interpolating between the previous and the next existing blobs in the family.

B. Face detection and analysis

Following a similar approach as the previous section, we detect the faces on each frame and track them within a shot. Faces f_{jk} are grouped in families F_i at shot level. For each detected face, its position, size, angle with respect to the camera and the state of the mouth and the eyes is extracted using the methods described by Ruf et al. [27].

We combine this information to determine which faces are important in the shot. This feature is key in the content targeted by this work, e.g. in a night show, the focus on the presenter rather than on the faces of audience members is preferred.

To determine key faces, we give a score to each family and then search for outliers in terms of behavior. To obtain a final score per family, the first step is to obtain the following four scores for each face f_{jk} found in a frame:

- A quality or flattery, $Q(f_{jk})$, score based on the rotation of the head, the state of the eyes and mouth and the reliability of that information.
- $P(f_{jk})$ based on the position of the face within the frame, giving more weight to centered positions.
- $D(f_{jk})$ based on the mean distance to other faces in the same frame.
- $M(f_{jk})$ based on the size of the face relative to the size of the other faces in the same frame.

All these scores are normalized between 0 and 1 where 1 represents the best score. We obtain the same scores at family level as the mean of the scores of the faces that belong to the family. The total score of a family is calculated as:

$$\begin{aligned} S(F_i) &= w_1 Q(F_i) + w_2 D(F_i) + w_3 M(F_i) \\ &+ w_4 \left(P(F_i) + \frac{1}{\#F_i} \sum_{f \in F_i} (P(f) - P(F_i))^2 \right) \\ &+ w_5 \left(\frac{\text{length}(F_i)}{\text{length}(\mathit{Shot})} - \frac{\#F_i}{\text{length}(F_i)} \right) \end{aligned} \quad (1)$$

Apart from considering the suitability of the detected position, the position variance also influences the score as faces with big movement are likely to be important. The last factor gives more importance to faces that appear more during the shot but penalizes long families with few actual detected faces to overcome possible false positives. w_i weights change the relative importance of the different scores. $S(F_i)$ is always kept normalized in our experiments. After some experimentation we set $w_1 = 0.20$, $w_2 = 0.35$, $w_3 = 0.25$, $w_4 = 0.05$ and $w_5 = 0.15$, which work well with different genres, from movies to news programs, or talk shows.

Once we have calculated the scores we decide which families F_i are categorized as important. Based on observations in TV content and the kind of shots used in it, we assume that all families are important if the number of families within the shot is small (three families or less) and have similar scores. If the number of families is bigger than 3, we select as important families the ones with outlying high scores. As all our scores are normalized, we consider as outliers those families with scores above or below the mean by 0.1 points.

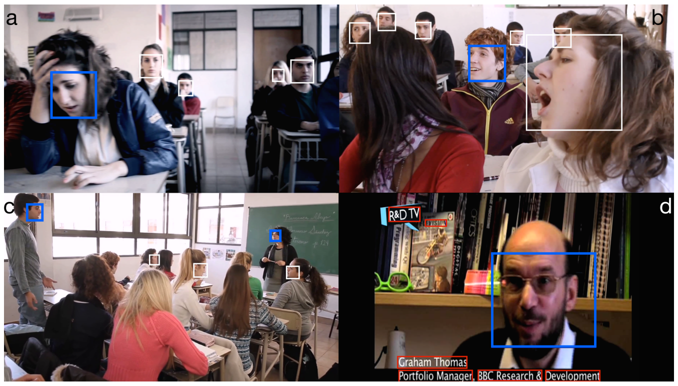

Fig. 2. Text and key face detection from publicly available content [28], [29]. Blue bounding boxes point to key faces, white bounding boxes to other faces and red boxes to text.

C. Keyframe extraction

For each frame j, $1 \leq j \leq m_i$ in the shot S_i we calculate a text score $S_{t,j}$, a faces score $S_{f,j}$ and a quality score $S_{e,j}$.

The text score is calculated as:

$$S_{t,j} = \frac{\sum_{k \geq 0} \text{Area}(b_{jk}) l(b_{jk}) g(b_{jk})}{\max_{1 \leq l \leq m_i} \left(\sum_{k \geq 0} \text{Area}(b_{lk}) l(b_{lk}) g(b_{lk}) \right)} \quad (2)$$

$l(b)$ and $g(b)$ are factors that depend on the start s_b and end e_b of the family the text blob b belongs to.

$$l(b) = \exp\left(\frac{\frac{j - s_b}{e_b - s_b + 1} - 0.5}{2.0 * 0.5^2} \right) \quad (3)$$

$$g(b) = \begin{cases} \frac{e_b - s_b + 1}{t_1} & e_b - s_b + 1 \leq t_1 \\ 0.5 & e_b - s_b + 1 > t_2 * \text{length(Shot)} \\ 1.0 & \text{otherwise} \end{cases} \quad (4)$$

The factor g penalizes overly short or long families of blobs. Families with a short duration are probably unimportant and might even be a false detection, while the families that continue for most of the duration of a given shot should not affect the selection of the keyframe. We set $t_1 = 48$, 2 seconds at 24 fps, and $t_2 = 0.85$. The factor l is an exponential that decreases the importance of a text blob when it is near the beginning or the end of its life to avoid possible fades or blurriness.

The face score can be calculated in two possible ways, depending on whether important families of faces have been found or not. If important families of faces are present, we define I_j as the set of faces in frame f_j that belong to an important face family. Then:

$$S_{f,j} = \frac{(\frac{r_i}{2} + \frac{1}{4}) |I_j|}{\max\limits_{1 \leq k \leq m_i} |I_k|} + \frac{(\frac{3}{4} - \frac{r_i}{2}) \sum_{f_{jk} \in I_j} Q(f_{jk})}{\max\limits_{1 \leq l \leq m_i} \sum_{f_{lk} \in I_l} Q(f_{lk})} \quad (5)$$

where

$$r = \frac{\max_{1 \leq k \leq m_i}(|I_j|)}{|F|}$$

and $Q(f)$ is the flattery score as defined in Section IV-B. The first term gives importance to the frames that have most of the important faces of the shot, the second term tries to maximize the flattery score of these faces. The r parameter balances between these two terms prioritizing getting all the important faces in one frame if possible or the most flattering of them if it is not.

If no face family has been detected as important the face score is simply calculated as:

$$S_{f,j} = \frac{\sum_k \text{score}(f_{jk})}{\max_j \sum_k \text{score}(f_{jk})} 0.3 + \frac{|f_j|}{\max_j |f_j|} 0.7 \tag{6}$$

We again prioritize frames with more faces. The 0.3 and 0.7 constants were chosen experimentally.

Finally, we compute an overall image quality score, which is inspired by recent research on automated image aesthetics [30], where sharpness is identified as the most crucial parameter of image quality. In order to include this in our keyframe extractor, we compute a simple measure of overall image sharpness. For each frame f_j a Laplacian filter is applied to its grayscale version, producing a new frame. An energy score, e_j, is calculated as the mean of the squared values of the resulting pixels. Finally the quality score of a frame is calculated as:

$$S_{e,j} = \frac{e_j}{\max_{1 \leq k \leq m_i}(e_k)}$$

The final score $S(f_j)$ for a frame is calculated as:

$$S(f_j) = w_f \cdot S_{f,j} + w_t \cdot S_{t,j} + w_e \cdot S_{e,j} \tag{7}$$

Where w_f, w_t, w_e are weights that change the relative importance of the different scores and can be adapted to fit the goals of the user. In our experiments we set $w_f = 2.0$, $w_t = 1.0$ and $w_e = 0.5$ as we focus on videos with people as protagonists. These values proved to work well in a wide variety of media.

V. Clustering

The clustering part of our system groups the shots identified in section III in segments that can be classified as short clips or anchor sections of the show. In order to detect anchoring sections we first analyze the color similarity of all the shots in the video and in a second step we apply a temporal clustering. Note that introducing temporal information in the first step as done in previous works [22] would not allows us to group shots belonging to the anchoring part of a show when substantially separate in time.

A. Shot similarity clustering

We select multiple equally k spaced frames in each shot and use them to create a measure of similarity between shots. We divide each k frame in a grid of $m \times n$ rectangles and compute their normalized histograms. The distance between two frames is the mean of the Euclidean distances of their histograms. Let $f_1, \ldots, f_k$ be the k selected frames from shot s, $f'_1, \ldots, f'_k$ the selected frames from shot s' and S_k the group of permutations of k elements. Then the distance between s and s' is calculated as:

$$\text{dist}(s, s') = \min_{\forall \sigma \in S_k} \sum_{i=1}^{k} \text{dist}(f_i, f'_{\sigma(i)}) \tag{8}$$

The distance distribution is clustered using DBSCAN [31] which is very robust against noise and allow us to find the related shots only requiring two parameters: the minimum number of shots to be considered a cluster and the maximum reachable distance between them. Hence, opposed to other clustering algorithms, DBSCAN allows us to consider different number of clusters from one video to another, making it very suitable for our task.

We set the minimum number of shots per cluster to one as some of the shots may appear only once and we define a maximum reachable distance tailored to each video: we iteratively decrease the maximum reachable distance starting from one until only a few temporally consecutive shots are grouped together. By doing this, we assume a correct shot segmentation because two shots would either have different color content or not be consecutive.

B. Temporal clustering

After the shot similarity clustering we find which of these clusters belong to the same sequence. Similarly to [22] we use the temporal relationship between shots to create a graph with the similarity clusters as nodes and temporal transitions as edges, *i.e.* a pair of nodes would have an edge between them if one has a shot that is adjacent in time of the shot from the other. After that, we search for closely related shot clusters looking at the number of edges between them.

Unlike the algorithm by Wah et al. [22], we do not use the Dijkstra algorithm to find the scenes as our graph would probably be cyclical. An example consists of news or discussion shows, which are commonly opened and ended with the same anchor. The first and last shot belong to the same cluster and the Dijkstra shortest path algorithm would consider the entire video as only one scene. In order to prevent this, we apply the HCS [32] recursively.

The HCS Clustering algorithm tries to find the more connected vertices regarding their connectivity. In order to do this it recursively applies the minimum cut to a connected graph or subgraphs until the minimum cut value, i.e. the number of edges of the mincut in an undirected graph, is greater than the half the number of nodes. As we want to minimize the number of clusters, we use the Karger's algorithm [33] to try to find the minimum cut that splits the graph in disjoint sets of the similar number of vertices. The method has some similarities with the method in [22] as they employ the second smallest eigenvalue to partition the graph representing the video decomposition, equivalent to minimize the Ratio cut from a graph A:

$$\min \ RatioCut = \frac{\text{cut}(A_i, \overline{A_i})}{|A_i||\overline{A_i}|} \tag{9}$$

While we look for:

$$\text{mincut}(A_i, \overline{A_i}) \text{ such that } \min \frac{1}{|A_i||\overline{A_i}|} \tag{10}$$

where A_i is a subset of nodes and its vertices from A, $|A_i|$ its number of vertices and $\overline{A_i}$ the subset formed by $A - A_i$. We perform HCS clustering iterations contracting the edges for which both vertices fall in the same cluster and we iterate again until there are no more cluster changes.

C. *Detection of anchoring sections*

After the temporal clustering we obtained groups of shots that are connected between them and we classify them into (i) those that are temporally adjacent, and so can be interpreted as video clips, and (ii) non-adjacent blocks of shots whose content is similar. We look for anchoring sections in the latter.

In order to detect which one of the clusters is the anchor we take into account two parameters: (i) their time on screen, i.e. the aggregated time of all their appearances, and (ii) their range, that is, the amount of time between their first and last appearance. Combining these two measures allows us to discard credit screen shots that may only appear at the beginning and at the end, as well as clusters with unique but long occurrences.

VI. Storyboard Generation and Interaction

For a given video asset, commonly several shots from the same camera are present. These shots share the same spatial location and usually the same action or characters. For example, in an interview is common to find an alternation between two kinds of shots: one focusing on the interviewer and one on the interviewed.

Unlike existing storyboards that try to present information in a spatially compact manner, our method places the keyframes that represent each shot chronologically from left to right and adds a second dimension vertically, where shots categorized as similar in Section V-A are placed in the same row. This gives the viewer a sense of the structure of the video. Furthermore, we add a colored label on the corner of each keyframe in the storyboard. Keyframes share the color of its label if they belong to the same clip or anchoring.

If necessary, users can correct the results of the similarity clustering moving the keyframes to a more appropriate row. After the user interaction, the temporal clustering described in Section V-B is computed again. An example of the storyboard and the result of an interaction on it is presented in Figure 4.

VII. Results

In our first set of experiments, we performed a user study which compared the visual appeal of our keyframes and the keyframes provided by the methods described in [8] and [12]. The study consisted in a standard three-alternate forced-choice procedure. Participants were instructed to select the keyframe that looked more enjoyable and appropriate to be posted online. Fifty volunteers participated, undertaking 20 trials each. To obtain comparable keyframes from the three methods we initially divided videos containing news and shows in shots and fed each individual shot to the different keyframe extraction methods.

The preference rates for this experiment are shown in Figure 3. We performed a two sample t-test between the preference rates averaged out over all responses provided by each participant between our method and [8] as well as between our method and [12]. We found that our method was selected significantly more often than both competing methods ($\sigma < 0.05$).

In our second set of experiments, we compared our anchoring block detection results against a manually annotated ground truth of anchoring blocks in multiple test sequences. Our approach performs very well with highly structured video such as news, morning TV and talent shows. Pushing the limits of the algorithm, we experimented on a movie [34], which does not have a clear anchoring structure like TV shows. We refined the initial results automatically computed with our system, by interactively moving five shots in the storyboard to a more appropriate position, which improved the results significantly. Detailed results are presented in Table I together with the storyboard of [34] in Figure 4.

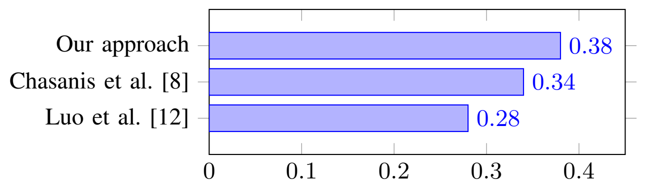

Fig. 3. Keyframe preference rates, user experiment results

TABLE I
Anchoring blocks detection compared to the ground truth

Video	Precision	Recall	Accuracy
News	0.99	0.88	0.93
Morning TV show	0.94	0.92	0.96
Talent show	0.83	0.48	0.85
Short movie [34]	0.43	0.34	0.71
Short movie [34] after 5 interactions	0.62	0.71	0.81

VIII. Conclusions

We presented a novel system and components for video content and structure description, which enables new forms of visualization and interaction with the data, as crucial in the era of the social networks. The new keyframe selection algorithm outperforms state of the art approaches in terms of visual appeal. The introduced clustering algorithm computes video clips and detects anchoring blocks in shows, which reveal structure and enable advanced playback such as content aware video skipping. Furthermore, results of the clustering are organized to a novel storyboard that visually preserves the structure of the video.

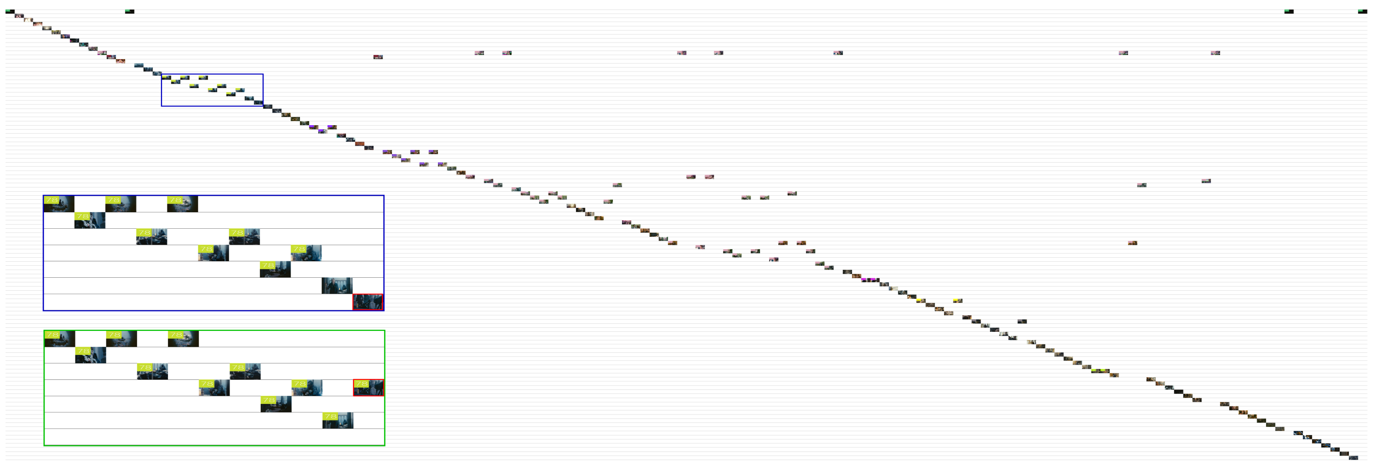

Fig. 4. In this figure, the storyboard generated for the short film "Tears of Steel" [34] is presented. Please zoom into the figure in order to see the details of our storyboard, such as the color labels of the shots. For example the shots labeled in pink can be understood as the anchor block of the movie. Inside the blue box we present a short clip of the movie where the similarity clustering was wrong. In the green box the user moved the last shot of the clip (in red) to a more appropriate row and as a consequence the temporal clustering algorithm corrected the labelling.

References

[1] Ba Tu Truong and Svetha Venkatesh, "Video abstraction: A systematic review and classification," *TOMCCAP*, vol. 3, no. 1, pp. 3, 2007.

[2] Arthur G. Money and Harry Agius, "Video summarisation: A conceptual framework and survey of the state of the art," *J. Vis. Comun. Image Represent.*, vol. 19, no. 2, pp. 121–143, Feb. 2008.

[3] W. Abd-Almageed, "Online, simultaneous shot boundary detection and key frame extraction for sports videos using rank tracing," in *ICIP 2008*, Oct 2008, pp. 3200–3203.

[4] Dr Shobha G. Azra Nasreen, "Key frame extraction using edge change ratio for shot segmentation," in *IJARCE Vol. 2, Issue 11, November 2013*, pp. 4421–4423. 2013.

[5] Zhong Qu, Lidan Lin, Tengfei Gao, and Yongkun Wang, "An improved keyframe extraction method based on hsv colour space," *Journal of Software*, vol. 8, no. 7, pp. 1751–1758, 2013.

[6] M. Furini, F. Geraci, M. Montangero, and M. Pellegrini, "Stimo: Still and moving video storyboard for the web scenario," *Multimedia Tools and Applications*, vol. 46, no. 1, pp. 47–69, 2010.

[7] Kadir A Peker and Faisal I Bashir, "Content-based video summarization using spectral clustering," in *International Workshop on Very Low-Bitrate Video, Sardinia, Italy*, 2005.

[8] Vasileios Chasanis, Aristidis Likas, and Nikolas P. Galatsanos, "Scene detection in videos using shot clustering and sequence alignment.," *IEEE Transactions on Multimedia*, vol. 11, no. 1, pp. 89–100, 2009.

[9] Tianming Liu, Hong-Jiang Zhang, and Feihu Qi, "A novel video key-frame-extraction algorithm based on perceived motion energy model," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 10, pp. 1006–1013, Oct 2003.

[10] Hong-Wen Kang and Xian-Sheng Hua, "To learn representativeness of video frames," in *Proceedings of the 13th annual ACM international conference on Multimedia*. ACM, 2005, pp. 423–426.

[11] G. Evangelopoulos, K. Rapantzikos, A. Potamianos, P. Maragos, A. Zlatintsi, and Y. Avrithis, "Movie summarization based on audiovisual saliency detection," in *ICIP*. IEEE, 2008, pp. 2528–2531.

[12] X. Luo, Q. Xu, M. Sbert, and K. Schoeffmann, "F-divergences driven video key frame extraction," in *ICME*, 2014, pp. 1–6.

[13] C Sujatha and Uma Mudenagudi, "A study on keyframe extraction methods for video summary," in *CICN*, 2011.

[14] J. Calic, D. Gibson, and N. Campbell, "Efficient layout of comic-like video summaries.," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 17, no. 7, pp. 931–936, 2007.

[15] J. Lokoc, K. Schoeffmann, and M. Del Fabro, "Dynamic hierarchical visualization of keyframes in endoscopic video," in *MMM 2015*.

[16] Herv Goau, Jrme Thivre, Marie-Luce Viaud, and Denis Pellerin, "Interactive visualization tool with graphic table of video contents.," in *ICME*. 2007, pp. 807–810, IEEE.

[17] C. Barnes, D. Goldman, E. Shechtman, and A. Finkelstein, "Video tapestries with continuous temporal zoom," *ACM Transactions on Graphics*, vol. 29, no. 3, Aug. 2010.

[18] L. Herranz and J. Martinez, "A framework for scalable summarization of video.," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 20, no. 9, pp. 1265–1270, 2010.

[19] L. Herranz and S. Jiang, "Scalable storyboards in handheld devices: applications and evaluation metrics," *Multimedia Tools and Applications*, pp. 1–29, 2015.

[20] Cailiang Liu, Dong Wang, Jun Zhu, and Bo Zhang, "Learning a contextual multi-thread model for movie/tv scene segmentation," *IEEE Transactions on Multimedia*, vol. 15, no. 4, pp. 884–897, 2013.

[21] M. M. Yeung, B.-L. Yeo, W. H. Wolf, and B. Liu, "Video browsing using clustering and scene transitions on compressed sequences," in *Multimedia Computing and Networking 1995*, Mar. 1995.

[22] Chong-Wah Ngo, Yu-Fei Ma, and HongJiang Zhang, "Video summarization and scene detection by graph modeling," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 15, no. 2, pp. 296–305, 2005.

[23] P. Sidiropoulos, V. Mezaris, I. Kompatsiaris, H. Meinedo, M. Bugalho, and I. Trancoso, "Temporal video segmentation to scenes using high-level audiovisual features.," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 21, no. 8, pp. 1163–1177, 2011.

[24] L. Baraldi, C. Grana, and R. Cucchiara, "Scene segmentation using temporal clustering for accessing and re-using broadcast video," in *IEEE ICME 2015*.

[25] C. Su, H.-Y.M. Liao, H. Tyan, K. Fan, and L. Chen, "A motion-tolerant dissolve detection algorithm," *Multimedia, IEEE Transactions on*, vol. 7, no. 6, pp. 1106–1113, Dec 2005.

[26] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting text in natural scenes with stroke width transform," in *CVPR*, 2010, pp. 2963–2970.

[27] Tobias Ruf, Andreas Ernst, and Christian Küblbeck, "Face detection with the sophisticated high-speed object recognition engine (shore)," in *Microelectronic Systems*, pp. 243–252. Springer, 2011.

[28] G. Doin, "http://www.educacionprohibida.com," .

[29] BBC Backstage and BBC RD, "RDTV episode 1," 2009, Available at http://ftp.kw.bbc.co.uk/backstage/rdtv.

[30] T. Aydin, A Smolic, and M. Gross, "Automated aesthetic analysis of photographic images," *Visualization and Computer Graphics, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.

[31] Martin Ester, Hans peter Kriegel, Jrg S, and Xiaowei Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," 1996, pp. 226–231, AAAI Press.

[32] E. Hartuv and R. Shamir, "A clustering algorithm based on graph connectivity," *Information Processing Letters*, 1999.

[33] David R. Karger, "Global min-cuts in rnc, and other ramifications of a simple min-out algorithm," in *Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, Philadelphia, PA, USA, 1993, pp. 21–30.

[34] Blender Foundation, "Tears of steel," http://mango.blender.org.