Computers & Operations Research 136 (2021) 105495

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

A new fast and accurate heuristic for the Automatic Scene Detection Problem

Daniele Catanzaro [a,*,1], Raffaele Pesenti [b,1], Roberto Ronco [c,1]

[a] Center for Operations Research and Econometrics (CORE), Université Catholique de Louvain, Louvain-la-Neuve, Belgium
[b] Department of Management, University Ca' Foscari, Venice, Italy
[c] Department of Computer Science, Bioengineering, Robotics, and Systems Engineering, University of Genoa, Genoa, Italy

ARTICLE INFO

Keywords:
Combinatorial optimization
Video processing
Segmentation
Scene detection
Heuristics
Dynamic programming

ABSTRACT

The *Automatic Scene Detection Problem* (ASDP) is a combinatorial optimization problem that arises in the context of video processing and that has a central role in the management, storing and content retrieval of videos. The problem consists of partitioning the shots of a given video into scenes by optimizing a measure related to the similarity between the given shots. In this article, we build up upon the results from the literature on the ASDP in order to design a new approximate solution algorithm able to outperform the current state-of-the-art both in terms of speed and quality of the solution.

1. Introduction

Given a positive integer $N \geq 1$, consider a video encoded as an ordered set $\mathcal{S} = \{1, 2, \dots, N\}$ of N groups of sequential frames, hereafter referred to as *shots* (see Fig. 1). Given a pair of shots $i, j \in \mathcal{S}$, $i \leq j$, let $\sigma_{i,j} = \{i, i+1, \dots, j-1, j\}$ denote a *scene*, i.e., a subset of $j - i + 1$ consecutive shots in $\mathcal{S}$ that starts at shot i and ends at shot j. For example, the scene $\sigma_{3,5}$ in Fig. 1 refers to the subset of three shots $\{3, 4, 5\}$. Whenever we need to refer to a generic scene in $\mathcal{S}$, we shall drop the indices and just write $\sigma \in \mathcal{S}$. Let $\mathbf{D}$ denote a $N \times N$ symmetric distance matrix, whose generic entry $d_{i,j} \in \mathbb{R}_{0+}$ encodes a measure of similarity between the pair of shots $i, j \in \mathcal{S}$. Let $\alpha(\cdot)$ denote a set function that associates a scene $\sigma \in \mathcal{S}$ with a nonnegative real, according to the following formula

$$\alpha(\sigma) := \sum_{p,q \in \sigma} d_{p,q}.$$

Fixed a positive integer $K \in \{1, \dots, N\}$, let $P = \{\sigma\}$ denote a *partition of $\mathcal{S}$ into K scenes*, i.e., a collection of scenes such that

$$|P| = K, \quad \sigma \cap \sigma' = \emptyset \quad \forall \sigma, \sigma' \in P, \quad \text{and} \quad \bigcup_{\sigma \in P} \sigma = \mathcal{S}.$$

Moreover, let $\mathcal{P}$ denote the set of the possible partitions of $\mathcal{S}$ into K scenes. Then, the *Automatic Scene Detection Problem* (ASDP) consists of finding a partition $P \in \mathcal{P}$ that minimizes the following cost function related to the quality of a partition P:

$$z(P) = \frac{\sum_{\sigma \in P} \alpha(\sigma)}{\sum_{\sigma \in P} |\sigma|^2}. \tag{1}$$

The numerator of $z(P)$ accounts for the similarity of the shots falling within each scene $\sigma \in P$; the denominator accounts for the number of shots in each scene σ. The shorter the scene and the higher the similarity of the shots that fall within it, the lower the value of $z(P)$. This cost function was proposed by Rotman et al. (2018), who reported several empirical considerations on the efficacy of $z(P)$ as opposed to other cost functions described in Rotman et al. (2016, 2017). Incidentally, we observe that the denominator presented in Rotman et al.'s article was $\sum_{\sigma \in P} |\sigma|^2 - N$ instead of $\sum_{\sigma \in P} |\sigma|^2$, as the authors intended to neglect the entries $d_{q,q}$, $q \in \{1, \dots, N\}$, which are zero by definition. As the authors observed, however, the subtraction by N in the denominator does not alter the meaning of the cost function $z(P)$ and can be omitted. Hence, in the rest of this article we will assume to work just with the cost function (1).

The value of K is part of the input of the problem and is usually fixed by means of heuristics, as described in Rotman et al. (2018). In the case $K = 1$ or $K = N$, solving the ASDP is trivial. In fact, in the former case, P is constituted by 1 scene, while in the latter case, P contains exactly K scenes, each consisting of a single shot. The problem instead becomes nontrivial for generic values of K and deciding its complexity in such a case still remains an open problem. An example of an ASDP instance is shown in Fig. 2.

The ASDP was introduced in the literature on video processing by Rotman et al. (2018) as a way to model the automatic segmentation of the shots from a given video into time-contiguous and semantically coherent scenes. This task is compelling for the management and storing of video contents. In particular, the massive quantity of videos

* Corresponding author.
E-mail addresses: daniele.catanzaro@uclouvain.be (D. Catanzaro), pesenti@unive.it (R. Pesenti), roberto.ronco@edu.unige.it (R. Ronco).
[1] The authors equally contributed to conceive the work and to write the article.

https://doi.org/10.1016/j.cor.2021.105495
Received 19 April 2021; Received in revised form 24 June 2021; Accepted 26 July 2021
Available online 31 July 2021

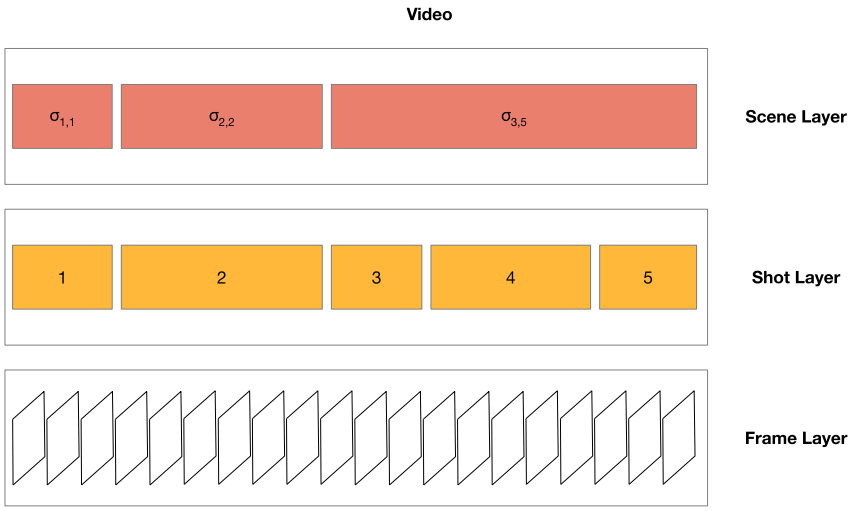

Fig. 1. An example of a video encoded as a sequence of frames, along with two possible logical partitions of the frames into shots (the shot layer) and of the shots into scenes (the scene layer), respectively. The ASDP involves the two topmost layers and consists of partitioning the shots into scenes, by optimizing a measure related to the similarity between the given shots.

$$\begin{bmatrix} 0 & 5.42 & 6.58 & 9.80 & 1.96 & 3.45 & 9.68 & 6.44 & 12.65 \\ 5.42 & 0 & 4.26 & 6.59 & 5.58 & 7.83 & 7.22 & 8.28 & 10.74 \\ 6.58 & 4.26 & 0 & 10.61 & 9.31 & 6.90 & 3.24 & 6.46 & 5.58 \\ 9.80 & 6.59 & 10.61 & 0 & 5.71 & 12.56 & 7.37 & 6.27 & 5.87 \\ 1.96 & 5.58 & 9.31 & 5.71 & 0 & 11.37 & 8.30 & 10.50 & 3.38 \\ 3.45 & 7.83 & 6.90 & 12.56 & 11.37 & 0 & 3.03 & 8.93 & 7.11 \\ 9.68 & 7.22 & 3.24 & 7.37 & 8.30 & 3.03 & 0 & 1.95 & 6.52 \\ 6.44 & 8.28 & 6.46 & 6.27 & 10.50 & 8.93 & 1.95 & 0 & 7.77 \\ 12.65 & 10.74 & 5.58 & 5.87 & 3.38 & 7.11 & 6.52 & 7.77 & 0 \end{bmatrix}$$

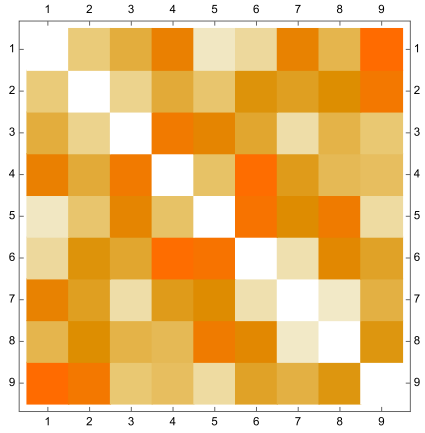

Fig. 2. An example of a possible distance matrix **D** (on the left) associated with an instance of the ASDP characterized by $N = 9$ shots and $K = 3$ scenes. The figure on the right shows the Mathematica Matrix Form of **D**: the lighter the colors the lower the entries of **D** they refer to. Conversely, the darker the colors the bigger the entries of **D** they refer to.

that are produced each day, e.g., by means of computers and smart devices, leads the broadcast companies that store them to automate the tedious and time-consuming manual operations that are involved in content management. As an example, one of the main tasks in the management of documentary, news and educational videos, consists of automatically generate metadata for each scene in order to enable an easy browsing of the videos as well as the re-use of (possibly part of) them (Zhai et al., 2005; Kurihara et al., 2019). In the context of a sport event, it may be often necessary to highlight the precise points of a video in which a specific athlete shows up, so as to enable faster video browsing (Ariki et al., 2003; Del Fabro and Böszörmenyi, 2010; Choroś, 2009). In the context of advertising insertions, spots are usually placed in a video in such a way to be as less intrusive as possible; it is therefore necessary to automatically identify which specific points of the considered video may minimize intrusiveness (Liang et al., 2018). In all of these contexts, scene detection algorithms prove of fundamental assistance: by segmenting a video into semantic units, these algorithms enable the extraction of metadata that can be subsequently used to manipulate and classify it.

The literature on scene detection provides several examples of algorithms that enable the segmenting of the shots from a given video into scenes. We may classify these algorithms into four main classes, based on the methodology used to carry out this task (Del Fabro and Böszörmenyi, 2013). Specifically, we may distinguish between the rule-based algorithms (Feng et al., 2008), the stochastic-based algorithms (Zhai and Shah, 2006; Han and Wu, 2011), the graph-based algorithms (Rasheed and Shah, 2005; Sakarya and Telatar, 2008), and finally the clustering algorithms (Baraldi et al., 2015c; Panda et al., 2017; Rotman et al., 2018). Discussing the foundational ideas at the core of each class is out of the scope of the present article. The interested reader, however, may refer to the article by Del Fabro and Böszörmenyi (2013) for an introduction to the topic. Here, we focus on the clustering algorithms. The idea at their core is to group frames into meaningful clusters based on a measure of the similarity among shots. Among the major contributions to this class, Baraldi et al.'s works (Baraldi et al., 2015c,a,b, 2016) stand out as one the most important examples that aim to pave the road toward the development of a general framework for scene detection. In particular, besides presenting a fast shot detection algorithm, as well as a scene detection algorithm based on hierarchical clustering (Baraldi et al., 2015c), Baraldi et al. further proposed a clustering algorithm to perform scene detection, grouping adjacent shots based on their color spectra (Baraldi et al., 2015a). The authors proposed the use of color information to generate a similarity matrix able to quantify the visual and temporal proximity between shots and then applied spectral clustering to this matrix in order to obtain the desired automatic scene detection. Baraldi et al. (2015b) further improved the accuracy of the scene detection, by enriching the color information of the shots with further features from the middle

frames extracted by means of a neural network. Rotman et al. (2018) significantly extended Baraldi et al.'s works, by proposing a framework for video post-processing consisting of four main stages: Baraldi et al.'s shot detection, middle frame selection and features extraction (performed by means of a neural network), and finally the automatic scene detection tout court, formulated in terms of the ASDP and in which the input distance matrix D was obtained by processing the outputs of the three previous steps.

Rotman et al. proposed a dynamic programming algorithm to solve the ASDP, and subsequently improved their overall framework by modifying the processing of the previous first three steps so as to generate input distance matrices having combinatorial properties able to vastly speed up the running time of their algorithm, at the cost of slightly losing in terms of accuracy (Rotman et al., 2020). Rotman et al. (2018) algorithm, and more in general their overall framework (Rotman et al., 2020), currently constitutes the current state-of-the-art for the ASDP.

In this article we build upon Rotman et al. (2018, 2020) seminal works to design a new improved solution algorithm for the ASDP able to outperform the current state-of-the-art both in terms of speed and quality of the provided solution. In particular, starting from a recall of the foundations of Rotman et al. (2018) algorithm, in Section 3 we exploit the combinatorics of the ASDP to design a new approximate algorithm characterized by a computational cost not higher than Rotman et al. (2018)'s algorithm. In Section 4, we report on the results obtained by running the new heuristic on an extensive battery of practical instances of the ASDP taken from the literature and in Section 5 we give a perspective on possible future developments.

2. On Rotman et al.'s algorithm

In this section, we recall some fundamental aspects of Rotman et al.'s algorithm. Before proceeding, we introduce some notation and definitions that will prove useful throughout the article. We start by observing that an instance $\mathcal{I}$ of the ASDP is characterized by the triplet $(S, \mathbf{D}, K)$ and that the total ordering of S allows to look at a scene $\sigma_{i,j}$ both as a subset of shots and as the discrete interval $\{i, i+1, \dots, i+j\}$, hereafter denoted as $[i, j]$. This duality proves useful not only to recall Rotman et al.'s algorithm but also to formalize the new heuristic discussed in the next sections.

Given an interval $[i, j] \subseteq [1, N] = S$ and a positive integer k such that $1 \le k \le \min\{j - i + 1, K\}$, let $P^k_{i,j}$ denote a partition of the interval $[i, j]$ into k scenes. Then the idea at the core of Rotman et al.'s algorithm consists of exploiting a bottom-up dynamic programming solution approach that progressively combines smaller instances (subproblems) of an input instance $\mathcal{I}$ of the ASDP until obtaining a solution to $\mathcal{I}$. The dynamic programming approach is enabled by the following key observation: if the shots in the interval $[i, N]$ are partitioned into k scenes, for some appropriate values of i and k, then the shots in the interval $[1, i-1]$ will have to be necessarily partitioned into $K-k$ scenes in order to ensure the feasibility of the solution. Hence, for each fixed shot $i \in S$ and $k \in \kappa_i := [\max\{1, K - i + 1\}, \min\{K, N - i + 1\}]$, Rotman et al.'s algorithm finds a partition $P^k_{i,N}$ that locally minimizes the following surrogate cost function

$$C^k_i(e) := \frac{\sum_{\sigma \in P^k_{i,N}} \alpha(\sigma)}{e + \sum_{\sigma \in P^k_{i,N}} |\sigma|^2},$$

where the positive integer

$$e \in \epsilon^k_i := \begin{cases} [\lceil (i-1)^2 / (K-k) \rceil, (i-K+k)^2 + K - k - 1] & k \in [1, K-1] \\ \{0\} & \text{otherwise,} \end{cases}$$

approximates the sum of the addends at the denominator of (1) related to the partition of the interval $[1, i-1]$. The authors observe that, for each fixed k, a lower bound on e is obtained when the shots in $[1, i-1]$ are grouped as evenly as possible into $K-k$ scenes. The upper bound, instead, is achieved when one of the $K-k$ scenes contains the largest admissible number $i - K + k$ of shots, while the other $K-k-1$ scenes contain just one shot each. We observe here that $|\epsilon^k_i|$ is, in the worst case, of order $O(N^2)$. This fact will turn out to be useful later on, when discussing the computational complexity aspects of Rotman et al.'s algorithm.

For a fixed shot $i \in S = [1, N]$, the generic base case for Rotman et al.'s algorithm can be easily determined by observing that there is one and only one partition of the shots in $[i, N]$ if and only if either $k = 1$ or $k = N - i + 1$. The case $k = N - i + 1$ is trivial because it implies that

$$\sum_{\sigma \in P^k_{i,N}} \alpha(\sigma) = \sum_{r=p}^{q} d_{r,r} = 0,$$

i.e., that $C^k_i(e) = 0$, for any $e \in \epsilon^k_i$. The case $k = 1$, instead, involves finding the partition $P^1_{i,N}$ that minimizes $C^1_i(e)$ over all of the possible values $e \in \epsilon^1_i$. It is easy to see that, because there is just one partition of the shots in $[i, N]$ into one scene, it holds that

$$C^1_i(e) = \frac{\alpha(\sigma_{i,N})}{e + (N - i + 1)^2}, \quad \forall\, e \in \epsilon^1_i.$$

Hence, the minimization of the surrogate cost function $C^1_i(e)$ with respect to e can be carried out in quadratic order at most. Note that there is no need to consider shots $i \in S$ with $i < K$ because it is not possible to partition $[1, i-1]$ into $K-1$ scenes. This fact allows Rotman et al.'s algorithm to skip the first $[1, K-1]$ shots and to focus just on the interval $[K, N]$.

The iterative step of Rotman et al.'s algorithm consists of considering all of the possible values of $k \in \kappa_i \setminus \{1\}$ and finding, for each of them, the partition $P^k_{i,N}$ that minimizes the surrogate cost $C^k_i(e)$, for all $e \in \epsilon^k_i$. To this end, for each shot $j \in [i, N-1]$, Rotman et al.'s algorithm first splits the target interval $[i, N]$ into $[i, j]$ and $[j+1, N]$. Then, it considers the partitions $P^1_{i,j}$ and $P^{k-1}_{j+1,N}$ for the intervals $[i, j]$ and $[j+1, N]$, respectively, and sets $P^k_{i,N} = P^1_{i,j} \cup P^{k-1}_{j+1,N}$. The partition $P^k_{i,N}$ that minimizes the function

$$G^k_{i,j}(e) := \frac{\sum_{\sigma \in P^1_{i,j}} \alpha(\sigma)}{e + \sum_{\sigma \in P^1_{i,j} \cup P^{k-1}_{j+1,N}} |\sigma|^2}$$

over all of the possible values of $e \in \epsilon^k_i$ is then selected as the best one for the interval $[i, N]$. Rotman et al. observe that the values of $C^k_i(e)$ and $G^k_{i,j}(e)$ are related by means of the two other quantities $X^k_i(e)$ and $A^k_i(e)$, corresponding to the last shot of the first scene of the partition associated with $C^k_i(e)$, and to the contribution of $P^k_{i,N}$ to the denominator of $C^k_i(e)$, respectively. In particular, the base case for $C^k_i(e)$, $X^k_i(e)$, and $A^k_i(e)$ is computed as

$$C^1_i(e) = \frac{\alpha(\sigma_{i,N})}{e + (N - i + 1)^2} \tag{2}$$

$$X^1_i(e) = N \tag{3}$$

$$A^1_i(e) = (N - i + 1)^2 \tag{4}$$

for all $i \in [K, N]$ and $e \in \epsilon^k_i$. The iterative step, instead, is characterized by the following relations:

$$C^k_i(e) = \min_{j \in [i, N-k+1]} \left\{ G^k_{i,j}(e) + C^{k-1}_{j+1}(e + (j - i + 1)^2) \right\} \tag{5}$$

$$G^k_{i,j}(e) = \frac{\alpha(\sigma_{i,j})}{e + (j - i + 1)^2 + A^{k-1}_{j+1}(e + (j - i + 1)^2)} \tag{6}$$

$$X^k_i(e) = \operatorname*{argmin}_{j \in [i, N-k+1]} \left\{ G^k_{i,j}(e) + C^{k-1}_{j+1}(e + (j - i + 1)^2) \right\} \tag{7}$$

$$A_i^k(e) = (X_i^k(e) - i + 1)^2 + A_{j+1}^{k-1}(e + (j - i + 1)^2) \tag{8}$$

for $i \in [1, N]$, $k \in \kappa_i \setminus \{1\}$, $p \in \epsilon_i^k$, and $j \in [i, N - k + 1]$. Specifically, for each $j \in [i, N-k+1]$, the partition of the shots $[i, j]$ into 1 scene and the one of the shots $[j+1, N]$ into $k-1$ scenes are combined according to the sum of their associated costs $G_{i,j}^k(e)$ and $C_{j+1}^{k-1}(e+(j-i+1)^2)$, respectively. As both share the same denominator, the numerator resulting from their sum corresponds to the sum of the distances between the shots in the two partitions. In other words, e allows to parameterize the number of shots in the partition of $\sigma_{1,i-1}$ into $K - k$ scenes, which is not considered when solving the problem of partitioning $\sigma_{i,N}$ into k scenes.

Algorithm 1 outlines the pseudo-code of Rotman et al.'s algorithm. With a little abuse of notation, we treat C, X, and A as tensors in the pseudo-code, consistently with Eqs. (2)–(8). In this way, we can save the computed values of $C_i^k(e)$, $X_i^k(e)$, and $A_i^k(e)$, and recall them whenever necessary in the iterations. Instead, we treat $G_{i,j}^k(e)$ as a function, so that computed values are not saved for further use. Lines 1–4 initialize the starting values for C, A and T. Line 1 initializes $C_i^k(e)$ to ∞ for each $(i, k, e) \in F$. Lines 2–4 compute the bases cases according to Eqs. (2)–(4). Lines 5–20 compute the step cases according to Eqs. (5)–(8). Lines 22–25 reconstruct the partition $\{[1, s_1], [s_1 + 1, s_2], \dots [s_{K-1} + 1, s_K]\}$ associated with the cost of the partition $P_{1,N}^K$ denoted as $C_1^K(0)$ and equal to

$$C_1^K(0) = \frac{\sum_{\sigma \in P_{1,N}^K} \alpha(\sigma)}{0 + \sum_{\sigma \in P_{1,N}^K} |\sigma|^2} = z(P_{1,N}^K).$$

In particular, observe that, by definition of $X_i^k(e)$, s_k is the last shot of kth scene at the end of the kth iteration of lines 23–25. Finally, at line 27, the algorithm returns the partition so computed and the relative value of the associated cost function.

Rotman et al. also describe a boolean look-up table $T_{n,e}^k$ that can be implemented in Algorithm 1 so as to exclude non admissible values of e. In particular, for $n \in [1, N]$, $k \in \kappa_i$, and $e \in \epsilon_{n+1}^k$, $T_{n,e}^k$ = true when n shots can be partitioned into k scenes and there exists $P_{i,j}^k$, $i, j \in S, i \le j$, such that $|P_{i,j}^k| = e$, and false otherwise. The look-up table can be initialized before Algorithm 1, and employed after line 12 to skip the iteration if $T_{n,e}^k$ = false. The base cases are computed as

$$T_{n,e}^1 = \begin{cases} \text{true} & \text{if } n^2 = e \\ \text{false} & \text{otherwise} \end{cases}$$

for $n \in S$ and $e \in \epsilon_{n+1}^k$, since n shots can be partitioned into 1 scene if and only if, for any $j - i + 1 = n$, $|P_{i,j}^1| = n^2$ is equal to e. The relation for $T_{n,e}^k$ is

$$T_{n,e}^k = \bigvee_{q=1}^{\lceil \sqrt{e} \rceil - 1} T_{n-q, e-q^2}^{k-1}$$

for $n \in S$, $k \in [2, \min\{K, n\}]$, and $e \in \epsilon_{n+1}^k$. Specifically, note that as $q < n$ shots can be partitioned into scene $\hat{\sigma}$ with $|\hat{\sigma}|^2 = q^2$, for each $q \in [1, \lceil \sqrt{e} \rceil - 1]$, then there is a partition P' of n shots into k scenes such that $\sum_{\sigma \in P'} \sigma = e$, i.e., $T_{n,e}^k$ = true, if there is a partition P'' of $n - q$ shots into $k - 1$ scenes such that $\sum_{\sigma \in P''} \sigma = e - q^2$.

Example 1. As an example of execution of Rotman et al.'s algorithm, consider the instance of the ASDP with $N = 6$, $K = 3$, and

$$\mathbf{D} = \begin{bmatrix} 0 & 2 & 1 & 2 & 1 & 1 \\ 2 & 0 & 2 & 2 & 1 & 0 \\ 1 & 2 & 0 & 0 & 0 & 2 \\ 2 & 2 & 0 & 0 & 0 & 2 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 2 & 2 & 0 & 0 \end{bmatrix}.$$

Algorithm 1: *Rotman et al.'s algorithm*

Input : Matrices **B** and **D**.
Output : A partition $P_{1,N}^K$ and the associate cost $C_1^K(0)$.
Internal Data : $F \leftarrow [1, N] \times [1, K] \times [0, N^2]$, $V, s_k \in \mathbb{Z}_{0^+}$.
Internal Functions : $C : F \to \mathbb{R}_{0^+}$, $X : F \to \mathbb{R}_{0^+}$, and $A : F \to \mathbb{Z}_{0^+}$.

1 Set $C_i^k(e) \leftarrow \infty$, for all $i \in [1, N]$, $k \in [1, K]$, $e \in [0, N^2]$;
2 **foreach** $i \in [K, N]$ **do**
3 **foreach** $e \in \epsilon_i^1$ **do**
4 Compute $C_i^1(e)$, $X_i^1(e)$, and $A_i^1(e)$ with equations Eqs. (2)–(4);
5 **foreach** $i \in S$ **do**
6 **foreach** $k \in \kappa_i$ **do**
7 **if** $k = K$ **then**
8 $e_l \leftarrow e_r \leftarrow 0$;
9 **else**
10 $e_l \leftarrow \lceil (i-1)^2/(K-k) \rceil$;
11 $e_r \leftarrow (i-1) - (K-k) + 1)^2 + K - k - 1$;
12 **foreach** $e \in [e_l, e_r]$ **do**
13 **foreach** $j \in [i, N - k + 1]$ **do**
14 **if** $C_i^k(e) = \infty$ **then**
15 **continue**;
16 $G_{i,j}^k(e) \leftarrow$ $\alpha(\sigma_{i,j}) \,/\, \left(e + (j - i + 1)^2 + A_{j+1}^{k-1}(e + (j - i + 1)^2)\right)$;
17 **if** $G_{i,j}^k(e) + C_{j+1}^{k-1}(e + (j - i + 1)^2) < C_i^k(e)$ **then**
18 $C_i^k(e) \leftarrow G_{i,j}^k(e) + C_{j+1}^{k-1}(e + (j - i + 1)^2)$;
19 $X_i^k(e) \leftarrow j$;
20 $A_i^k(e) \leftarrow (j - i + 1)^2 + A_{j+1}^{k-1}(e + (j - i + 1)^2)$;
21 // Recovering the partition into scenes with Eq. (7);
22 $V \leftarrow 0, s_0 \leftarrow 0$;
23 **foreach** $k \in [1, K]$ **do**
24 $s_k \leftarrow X_{s_{k-1}+1,V}^{K-k+1}$;
25 $V \leftarrow V + (s_k - s_{k-1})^2$;
26 $P_{1,N}^K \leftarrow \{[1, s_1], [s_1 + 1, s_2], \dots [s_{K-1} + 1, s_K]\}$;
27 **return** $P_{1,N}^K, C_1^K(0)$;

As a first step of Rotman et al.'s algorithm, we compute the base cases

$$C_3^1(2) = \frac{8}{18}, \quad C_4^1(5) = \frac{4}{14}, \quad C_5^1(8) = C_5^1(10) = C_6^1(13) = C_6^1(17) = 0$$

The subsequent steps exploit (5)–(8), so we have

$$\begin{aligned} C_3^2(4) &= \min\{G_{3,3}^2(4) + C_4^1(5),\ G_{3,4}^2(4) + C_5^1(8),\ G_{3,5}^2(4) + C_6^1(13)\} \\ &= \min\left\{\frac{4}{14}, \frac{0}{12}, \frac{0}{14}\right\} \\ &= 0. \end{aligned} \tag{9}$$

Fig. 3 shows a visual representation of such iterative step. Observe that the last two terms of the expression (9) are equal to 0. If we choose $G_{3,4}^1(4) + C_5^1(8) = 0$, then $A_3^1(4) = 12$, and $X_3^1(4) = 4$. Moreover, with simple arithmetic steps we also obtain

$$\begin{aligned} C_2^2(1) &= \frac{4}{9}, & A_2^2(1) &= 17, & X_2^2(1) &= 2 \\ C_4^2(9) &= 0, & A_4^2(9) &= 5, & X_4^2(9) &= 5 \\ C_5^2(16) &= 0, & P_5^2(16) &= 2, & I_5^2(16) &= 5. \end{aligned}$$

Therefore,

$$\begin{aligned} C_1^3(0) &= \min\{G_{1,1}^3(0) + C_2^2(1),\ G_{1,2}^3(0) + C_3^2(4),\ G_{1,3}^3(0) \\ &\quad + C_4^2(9),\ G_{1,4}^4(0) + C_5^2(16)\} \\ &= \min\left\{\frac{4}{9}, \frac{1}{3}, \frac{3}{7}, 1\right\} = \frac{1}{3}, \end{aligned}$$

corresponding to the partition $\{\{1, 2\}, \{3, 4\}, \{5, 6\}\}$. Incidentally, it is worth noting that in the considered example, a better partition can be

$$\begin{bmatrix} 0 & 2 & 1 & 2 & 1 & 1 \\ 2 & 0 & 2 & 2 & 1 & 0 \\ 1 & 2 & 0 & 0 & 0 & 2 \\ 2 & 2 & 0 & 0 & 0 & 2 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 2 & 2 & 0 & 0 \end{bmatrix}$$

Fig. 3. An iterative step performed by Rotman et al.'s algorithm to compute $C_3^2(4)$ in Example 1. We mark in orange and yellow the entries of the distance matrix that contribute to the numerator of $G_{3,4}^2(4)$ and $C_5^1(8)$, respectively. Instead, we mark in red the entries corresponding to the distances between the shots in $\sigma_{1,2}$. We observe that, at such iterative step, e is indeed equal to $|\sigma_{1,2}|^2 = 4$.

obtained by considering $\{\{1,2\},\{3,4,5\},\{6\}\}$ which allows to achieve a value $2/7$ of the cost function. This fact further confirms the heuristic nature of Rotman et al.'s algorithm.

As regards the computational complexity of Algorithm 1, we first observe that the values $\alpha(\sigma_{i,j})$, $i,j \in S$, can be precomputed before running Algorithm 1 by means of a simple $O(N^2)$ dynamic programming algorithm described by Rotman et al. (2016), and outlined in the following. Hereinafter, we denote by $\mathbf{D}_{i,j}$, $i \le j$, the sub-matrix $[d_{q,r}]$ of $\mathbf{D}$, where $q,r \in [i,j]$. We observe that $\mathbf{D}_{i,j}$ is a square sub-matrix on the main diagonal of $\mathbf{D}$. The algorithm first initializes $\alpha(\sigma_{i,i}) = 0$, $i \in S$. Then, for $l \in [2,N]$ and $i \in [1, N-l+1]$, the algorithm computes $\alpha(\sigma_{i,i+l-1})$ as

$$\alpha(\sigma_{i,i+l-1}) = \alpha(\sigma_{i,i+l-2}) + \alpha(\sigma_{i+1,i+l-1}) - \alpha(\sigma_{i+1,i+l-2}) + 2d_{i,i+l-1}. \tag{10}$$

The first two terms of (10) are the sums of the entries belonging to the sub-matrices $\mathbf{D}_{i,i+l-2}$ and $\mathbf{D}_{i+1,i+l-1}$. Since these two sub-matrices share the entries in $\mathbf{D}_{i+1,i+l-2}$, then $\alpha(\sigma_{i+1,i+l-2})$ is counted twice in the sum $\alpha(\sigma_{i,i+l-2}) + \alpha(\sigma_{i+1,i+l-1})$, hence $\alpha(\sigma_{i+1,i+l-2})$ has to be subtracted once from (10). Finally, the last term accounts for the fact that $d_{i,i+l-1}$ and its transpose value $d_{i+l-1,i}$ are the only two entries in $\mathbf{D}_{i,i+l-1}$ that are not considered in $\alpha(\sigma_{i,i+l-2}) + \alpha(\sigma_{i+1,i+l-1}) - \alpha(\sigma_{i+1,i+l-2})$. Hence, their values have to be added in (10). Now, even in the case in which the boolean look-up table is used, the computational complexity of Algorithm 1 is equal to

$$O\left(\sum_{i=1}^{N}\sum_{k=M_{1,i}}^{m_{K,i}}\sum_{e=\lceil\frac{i^2}{k}\rceil}^{(i-k+1)^2+k-1}(N-(k-1)-i+1)\right),$$

where $M_{1,i} = \max\{1, K-i+1\}$ and $m_{K,i} = \min\{K, N-i+1\}$. This notation can be further simplified by expanding the innermost sum as follows:

$$O\left(\sum_{i=1}^{N}\sum_{k=M_{1,i}}^{m_{K,i}}(N-k-i)\left((i-k+1)^2+k-\frac{i^2}{k}\right)\right)$$

that leads to

$$\begin{aligned} &O\left(\sum_{i=1}^{N}\sum_{k=M_{1,i}}^{m_{K,i}}(N-k-i)\left(\frac{k-1}{k}i^2+k^2-2ik-2k+2i\right)\right) \\ &\sim O\left(\sum_{i=1}^{N}\sum_{k=M_{1,i}}^{m_{K,i}} N\frac{k-1}{k}i^2+Nk^2-2Nki-2Nk+2Ni \right. \\ &\left. -ki^2\frac{k-1}{k}-k^3-2k^2i-2k^2-2ki-\frac{k-1}{k}i^3-k^2i+2ki^2+2ki-2i^2\right). \end{aligned} \tag{11}$$

Now, observe that $\sum_{q=1}^{N} q^2 = N(N+1)(2N+1)/6$. Then, we can rewrite (11) as

$$O\left(\sum_{i=1}^{N}\sum_{k=M_{1,i}}^{m_{K,i}}(N+k)i^2+(N-3i)k^2-\frac{k-1}{k}i^3-k^3+2Ni(1-k)\right)$$

$$\begin{aligned} &\sim O\left(\sum_{i=1}^{N}\frac{m_{K,i}(m_{K,i}+2N+1)}{2}i^2+(N-3i)\frac{m_{K,i}(m_{K,i}+1)(2m_{K,i}+1)}{6} \right. \\ &\left. -i^3m_{K,i}-\frac{m_{K,i}^2(m_{K,i}+1)^2}{4}\right). \end{aligned} \tag{12}$$

By recalling that

$$\sum_{q=1}^{N} q^3 = \frac{N^2(N+1)^2}{4} \sim O(N^4),$$

$$\sum_{q=1}^{N} q^4 = \frac{N(N+1)(2N+1)(3N^2+3N-1)}{30} \sim O(N^5),$$

and by observing that $m_{K,i} = K$ if $i < N-K+1$ and $m_{K,i} = N-i+1$ otherwise, (12) reduces to

$$\begin{aligned} &O\left(\sum_{i=1}^{N-K}\frac{K(K+2N+1)}{2}i^2+(N-3i)\frac{K(K+1)(2K+1)}{6} \right. \\ &-Ki^3-\frac{K^2(K+1)^2}{4} \\ &+\sum_{i=N-K+1}^{N}\frac{(N-i+1)(3N-i+2)}{2}i^2 \\ &+(N-3i)\frac{(N-i+1)(N-i+2)(2N-2i+3)}{6} \\ &\left.-i^3(N-i+1)-\frac{(N-i+1)^2(N-i+2)^2}{4}\right) \\ &\sim O\left(\sum_{i=1}^{N-K}(K^2i^2+NKi^2+K^3N-K^3i-Ki^3-K^4) \right. \\ &\left.+\sum_{i=N-K+1}^{N}((N^2-i^2)i^2+N^4-N^3i+N^2i^2-Ni^3+i^4)\right). \end{aligned} \tag{13}$$

The second sum in (13) is $O(KN^4)$, which yields

$$\begin{aligned} &O(K^2(N-K)^2+NK(N-K)^2+K^3N \\ &-K^3(N-K)-K(N-K)^3-K^4+KN^4) \sim O(KN^4). \end{aligned}$$

3. A novel heuristic for the ASDP

In this section, we present a novel heuristic for the ASDP that proves able to outperform Rotman et al. (2018)'s algorithm, which currently constitutes the state-of-the-art for the problem. We start by describing the main idea at the core of the heuristic. Subsequently, we will enter into the details of its pseudo-code and analyze its computational complexity. Before proceeding, we introduce some notation and definitions that will prove useful throughout the section.

Given an instance $\mathcal{I} = (S, \mathbf{D}, K)$ of the ASDP, a subset of shots $[i,j] \subseteq S$ and a positive integer k such that $1 \le k \le \min\{K, j-i+1\}$, we denote by $\mathcal{I}_{i,j}^k := ([i,j], \mathbf{D}, K, k)$ a *sub-instance* of $\mathcal{I}$ that involves the partitioning of the shots in $[i,j]$ into k scenes. We observe that if $i=1$, $j=N$, and $k=K$, then $\mathcal{I}_{i,j}^k = \mathcal{I}_{1,N}^K = \mathcal{I}$, i.e., the sub-instance $\mathcal{I}_{i,j}^k$ coincides with the given input instance $\mathcal{I}$ to solve. We denote by $\bar{P}_{i,j}^k$ the partition of the feasible sub-instance $\mathcal{I}_{i,j}^k$ with (locally) minimum cost $z(\bar{P}_{i,j}^k)$, obtained by recursively splitting $\mathcal{I}_{i,j}^k$ into the two feasible sub-instances $\mathcal{I}_{i,v}^h$ and $\mathcal{I}_{v+1,j}^{k-h}$, for $v \in [i, j-1]$ and $h \in [1, k-1]$. Then, a possible approach to solution of the ASDP consists of (i) recursively splitting a sub-instance $\mathcal{I}_{i,j}^k$ into $\mathcal{I}_{i,v}^h$ and $\mathcal{I}_{v+1,j}^{k-h}$, for each $v \in [i, j-1]$ and $h \in [1, k-1]$, (ii) finding the (locally) minimum cost partitions $\bar{P}_{i,v}^h$ and $\bar{P}_{v+1,j}^{k-h}$ for $\mathcal{I}_{i,v}^h$ and $\mathcal{I}_{v+1,j}^{k-h}$, respectively, and finally (iii) choosing a partition $\bar{P}_{i,j}^k$ for $\mathcal{I}_{i,j}^k$ that can be written as $\bar{P}_{i,j}^k = \bar{P}_{i,v}^h \cup \bar{P}_{v+1,j}^{k-h}$ and that (locally) minimizes the cost function $z(\bar{P}_{i,j}^k)$. The term "locally" remarks the fact that this particular recursive splitting behaves as a greedy solution approach to the ASDP, but in general it proves unable to guarantee the optimality of the overall solution computed. This fact is clarified by means of the following example.

Example 2. Consider the instance $\mathcal{I} = ([1, 12], \mathbf{D}, 7)$ of the ASDP, for some distance matrix $\mathbf{D}$. Suppose that the solution to $\mathcal{I}$ can be obtained by splitting $\mathcal{I}$ as $\mathcal{I}^4_{1,6} \cup \mathcal{I}^3_{7,12}$ and by computing the partition $P^7_{1,12}$ as the union of the partitions $\bar{P}^4_{1,6}$ and $\bar{P}^3_{7,12}$, respectively. Finally, suppose that $z(\bar{P}^3_{7,12}) = 11/12$, and that $\mathbf{D}_{1,6} = [d_{q,r}]$, $q, r \in [1, 6]$ be as follows

$$\mathbf{D}_{1,6} = \begin{bmatrix} 0 & 5 & 5 & 0 & 0 & 0 \\ 5 & 0 & 5 & 0 & 0 & 0 \\ 5 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0.5 & 0 \end{bmatrix}.$$

It is easy to see that the optimal partition

$$\bar{P}^4_{1,6} = \{\{1\}, \{2\}, \{3, 4\}, \{5, 6\}\}$$

for $\mathcal{I}^4_{1,6}$ is characterized by a cost $z(\bar{P}^4_{1,6}) = 1/10$. Hence, we have that $z(\bar{P}^7_{1,12}) = (1 + 11)/(10 + 12) = 12/22$. Now, consider the partition

$$\hat{P}^4_{1,6} = \{\{1\}, \{2\}, \{3\}, \{4, 5, 6\}\}$$

and observe that it is locally suboptimal for $\mathcal{I}^4_{1,6}$, as characterized by a cost $z(\hat{P}^4_{1,6}) = 2/12 > z(\bar{P}^4_{1,6})$. However, the union of $\hat{P}^4_{1,6}$ with $\bar{P}^3_{7,12}$ gives rise to the partition $\hat{P}^7_{1,12} = \bar{P}^4_{1,6} \cup \hat{P}^3_{7,12}$ with cost $z(\hat{P}^7_{1,12}) = (2 + 11)/(12 + 12) = 13/24 < z(\bar{P}^7_{1,12})$. Hence, the recursive splitting in which locally optimal partitions are concatenated does not generally guarantee the optimality of the overall solution to the ASDP.

Although the recursive splitting in general does not preserve the global optimality of the overall solution to a given instance, it still proves able to generalize Rotman et al. (2018)'s splitting strategy. Specifically, note that Rotman et al. (2018)'s solution space, consisting of the set of partitions that can be written as $P^k_{i,N} = P^1_{i,j} \cup P^{k-1}_{j+1,N}$, is contained in the solution space constituted by the partitions that can be written as $P^k_{i,N} = P^h_{i,j} \cup P^{k-h}_{j+1,N}$. Hence, provided that both $k > 2$ and $1 < h < k$ hold, the above splitting strategy potentially allows to search for solutions to the ASDP in a larger space. In the following example, we show that the values of h and k need to be appropriately determined to avoid incurring in an infeasible partitioning of the given input interval $\mathcal{S}$.

Example 3. Consider the instance $\mathcal{I} = ([1, 8], \mathbf{D}, 4)$ and the sub-instance $\mathcal{I}^3_{2,7}$. It is easy to see that the union of $\bar{P}^3_{2,7}$ (independently of its combinatorial structure) with any other partitioning of the remaining non-sequential shots $\{1, 8\}$ would force considering more than the required $K = 4$ scenes. In this sense, we say that the sub-instance $\mathcal{I}^3_{2,7}$ is *infeasible*.

In order to characterize the concept of feasibility (or infeasibility) of a sub-instance $\mathcal{I}^k_{i,j}$, we observe that a partition $P^k_{i,j}$ constitutes a *feasible solution* for a sub-instance $\mathcal{I}^k_{i,j}$ of a given instance $\mathcal{I}$ of the ASDP if and only if

$$\begin{aligned} k \in \kappa_{i,j} := [&\max\{K - N + j - i + 1, 1\}, \min\{K - \min\{i - 1, 1\} \\ &- \min\{N - j, 1\}, j - i + 1\}] \\ = [&\max\{K - N + j - i + 1, 1\}, \min\{K - N + j - i + 1, K - 2, \\ &K - i, j - i + 1\}]. \end{aligned} \tag{14}$$

In particular, because the shots in $[i, j]$ must be partitioned in k scenes and the shots in $\mathcal{S} \setminus [i, j]$ must be partitioned in $K - k$ scenes, it holds that (i) k cannot exceed the cardinality of $[i, j]$ and must be greater than or equal to 1 and (ii) $K - k$ cannot exceed the cardinality of $\mathcal{S} \setminus [i, j]$ and must be greater than or equal to 2 in the case $i > 1$ and $j < N$, greater than or equal to 1 if only one of the last two inequality holds, and 0 otherwise. We say that an instance $\mathcal{I}^k_{i,j}$ is *feasible* when the indices i, j, and k satisfy (14).

Proposition 1. *Given a feasible sub-instance $\mathcal{I}^k_{i,j}$, with $k \geq 2$, the number of pairs (v, h) that define two feasible sub-instances $\mathcal{I}^h_{i,v}$ and $\mathcal{I}^{k-h}_{v+1,j}$ is less than or equal to $(j - i - k + 2)(k - 1)$. Moreover, each of such pairs satisfies the following conditions:*

$$v \in [i, j - 1], \tag{15a}$$

$$\begin{aligned} h \in \eta^k_{i,j,v} := [&\max\{k - i + 1 - N + v,\ 1, \\ &k - K + 1 + \min\{N - j, 1\},\ k - j + v\}, \\ &\min\{k - K + N - j + v,\ k - 1, \\ &K - \min\{i - 1,\ 1\} - 1,\ v - i + 1\}]. \end{aligned} \tag{15b}$$

Proof. Since $\mathcal{I}^k_{i,j}$ is feasible, then $k \in \kappa_{i,j}$, and $k \leq j - i + 1$. Hence, there exists a pair (v, h) such that $i \leq v \leq j - 1$ for $h \in [1, k - 1]$. Moreover, each of such pair (v, h) must satisfy the properties $h \leq v - i + 1$ and $k - h \leq j - v$, because both $\mathcal{I}^h_{i,v}$ and $\mathcal{I}^{k-h}_{v+1,j}$ are feasible, i.e., $h \in \kappa_{i,v}$ and $k - h \in \kappa_{v+1,j}$. Thus, $k - j + v \leq h \leq v - i + 1$, and the number of pairs (v, h) is no greater than $(j - i - k + 2)(k - 1)$. Concerning conditions (15), observe that $i \leq v$ and $v + 1 \leq j$ hold for $\mathcal{I}^h_{i,v}$ and $\mathcal{I}^{k-h}_{v+1,j}$ if and only if $i \leq v \leq j - 1$. This proves (15a). By applying (14) to $\mathcal{I}^h_{i,v}$ and $\mathcal{I}^{k-h}_{v+1,j}$, we obtain:

$$\begin{cases} h \geq \max\{K - N + v - i + 1, 1\} \\ h \leq \min\{K - \min\{i - 1, 1\} - 1, v - i + 1\} \\ k - h \geq K - N + j - v \\ k - h \geq 1 \\ k - h \leq K - 1 - \min\{N - j, 1\} \\ k - h \leq j - v. \end{cases}$$

It is easy to see that this set of conditions leads to its compact form (15b). □

We introduce now some notation that will allow to compactly express the cost of partitioning a feasible sub-instance $\mathcal{I}^k_{i,j} = \mathcal{I}^h_{i,v} \cup \mathcal{I}^{k-h}_{v+1,j}$ of $\mathcal{I}$ as $P^k_{i,v} = P^h_{i,v} \cup P^{k-h}_{v+1,j}$, with (v, h) satisfying (15). This notation will prove useful to outline the new heuristic for the ASDP shown in Algorithm 2. Specifically, given a partition $P^k_{i,j}$ for $\mathcal{I}^k_{i,j}$, we denote

$$\varphi(P^k_{i,j}) := \sum_{\sigma \in P^k_{i,j}} \alpha(\sigma),$$

$$\gamma(P^k_{i,j}) := \sum_{\sigma \in P^k_{i,j}} |\sigma|^2,$$

and we rewrite $z(P^k_{i,j})$ as

$$z(P^k_{i,j}) = \frac{\varphi(P^k_{i,j})}{\gamma(P^k_{i,j})}.$$

We also denote by

$$\varphi^k_{i,j} := \varphi(\bar{P}^k_{i,j}), \text{ and } \gamma^k_{i,j} := \gamma(\bar{P}^k_{i,j}),$$

the values of $\varphi(\cdot)$ and $\gamma(\cdot)$ when computed in the (locally) optimal partition $\bar{P}^k_{i,j}$, that is, the one obtained with the recursive splitting described at the beginning of this section. Finally, we denote $\Lambda^k_{i,j}$ as the set of the costs associated with each solution to $\mathcal{I}^k_{i,j}$ obtained as $\bar{P}^h_{i,v} \cup \bar{P}^{k-h}_{v+1,j}$, with (v, h) satisfying (15):

$$\Lambda^k_{i,j} = \begin{cases} \left\{ \Omega^{k,h}_{i,j,v} : (v, h) \text{ satisfies (15)} \right\} & \text{if } k \geq 2 \\ \left\{ \frac{\varphi^1_{i,j}}{\gamma^1_{i,j}} \right\} & \text{if } k = 1 \end{cases} \tag{16}$$

with

$$\Omega^{k,h}_{i,j,v} = \frac{\varphi^h_{i,v} + \varphi^{k-h}_{v+1,j}}{\gamma^h_{i,v} + \gamma^{k-h}_{v+1,j}}. \tag{17}$$

Algorithm 2: *Recursive-Solver*

Input : A feasible sub-instance $([i,j],\mathbf{D},K,k)$, and matrix $\mathbf{B}$.
Output : The values $\varphi^k_{i,j}$, $\gamma^k_{i,j}$, and $\bar{P}^k_{i,j}$.
Internal Data : $\varphi' \in \mathbb{R}_{0^+}$, $\gamma' \in \mathbb{Z}_{0^+}$, and set Q.
Internal Functions : $\varphi^r_{p,q}, \gamma^r_{p,q}, \bar{P}^r_{p,q}$, $\forall\, p,q,r : i \le p \le q \le j, r \in [1,k]$ (global scope).

1 // Returns the solution to processed subproblems
2 **if** $\varphi^k_{i,j} < \infty$ **then**
3 **return** $\varphi^k_{i,j}, \gamma^k_{i,j}, \bar{P}^k_{i,j}$;
4 // Recursion base case
5 **if** $k == 1$ **then**
6 $\varphi^1_{i,j} \leftarrow \alpha(\sigma_{i,j})$;
7 $\gamma^1_{i,j} \leftarrow (j-i+1)^2$;
8 $\bar{P}^1_{i,j} \leftarrow [i,j]$;
9 **else**
10 // Recursion step case
11 $\varphi' \leftarrow \infty$; $\gamma' \leftarrow 1$;
12 **foreach** $v \in [i, j-1]$ **do**
13 **foreach** *h satisfying* (15b) **do**
14 $\varphi^h_{i,v}, \gamma^h_{i,v}, \bar{P}^h_{i,v} \leftarrow$ Recursive-Solver$(([i,v],\mathbf{D},K,h),\mathbf{B})$;
15 $\varphi^{k-h}_{v+1,j}, \gamma^{k-h}_{v+1,j}, \bar{P}^{k-h}_{v+1,j} \leftarrow$ Recursive-Solver$(([v+1,j],\mathbf{D},K,k-h),\mathbf{B})$;
16 **if** $\frac{\varphi^h_{i,v}+\varphi^{k-h}_{v+1,j}}{\gamma^h_{i,v}+\gamma^{k-h}_{v+1,j}} < \frac{\varphi'}{\gamma'}$ **then**
17 $\varphi' \leftarrow \varphi^h_{i,v} + \varphi^{k-h}_{v+1,j}$;
18 $\gamma' \leftarrow \gamma^h_{i,v} + \gamma^{k-h}_{v+1,j}$;
19 $Q \leftarrow \bar{P}^h_{i,v} \cup \bar{P}^{k-h}_{v+1,j}$;
20 $\varphi^k_{i,j} \leftarrow \varphi'$;
21 $\gamma^k_{i,j} \leftarrow \gamma'$;
22 $\bar{P}^k_{i,j} \leftarrow Q$;
23 **return** $\varphi^k_{i,j}, \gamma^k_{i,j}, \bar{P}^k_{i,j}$;

Algorithm 3: *Heuristic-Partitioner*

Input : Instance $\mathcal{I} = ([1,N],\mathbf{D},K)$, matrix $\mathbf{B}$.
Output : $z^K_{1,N}, \bar{P}^K_{1,N}$
Internal Functions : $\varphi^k_{i,j}, \gamma^k_{i,j}, \bar{P}^k_{i,j}$, $i \in [1,N], j \in [i,N], k \in [1,K]$ (global scope).

1 $\varphi^k_{i,j} \leftarrow \infty$ for $i = 1,\dots,N-1$, $j = 2,\dots,N$, $k = 1,\dots,N$; // Initialization
2 $\varphi^K_{1,N}, \gamma^K_{1,N}, \bar{P}^K_{1,N} \leftarrow$ Recursive-Solver$(\mathcal{I},\mathbf{B})$; // Main
3 **return** $\varphi^K_{1,N} / \gamma^K_{1,N}$, $\bar{P}^K_{1,N}$;

Observe that, by definition, $\varphi^k_{i,j}$ and $\gamma^k_{i,j}$ are such that

$$\min \Lambda^k_{i,j} = \frac{\varphi^k_{i,j}}{\gamma^k_{i,j}}. \tag{18}$$

In light of this notation, we can now discuss the new heuristic for the ASDP, called *Recursive-Solver*, whose pseudo-code is provided in Algorithm 2. Recursive-Solver exploits (16)–(18) to compute the partition $\bar{P}^k_{i,j}$ for the instance $\mathcal{I}^k_{i,j}$. The input instance $\mathcal{I}$ is then solved by means of the *Heuristic-Partitioner* whose pseudo-code is provided in Algorithm 3. Heuristic-Partitioner makes use of Algorithm 2 to compute $z^K_{1,N}$ and $\bar{P}^K_{1,N}$. We observe that both Recursive-Solver and Heuristic-Partitioner treat $\varphi^k_{i,j}$, $\varphi^k_{i,j}$, and $\bar{P}^k_{i,j}$ as tensors in order to store the solutions to the sub-instances of $\mathcal{I}$ already processed. In particular, Heuristic-Partitioner first initializes $\varphi^k_{i,j}$, for all $i,j \in \mathcal{S}$, $i \le j$, $k \in [1,K]$, at line 1, then it solves $\mathcal{I} = \mathcal{I}^K_{1,N}$ by calling Recursive-Solver$(\mathcal{I}^K_{1,N},\mathbf{B})$ at line 2, and finally returns the computed values of $z^K_{1,N}$ and $\bar{P}^K_{1,N}$. We observe that $\varphi^k_{i,j}$, $\gamma^k_{i,j}$, and $\bar{P}^k_{i,j}$ have global scope so that their values can be directly accessed. Recursive-Solver first sets $\varphi^k_{i,j} = \infty$ for all $i,j \in \mathcal{S}$, $i \le j$, $k \in [1,K]$. The same operation is done also for $\gamma^k_{i,j}$ and $\bar{P}^k_{i,j}$. This convention is used to indicate that the feasible sub-instance $\mathcal{I}^k_{i,j}$ is yet to be solved, or infeasible. As suggested by its name, Recursive-Solver computes recursively the values $\varphi^k_{i,j}$, $\gamma^k_{i,j}$, and $\bar{P}^k_{i,j}$ in correspondence to a given input feasible sub-instance $\mathcal{I}^k_{i,j}$ of $\mathcal{I}$. In particular, the algorithm traverses the recursion tree backwards from base cases, by considering sub-instances with a progressively larger number of shots and scenes. When called on a sub-instance $\mathcal{I}^k_{i,j}$, $k \ge 2$, Recursive-Solver computes $\Omega^{k,h}_{i,j,v}$ for each pair (v,h) satisfying (15), by recursively calling itself on $\mathcal{I}^h_{i,v}$ and $\mathcal{I}^{k-h}_{i,v}$, so as to determine $\min\{\Lambda^k_{i,j}\}$. When the recursion unfolds, the same sub-instances may arise multiple times while splitting distinct $\mathcal{I}^k_{i,j}$. In this case, Recursive-Solver reuses the solution to already processed sub-instances, thus diminishing the computation load. By entering in the merit of its pseudo-code, we can see that lines 2–3 check whether $\varphi^k_{i,j}$ is assigned a finite value: in the positive case, the sub-instance $\mathcal{I}^k_{i,j}$ has already been solved, and its computed solution can be immediately returned. Lines 4–8 compute the only possible solution of $\mathcal{I}^1_{i,j}$. Lines 9–22 tackle the problem of solving $\mathcal{I}^k_{i,j}$ when $k \ge 2$ by computing all the elements of $\Lambda^k_{i,j}$ and saving the one with the minimal cost. In particular, at each iteration of the doubly nested for cycle, lines 14 and 15 solve each pair of sub-instances $\mathcal{I}^h_{i,v}$ and $\mathcal{I}^{k-h}_{v+1,j}$ with (v,h) satisfying (15). Their solutions are combined into $\Omega^{k,h}_{i,j,v}$ with (17) (see line 16). If $\Omega^{k,h}_{i,j,v}$ improves the estimate of $\varphi^k_{i,j} / \gamma^k_{i,j}$ given by φ' / γ', lines 17 and 18 assign the values $\varphi^h_{i,v} + \varphi^{k-h}_{v+1,j}$ and $\gamma^h_{i,v} + \gamma^{k-h}_{v+1,j}$ to φ' and γ', respectively, so that $\varphi' / \gamma' = (\varphi^h_{i,v} + \varphi^{k-h}_{v+1,j}) / (\gamma^h_{i,v} + \gamma^{k-h}_{v+1,j})$. Moreover, line 19 saves the associated partition $\bar{P}^h_{i,v} \cup P^{k-h}_{v+1,j}$ by assigning it to the local variable Q. At the end of the algorithm, $z^k_{i,j}$ is the value given by Eq. (18), associated with the partition $\bar{P}^k_{i,j}$. At line 23, the algorithm finally returns $\varphi^k_{i,j}$, $\gamma^k_{i,j}$, and the computed partition $\bar{P}^k_{i,j}$. The computational complexity of Algorithm 3 is dominated by line 2. Therefore, to derive the computational complexity of Algorithm 3, it is sufficient to characterize the one of Algorithm 2.

Proposition 2. *The computational complexity of Recursive-Solver*$(\mathcal{I}^K_{1,N})$ *is* $O(\min\{N-K,K\}^2 N^3)$.

Proof. Since each admissible sub-instance is solved once, the computed solutions to already processed sub-instances can be recalled in $O(1)$, and both the base case $k = 1$ and the recombination of subproblems with $k > 1$ also take $O(1)$. Hence, to prove the statement of the proposition, it is sufficient to count the number of feasible sub-instances that may arise when tackling each feasible sub-instance $\mathcal{I}^k_{i,j}$. It is easy to see that this number is

$$O\left(\sum_{i=1}^{N}\sum_{j=i}^{N}\sum_{k\in\kappa_{i,j}} |\Lambda^k_{i,j}|\right). \tag{19}$$

By Proposition 1, (19) is equal to

$$O\left(\sum_{i=1}^{N}\sum_{j=i}^{N}\sum_{k\in\kappa_{i,j}} (j-i-k+2)(k-1)\right)$$

By observing that $|\kappa_{i,j}| \sim O(\min\{N-K,K\})$, and by recalling that $\sum_{q=1}^{N} q = \frac{N(N+1)}{2}$ and $\sum_{q=1}^{N} q^2 = \frac{N(N+1)(2N+1)}{6}$, we get

$$O\left(\max\{\min\{N-K,K\}^2 N^3, \min\{N-K,K\}^3 N^2\}\right). \tag{20}$$

Because $K \le N$, (20) further reduces to

$$O(\min\{N-K,K\}^2 N^3)$$

which concludes the proof. □

4. Computational experiments

In this section, we analyze the performance of the novel heuristic algorithm with respect to Rotman et al.'s algorithm. The experiments reported in this section were motivated by the main goal of evaluating

the performance improvement introduced by Heuristic-Partitioner with respect to Rotman et al.'s algorithm. The extensive set of application instances that we employed for this purpose contains some of the most relevant datasets used in the literature of scene detection. Among them, we included the *Open Video Scene Detection Dataset* (OVSD), that is the reference dataset used by Rotman et al. (2018). In order to generate the input data of the ASDP, i.e., the distance matrices associated with each test video, we reimplemented the relevant stages of Rotman et al.'s scene detection pipeline: shot detection Baraldi et al. (2015c), middle frame selection, and visual features extraction with an Inception-v3 neural network Szegedy et al. (2016) pre-trained on the ImageNet dataset. We restrained the attention to the visual features of the shots, since integrating audio features introduces an additional computational burden which is out of the scope of this work. In fact, our aim in reimplementing Rotman et al.'s pipeline is to provide a common ground for a comparative analysis of the novel heuristic with Rotman et al.'s algorithm on application cases. In Section 4.1, we discuss the details of the algorithms' implementations. In Section 4.2, we describe the datasets included in the set of instances used for the experimental tests. Finally, in Section 4.3, we empirically evaluate the efficacy of the implementation of Heuristic-Partitioner (hereafter denoted as HP for the sake of notation) against two different implementations of Rotman et al.'s algorithm.

4.1. Implementation

We implemented all of the algorithms in Python 3.7 and carried out the experiments on a 64-bit Windows 10 PC equipped with a 3.6 GHz Intel Core i7-3820 CPU and 24 GB of RAM. We implemented Rotman et al.'s algorithm accordingly to the information provided in Rotman et al. (2018). Our first implementation, denoted as RT, makes use of the tensors C, X, A, and T of size $|[1,N]\times[1,K]\times[1,N^2]|$, with $C_i^k(e)$, $X_i^k(e)$, $A_i^k(e)$, and $T_i^k(e)$ as entries for $i\in[1,N]$, $k\in[1,K]$, and $e\in[1,N^2]$. Since not all the triplets (i,k,e) are feasible, as detailed in Section 2, tensors allocate memory space inefficiently. Therefore, in the second implementation of the algorithm, denoted as RH, we used hash-maps to allocate memory space just for feasible entries and save them once computed. While the memory usage of RH is more convenient, solving each feasible sub-instance of the ASDP requires dynamically allocating new entries in the hash-maps. In contrast, RT does not suffer from this issue, since it performs the needed allocations at once, before starting to solve the instance at hand. However, frequent transfer from and to the processor cache, due to possibly far entries in the table, may require additional computational time. We experimentally study the difference in the two implementations in the next subsection. Finally, we observe that the choice of very large values for K may cause a high number of recursion calls, which in turn translates into a nonnegligible computational overhead. A way around this phenomenon (which is out of the scope of the present work) consists of implementing Recursive-Solver in a bottom-up fashion.

It is important to observe that, for $K \simeq N/2$, the asymptotic complexity of HP is equivalent to the one of Rotman et al.'s algorithm, i.e., $O(KN^4)$. This is however the worst case scenario. In fact, in practical applications, the instances of the ASDP are typically characterized by a number of scenes K at least one order of magnitude smaller than N. We also observe that the tensor data structures encoding $\varphi_{i,j}^k$, $\gamma_{i,j}^l$, and $\bar{P}_{i,j}^k$ allow to set the space complexity of Recursive-Solver to $O(KN^2)$. Rotman et al.'s algorithm implements $C_i^k(e)$, $X_i^k(e)$, and $A_i^k(e)$ as tensors as well. However, as $i\sim O(N)$, $k\sim O(K)$, and $e\sim O(N^2)$, the space complexity of Rotman et al.'s algorithm is $O(KN^3)$. We will see in Section 4 that such space complexity de facto poses limitations on the size of the instances that Rotman et al.'s algorithm is able to process. However, since not all the values of e are feasible, not all the entries of the tensors are used for saving computed values. Hence, the space efficiency can be improved by using hash-tables instead of tensors, at the expense of degrading the computational performance, due to the fact that the dynamic memory allocations needed to store new entries. Yet, since the required tensors are three-dimensional, the accessed entries are not necessarily adjacent, and transfers from and to the processor cache might occur frequently. These implementation issues are experimentally addressed in Section 4.3.

In order to enrich the computational assessments, we implemented a further algorithm, hereinafter referred to as *Additive-Heuristic-Partitioner* (AHP), by slightly modifying HP so as to obtain the partition $P\in\mathcal{P}$ that minimizes the additive cost function

$$H(P)=\sum_{\sigma\in P}\alpha(\sigma), \tag{21}$$

proposed by Rotman et al. (2016). In order to perform a proper comparison with HP, RT, and RH, we evaluated the partition P that minimizes $H(P)$ with the cost function $z(P)$. Because Rotman et al. considered the weighting factors in their original formulation of the additive cost function as optional, we neglect them in (21). The pseudo-code of AHP can be derived by Algorithm 2 by disregarding the computation of $\gamma_{i,j,k}$ for each feasible $I_{i,j,k}$, and by comparing the sum of $\phi_{i,v}^h$ and $\phi_{v+1,j}^{k-h}$ with ϕ' in line 16.

The implementation of the algorithms used in this article can be downloaded at the link "https://github.com/ORresearcher/A-New-Fast-and-Accurate-Heuristic-for-the-Automatic-Scene-Detection-Problem".

4.2. Datasets

The datasets used for performance evaluation are the OVSD, the Rai, and the BBC datasets. The OVSD dataset, the one used by Rotman et al. (2018, 2020), was created from 21 Creative Commons licensed videos freely available for download and use. This set of videos contains short and full-length movies including animation, documentary, drama, crime, comedy, and sci-fi. Their ground-truth scene division was obtained from the director script and from the work of several independent human annotators. The Rai dataset contains a collection of ten videos, mainly documentaries and talk shows, taken from the Rai Scuola video archive, notably used by Baraldi et al. (2015a) to evaluate their scene detection algorithm. The BBC dataset, introduced by Baraldi et al. (2015b), is based on the BBC documentary series "Planet Earth" which consists of eleven episodes, each about 50 minutes long.

Due to hardware limitations, we did not consider some of the videos in their entirety, by restricting them to a subsets of their shots. Despite the reduction, our hardware did not satisfy the memory demands of RT and RH for some instances. Hence, we first merged the three datasets into a single one, and then split it into two set of instances I_1 and I_2: the former could be tackled by HP, RT, RH, and AHP, while the latter could be solved in its entirety only by HP and AHP.

The OVSD dataset can be found at "https://www.research.ibm.com/haifa/projects/imt/video/Video_DataSetTable.shtml", while the Rai and the BBC datasets, proposed by Baraldi et al., can be downloaded at "http://imagelab.ing.unimore.it". The distance matrices associated with the OVSD, the Rai and the BBC datasets, obtained by using the reimplementation of the first stages of Rotman et al.'s scene detection pipeline, are available at "https://github.com/ORresearcher/A-New-Fast-and-Accurate-Heuristic-for-the-Automatic-Scene-Detection-Problem".

4.3. Performance evaluation

In Fig. 4, we report a box-and-whiskers plot that compares the computational times (in seconds) achieved by HP, RT, and RH on the instances in I_1. We can observe the significantly higher computational efficiency of HP with respect to RT and RH. Moreover, the plot shows the statistical equivalence between the computational times of RT and RH, yet highlighting the slightly better performance of the former implementation of Rotman et al.'s algorithm. In Table 1, we report the numerical values of the computational times and the cost function

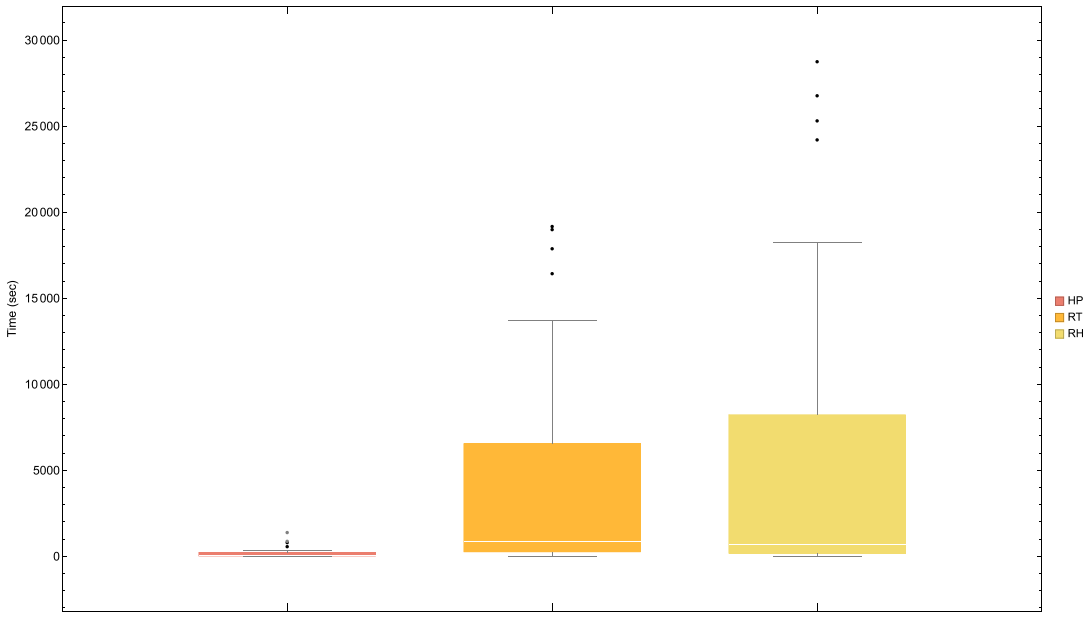

Fig. 4. Box-and-whiskers plot of the CPU times (expressed in seconds) obtained by HP, RT, and RH when solving the instances I_1.

Table 1
Results obtained by HP, RT, RH, and AHP when solving the instances in I_1.

Instance	N	K	Time (s)				$\Delta_{HP,R}$	Cost function		
			HP	RT	RH	AHP		HP	RT/RH	AHP
1000 Days (cut)	204	15	224.4706	4582.7344	5194.4145	157.1406	0.9510	17.8101	18.0011	18.1444
Big Buck Bunny	146	15	75.2031	1166.9219	1017.8210	53.2500	0.9261	17.4070	17.5365	17.7059
Boy Who Never Slept (cut)	193	15	188.1875	3659.8351	3978.9020	132.9844	0.9486	16.9341	17.0847	17.3552
CH7 (cut)	276	10	237.4844	9382.3037	11597.1574	169.0313	0.9747	18.1246	18.2108	18.3013
Cosmos Laundromat — First Cycle	113	7	5.6406	151.6250	69.5630	4.2656	0.9189	18.6779	18.7730	19.0047
Elephants Dream	142	9	22.0000	557.1406	393.8910	15.9375	0.9441	19.5336	19.5726	19.7780
Fires Beneath Water (cut)	143	15	71.6719	1078.0625	915.5120	49.6563	0.9217	17.7851	17.8617	17.9191
Honey (cut)	303	10	306.0000	13718.0156	18240.5745	219.8125	0.9777	16.8148	16.9349	17.0337
Jathia's Wager	181	15	150.9688	2819.5215	2931.1050	106.6250	0.9465	15.9359	15.9810	16.5766
La Chute d'une Plume	83	11	6.3750	81.7500	44.2190	4.7031	0.8558	17.1318	17.2601	17.7979
Lord Meia (cut)	103	15	23.9687	281.1562	183.5780	17.0937	0.8694	14.5889	14.6457	15.2260
Meridian	66	9	2.0000	25.1250	10.8210	1.5156	0.8152	13.8457	13.9322	14.1778
Oceania (cut)	114	20	60.3438	586.1250	381.3300	40.5469	0.8418	11.9540	12.0666	12.4388
Pentagon (cut)	263	20	903.0781	17905.1563	26797.1501	630.6249	0.9496	15.8659	16.1754	16.3271
Route 66 (cut)	279	15	593.2500	16456.9688	25337.4979	417.2969	0.9640	18.0811	18.3828	18.5722
Seven Dead Men (cut)	111	20	53.8906	524.6663	336.2300	37.0781	0.8397	12.7635	12.9786	13.0114
Sintel	154	8	21.4687	645.2187	443.4400	15.6563	0.9516	20.0329	20.1280	20.2041
Sita Sings the Blues (cut)	275	10	229.3125	9284.6361	11842.6030	166.3750	0.9753	17.2451	17.4384	18.0076
Star Wreck (cut)	237	15	354.0781	8504.5689	11263.1990	253.0312	0.9584	17.7036	17.8282	17.9676
Tears of Steel	158	11	51.4219	1111.9531	990.4980	35.3594	0.9481	19.0229	19.1953	19.3423
Valkaama (cut)	275	18	828.1875	19198.9734	28776.2360	584.2969	0.9569	15.9854	16.2155	16.4283
Rai01	114	7	5.7812	160.7656	77.1719	4.4062	0.9251	18.3110	18.3741	18.5637
Rai02	57	12	2.2500	20.0469	8.0625	1.6562	0.7209	15.2982	15.3541	15.4367
Rai03	107	16	31.3906	359.2500	246.0781	21.8437	0.8724	15.9594	16.0769	16.2140
Rai04	143	22	153.3125	1654.1719	1354.4062	103.4844	0.8868	15.9356	15.9678	16.0353
Rai05	66	13	4.2344	39.7500	18.0781	3.0937	0.7658	10.0023	10.2182	10.8087
Rai06	59	5	0.2969	6.6719	1.0000	0.2656	0.7031	18.1266	18.3261	18.6579
Rai07	116	9	11.6250	247.0313	171.0781	8.6406	0.9320	18.5597	18.6649	18.7498
Rai08	205	12	137.5938	3632.6094	3975.9375	98.0000	0.9621	19.6558	19.6984	19.8605
Rai09	106	14	22.9375	300.0469	206.0312	16.5156	0.8887	11.6464	11.8385	12.0166
Rai10	100	16	24.5938	272.6094	174.6875	17.5469	0.8592	10.5562	10.7233	11.4987
BBC08	242	29	1412.6875	19016.8960	24232.4531	979.7500	0.9257	14.6778	14.7678	15.0144

values achieved by the aforementioned three algorithms and AHP on the instances in I_1. In the column named $\Delta_{HP,R}$, Table 1 shows the improvement introduced by HP with respect to the minimum between the times achieved by RH and RT. Specifically, for each instance, if t_0 is the time achieved by HP, and t_1 is the minimum of the times achieved by RH and RT, the value reported for $\Delta_{HP,R}$ is equal to $1 - t_0/t_1$. Such value provides a measure of the ratio between the times achieved by HP and Rotman et al.'s algorithm. The average value of $\Delta_{HP,R}$ is 0.9024, providing an experimental evidence to the sensible improvement introduced by HP. We also observe that the novel heuristic outperforms Rotman et al.'s algorithm in terms of cost function values in any instance of I_1. The second-to-last column of Table 1 is called "RT/RH" since RT and RH produce identical results, and they only differ in the computational requirements. In Table 2 we instead report the performances achieved by HP and AHP on the instances in I_2. We remark that, as anticipated in Section 4.2, we could not produce the Rotman et al.'s algorithm results on I_2 since our hardware was not able to satisfy the memory requirements of RT and

Table 2
Results obtained by HP and AHP when solving the instances in I_2.

Instance	N	K	Time (sec)		Cost function	
			HP	AHP	HP	AHP
1000 Days	329	22	2190.4531	1529.4262	17.7323	18.1663
Boy Who Never Slept	375	36	8813.8250	6121.0312	16.4963	16.9029
Fires Beneath Water	358	62	19860.2090	13582.7812	15.4998	15.6269
Honey	371	20	2620.5625	1842.7500	16.1890	16.5846
Lord Meia	325	27	3166.1719	2202.7812	15.5539	16.085
Oceania	279	31	2545.3438	1769.0000	13.8514	14.1807
Pentagon	333	31	4544.3281	3111.7812	15.4119	15.9333
Seven Dead Men	164	34	507.8125	351.2031	11.9474	12.6003
BBC01	458	23	6742.3125	4721.3514	17.3984	17.7717
BBC02	382	36	9232.1094	6447.0469	15.3077	15.4840
BBC03	412	33	10051.9688	6932.3594	15.9327	16.2966
BBC04	461	30	11854.6105	8213.3410	16.2738	16.9464
BBC05	436	25	6903.8490	4824.6719	16.4621	16.8604
BBC06	506	33	19177.9616	13253.9807	17.4224	17.6644
BBC07	549	37	31293.5912	21476.5831	16.3247	16.7221
BBC09	348	33	5808.5000	4030.7187	15.1449	15.6124
BBC10	346	22	2565.9531	1795.3906	17.2508	17.5522
BBC11	509	26	12312.1875	8419.4657	17.1233	17.7487

RH on such instances. Tables 1 and 2 show that AHP obtains the worst results for the cost function value associated with each instance in I_1 and I_2, respectively, while achieving a slightly better computational efficiency with respect to HP. In fact, we observe that although the computational complexity of AHP and HP is the same, the several floating-point operations performed by HP to compute the surrogate cost function values may be a critical factor in burdening the actual running time of the algorithm.

Finally, Table 3 reports the *Differential Edit Distance* (DED) (Sidiropoulos et al., 2012) scores obtained by the considered algorithm implementations on the instances in I_1 and I_2. DED is a state-of-art performance index used to evaluate the differences between a procedurally generated partition into scenes with respect to a ground-truth partition. In this way, the DED score achieved by an algorithm on a specific instance allows to assess the capability of such algorithm to generate an accurate partition for that instance. For each instance, Table 3 highlights the best scores in bold. HP achieves the best DED score on 37 instances over the 50 instances in I_1 and I_2; among such instances, HP obtains the same best DED score as RT and RH on "Rai03".

5. Conclusions

Detecting scenes in the context of video processing has a central role in the management, storing and content retrieval of videos. The literature proposes different strategies to cope with this task, one of these consisting of modeling scene detection in terms of a combinatorial optimization problem, called the Automatic Scene Detection Problem (ASDP), in which the shots of a given video must be partitioned into scenes so as to optimize a measure related to the similarity between the given shots (Rotman et al., 2018). The proxy nature of the objective function of the ASDP together with the need to run scene detection over very large repositories containing thousands or even million videos justified, in recent times, the development of heuristics able to approximate the optimal solution to the problem as fast as possible (Rotman et al., 2018). In this article we built upon the results from the literature on the ASDP in order to design a new heuristic, called HP, able to outperform the current state-of-the-art both in terms of speed and quality of the provided solution. The empirical derivation of the objective function of the ASDP leaves room for further refinements in terms of modeling of scene detection and motivates the development of improved heuristics for the problem. Investigating these issues will be the subject of future research efforts.

Table 3
DED scores achieved by HP, RT and RH, and AHP on the instances in I_1 and I_2.

Instance	HP	RT/RH	AHP
1000 Days	**0.3860**	–	0.4742
1000 Days (cut)	0.3627	**0.3529**	0.4069
Big Buck Bunny	0.4110	**0.3973**	0.4247
BBC01	**0.3777**	–	0.4105
BBC02	0.4319	–	**0.4084**
BBC03	**0.3495**	–	0.3738
BBC04	0.4534	–	**0.4230**
BBC05	0.4725	–	**0.4656**
BBC06	0.4506	–	**0.4269**
BBC07	**0.3752**	–	0.4463
BBC08	0.4380	**0.4339**	0.4421
BBC09	**0.3994**	–	0.4511
BBC10	**0.4017**	–	0.4682
BBC11	**0.3831**	–	0.4676
Boy Who Never Slept	**0.3973**	–	0.4587
Boy Who Never Slept (cut)	**0.2953**	0.4249	0.4508
CH7 (cut)	**0.1522**	0.1775	0.1993
Cosmos Laundromat — First Cycle	**0.3540**	0.4159	0.5044
Elephants Dream	**0.3592**	0.3732	0.4225
Fires Beneath Water	**0.4106**	–	0.4246
Fires Beneath Water (cut)	**0.3916**	0.4196	0.4196
Honey	**0.4378**	–	0.6270
Honey (cut)	**0.2838**	0.3465	0.4158
Jathia's Wager	**0.3757**	0.3812	0.4254
La Chute d'une Plume	**0.5663**	0.6386	0.6627
Lord Meia	**0.4031**	–	0.5231
Lord Meia (cut)	**0.4757**	0.5243	0.5728
Meridian	**0.4242**	0.5000	0.5303
Oceania	**0.3692**	–	0.4516
Oceania (cut)	**0.2807**	0.3421	0.3772
Pentagon	**0.3964**	–	0.4865
Pentagon (cut)	**0.4259**	0.4867	0.5285
Rai01	0.4825	0.4912	**0.4474**
Rai02	0.5000	**0.4828**	**0.4828**
Rai03	**0.1869**	**0.1869**	0.2617
Rai04	**0.5503**	0.6040	0.6040
Rai05	0.7714	**0.7143**	0.7714
Rai06	**0.1186**	0.1864	0.3051
Rai07	**0.3621**	0.3966	0.3707
Rai08	0.3659	**0.3415**	0.3951
Rai09	0.2736	0.2453	**0.1981**
Rai10	**0.2745**	0.3333	0.5784
Route 66 (cut)	**0.3692**	0.4444	0.4731
Seven Dead Men	**0.3598**	–	0.4024
Seven Dead Men (cut)	**0.2883**	0.3423	0.3604
Sintel	**0.4156**	0.4351	0.4675
Sita Sings the Blues (cut)	0.2836	**0.2800**	0.3055
Star Wreck (cut)	**0.3882**	0.3966	0.4430
Tears of Steel	**0.4051**	0.4620	0.4747
Valkaama (cut)	**0.1600**	0.2400	0.2727

Acknowledgments

The first author acknowledge support from the Université Catholique de Louvain, Belgium via the Fonds Spéciaux de Recherche (FSR) 2017–2021 and the Fondation Louvain, Belgium via the research grant COALESCENS of the funding program "Le numérique au service de l'humain". The authors are in debt with Alessandro Birago and Igor Pio Bianchi for their valuable help in reimplementing the components of the scene detection pipeline used to generate the distance matrices.

References

Ariki, Y., Kumano, M., Tsukada, K., 2003. Highlight scene extraction in real time from baseball live video. In: Proceedings of the 5th ACM SIGMM International Workshop on Multimedia Information Retrieval, pp. 209–214.

Baraldi, L., Grana, C., Cucchiara, R., 2015a. Analysis and re-use of videos in educational digital libraries with automatic scene detection. In: Proceedings of the Italian Research Conference on Digital Libraries, pp. 155–164.

Baraldi, L., Grana, C., Cucchiara, R., 2015b. A deep siamese network for scene detection in broadcast videos. In: Proceedings of the 23rd ACM International Conference on Multimedia, pp. 1199–1202.

Baraldi, L., Grana, C., Cucchiara, R., 2015c. Shot and scene detection via hierarchical clustering for re-using broadcast video. In: International Conference on Computer Analysis of Images and Patterns, pp. 801–811.

Baraldi, L., Grana, C., Cucchiara, R., 2016. Recognizing and presenting the storytelling video structure with deep multimodal networks. IEEE Trans. Multimed. 19 (5), 955–968.

Choroś, K., 2009. Video shot selection and content-based scene detection for automatic classification of TV sports news. In: Tkacz, E., Kapczynski, A. (Eds.), Internet – Technical Development and Applications. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 73–80.

Del Fabro, M., Böszörmenyi, L., 2010. Video scene detection based on recurring motion patterns. In: Proceedings of the Second International Conferences on Advances in Multimedia. IEEE, pp. 113–118.

Del Fabro, M., Böszörmenyi, L., 2013. State-of-the-art and future challenges in video scene detection: A survey. Multimedia Syst. 19 (5), 427–454.

Feng, Y., Ren, R., Jose, J., 2008. Rule-based scene boundary detection for semantic video segmentation. In: Proceedings of the 5th International Conference on Visual Information Engineering, VIE. IET Digital Library, pp. 667–672.

Han, B., Wu, W., 2011. Video scene segmentation using a novel boundary evaluation criterion and dynamic programming. In: Proceedings of the IEEE International Conference on Multimedia and Expo, pp. 1–6.

Kurihara, K., Imai, A., Seiyama, N., Shimizu, T., Sato, S., Yamada, I., Kumano, T., Tako, R., Miyazaki, T., Ichiki, M., Takagi, T., Sumiyoshi, H., 2019. Automatic generation of audio descriptions for sports programs. SMPTE Motion Imag. J. 128, 41–47.

Liang, Y., Liu, W., Liu, K., Ma, H., 2018. Automatic generation of textual advertisement for video advertising. In: Proceedings of the IEEE Fourth International Conference on Multimedia Big Data, BigMM, pp. 1–5.

Panda, R., Kuanar, S.K., Chowdhury, A.S., 2017. Nyström approximated temporally constrained multisimilarity spectral clustering approach for movie scene detection. IEEE Trans. Cybern. 48 (3), 836–847.

Rasheed, Z., Shah, M., 2005. Detection and representation of scenes in videos. IEEE Trans. Multimed. 7 (6), 1097–1105.

Rotman, D., Porat, D., Ashour, G., 2016. Robust and efficient video scene detection using optimal sequential grouping. In: Proceedings of the 2016 IEEE International Symposium on Multimedia, ISM, pp. 275–280.

Rotman, D., Porat, D., Ashour, G., 2017. Robust video scene detection using multimodal fusion of optimally grouped features. In: Proceedings of the IEEE 19th International Workshop on Multimedia Signal Processing, MMSP, pp. 1–6.

Rotman, D., Porat, D., Ashour, G., Barzelay, U., 2018. Optimally grouped deep features using normalized cost for video scene detection. In: Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval, pp. 187–195.

Rotman, D., Yaroker, Y., Amrani, E., Barzelay, U., Ben-Ari, R., 2020. Learnable optimal sequential grouping for video scene detection. In: Proceedings of the 28th ACM International Conference on Multimedia.

Sakarya, U., Telatar, Z., 2008. Video scene detection using dominant sets. In: Proceedings of the 15th IEEE International Conference on Image Processing, pp. 73–76.

Sidiropoulos, P., Mezaris, V., Kompatsiaris, Y., Kittler, J., 2012. Differential edit distance: A metric for scene segmentation evaluation. IEEE Trans. Circuits Syst. Video Technol. 22, 904–914.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2016. Rethinking the Inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, pp. 2818–2826.

Zhai, Y., Shah, M., 2006. Video scene segmentation using Markov chain Monte Carlo. IEEE Trans. Multimed. 8 (4), 686–697.

Zhai, Y., Yilmaz, A., Shah, M., 2005. Story segmentation in news videos using visual and text cues. In: Proceedings of the International Conference on Image and Video Retrieval. Springer, pp. 92–102.