2016 IEEE International Symposium on Multimedia

Robust and Efficient Video Scene Detection Using Optimal Sequential Grouping

Daniel Rotman, Dror Porat, Gal Ashour
IBM Research
Haifa, Israel
E-mail: [danieln,drorp,ashour]@il.ibm.com

***Abstract*— Video scene detection is the task of dividing a video into semantic sections. We propose a novel and effective method for temporal grouping of scenes using an arbitrary set of features computed from the video. We formulate the task of video scene detection as a general optimization problem and provide an efficient solution using dynamic programming. Our unique formulation allows us to directly obtain a temporally consistent segmentation, unlike many existing methods, and has the advantage of being parameter-free. We also present a novel technique to estimate the number of scenes in the video using Singular Value Decomposition (SVD) as a low-rank approximation of a distance matrix. We provide detailed experimental results, showing that our algorithm outperforms current state of the art methods. In addition, we created a new Open Video Scene Detection (OVSD) dataset which we make publicly available on the web. The ground truth scene annotation was objectively created based on the movie scripts, and the open nature of the dataset makes it available for both academic and commercial use, unlike existing datasets for video scene detection.**

Keywords-video; scene; temporal segmentation; dynamic programming; video dataset;

I. Introduction

When attempting to extract information from a video, or deduce the semantics and settings of a video scene, it is usually crucial that the video being used be semantically uniform. For this purpose, the task of video scene detection is an essential pre-processing stage to divide longer videos with multiple scenes into semantically coherent segments. Besides this, for very long videos, video scene detection can be essential for video summarization and as an indexing tool to allow fast browsing and retrieval of a relevant part of a video.

A complete video can often be divided at different temporal resolutions, creating a hierarchical decomposition. The individual frames that compose an entire video are the lowest level of such a division. A sequence of frames taken from the same camera at a specific period of time is called a *shot*. A semantic level of division above shots divides a video into scenes, which is the goal of this paper. One of the more general definitions for a video scene was given by [1], where the authors define a scene as a sequence of semantically related and temporally adjacent shots depicting a high-level concept or story.

Many existing methods for video scene detection rely on a division into shots, and then group those shots into scenes by using different clustering methods [2,3,4]. The main drawback of such methods is that temporal consistency is not guaranteed – when clustering shots only according to some extracted feature, shots that are not actually adjacent in the video may be grouped together. Even if temporal closeness is taken into account, there is no guarantee that two very similar shots at opposite sides of the video will not be joined to a scene despite their temporal distance.

We propose a method that divides a video utilizing the inherent structure of shot locations. The only possible groupings examined by the algorithm are those that would lead to a valid scene partitioning. Additional methods exist today that can also divide a video in this way, including the recent and popular method described in [5]. Our methodology is unique in that we propose a much simpler and general approach that can easily be applied to many different types of abstraction, and which outperforms other methods, even when based on relatively simple features. The use of dynamic programming makes our method efficient and with practical running times. Additionally, our method is robust in that it is parameter-free, and requires no training or fine tuning to a specific dataset.

As concluded in a recent survey work [6], one of the major setbacks of the study of video scene detection is the unavailability of a common dataset on which methods can be tested. Therefore, we present a new Open Video Scene Detection (OVSD) dataset. The open nature of the dataset makes it the first of its kind in that it is available also for commercial use by industrial entities, which may have stricter legal constraints regarding the use of existing datasets, such as [7].

The paper is organized as follows: In Section II, a review is given of the relevant literature on the problem of video scene detection. In Section III, we detail our problem definition and formulation for solving the division problem, and the specifics are expanded upon in Section IV. In Section V, we present our new video scene detection dataset and evaluate the results of our method, and in Section VI conclusions are drawn.

II. Related Work

This section provides a review of some recent methods. Note, however, that the goal of video scene detection is not always the same in all of the related literature, and we limit ourselves to considering those that attempt to perform a complete temporal segmentation of a video (as opposed to scene extraction) and are based on visual features. For a recent and more extensive review, see [6].

A. Shot Boundary Detection

As described above, shots are the low-level segments comprising a video, and shot boundary detection is the task of dividing a video into its comprising shots. When performing scene segmentation, shot boundary detection can be utilized as a preliminary step. Due to the very precise and objective definition of a shot and the typical uniformness of the frames in a shot, it is typically considered a solved problem [8].

Various effective methods for shot boundary detection exist. One notable method [9] uses histogram similarity and feature matching to measure the distance between frames, denoting a boundary where the distance between adjacent frames is large. It also measures the distance per frame as a function of time to detect longer stretches where large distances between frames are abundant, in order to identify gradual transitions between shots. Another method [2] uses pixel and histogram distances. It detects gradual transitions by measuring distances between non-adjacent frames with increasingly larger gaps. During our experimentations we used a slightly modified version of [2] with the different distances weighted in a probabilistic way using histogram matching.

We note that although single-shot videos cannot be divided into shots, they may nonetheless consist of a number of scenes. A good example is home videos from handheld cameras, which can be very long and without breaks. For these cases, since precise scene transitions are highly ambiguous, it is sufficient to simply denote shot boundaries every few seconds and proceed with scene detection as needed.

B. Video Scene Detection

In this section, we review a number of methods for video scene detection.

In [4], the authors measure similarity between shots by calculating color and motion similarity aided by key-frames. They construct a shot similarity graph, where the weights are the calculated similarities weighted inversely by the temporal distance between shots. Normalized cuts are performed recursively until a stopping criterion is met, allowing only cuts that divide along the temporal axis.

In [3], shots are clustered only according to visual similarity. Sequence alignment is performed to isolate sections of temporally adjacent shots that display clustering patterns. These patterns are motivated by production rules to identify repetitions of a particular shot of a person or area.

The authors in [5] offer an advanced method using low-level and high-level audio and visual features. After a division into shots, they create multiple Scene Transition Graphs for each type of feature. They perform a probabilistic merging of all of the graphs to arrive at the final division.

In [7], the authors measure shot similarity by weighing the distance between normalized shot histograms, and the temporal distance between shot centers. They use spectral clustering to group shots together, and assign scene transitions between adjacent shots that belong to different clusters.

C. Temporal Clustering

It can be illuminating to explore the literature outside of the field of video scene detection to understand other possible approaches to the very general problem of partitioning elements with a temporal constraint.

In the field of time series analysis, a common task is change point analysis. In a recent work [10], the authors attempt to detect a change in an online stream of data points. They use a kernel discriminant ratio to test the likelihood of a change, and estimate the location of such a change point if it exists.

Switching Dynamical Linear Systems are sometimes used to model a series of observations, each belonging to one of a group of states [11]. Using this formulation one can learn segmentations of data to understand the underlying structure of the samples.

Though these methods are well founded, they lack an important element for scene detection: they fail to compare elements that are far from one another. In a video scene it is sometimes prudent to specifically rely on features that are similar but placed at different ends of the scene. Only a formulation that looks at the series as a whole and measures complete feature distances from beginning to end can hope to correctly segment difficult examples.

III. Optimal Sequential Grouping

We now present our method for video scene detection using optimal sequential grouping. The formulation is represented as a general sequential feature grouping method in order to extend the applicability to numerous feature sets and problem domains. In our problem of video scene detection, the features can be, for example, a color histogram representing each shot.

A. Notations and Definitions

Let us define the information retrieved from a video as a series of N feature vectors of length m : $\{x_1, \dots, x_N\} \mid x_i \in \mathbb{R}^m$. This can represent, for example, a normalized HSV histogram of a key frame representing a shot, the average histogram of all the frames in each shot, or audio features extracted at detected locations. Let us define a distance metric $\mathcal{D} : \mathbb{R}^m \times \mathbb{R}^m \longrightarrow \mathbb{R}$, where $\mathcal{D}$ fulfils all of the conditions of a mathematical distance metric and gives an indication of the dissimilarity between feature vectors.

We wish to detect feature instances that are similar and group them into clusters. Instead of clustering these features disregarding their sequential order or weighting the clustering method by temporal distance (as is typically done in other methods), we divide the video inherently using the following formulation:

Estimating the number of scenes comprising the video, K, is described in Section IV-C. Given K, we must find a monotonically increasing series of indices denoted by $\bar{t} = \{t_1, \dots, t_K\} \mid t_i \in \mathbb{N}$ to represent the division to scenes. t_i is the index of the last feature belonging to scene i, thus $\bar{t}$ marks the endpoints of all of the sections. We note that the monotonously increasing property is essential so that each scene is assigned a number of features that is greater than zero. We set $t_K = N$ to enforce encompassment of all the features detected, and in addition we can define $t_0 = 0$ to simplify the mathematical notation. We denote by $\Omega^K \subsetneq \mathbb{N}^K$ the set of all possible series of length K with the restrictions described above.

B. Intuition

To provide intuition about the solution developed below in Sections III-C and III-D, we first present here a visualization of our method. When calculating $\mathcal{D}$ between all of the N features, we can consolidate the values into a matrix: $D(j,j') = \mathcal{D}(x_j, x_{j'})$. If the features are chosen well so that features in the same scene have a low distance value and features from different scenes have a high value, we can presume that D will look similar to the portrayal in Fig. 1. The blocks on the diagonal represent scenes where groups of sequential features have small distance values between them but large distances to other features in the video. The main diagonal has values of distance zero since $\mathcal{D}(x_j, x_j) = 0$.

The ideal example given in Fig. 1 illustrates that an objective function which exclusively sums values from inside the blocks on the diagonal can be advantageous. An optimal value will be obtained when the divisions are placed such that the objective function sums the blocks with low feature distances.

C. Objective Function

With the distance metric mentioned above and taking the scene division definition into account, we now wish to define an objective function $\mathcal{H}_{\mathcal{D}}^K : \Omega^K \longrightarrow \mathbb{R}$, which assigns a score to any possible division with regard to $\mathcal{D}$ and K. We present an objective function that takes the form

$$\mathcal{H}_{\mathcal{D}}^K(\bar{t}) = \sum_{i=1}^{K} \left\{ \left(\sum_{j=t_{i-1}+1}^{t_i} \sum_{j'=t_{i-1}+1}^{t_i} \mathcal{D}(x_j, x_{j'}) \right) \Big/ S_i \right\}. \quad (1)$$

This objective function sums the intra-group distances for all features grouped into scene i and sums over these group distances for all scenes, where S_i is an optional weighting factor (e.g., to incorporate a domain-specific prior, if such exists). This design is similar to some classic formulations for clustering but with the added inherent grouping of only sequential elements.

For the optimization process below, we generalize using an additional superscript $\mathcal{H}_{\mathcal{D}}^{n,k}$ to indicate that we are using only a subset of features from n to N. In this case, the domain is altered to $\Omega^{n,k}$, which is the set of all possible monotonically increasing series of length k beginning no lower than n and the final value being N. When $\mathcal{H}$ appears with only one superscript, the implication is that $n = 1$.

D. Objective Function Optimization

We now need to find the $\bar{t}$ that minimizes $\mathcal{H}_{\mathcal{D}}^{\mathcal{K}}$. Since the formulation is not a typical definition of clustering, classical clustering solutions cannot be applied. We propose a dynamic programming scheme to arrive at the optimal solution $\bar{t}^*$ efficiently. In the following, we use $S_i = 1$ for simplification. We now describe the stages of the dynamic programming method:

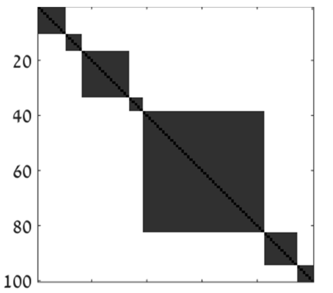

Figure 1. An example of an ideal matrix D. Brighter pixels represent larger distance values.

Let us define a cost table $\mathcal{C}(n,k)$ as the optimal objective function value of $\mathcal{H}_{\mathcal{D}}^{n,k}$. The optimal objective function value of the entire problem is then $\mathcal{C}(1,K)$. Since the value of the objective function alone does not contain information regarding the locations of the divisions, we define $\mathcal{I}(n,k)$ as t_1 of the sub problem of dividing the $N-n+1$ features into k scenes. To reiterate, $\mathcal{I}(n,k)$ specifies where best to place the next boundary. Proof of optimality is shown in IV-A.

We initialize the tables with

$$\mathcal{C}(n,1) = \sum_{j=n}^{N} \sum_{j'=n}^{N} \mathcal{D}(x_j, x_{j'}) \quad (2)$$

and

$$\mathcal{I}(n,1) = N, \quad (3)$$

since dividing the remaining features into one section consists of only one possible solution $\mathcal{C}(n,1) = \mathcal{H}_{\mathcal{D}}^{n,1}(\bar{t} = t_1 = N)$.

Utilizing the linearity of the outer sum of $\mathcal{H}_{\mathcal{D}}^K$ in (1), the rest of the table is defined recursively for every $1 < k \le K$:

$$\mathcal{C}(n,k) = \min_i \left\{ \sum_{j=n}^{i} \sum_{j'=n}^{i} \mathcal{D}(x_j, x_{j'}) + \mathcal{C}(i+1, k-1) \right\} \quad (4)$$

$$\mathcal{I}(n,k) = \operatorname*{argmin}_i \left\{ \sum_{j=n}^{i} \sum_{j'=n}^{i} \mathcal{D}(x_j, x_{j'}) + \mathcal{C}(i+1, k-1) \right\}. \quad (5)$$

The tables have to be constructed serially with k going from 2 to K. The scene boundaries can be reproduced by running i serially from 1 to K with the following formula:

$$t_i^* = \mathcal{I}(t_{i-1}+1, K-i+1). \quad (6)$$

For further efficiency, we can compute the intermediate sums separately. We construct a table Σ of size $N \times N$ that contains the sums

$$\Sigma(i,i') = \sum_{j=i}^{i'} \sum_{j'=i}^{i'} \mathcal{D}(x_j, x_{j'}) \quad (7)$$

for fast referencing. This table does not have to be constructed directly, but instead can itself be calculated using dynamic programming: We initialize $\Sigma(i,i) = \mathcal{D}(x_i, x_i) = 0$ and then recursively define the rest of the table,

$$\Sigma(i,i') = \Sigma(i-1,i') + \Sigma(i,i'-1) - \Sigma(i-1,i'-1) + \mathcal{D}(x_i,x_{i'}) + \mathcal{D}(x_{i'},x_i), \tag{8}$$

which allows Σ to be constructed along the diagonals from the main diagonal and in increasing order.

IV. Method Details

A. Proof of Correctness

We now prove the correctness of the dynamic programming method detailed above. Specifically, we must prove that $\mathcal{C}(1,K)$ is the minimal value of $\mathcal{H}_{\mathcal{D}}^{K}$, and subsequently the indices of the optimal division can be retrieved from $\mathcal{I}$, which holds the optimal division locations. We prove this using mathematical induction, where k will be our induction variable and n will be a parameter, making the proof valid for every n.

1) Statement

$\mathcal{C}(n,k)$ is the optimal value of $\mathcal{H}_{\mathcal{D}}^{n,k}$.

2) Basis

For $k=1$, we have a subset of $N-n+1$ features that must be grouped into one group. There is only one possible division for such a problem for every n, and the solution is to assign all of the features to the same group. This leads to an objective function value of

$$\mathcal{H}_{\mathcal{D}}^{n,1}(\bar{t}=t_1=N) = \sum_{j=n}^{N}\sum_{j'=n}^{N}\mathcal{D}(x_j,x_{j'}), \tag{9}$$

which is precisely the value of $\mathcal{C}(n,1)$ initialized in (2).

3) Inductive Step

Assuming that $\mathcal{C}(n,k)$ is the optimal value of $\mathcal{H}_{\mathcal{D}}^{n,k}$ for every n, we now prove by contradiction that $\mathcal{C}(n,k+1)$ is the optimal value of $\mathcal{H}_{\mathcal{D}}^{n,k+1}$.

Suppose that $\mathcal{C}(n,k+1)$ is *not* the optimal value of $\mathcal{H}_{\mathcal{D}}^{n,k+1}$. We denote $\widetilde{\mathcal{H}}_{\mathcal{D}}^{n,k+1}$ as the optimal $\mathcal{H}_{\mathcal{D}}^{n,k+1}$, $\tilde{t}$ as the solution of $\widetilde{\mathcal{H}}_{\mathcal{D}}^{n,k+1}$, $\tilde{\imath}=\tilde{t}_1$, and i the index chosen in (4) of $\mathcal{C}(n,k+1)$, while under our supposition $\tilde{\imath}\neq i$.

Utilizing the linearity of the outer sum in (1) we obtain

$$\widetilde{\mathcal{H}}_{\mathcal{D}}^{n,k+1} = \sum_{j=n}^{\tilde{\imath}}\sum_{j'=n}^{\tilde{\imath}}\mathcal{D}(x_j,x_{j'}) + \sum_{l=2}^{k+1}\left\{\sum_{j=\tilde{t}_{l-1}+1}^{\tilde{t}_l}\sum_{j'=\tilde{t}_{l-1}+1}^{\tilde{t}_l}\mathcal{D}(x_j,x_{j'})\right\}. \tag{10}$$

The second element is the solution to a sub-problem of dividing $N-(\tilde{\imath}+1)+1$ features into k groups. Because of the linearity of the problem, this second element is in itself an optimal solution of the sub-problem, which we can denote as $\widetilde{\mathcal{H}}_{\mathcal{D}}^{\tilde{\imath}+1,k}$. If this was not so, then by contradiction $\widetilde{\mathcal{H}}_{\mathcal{D}}^{n,k+1}$ could simply replace its own sub-problem solution with $\widetilde{\mathcal{H}}_{\mathcal{D}}^{\tilde{\imath}+1,k}$ and arrive at an improved solution. We now have

$$\widetilde{\mathcal{H}}_{\mathcal{D}}^{n,k+1} = \sum_{j=n}^{\tilde{\imath}}\sum_{j'=n}^{\tilde{\imath}}\mathcal{D}(x_j,x_{j'}) + \widetilde{\mathcal{H}}_{\mathcal{D}}^{\tilde{\imath}+1,k}. \tag{11}$$

When observing the definition in (4)

$$\mathcal{C}(n,k+1) = \min_{i}\left\{\sum_{j=n}^{i}\sum_{j'=n}^{i}\mathcal{D}(x_j,x_{j'}) + \mathcal{C}(i+1,k)\right\}, \tag{12}$$

we see that we can substitute $\widetilde{\mathcal{H}}_{\mathcal{D}}^{i+1,k}$ for $\mathcal{C}(i+1,k)$ according to our induction assumption

$$\mathcal{C}(n,k+1) = \min_{i}\left\{\sum_{j=n}^{i}\sum_{j'=n}^{i}\mathcal{D}(x_j,x_{j'}) + \widetilde{\mathcal{H}}_{\mathcal{D}}^{i+1,k}\right\}. \tag{13}$$

We assumed by contradiction that $\mathcal{C}(n,k+1) > \widetilde{\mathcal{H}}_{\mathcal{D}}^{n,k+1}$, but (11) is one of the possible values in the minimum function in (13), which means that $\mathcal{C}(n,k+1) \leq \widetilde{\mathcal{H}}_{\mathcal{D}}^{n,k+1}$. We have arrived at a contradiction, which confirms that $\mathcal{C}(n,k+1)$ is the optimal value of $\mathcal{H}_{\mathcal{D}}^{n,k+1}$. ■

B. Computational Complexity

We propose the dynamic programming method detailed above to overcome the inability to estimate all possible divisions in a brute-force manner. The number of possible divisions is the number of possibilities to choose $K-1$ boundaries from the $N-1$ places between features, given by

$$\mathcal{C}\binom{N-1}{K-1} = \frac{(N-1)!}{(K-1)!\,(N-K)!}, \tag{14}$$

which is impractical to compute for real-world problems.

On the other hand, using our method, two tables of size $K\times N$ are constructed. For each element in the table the minimum function considers at most N values using the Σ table. The construction of the Σ table is negligible to this and we arrive at a time complexity of $O(K\cdot N^2)$.

C. Determining the Number of Scenes K

Identifying the number of clusters is a common problem in data clustering. In our description above, we assumed that K was known. Here we discuss how K can be estimated, and we present a new novel approach.

Since the matrix D resembles a dissimilarity matrix in classical clustering problems, it is possible to use a number of methods for predicting K taken from the field of data clustering. A noteworthy method is the gap statistic [12], which compares the improvement in clustering distance as K increases between the existing data and a null distribution (i.e., an artificial dataset that is uniformly distributed).

Motivated by the visualization of the problem mentioned in Section III-B, we further develop a novel method for determining K that is unique to our formulation. Clustering problems which do not require sequential grouping cannot utilize the convenient structure of the matrix D in this problem. As can be seen in the ideal matrix D in Fig. 1, due to the linear dependency between some of the rows, the rank of the matrix is approximately the number of blocks on the diagonal. In practice, when noise is present, performing a low-rank approximation of D [13] will leave the block structure and eliminate noise artifacts. This allows us to estimate K using Singular Value Decomposition (SVD), a commonly used tool for low-rank approximation.

Using SVD, we obtain the singular values of D. We wish to determine the number of significant singular values in order to decide on the correct level for low-rank approximation. By plotting the sorted singular values, we can identify the point in the graph where the difference in magnitude of the values begins to level off. As in the problem of estimating the number of clusters, we attempt to identify the "elbow" in the singular values graph [14].

Since the singular values decrease in an exponential manner, we compute the log of the singular values that are greater than 1. Here we detect the point on the graph that is farthest from the line drawn from beginning to end of the graph. This gives a good indication of the elbow point as can be seen in Fig. 2. If we denote the vector s as the m values in the log graph of the singular values, then the two-dimensional vector from the beginning of s to a point at index i on s is $I_i \triangleq [i\ ,\ s_i]^T$. The diagonal over the graph is $H \triangleq [m-1\ ,\ s_m - s_1]^T$, and the elbow point is obtained by

$$elbow\_index = \underset{i}{\operatorname{argmax}} \left\{ I_i - \frac{I_i^T H}{\|H\|} \cdot \frac{H}{\|H\|} \right\}. \tag{15}$$

In Section V-B, we assess experimentally the results of this method for determining K.

V. Experimental Results

As mentioned above, the availability of evaluation datasets in the field of video scene detection is limited. To aid in overcoming the lack of an available, comprehensive and objective benchmark, we present the Open Video Scene Detection (OVSD) dataset based on open videos, allowing both academic and industrial research and use.

The OVSD dataset consists of five short films from Blender Institute [15], and an open full length film – 'Valkaama' [16]. The ground truth shot boundaries were annotated by hand, while the scene divisions were objectively extracted from the movie scripts, as detailed in the dataset information. Table I consolidates the details regarding the videos, and Fig. 3 shows some representative frames. The entire annotated dataset can be downloaded at: `https://www.research.ibm.com/haifa/av/OS_VSD_Dataset.zip`.

For our method, we begin by running Shot Boundary Detection, and as features we used the normalized RGB histograms for each shot, while $\mathcal{D}$ was taken as the Bhattacharyya distance.

To assess video scene detection performance, we use the widely accepted Coverage C and Overflow O measures proposed in [17]. The harmonic mean of C and $1-O$, denoted by F, represents a single score for the quality of the scene detection.

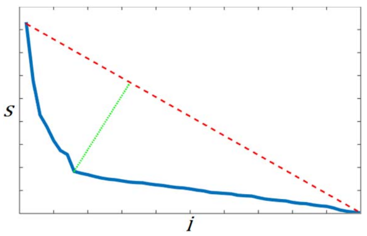

Figure 2. Detecting K using the elbow in the singular value log curve s.

TABLE I. OVSD Dataset

Video Name	Duration hh:mm:ss	Number of Scenes
Elephants Dream	00:09:22	8
Big Buck Bunny	00:08:08	13
Sintel	00:12:24	7
Tears of Steel	00:09:48	9
Cosmos Laundromat	00:09:59	6
Valkaama	01:33:05	49

A. Results

We evaluated our method and our implementation of [7] on the OVSD dataset. In addition, we compared our results to the method of [5] using the authors' provided executable, with the following limitations: Since the provided executable works on videos up to 10 minutes in length, when running on 'Sintel' the movie was shortened by a scene, and 'Valkaama' was divided into 10 parts, with the results averaged. Table II shows the results, with the highest F score for each video in bold. Regarding the implementation of [7], we chose the value of σ so as to maximize the performance of [7] on this dataset. The other parameters were detailed in the paper.

As shown in Table II, our method clearly outperforms the current state of the art methods. In addition to the objective improvement in F score, our method also has the advantage of being parameter-free, and unlike the other methods, requires no training or fine-tuning for a specific dataset. In fact, our method was developed using different internal proprietary videos, and was only evaluated on the OVSD dataset for a completely fair comparison of the other methods.

B. Accuracy of $elbow\_index$

In order to evaluate the accuracy of our method for approximating K, we performed automated simulations. We created randomly generated D matrices, with K chosen randomly between 1 and 10, and N chosen to be $10 \cdot \alpha \cdot K$, where α was chosen randomly between 1 and 10

TABLE II. Results

Video Name	Ours			[5]			[7]		
	F	***C***	***O***	***F***	***C***	***O***	***F***	***C***	***O***
ED	**0.69**	0.63	0.23	0.56	0.69	0.52	0.55	0.73	0.56
BBB	**0.69**	0.71	0.33	0.49	0.77	0.64	0.46	0.65	0.65
Sintel	**0.66**	0.55	0.17	0.59	0.62	0.43	0.51	0.81	0.63
ToS	0.62	0.57	0.32	**0.75**	0.92	0.37	0.23	0.88	0.87
CL	**0.53**	0.47	0.40	0.45	0.43	0.53	0.07	0.85	0.97
Valkaama	**0.73**	0.96	0.42	**0.73**[a]	0.88[a]	0.32[a]	0.16	0.94	0.91

a. Score received by averaging the results over 10 parts of the film.

Figure 3. Two videos from the presented OVSD dataset. Representative frames were chosen as the center frame of each scene as detected by our method.

as well. This gives us a variability between the number of scenes and the number of shots. We generated random D matrices with values drawn uniformly from the interval $[0,1]$. The values on the block diagonal were squared and the other values were square-rooted, thus leaving low values on the block diagonal and high values outside.

We estimated K with our SVD log-elbow method, and compared to the real value of K. In Fig. 4 we present a chart of the estimated $elbow\_index$ versus the ground truth K. The results show a strong correlation across the diagonal with a slightly increasing deviation as K rises.

VI. Conclusions

In this paper, we presented a method for temporal grouping of scenes using an arbitrary set of features computed from the video. We showed how our unique and parameter-free formulation of the problem can lead to an efficient solution using dynamic programming. Our experimental results clearly demonstrate the effectiveness of the proposed method, which outperforms current state of the art methods.

In addition, we created a new video scene detection dataset which we make publicly available, allowing both academic and commercial use. Our hope is that this dataset will accelerate future work in video scene detection.

References

[1] Y. Rui, T. S. Huang, and S. Mehrotra, “Constructing table-of-content for videos,” *Multimedia systems*, vol. 7, no. 5, pp. 359–368, 1999.

[2] L. Baraldi, C. Grana, and R. Cucchiara, “Shot and scene detection via hierarchical clustering for re-using broadcast video,” in *International Conference on Computer Analysis of Images and Patterns*. Springer, 2015, pp. 801–811.

[3] V. T. Chasanis, A. C. Likas, and N. P. Galatsanos, “Scene detection in videos using shot clustering and sequence alignment,” *IEEE Transactions on Multimedia*, vol. 11, no. 1, pp. 89–100, 2009.

[4] Z. Rasheed and M. Shah, “Detection and representation of scenes in videos,” *IEEE transactions on Multimedia*, vol. 7, no. 6, pp. 1097–1105, 2005.

[5] P. Sidiropoulos, V. Mezaris, I. Kompatsiaris, H. Meinedo, M. Bugalho, and I. Trancoso, “Temporal video segmentation to scenes using high-level audiovisual features,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 8, pp. 1163–1177, 2011.

[6] M. Del Fabro and L. Böszörmenyi, “State-of-the-art and future challenges in video scene detection: a survey,” *Multimedia systems*, vol. 19, no. 5, pp. 427–454, 2013.

[7] L. Baraldi, C. Grana, and R. Cucchiara, “Analysis and re-use of videos in educational digital libraries with automatic scene detection,” in *11th Italian Research Conference on Digital Libraries*. Springer, 2015, pp. 155-164.

[8] A. F. Smeaton, P. Over, and A. R. Doherty, “Video shot boundary detection: Seven years of trecvid activity,” *Computer Vision and Image Understanding*, vol. 114, no. 4, pp. 411–418, 2010.

[9] E. Apostolidis and V. Mezaris, “Fast shot segmentation combining global and local visual descriptors,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 6583–6587.

[10] Z. Harchaoui, E. Moulines, and F. R. Bach, “Kernel change-point analysis,” in *Advances in Neural Information Processing Systems*, 2009, pp. 609–616.

[11] E. Fox, E. B. Sudderth, M. I. Jordan, and A. S. Willsky, “Nonparametric bayesian learning of switching linear dynamical systems,” in *Advances in Neural Information Processing Systems*, 2009, pp. 457–464.

[12] R. Tibshirani, G. Walther, and T. Hastie, “Estimating the number of clusters in a data set via the gap statistic,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001.

[13] I. Markovsky, “Low rank approximation: algorithms, implementation, applications,” 2011.

[14] C. A. Sugar, L. A. Lenert, and R. A. Olshen, “An application of cluster analysis to health services research: Empirically defined health states for depression from the sf-12,” 1999.

[15] “Blender institute.” [Online]. Available: https://www.blender.org/-institute/

[16] “Valkaama.” [Online]. Available: http://www.valkaama.com/

[17] J. Vendrig and M. Worring, “Systematic evaluation of logical story unit segmentation,” *IEEE Transactions on Multimedia*, vol. 4, no. 4, pp. 492–499, 2002.

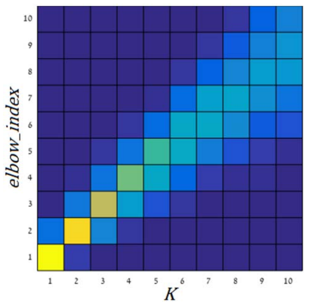

Figure 4. Brighter squares represent a higher probability of estimating a particular $elbow\_index$ for a specific K.