Optimally Grouped Deep Features Using Normalized Cost for Video Scene Detection

Daniel Rotman, Dror Porat, Gal Ashour, Udi Barzelay
IBM Research
Haifa, Israel
[danieln,drorp,ashour,udib]@il.ibm.com

ABSTRACT

Video scene detection is the task of temporally dividing a video into its semantic sections. This is an important preliminary step for effective analysis of heterogeneous video content. We present a unique formulation of this task as a generic optimization problem with a novel normalized cost function, aimed at optimal grouping of consecutive shots into scenes. The mathematical properties of the proposed normalized cost function enable robust scene detection, also in challenging real-world scenarios. We present a novel dynamic programming formulation for efficiently optimizing the proposed cost function despite an inherent dependency between subproblems. We use deep neural network models for visual and audio analysis to encode the semantic elements in the video scene, enabling effective and more accurate video scene detection. The proposed method has two key advantages compared to other approaches: it inherently provides a temporally consistent division of the video into scenes, and is also parameter-free, eliminating the need for fine-tuning for different types of content. While our method can adaptively estimate the number of scenes from the video content, we also present a new non-greedy procedure for creating a hierarchical consensus-based division tree spanning multiple levels of granularity. We provide comprehensive experimental results showing the benefits of the normalized cost function, and demonstrating that the proposed method outperforms the current state of the art in video scene detection.

CCS CONCEPTS

• **Computing methodologies** → **Computer vision**; **Computer vision tasks**; **Visual content-based indexing and retrieval**; Hierarchical representations; • **Theory of computation** → *Discrete optimization*; *Dynamic programming*;

KEYWORDS

video scene detection; temporal segmentation; optimization; normalized cost; dynamic programming; hierarchy creation

ACM Reference Format:
Daniel Rotman, Dror Porat, Gal Ashour, Udi Barzelay. 2018. Optimally Grouped Deep Features Using Normalized Cost for Video Scene Detection. In *ICMR '18: 2018 International Conference on Multimedia Retrieval, June 11–14, 2018, Yokohama, Japan.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3206025.3206055

ICMR '18, June 11–14, 2018, Yokohama, Japan

ACM ISBN 978-1-4503-5046-4/18/06…$15.00
https://doi.org/10.1145/3206025.3206055

1 INTRODUCTION

Video scene detection is the task of temporally dividing a heterogenous video into its semantic sections, called *scenes*. Scenes typically relay a specific concept or theme which acts as a component of the story delivered by the video. Formally, scenes are defined as a sequence of semantically related and temporally adjacent shots depicting a high-level concept or story [23]. The *shots* that compose the scenes are a series of frames taken from the same camera at the same time; different shots are separated by a shot boundary, usually in the form of an abrupt cut or gradual fade transition. Due to the relative uniformity of the frames in a shot, detecting the boundaries of shots is relatively simple and usually considered to be a solved problem [27]. There are a number of available methods for this task that can be used off-the-shelf with impressive performance [1, 4]. However, dividing videos into scenes, which is the goal of this work, is considered to be a much more challenging task on account of the semantic nature of scenes.

The rise in video content volume in recent years [17] has increased the importance of developing technologies that can help manage, recommend, and consume video content. For long heterogeneous videos, it can be difficult for consumers, and in particular for data analysts (in a professional context), to understand the contents of the video and the location of the information they seek. For this, video scene detection is a powerful tool. We can use the ability to partition the video into scenes to help build a table-of-contents, enable fast browsing and skipping between scenes, and identify the contextual boundaries of the content in the video. In addition, the tasks of summarizing, retrieving, understanding, and classifying video content can benefit greatly when dealing with multi-chaptered videos. For video navigation in particular, creating a hierarchy of division over multiple levels of granularity can be especially beneficial for providing insight as to the structure of the video (see Section 6).

A common approach for video scene detection is to equate the problem to a standard clustering task. Such an approach neglects an important element, which is the *temporal consistency* of video scenes: Since video scenes have an obvious ingrained temporal structure and sequence, clustering shots independently of their location and order in the video can lead to situations of marking a scene as a non-contiguous segment. Not only does this not make sense, but it also contradicts the definition of a scene. One obvious problematic example is classic ping-pong shot sequences, in which the camera alternates between two entities/views. Another involves

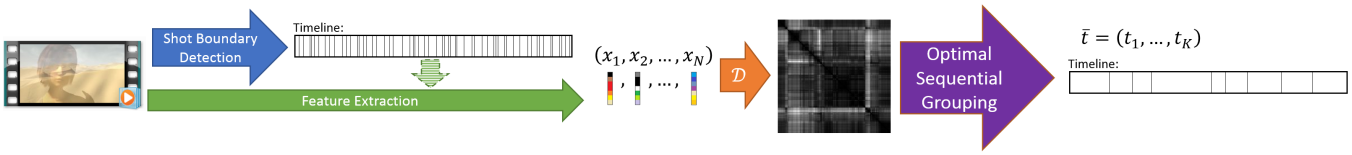


Figure 1: An overview of the proposed video scene detection method. Given a video shot boundary detection is performed (blue arrow) and from each shot a representative feature is extracted (green arrow). A distance matrix D is created from the features (orange arrow) on which the optimal sequential grouping algorithm can be performed (purple arrow) to result in the optimal division into scenes. (Best viewed in color.)[1]

outlier shots, where a single shot in a sequence looks completely different because of a close-up or angle shift. Even when a method considers the temporal distance of shots and enforces temporal consistency using some heuristics, the result remains suboptimal since the video structure is treated as an enforcement instead of as an integral property.

In contrast, in this paper we extend our work in [20], which formalizes video scene detection as an optimization process of sequential grouping. Utilizing the structure of the video, we choose optimal locations for division based on features that represent the shots in the video, and a distance metric to measure the (dis-)similarity between them. As opposed to our previous approach [20], we incorporate a normalized cost function which has important mathematical properties that make it appropriate for difficult situations such as widely varying scene lengths and weak scene indications (as detailed in Section 3.3). In addition, we incorporate the use of deep neural networks to better capture the semantic elements in a video scene. Using deep features helps to more accurately identify the underlying and correlative elements in a scene, enabling scene partitioning which requires high-level understanding.

A general overview of the proposed method is shown in Figure 1. Given a video, we perform shot boundary detection [4] and extract a representative feature vector from each shot, which allows a distance between each pair of shots to be calculated. We formulate an optimization problem with a normalized cost function over the computed distance values. We minimize the cost function to obtain the optimal division into scenes.

The proposed method for video scene detection has substantial advantages. It is parameter-free, and thus avoids the need for fine-tuning to make the method applicable to different types of content. It is also efficient due to a novel dynamic programming scheme which allows fast scene partitioning. Furthermore, we formulate the method as a general optimization process for sequential data, thus making it applicable to various other tasks, such as audio partitioning or data analysis for change point detection. For video scene detection, our comprehensive evaluation shows that the proposed method outperforms the state of the art on real-world videos of a variety of genres.

The major contributions in this work are as follows: 1. Development of a novel normalized cost function, which we show to be superior for performing the sequential grouping task, both analytically and in practice. 2. A novel dynamic programming formulation for efficiently optimizing the proposed cost function despite an inherent dependency between subproblems (see Section 4.1). 3. Incorporation of deep neural networks for visual and audio analysis to identify semantic elements that compose a scene. 4. A novel non-greedy algorithm for constructing a hierarchical division tree using a set of optimal divisions.

This paper is organized as follows: In Section 2, we review the relevant literature on the problem of video scene detection. In Section 3, we detail the problem definition and our formulation, and introduce the normalized cost function. Section 4 details the optimization process. In Section 5 we describe the incorporation of deep neural networks for multimodal feature extraction. Section 6 details our non-greedy algorithm for constructing a hierarchical division tree. Section 7 presents our evaluation, and in Section 8 we draw conclusions.

2 PREVIOUS WORK

In this section, we review some of the previous work in video scene detection. We limit ourselves to methods that aim to perform an entire partitioning of a video as opposed to scene extraction. For a more general review, see [8].

One methodology for video scene detection makes assumptions about video editing and uses *editing rules* to isolate video scenes [7, 10]. These methods use heuristics derived from the film industry to attempt to identify scene changes by recognizing the types of shots (wide shot, close-up, etc.) and learning the predictable behavior. Many of these rules are highly prevalent in the mainstream film industry (e.g., Hollywood movies). However, relying on them drastically limits the capabilities to a very particular genre and production type. For example, the rules learned for documentary films will most likely be completely different than the rules for news broadcasts or sports videos. This limitation makes these methods suitable only for application-specific tasks.

Clustering methods [2, 3, 5, 18] use a variety of advanced schemes to depict shots and the relationship between them. These methods then typically group the shots together using standard clustering techniques. As mentioned above, this results in the need to introduce heuristics that enforce temporal consistency, since the shot clusters may not all be temporally aligned. Specifically, situations such as outlier shots and ping-pong shot sequences are likely to cause complete mislabeling of the scene (see Figure 2).

Graph methods for video scene detection are also quite prevalent [19, 24, 25, 28]. These model the video as a connected graph and employ known algorithms such as graph cuts or dominant sets to detect the video scene boundaries. One state-of-the-art method [25] uses high-level and low-level audio-visual representations over

[1]Image taken from Sintel ©copyright Blender Foundation | durian.blender.org

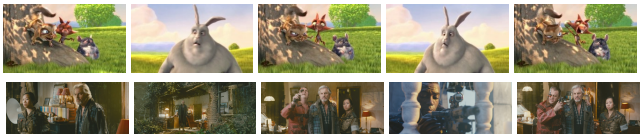


Figure 2: Examples of shot sequences which can lead to a temporally inconsistent division to scenes. (Top) Ping-pong sequences where there is an interaction between two entities in a scene. (Bottom) Outlier shots, where a single shot (fourth from the left) in the middle of a scene looks very different. (Best viewed in color.)[2]

multiple scene transition graphs to calculate the probabilistic confidence of a scene transition. It is interesting to note that these methods lack a unified agreement on how to represent the video. For example, some represent shots as nodes while others represent shot clusters as nodes. Since there is a lack of a joint formulation, the motivation, rationalities, and conclusions from experiments cannot be shared or migrated between methods.

More advanced methods use probabilistic models and stochastic processes [6, 15, 32] to describe a video. Developing or learning a probabilistic model enables a mathematical representation of the video and calculating the probabilities for the existence of scene boundaries. These methods typically cannot guarantee an optimal solution, and often rely on multiple initializations in order to explore and test multiple probabilistic configurations.

3 PROBLEM DEFINITION AND NOTATION

We now present our method for video scene detection with normalized cost. Since the formulation and the optimization process can be applied to multiple types of features and even to different problems, we represent our method in a general formulation as a sequential grouping problem. For the problem of video scene detection, the features can be, for example, color histograms representing each shot, high-level visual concepts, audio features, a multimodal combination, and so forth [21]. In Section 5 we discuss the deep features we use in this work to achieve accurate and robust video scene detection.

3.1 Notation and Definitions

Let us define a series of N feature vectors of length m with a defined order $X_1^N = \{x_1, \ldots, x_N\}$, where $\forall i, x_i \in \mathbb{R}^m$. To measure the distance between feature vectors, we define: $\mathcal{D} : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$ such that $\mathcal{D}$ fulfills the mathematical conditions of a distance metric. $\mathcal{D}$ is chosen to return a low value for feature vectors that are similar.

The goal is to find a vector of indexes: $\bar{t} = (t_1, \ldots, t_K)$ that denotes the partitioning into segments. K represents the number of segments (scenes, in the problem of video scene detection). We discuss estimating this parameter in Section 4.2. The vector $\bar{t}$ is limited to $\forall i, t_i \in \mathbb{N}$, and must be monotonically increasing ($t_i > t_j$ for $i > j$), with t_i indicating the last feature vector index in segment i. Thus $t_K = N$, and we can define $t_0 = 0$ to simplify

[2]Images taken from (top) Big Buck Bunny ©copyright Blender Foundation | peach.blender.org and (bottom) Tears of Steel ©copyright Blender Foundation | mango.blender.org

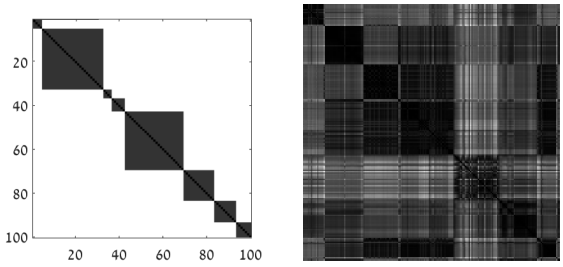


Figure 3: D matrices: illustration (left) and real data (right). Brighter pixels represent larger distance values.

mathematical notation. The monotonicity is enforced to ensure that every segment has at least one feature. We use $\Omega^K \subsetneq \mathbb{N}^K$ to denote the set of all possible $\bar{t}$'s that fulfill these restrictions.

3.2 Intuition

Before entering the technical mathematical formulation, we provide a visualization of the problem to explain our intuition regarding the effectiveness of the method. Taking the values of $\mathcal{D}$ and placing them in a two-dimensional array, we arrive at a matrix D in which the value at the i-th row and j-th column is the value obtained by $\mathcal{D}(x_i, x_j)$. This matrix is symmetrical due to the symmetry property of the distance metric, and the elements on the main diagonal are zeros because $\mathcal{D}(x_i, x_i) = 0$.

Our goal in dealing with the problem of video scene detection is to choose the features and distance metric to return a small distance between feature vectors belonging to the same scene, and large distances for features from different scenes. In an ideal case, we would expect the matrix D to look similar to the illustration in Figure 3. In this visualization, larger values are portrayed by brighter pixels. In the ideal matrix D there is an obvious *block-diagonal* structure, in which groups of features extracted from adjacent shots return low distance values between them, resulting in low valued blocks on the diagonal. It is necessary to create a cost function in which placing the elements of $\bar{t}$ on the edges of the blocks on the diagonal would return the global minimum when minimizing this cost.

3.3 Cost Function

Given the motivation described above, we define a cost function $\mathcal{H} : \Omega^K \rightarrow \mathbb{R}$ that maps a possible $\bar{t}$ to a value indicating the quality of the division.

There are various possibilities for the cost function $\mathcal{H}$. The most obvious form is an *additive cost function*

$$\mathcal{H}_{add}(\bar{t}) = \sum_{i=1}^{K} \sum_{j_1=t_{i-1}+1}^{t_i} \sum_{j_2=t_{i-1}+1}^{t_i} \mathcal{D}(x_{j_1}, x_{j_2}). \quad (1)$$

This cost function, used in our previous work [20], sums up all the values in the blocks on the diagonal. The assumption is that these values bordered by $\bar{t}$ should be substantially lower than the values outside the blocks on the diagonal. Another possible cost

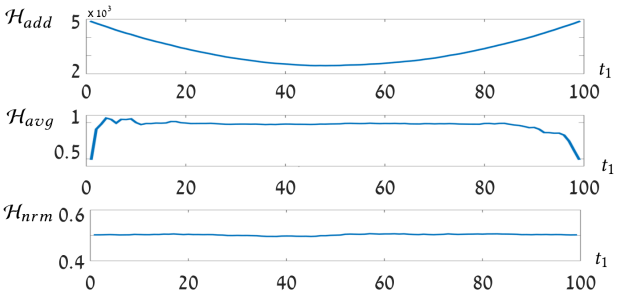


Figure 4: From top to bottom: $\mathcal{H}_{add}$, $\mathcal{H}_{avg}$, and $\mathcal{H}_{nrm}$ for uniformly distributed values of D. Presented here is the cost function value for $N = 100$ and $K = 2$ as a function of the point of division t_1.

function is the *average cost function*

$$\mathcal{H}_{avg}(\bar{t}) = \sum_{i=1}^{K} \frac{\sum_{j_1=t_{i-1}+1}^{t_i} \sum_{j_2=t_{i-1}+1}^{t_i} \mathcal{D}(x_{j_1}, x_{j_2})}{(t_i - t_{i-1})^2}, \tag{2}$$

which sums up the average value of each block on the diagonal.

In this paper we propose the *normalized cost function*

$$\mathcal{H}_{nrm}(\bar{t}) = \frac{\sum_{i=1}^{K} \sum_{j_1=t_{i-1}+1}^{t_i} \sum_{j_2=t_{i-1}+1}^{t_i} \mathcal{D}(x_{j_1}, x_{j_2})}{\sum_{i=1}^{K} (t_i - t_{i-1})^2 - N}, \tag{3}$$

which measures the average value of the entire block diagonal as partitioned by $\bar{t}$. Note that the N subtracted in the denominator is optional and is intended to exclude the values on the main diagonal itself (which are zero) from being taken into account.

To better understand the properties of the different cost functions, we conduct the following experiment: We create a synthetic simulation under a null hypothesis of the data showing no inclination toward a specific division. This is achieved by creating values of D that are uniformly distributed, and thus all points of division should be equally acceptable. Figure 4 shows the three cost functions as a function of t_1 for a division into $K = 2$. We expect the cost function to have a constant response over the point of division due to the null hypothesis stated above. However, we can see that only $\mathcal{H}_{nrm}$ results in the expected behavior, while $\mathcal{H}_{add}$ and $\mathcal{H}_{avg}$ introduce certain biases for the division location.

These biases are mathematical properties of the $\mathcal{H}_{add}$ and $\mathcal{H}_{avg}$ cost functions. For the additive cost function, there is a strong inclination to divide the sequence roughly in the middle since the number of values being summed is the smallest ($N^2/2$ compared to roughly N^2 at the borders). There are two typical scenarios in which the additive cost function will fail to locate a correct partitioning: 1. when the blocks vary greatly in size (see Figure 8), and 2. when the block structure is not very pronounced (see Figure 5). Portraying the opposite behavior, the average cost function prefers partitioning near the borders. This is because a small block has a much higher probability of giving an extremely low average value compared to bigger blocks (since there are less values being accumulated). This results in the algorithm always partitioning as many sequences into single-feature blocks as possible. Due to this limitation it is unusable in practice and therefore not considered further in this paper.

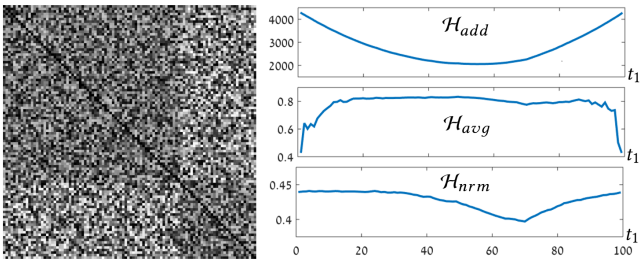


Figure 5: (Left) A synthetic D matrix with a slight inclination for division. (Right) From top to bottom, $\mathcal{H}_{add}$, $\mathcal{H}_{avg}$, and $\mathcal{H}_{nrm}$ in a similar representation as in Figure 4. Only $\mathcal{H}_{nrm}$ allows the identification of the subtle inclination for a division in D.

The normalized cost function gives the desired mathematical formulation to allow unbiased and correct partitioning overcoming the difficulties mentioned above. We evaluate the normalized cost function versus the additive cost function over large permutations of synthetic data in Section 7.1.

4 OPTIMAL SEQUENTIAL GROUPING WITH NORMALIZED COST

Given the normalized cost function, we perform an optimization to find the $\bar{t}$ that globally minimizes $\mathcal{H}_{nrm}$ using dynamic programming.

Dynamic programming is a method to solve complex problems by breaking them down into subproblems and conserving their solutions. Minimizing $\mathcal{H}_{nrm}$, as opposed to $\mathcal{H}_{add}$ and $\mathcal{H}_{avg}$, is especially difficult since the solution to any given subproblem depends on the solution of the complete problem. For example, given $X_n^N = \{x_n, \ldots, x_N\}$, we want to find the solutions to the subproblems of dividing a subset of the features $X_{i>n}^N$. This is possible with $\mathcal{H}_{add}$ and $\mathcal{H}_{avg}$ due to the linearity of their outer sum. However, in $\mathcal{H}_{nrm}$, finding the optimal t_i depends on $(t_1, \ldots, t_n, \ldots, t_{i-1})$ since they affect the denominator.

In the next subsection (4.1), we develop a novel and unique dynamic programming scheme to solve the optimization problem, while overcoming this difficulty.

4.1 Optimization

In our dynamic programming scheme, the only external element that affects the sub-sequence optimizations is a single value—the denominator in (3). The denominator can be treated as an unknown parameter $p \in \mathbb{N}$, and the optimization process can be applied to every possible value of this parameter. We call this parameter *area*, since in effect this is the area of the blocks on the diagonal. We show in Section 4.3 that although the evaluation over all possible values of the parameter might seem costly, in practice the number of possibilities can be greatly limited. In Section 7.1 we present actual running times showing that the method is very practical even in real-world scenarios.

We now detail the optimization procedure using our novel dynamic programming scheme. We define superscripts $\mathcal{H}^{n,k}$ to indicate a subproblem in which a sub-sequence of the feature vectors

$X_n^N = \{x_n, \ldots, x_N\}$ is divided into k segments. When we omit the superscripts, the assumption is that $n = 1$ and $k = K$. Three tables C, $\mathcal{I}$, and $\mathcal{P}$ are defined as follows: $C(n,k,p)$ is the optimal value of $\mathcal{H}_{nrm}^{n,k}$ where the unknown parameter equals p. p is assumed to be the area added to the denominator from dividing the rest of the sequence X_1^{n-1}. $\mathcal{I}(n,k,p)$ preserves the optimal t_1 of the solution to $\mathcal{H}_{nrm}^{n,k}$ in order to enable recreation of the partitioning, and $\mathcal{P}(n,k,p)$ is the area used in this optimal solution contributed by X_n^N.

We initialize the tables for $k = 1$. This leaves only one possibility, grouping all the features together, which results in:

$$C(n,1,p) = \frac{\sum_{j_1=n}^{N}\sum_{j_2=n}^{N}\mathcal{D}(x_{j_1},x_{j_2})}{p+(N-n+1)^2-N}, \tag{4}$$

$$\mathcal{I}(n,1,p) = N, \tag{5}$$

and

$$\mathcal{P}(n,1,p) = (N-n+1)^2, \tag{6}$$

for every n and p. We then fill the rest of the table for k in ascending order.

The logic is as follows. Given a sequence of features X_n^N, allocate the group of features X_n^i into a single group, and the rest X_{i+1}^N into $k-1$ groups. By choosing the i that minimizes the cost, we ensure that as k grows, the optimal solution is chosen. For mathematical neatness, we define:

$$G(n,k,p)(i) = \frac{\sum_{j_1=n}^{i}\sum_{j_2=n}^{i}\mathcal{D}(x_{j_1},x_{j_2})}{p+(i-n+1)^2+\mathcal{P}\left(i+1,k-1,p+(i-n+1)^2\right)-N} \tag{7}$$

as the contribution of X_n^i to the cost function. The denominator includes the entire area of the blocks on the diagonal, composed of three elements: p, which is the area of solving for X_1^{n-1}; $(i-n+1)^2$, which is the area for X_n^i; and $\mathcal{P}\left(i+1,k-1,p+(i-n+1)^2\right)$, which is the area for X_{i+1}^N.

Therefore, the tables are filled with the following formulas:

$$C(n,k,p) = \min_i \left\{G_{n,k,p}(i) + C\left(i+1,k-1,p+(i-n+1)^2\right)\right\}, \tag{8}$$

$$\mathcal{I}(n,k,p) = \operatorname*{argmin}_i \left\{G_{n,k,p}(i) + C\left(i+1,k-1,p+(i-n+1)^2\right)\right\}, \tag{9}$$

and

$$\mathcal{P}(n,k,p) = (\mathcal{I}(n,k,p)-n+1)^2 + \mathcal{P}\left(\mathcal{I}(n,k,p)+1,k-1,p+(\mathcal{I}(n,k,p)-n+1)^2\right). \tag{10}$$

See Figure 6 for an illustration of this dynamic programming scheme. The value of $\mathcal{P}(n,k,p)$ is composed of contributions to the denominator from X_n^i and X_{i+1}^N, where the optimal point for division i is chosen as $\mathcal{I}(n,k,p)$.

Once the tables have been filled, the optimal cost function value is then $\tilde{\mathcal{H}}_{nrm}^K = C(1,K,0)$. To recreate the optimal division locations $\tilde{t}$, we use the following recursive formula: initialize $\mathcal{P}_{tot} = 0$ and $\tilde{t}_0$, and then for i running from 1 to K:

$$\tilde{t}_i \leftarrow \mathcal{I}\left(\tilde{t}_{i-1}+1, K-i+1, \mathcal{P}_{tot}\right), \tag{11}$$

$$\mathcal{P}_{tot} \leftarrow \mathcal{P}_{tot} + \left(\tilde{t}_i - \tilde{t}_{i-1}\right)^2. \tag{12}$$

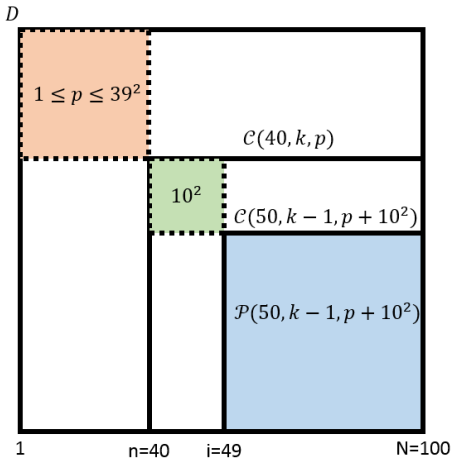


Figure 6: An illustration depicting the operation of the dynamic programming scheme. The entire block represents D, and presented is solving the subproblem beginning at $n = 40$. One of the possibilities $i = 49$ is calculated over possible values of p—the area contributed from the preceding feature vectors (orange). The calculated value $C(40,k,p)$ uses a subsolution $C(50,k-1,p+10^2)$ (blue) which begins at index 50, divides into $k-1$ blocks, and the area outside of the subsolution is p plus the area for the segmented block of features 40-49 (green). (Best viewed in color.)

As a sanity check, the process should end meeting the following equality:

$$\mathcal{P}_{tot} = \mathcal{P}(1,K,0). \tag{13}$$

4.2 Estimating the Number of Scenes K

In the process above, we performed the sequential grouping given a particular number of groups K. In many applications, K can be left as an adjustable parameter used to customize the partitioning to a particular level of granularity. Furthermore, a series of results over multiple values of K could be returned denoting a hierarchy of group partitioning. In Section 6 we detail our novel non-greedy algorithm for creating a hierarchical division tree.

When a hard decision has to be made, such as for the analytical evaluation of performance, K can be estimated similar to the estimation of the number of clusters in a classic clustering problem. The gap statistic [30] is an especially noteworthy candidate; it calculates the number of scenes according to the improvement in clustering distance compared with data under a null hypothesis. In practice, we use our method in [20] which automatically detects the *elbow_index*—the elbow in the graph of the log singular values of the matrix D. This index has been shown to reliably estimate the number of blocks on the diagonal of the matrix D due to the equivalence between the problem and low-rank matrix approximation.

4.3 Limiting p

The tables detailed in the optimization process above have to be calculated for all possible values of the parameter p. However, it

can be useful to remove values of p that are not plausible. Given n for a specific subproblem, one way to limit p is by keeping it within the range $\left[n, n^2\right]$. If k is also known, the range can be further limited to $\left[\left\lceil \frac{n^2}{k} \right\rceil, (n-k+1)^2 + k - 1\right]$. This is due to the fact that the maximum area is obtained for one large group of size $n-k+1$, and $k-1$ single-featured groups. The minimum area is obtained by dividing the n features into k groups as equally as possible, resulting in an area of $\left\lceil k \cdot (\frac{n}{k})^2 \right\rceil$.

The process can be improved further by calculating exactly which values of p are possible, and which are not: We can define a static binary lookup table $\mathcal{B}(n,k,p)$ that denotes when p is a possible area of dividing n features into k groups. This table can be calculated completely offline and is independent of the choice of feature or distance metric. We initialize:

$$\mathcal{B}(n,1,p) = true \quad \text{iff} \quad n = p^2. \tag{14}$$

The rest of the table can be filled as k increases with the recursive formula (utilizing the logical OR operation):

$$\mathcal{B}(n,k,p) = \bigvee_i \mathcal{B}\left(n-i, k-1, p-i^2\right). \tag{15}$$

This essentially attempts to divide n features into one group of i features and the rest of the features into $k-1$ groups, for any possible i.

5 FEATURE EXTRACTION USING DEEP NEURAL NETWORKS

In recent years, deep Convolutional Neural Networks (CNNs) have enjoyed great success in large-scale image and video recognition, yielding substantial gains in various benchmarks. Deep CNN architectures such as AlexNet [13], VGG [26], Inception [29], and ResNet [11] can be trained on large datasets like ImageNet [9] and then used as powerful feature extractors, with the extracted features incorporating useful semantic information. Similarly, it has been shown that CNN architectures can also offer very promising results on audio classification (e.g., [12]).

For enabling effective video scene detection, the features extracted from the different shots must encode high-level, semantic information about the video content. Given that the low-level information in the video can vary greatly even with small camera shifts, it can be highly useful to identify a shot with the high-level concepts, which will make comparison to adjacent shots more meaningful. Therefore, our hypothesis in this work is that deep features (encoding both the visual and audio information in the video) can prove useful also for the task of video scene detection.

More specifically, for encoding the visual information, we use the Inception-v3 architecture [29] (pre-trained on ImageNet) and apply it to frames in the video (middle frame of each shot) to extract semantic information representing each shot. This is done by removing the final classification layer of the network and extracting the next-to-last layer of the CNN to obtain a high-level representation. The resulting 2048-dimensional vector per shot allows distances between shots to be measured in a semantic and more meaningful way.

In addition, we evaluated the performance when incorporating also audio features to obtain a multimodal representation of the video. For this, we use the VGGish model [12] which represents each 0.96 seconds of audio with a 128-dimensional deep feature vector. When using both visual and audio together, we create separate D matrices for each modality, which are then normalized and averaged together. We note that this merging is not equivalent to averaging the normalized feature vectors together, since the D matrices already incorporate the similarities and dissimilarities between vectors. The merging essentially averages the relative similarity between each pair of shots, allowing both modalities to contribute to the preference of partitioning.

6 HIERARCHY CREATION

In this section we detail our novel non-greedy hierarchy creation method, an optional component when a hierarchical division tree is desired.

In the field of video browsing it can often be beneficial to compute more than a single partitioning for a video, where a hierarchy of the divisions across multiple granularity levels (from coarse to fine temporal partitioning) allows the consumer to better understand the high-level structure of the content in the video. For this, it is particularly important that the multitude of divisions over various granularity levels adhere to a hierarchical structure.

In general, there are two main approaches for hierarchy creation: Divisive methods (e.g., [14]) begin with the coarsest level (no division) and sequentially choose the best location for a division at each finer granularity level. Agglomerative methods (e.g., [16]) begin at the finest granularity level (a partition at each possible location) and choose the best location to remove a partition at each coarser granularity level. The problem with these methods is that they are inherently greedy and typically lead to suboptimal solutions, since at each stage the best location for division or grouping is chosen and cannot be retracted.

To overcome this limitation, we propose a non-greedy hierarchy creation procedure: Using multiple sets of optimal divisions at different granularity levels, we measure the consensus for division at particular indexes and determine the hierarchy of divisions according to this consensus score. If in a classical hierarchy creation method an index of division/grouping indicates that it was the best choice at that granularity level, in our method it indicates that there was a strong correlation over the set of optimal divisions to divide/group at that particular index. This method leads to a much more consistent and reliable hierarchy, where points of division are more accurate due to the larger consensus between multiple sets of optimal divisions.

6.1 Notation

We now define the notation used for our hierarchy tree creation method (see Figure 7 for a visualization of the process). We assume that we have consolidated a collection of optimal divisions $\{\overline{t}^{(K)}\}_{K \in \mathcal{K}}$, with K (as defined in Section 3.1) running over a set of different values $\mathcal{K}$. We wish to determine the hierarchy boundary locations $\overline{T}$, with T_i indicating the i-th chosen boundary. This boundary, per the definition of a hierarchy, exists in the hierarchical tree beginning from level i and onward. Thus, the resulting division in the i-th level of the hierarchy includes the boundaries: $(T_1, \ldots, T_i)$.

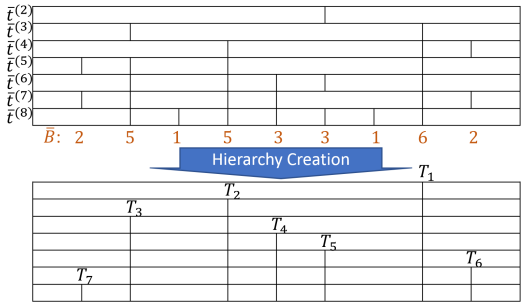
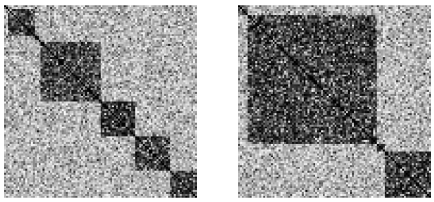


Figure 7: Visualization of our hierarchy creation method. The optimal divisions $\{\bar{t}^{(K)}\}_{K\in\mathcal{K}}$ are used to calculate the hierarchical division tree $\overline{T}$.



(a) **(b)**

Figure 8: Two examples of synthetic D matrices. $\mathcal{H}_{nrm}$ returned an F-score of 0.92 and 1 for D in 8a and 8b respectively, while $\mathcal{H}_{add}$ returned 0.77 and 0.4. Specifically, the low performance of $\mathcal{H}_{add}$ for 8b is a result of the inherent undesired mathematical bias which tends toward same-sized blocks (as described in Section 3.3).

To avoid confusion, we note that unlike $\bar{t}$ above, $\overline{T}$ is not monotonically increasing, and that $\overline{T}$'s last value is not necessarily N, rather $0 < T_i < N$ indicates a boundary *after* element i.

6.2 Non-Greedy Hierarchical Division Tree Creation

We define the indicator function $\mathbf{1}_A = 1$ if A is true, and 0 otherwise. This enables counting the number of instances that a boundary was chosen over the entire set of optimal divisions:

$$B_l = \sum_{K\in\mathcal{K}} \sum_{i=1}^{K} \mathbf{1}_{t_i^{(K)}=l}. \tag{16}$$

The values $0 \le B_l \le |\mathcal{K}|$ indicate how many of the optimal divisions in the collection chose to place a boundary at index $l \in [1,\ldots,N-1]$. Dividing by $|\mathcal{K}|$ (this is not mathematically necessary and is not performed in practice) would indicate the probability that a division should be placed at a particular location. With $\hat{B}$ containing the elements in B sorted in descending order, we can determine the indexes that have the most consensus for placing a division. The hierarchy division locations $\overline{T}$ are therefore calculated by assigning partition locations to higher consensus values:

$$T_i = l, \text{ such that } b_l = \hat{b_i}. \tag{17}$$

This results in a hierarchical division tree $\overline{T}$ where the existence of a particular division is determined by the consensus across the set of optimal divisions.

7 EVALUATION

7.1 Cost Function Evaluation

As explained in Section 4.1, $\mathcal{H}_{nrm}$ is computationally more difficult to minimize compared to $\mathcal{H}_{add}$. However, the latter adds an undesired bias for the point of division. We perform an evaluation between these two cost functions to assess the benefit of the normalized cost function in practice. By creating synthetic simulations of the fundamental optimization problem, we can run a large-scale evaluation using randomized permutations of a typical matrix D, and test the quality of division.

We randomly generated instances of the matrix D with a block-diagonal structure by creating a random matrix D_s with $N = 100$ and uniformly distributed values in the range $[0, 1]$. We chose a random K between 1 and 10, and generated a random permutation of K numbers that sum to N to denote scene blocks. We squared the values inside the blocks on the diagonal, and square-rooted the values outside to keep the values within the $[0, 1]$ range. Finally, we zeroed the values on the diagonal itself $D(i, i) = 0$. Figure 3 shows an example of a synthetic matrix D, and a matrix D created from real video data.

For evaluation, we used the widely accepted Coverage C and Overflow O metrics [31], with a single F-score as an overall quality measure for scene detection computed as the harmonic mean of C and $1 - O$. Performing 1000 random iterations of the process and assuming a known K, we performed the optimal sequential grouping with the normalized cost function $\mathcal{H}_{nrm}$ and the additive cost function $\mathcal{H}_{add}$.

The results show an average F-score of 0.9 for $\mathcal{H}_{nrm}$ and 0.73 for $\mathcal{H}_{add}$. This shows an obvious advantage for the normalized cost function over the additive cost function. Figure 8 shows two examples of the experiments which stress in particular that the normalized cost function is particularly favorable for varying block sizes. For video scene detection, one real-world example where this is important is when attempting to correctly partition between a video and embedded advertisements, where typically the advertisements are substantially shorter than the video content.

Runtimes averaged at 0.92 seconds for $\mathcal{H}_{nrm}$ with C++ code on a 2.8 GHz Intel ®Core™i7 processor, while $\mathcal{H}_{add}$ resulted in an approximate 5x acceleration. With these numbers, it is possible to decide on the tradeoff between quality and runtime for a specific application. For offline video scene detection, a runtime of under a second for a video with $N = 100$ shots (which in practice relates to a video roughly 10 minutes long) is extremely fast even for large corpuses of videos, which one might find in industry.

7.2 Scene Detection Evaluation

The availability of common datasets to evaluate video scene detection algorithms is limited [8]. Many methods rely on performing evaluation on feature films that are not publicly available. Even some published video scene detection datasets feature videos with no clear licenses or permissions; this can present legal difficulties especially for researchers outside academia.

Consequently, we use the Open Video Scene Detection dataset (OVSD) [22], which comprises 21 full-length freely available videos

Figure 9: The first six scenes of the video ToS from the OVSD dataset as partitioned by our method. In this visualization each cluster consists of a single frame from each shot in the scene. (Best viewed in color.)[3]

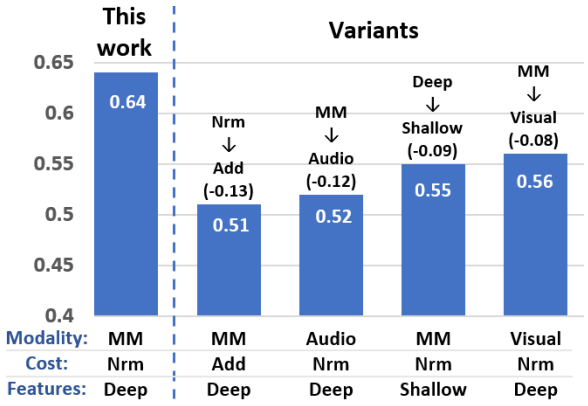


Figure 10: An evaluation of the contribution of each of the elements in our improved method. When using the additive ('Add') cost instead of the normalized ('Nrm') cost function we see the largest drop in F-score value, and so forth replacing the multimodal approach with only audio features, replacing deep features with shallow features, and replacing the multimodal approach with only visual features.

from a variety of genres. We performed shot boundary detection and extracted deep features (see Section 5), and as a distance metric we used L_2-norm. We distinguish between a method that uses only visual results versus methods that incorporate multiple modalities.

Table 1 presents our results over the OVSD dataset against two state-of-the-art methods, and Figure 9 shows some visual results. We compare our normalized cost method using deep visual features to the visual-features-based method in [2] and to our previous method [20], and we compare our multimodal method incorporating the additional use of deep audio features (see Section 5) to the multimodal method in [25]. The results show a significant performance boost for our methods compared to the state of the art.

Additionally, we evaluate the relative gain obtained from various potential design choices of our method. In Figure 10 changing a single element of the algorithm (normalized cost function, multimodal features, and deep features) is tested. We see the largest drop in F-score performance (-0.13) when replacing the normalized cost with the additive cost. This demonstrates again the superiority of the normalized cost function. Next is using only deep audio features (-0.12), using shallow features (RGB histograms and Mel Frequency Cepstrum Coefficients) instead of deep features (-0.09), and using only deep visual features (-0.08). This gives an indication of the relative contribution of each element to the final performance of the complete method.

[3]Images taken from Tears of Steel ©copyright Blender Foundation | mango.blender.org

Table 1: Results on OVSD dataset. Best F-score in bold.

Video Name	Multimodal		Visual		
	Ours MM	[25]	Ours V	[20]	[2]
1000	**0.60**	0.50	**0.64**	0.57	0.39
BBB	**0.63**	0.49	0.38	**0.69**	0.46
BWNS	**0.70**	0.61	**0.55**	0.20	0.43
CH7	**0.60**	0.52	**0.58**	0.49	0.26
CL	**0.51**	0.45	0.37	**0.53**	0.07
ED	**0.61**	0.56	0.33	**0.69**	0.55
FBW	0.59	**0.61**	**0.67**	0.14	0.52
Honey	**0.63**	0.38	**0.64**	0.38	0.36
JW	**0.63**	0.28	**0.69**	0.64	0.22
LCDP	**0.72**	0.18	**0.49**	0.42	0.22
LM	0.64	**0.71**	**0.57**	0.25	0.28
Meridian	**0.79**	0.63	0.66	0.71	**0.82**
Oceania	**0.61**	0.51	**0.61**	0.45	0.26
Pentagon	**0.65**	0.48	**0.63**	0.18	0.16
Route 66	**0.60**	0.36	**0.50**	0.31	0.05
SDM	0.70	**0.81**	**0.67**	0.18	0.55
Sintel	0.58	**0.59**	0.36	**0.66**	0.51
SStB	**0.62**	0.43	**0.57**	0.48	0.22
SW	**0.61**	0.40	**0.61**	0.36	0.13
ToS	0.73	**0.75**	0.46	**0.62**	0.23
Valkaama	0.70	**0.73**	0.70	**0.73**	0.16
Average	**0.64**	0.52	**0.56**	0.46	0.33

8 CONCLUSION

In this paper, we presented a unique formulation of the task of video scene detection as a generic optimization problem with a novel normalized cost function to optimally group consecutive shots into scenes. The proposed normalized cost function enables robust scene detection, and is efficiently optimized using a novel dynamic programming scheme despite an inherent dependency between subproblems. In addition, we incorporated the use of deep neural network models for visual and audio analysis to obtain a semantic representation of the elements in the video scene, enabling effective and more accurate video scene detection. We offered two ways to determine the granularity of division: an adaptive estimation of the number of scenes, and a novel non-greedy procedure for creating a hierarchical consensus-based division tree spanning multiple levels of granularity. Our comprehensive experimental analysis showed the clear advantages of the normalized cost function, and demonstrated that the proposed method outperforms the current state of the art in video scene detection.

REFERENCES

[1] Evlampios Apostolidis and Vasileios Mezaris. 2014. Fast shot segmentation combining global and local visual descriptors. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 6583–6587.

[2] Lorenzo Baraldi, Costantino Grana, and Rita Cucchiara. 2015. Analysis and Re-Use of Videos in Educational Digital Libraries with Automatic Scene Detection. In *11th Italian Research Conference on Digital Libraries*. Springer, 155–164.

[3] Lorenzo Baraldi, Costantino Grana, and Rita Cucchiara. 2015. A deep siamese network for scene detection in broadcast videos. In *Proceedings of the 23rd ACM international conference on Multimedia*. ACM, 1199–1202.

[4] Lorenzo Baraldi, Costantino Grana, and Rita Cucchiara. 2015. Shot and scene detection via hierarchical clustering for re-using broadcast video. In *International Conference on Computer Analysis of Images and Patterns*. Springer, 801–811.

[5] Lorenzo Baraldi, Costantino Grana, and Rita Cucchiara. 2017. Recognizing and presenting the storytelling video structure with deep multimodal networks. *IEEE Transactions on Multimedia* 19, 5 (2017), 955–968.

[6] Vasileios T Chasanis, Aristidis C Likas, and Nikolaos P Galatsanos. 2009. Scene detection in videos using shot clustering and sequence alignment. *IEEE transactions on multimedia* 11, 1 (2009), 89–100.

[7] Lei Chen and M Tamer Ozsu. 2002. Rule-based scene extraction from video. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, Vol. 2. IEEE, II–II.

[8] Manfred Del Fabro and Laszlo Böszörmenyi. 2013. State-of-the-art and future challenges in video scene detection: a survey. *Multimedia systems* 19, 5 (2013), 427–454.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 248–255.

[10] Mehdi Ellouze, Nozha Boujemaa, and Adel M Alimi. 2010. Scene pathfinder: unsupervised clustering techniques for movie scenes extraction. *Multimedia Tools and Applications* 47, 2 (2010), 325–346.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[12] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. 2017. CNN architectures for large-scale audio classification. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 131–135.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[14] Kaiyang Liao, Fan Zhao, Yuanlin Zheng, Congjun Cao, and Mingzhu Zhang. 2017. Parallel N-Path Quantification Hierarchical K-means Clustering Algorithm for Video Retrieval. *International Journal of Pattern Recognition and Artificial Intelligence* (2017), 1750029.

[15] Tong Lin, HongJiang Zhang, and Qing-Yun Shi. 2001. Video Scene Extraction by Force Competition.. In *IEEE International Conference on Multimedia and Expo, 2001. ICME 2001.* 753–756.

[16] Jakub Lokoč, Klaus Schoeffmann, and Manfred del Fabro. 2015. Dynamic hierarchical visualization of keyframes in endoscopic video. In *International Conference on Multimedia Modeling*. Springer, 291–294.

[17] Mary Meeker. 2015. Internet trends 2015-code conference. *Glokalde* 1, 3 (2015).

[18] Rameswar Panda, Sanjay K Kuanar, and Ananda S Chowdhury. 2017. Nyström Approximated Temporally Constrained Multisimilarity Spectral Clustering Approach for Movie Scene Detection. *IEEE Transactions on Cybernetics* (2017).

[19] Zeeshan Rasheed and Mubarak Shah. 2005. Detection and representation of scenes in videos. *IEEE transactions on Multimedia* 7, 6 (2005), 1097–1105.

[20] Daniel Rotman, Dror Porat, and Gal Ashour. 2016. Robust and Efficient Video Scene Detection Using Optimal Sequential Grouping. In *Multimedia (ISM), 2016 IEEE International Symposium on*. IEEE, 275–280.

[21] Daniel Rotman, Dror Porat, and Gal Ashour. 2017. Optimal Sequential Grouping for Robust Video Scene Detection Using Multiple Modalities. *International Journal of Semantic Computing* 11, 02 (2017), 193–208.

[22] Daniel Rotman, Dror Porat, and Gal Ashour. 2017. Robust video scene detection using multimodal fusion of optimally grouped features. In *Multimedia Signal Processing (MMSP), 2017 IEEE 19th International Workshop on*. IEEE, 1–6.

[23] Yong Rui, Thomas S Huang, and Sharad Mehrotra. 1999. Constructing table-of-content for videos. *Multimedia systems* 7, 5 (1999), 359–368.

[24] Ufuk Sakarya and Ziya Telatar. 2008. Video scene detection using dominant sets. In *2008 15th IEEE International Conference on Image Processing*. IEEE, 73–76.

[25] Panagiotis Sidiropoulos, Vasileios Mezaris, Ioannis Kompatsiaris, Hugo Meinedo, Miguel Bugalho, and Isabel Trancoso. 2011. Temporal video segmentation to scenes using high-level audiovisual features. *IEEE Transactions on Circuits and Systems for Video Technology* 21, 8 (2011), 1163–1177.

[26] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[27] Alan F Smeaton, Paul Over, and Aiden R Doherty. 2010. Video shot boundary detection: Seven years of TRECVid activity. *Computer Vision and Image Understanding* 114, 4 (2010), 411–418.

[28] Jeong-Woo Son, Wonjoo Park, Minho Han, and Sun-Joong Kim. 2017. Scene boundary detection with graph embedding. In *Advanced Communication Technology (ICACT), 2017 19th International Conference on*. IEEE, 451–453.

[29] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2818–2826.

[30] Robert Tibshirani, Guenther Walther, and Trevor Hastie. 2001. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63, 2 (2001), 411–423.

[31] Jeroen Vendrig and Marcel Worring. 2002. Systematic evaluation of logical story unit segmentation. *IEEE Transactions on Multimedia* 4, 4 (2002), 492–499.

[32] Yun Zhai and Mubarak Shah. 2006. Video scene segmentation using Markov chain Monte Carlo. *IEEE Transactions on Multimedia* 8, 4 (2006), 686–697.