A New Fast and Accurate Heuristic for the Automatic Scene Detection Problem

Daniele Catanzaro\textsuperscript{a,1}, Raffaele Pesenti\textsuperscript{b,2} and Roberto Ronco\textsuperscript{c,***,3}

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

\textbf{ARTICLE INFO}

\textbf{Keywords:}
Combinatorial Optimization; Video Processing; Segmentation; Scene Detection; Heuristics; Dynamic Programming.

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

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

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

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

Moreover, let $\mathcal{P}$ denote the set of the possible partitions of $S$ 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 $P$:

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

\*Corresponding author

ORCID(s): 0000-0001-9427-1562 (D. Catanzaro); 0000-0001-5890-4238 (R. Pesenti); 0000-0001-6846-6374 (R. Ronco)

1Email: daniele.catanzaro@uclouvain.be (Daniele Catanzaro)

2Email: pesenti@unive.it (Raffaele Pesenti)

3Email: roberto.ronco@edu.unige.it (Roberto Ronco)
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 $\sigma$. 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. [17], who reported several empirical considerations on the efficacy of $z(P)$ as opposed to other cost functions described in [15, 16]. 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 $d_{q,q}$, $q \in \{1, \ldots, N\}$, 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 [17]. 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 Figure 2.

The ASDP was introduced in the literature on video processing by Rotman et al. [17] 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 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 [11, 23]. 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 [1, 6, 7]. 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 [12]. 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...
A New Heuristic for the Automatic Scene Detection Problem

Figure 2: An example of a possible distance matrix $\mathbf{D}$ (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 MatrixForm of $\mathbf{D}$: the lighter the colors the lower the entries of $\mathbf{D}$ they refer to. Conversely, the darker the colors the bigger the entries of $\mathbf{D}$ they refer to.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5.42</td>
<td>6.58</td>
<td>9.80</td>
<td>1.96</td>
<td>3.45</td>
<td>9.68</td>
<td>6.44</td>
<td>12.65</td>
<td></td>
</tr>
<tr>
<td>5.42</td>
<td>0</td>
<td>4.26</td>
<td>6.59</td>
<td>5.58</td>
<td>7.83</td>
<td>7.22</td>
<td>8.28</td>
<td>10.74</td>
<td></td>
</tr>
<tr>
<td>6.58</td>
<td>4.26</td>
<td>0</td>
<td>10.61</td>
<td>9.31</td>
<td>6.90</td>
<td>3.24</td>
<td>6.46</td>
<td>5.58</td>
<td></td>
</tr>
<tr>
<td>9.80</td>
<td>6.59</td>
<td>10.61</td>
<td>0</td>
<td>5.71</td>
<td>12.56</td>
<td>7.37</td>
<td>6.27</td>
<td>5.87</td>
<td></td>
</tr>
<tr>
<td>1.96</td>
<td>5.58</td>
<td>9.31</td>
<td>5.71</td>
<td>0</td>
<td>11.37</td>
<td>8.30</td>
<td>10.50</td>
<td>3.38</td>
<td></td>
</tr>
<tr>
<td>3.45</td>
<td>7.83</td>
<td>6.90</td>
<td>12.56</td>
<td>11.37</td>
<td>0</td>
<td>3.03</td>
<td>8.93</td>
<td>7.11</td>
<td></td>
</tr>
<tr>
<td>9.68</td>
<td>7.22</td>
<td>3.24</td>
<td>7.37</td>
<td>8.30</td>
<td>3.03</td>
<td>0</td>
<td>1.95</td>
<td>6.52</td>
<td></td>
</tr>
<tr>
<td>6.44</td>
<td>8.28</td>
<td>6.46</td>
<td>6.27</td>
<td>10.50</td>
<td>8.93</td>
<td>1.95</td>
<td>0</td>
<td>7.77</td>
<td></td>
</tr>
<tr>
<td>12.65</td>
<td>10.74</td>
<td>5.58</td>
<td>5.87</td>
<td>3.38</td>
<td>7.11</td>
<td>6.52</td>
<td>7.77</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

To carry out this task [8]. Specifically, we may distinguish between the rule-based algorithms [9], the stochastic-based algorithms [10, 22], the graph-based algorithms [14, 19], and finally the clustering algorithms [4, 13, 17]. 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 Bőszörményi [8] 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 [2, 3, 4, 5] stand out as one the most important examples that aim to pave the road towards 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 [4], Baraldi et al. further proposed a clustering algorithm to perform scene detection, grouping adjacent shots based on their color spectra [2]. 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. [3] 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. [17] 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 $\mathbf{D}$ 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 loosing in terms of accuracy [18]. Rotman et al. [17] algorithm, and more in general their overall framework [18], currently constitutes the current state-of-the-art for the ASDP.

In this article we build upon Rotman et al. [17, 18] 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. [17] 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. [17]’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 $I$ of the ASDP is characterized by the triplet $(S, D, K)$ 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 $\{i, i+1, \ldots, i+j\}$, 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 $1 \leq k \leq \min\{j-i+1, K\}$, let $P_{i,j}^k$ 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 $I$ of the ASDP until obtaining a solution to $I$. 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 $P_{i,N}^k$ that locally minimizes the following surrogate cost function

$$C_i^k(e) := \frac{\sum_{\sigma \in P_{i,N}^k} a(\sigma)}{e + \sum_{\sigma \in P_{i,N}^k} |\sigma|^2},$$

where the positive integer

$$e \in \epsilon_i^k := \begin{cases} \left\lfloor \frac{(i-1)^2}{(K-k)} \right\rfloor, (i-K+k)^2 + K-k-1 & k \in [1, K-1] \\ 0 & \text{otherwise,} \end{cases}$$

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 $|\epsilon_i^k|$ 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

$$\sum_{\sigma \in P_{i,N}^1} a(\sigma) = \sum_{r=p}^q d_{r,r} = 0,$$

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

$$C_i^1(e) = \frac{a(\sigma_{i,N})}{e + (N-i+1)^2}, \quad \forall e \in \epsilon_i^1.$$

Hence, the minimization of the surrogate cost function $C_i^1(e)$ 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

D. Catanzaro et al.: Preprint submitted to Elsevier
finding, for each of them, the partition \( P_{i,N}^k \) that minimizes the surrogate cost \( C_i^k(e) \), for all \( e \in e_i^k \). 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 \( P_{i,j}^1 \) and \( P_{j+1,N}^{k-1} \) for the intervals \([i, j]\) and \([j + 1, N]\), respectively, and sets \( P_{i,N}^k = P_{i,j}^1 \cup P_{j+1,N}^{k-1} \). The partition \( P_{i,N}^k \) that minimizes the function

\[
G_{i,j}^k(e) := \frac{\sum_{\sigma \in P_{i,j}^1} a(\sigma)}{e} + \frac{\sum_{\sigma \in P_{i,j}^1 \cup P_{j+1,N}^{k-1}} |\sigma|^2}{e}
\]

over all of the possible values of \( e \in e_i^k \) is then selected as the best one for the interval \([i, N]\). Rotman et al. observe that the values of \( C_i^k(e) \) and \( G_{i,j}^k(e) \) are related by means of the two other quantities \( X_i^k(e) \) and \( A_i^k(e) \), corresponding to the last shot of the first scene of the partition associated with \( C_i^k(e) \), and to the contribution of \( P_{i,N}^k \) to the denominator of \( C_i^k(e) \), respectively. In particular, the base case for \( C_i^k(e) \), \( X_i^k(e) \), and \( A_i^k(e) \) is computed as

\[
C_i^1(e) = \frac{a(\sigma_{i,N})}{e + (N - i + 1)^2} \quad (2)
\]

\[
X_i^1(e) = N \quad (3)
\]

\[
A_i^1(e) = (N - i + 1)^2 \quad (4)
\]

for all \( i \in [K, N] \) and \( e \in e_i^k \). The iterative step, instead, is characterized by the following relations:

\[
C_i^k(e) = \min_{j \in [i, N - k + 1]} \left\{ G_{i,j}^k(e) + C_{j+1}^{k-1}(e + (j - i + 1)^2) \right\} \quad (5)
\]

\[
G_{i,j}^k(e) = \frac{a(\sigma_{i,j})}{e + (j - i + 1)^2 + A_{j+1}^{k-1}(e + (j - i + 1)^2)} \quad (6)
\]

\[
X_i^k(e) = \arg\min_{j \in [i, N - k + 1]} \left\{ G_{i,j}^k(e) + C_{j+1}^{k-1}(e + (j - i + 1)^2) \right\} \quad (7)
\]

\[
A_i^k(e) = (X_i^k(e) - i + 1)^2 + A_{j+1}^{k-1}(e + (j - i + 1)^2) \quad (8)
\]

for \( i \in [1, N] \), \( k \in \mathbb{K} \setminus \{1\} \), \( p \in e_i^k \), 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 \( \sigma_{1,j-1} \) into \( K - k \) scenes, which is not considered when solving the problem of partitioning \( \sigma_{1,N} \) 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 equations (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 \( \infty \) for each \((i, k, e) \in F \). Lines 2–4 compute the bases cases according to equations (2)–(4). Lines 5–20 compute the step cases according to equations (5)–(8). Lines 22–25 reconstruct the partition \([\{1, s_1\}, [s_1 + 1, s_2], \ldots, [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

\[
C_1^K(0) = \frac{\sum_{\sigma \in P_{1,N}^K} a(\sigma)}{0 + \sum_{\sigma \in P_{1,N}^K} |\sigma|^2} = z(P_{1,N}^K).
\]
Algorithm 1: Rotman et al.'s algorithm

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

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 \( e \in e_i^k \) do
      4.     Compute \( C_i^j(e) \), \( X_i^j(e) \), and \( A_i^j(e) \) with equations (2)--(4);
3. foreach \( i \in S \) do
   4.   foreach \( k \in \kappa_i \) do
      5.     if \( k = K \) then
         6.       \( e_i \leftarrow e_j \leftarrow 0 \);
      7.     else
         8.       \( e_i \leftarrow [(i-1)^2/(K-k)] ; \)
         9.       \( e_j \leftarrow (i-1) - (K-k + 1)^2 + K - k - 1 \);
      10.    foreach \( e \in [e_i, e_j] \) do
        11.       if \( C_i^k(e) = \infty \) then continue;
        12.         \( G_i^k_j(e) \leftarrow a(\sigma_{i,j}) / \left( e + (j - i + 1)^2 + A_i^{k-1}(e + (j - i + 1)^2) \right) ; \)
        13.         if \( G_i^k_j(e) + C_i^{k-1}(e + (j - i + 1)^2) < C_i^k(e) \) then
           14.             \( C_i^k(e) \leftarrow G_i^k_j(e) + C_j^{k-1}(e + (j - i + 1)^2) \);
           15.             \( X_i^k(e) \leftarrow j \);
           16.             \( A_i^k(e) \leftarrow (j - i + 1)^2 + A_j^{k-1}(e + (j - i + 1)^2) \);
        17.    // Recovering the partition into scenes with equation (7);
   18.    \( V \leftarrow 0, x_0 \leftarrow 0 \);
   19.    foreach \( k \in [1, K] \) do
        20.      \( s_k \leftarrow X_{s_{k-1} + 1}^{K-k+1} \);
        21.      \( V \leftarrow V + (s_k - s_{k-1})^2 \);
   22.    \( P_{i,N}^k \leftarrow \{[1, s_1], [s_1 + 1, s_2], \ldots, [s_{K-1} + 1, s_K] \} \);
23. return \( P_{i,N}^k, C_i^k(0) \);
```

In particular, observe that, by definition of \( X_i^k(e) \), \( s_k \) is the last shot of \( k \)-th scene at the end of the \( k \)-th 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 \( T_{n,e} \) 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 \( e \in e_{n+1}^k \), \( T_{n,e} = true \) when \( n \) shots can be partitioned into \( k \) scenes and there exists \( P_{i,j}^k \), \( i, j \in S, i \leq j \), 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 \( T_{n,e} = false \). The base cases are computed as

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

for \( n \in S \) and \( e \in e_{n+1}^k \), 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 $T^k_{n,e}$ is

$$T^k_{n,e} = \sqrt[k]{e} - 1 \sum_{q=1}^{\left\lfloor \sqrt[k]{e} \right\rfloor} T^{k-1}_{n-q,e-q^2}$$

for $n \in S$, $k \in [2, \min\{K, n\}]$, and $e \in E_n$. Specifically, note that as $q < n$ shots can be partitioned into scene $\hat{s}$ with $|\hat{s}|^2 = q^2$, for each $q \in [1, \left\lfloor \sqrt[e]{e} \right\rfloor - 1]$, then there is a partition $P'$ of $n$ shots into $k$ scenes such that $\sum_{\sigma \in P'} \sigma = e$, i.e., $T^k_{n,e}$ is 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

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

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

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

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

$$C_3^2(4) = \min\{G_{3,3}^1(4) + C_3^1(5), \ G_{3,4}^1(4) + C_3^1(8), \ G_{3,5}^1(4) + C_3^1(13)\} = \min\left\{\frac{4}{14}, \frac{0}{12}, \frac{0}{14}\right\} = 0. $$

Figure 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 $G_{3,4}^1(4) + C_3^1(8) = 0$, then $A_3^1(4) = 12$, and $X_3^1(4) = 4$. Moreover, with simple arithmetic steps we also obtain

$$C_2^2(1) = \frac{4}{9}, \quad A_2^1(1) = 17, \quad X_2^2(1) = 2$$

$$C_4^2(9) = 0, \quad A_4^2(9) = 5, \quad X_4^2(9) = 5$$

$$C_5^2(16) = 0, \quad P_5^2(16) = 2, \quad I_5^2(16) = 5.$$ 

Therefore,

$$C_1^3(0) = \min\{G_{1,1}^1(0) + C_{1,2}^2(1), \ G_{1,2}^1(0) + C_{1,3}^2(4), \ G_{1,3}^1(0) + C_{1,4}^2(9), \ G_{1,4}^1(0) + C_{1,5}^2(16)\} = \min\left\{\frac{4}{9}, \frac{1}{3}, \frac{3}{7}, \frac{1}{1}\right\} = \frac{1}{3},$$

corresponding to the partition \{\{1, 2\}, \{3, 4\}, \{5, 6\}\}.

As regards the computational complexity of Algorithm 1, we first observe that the values $a(\sigma_{i,j})$, $i, j \in S$, can be precomputed before running Algorithm 1 by means of a simple $O(N^2)$ dynamic programming algorithm described by Rotman et al. [15], and outlined in the following. Hereinafter, we denote by $D_{i,j}$, $i \leq j$, the sub-matrix $[d_{q,r}]$ of $D$, where $q, r \in [i, j]$. We observe that $D_{i,j}$ is a square sub-matrix on the main diagonal of $D$. The algorithm first
initializes $a(\sigma_{i,j}) = 0$, $i \in S$. Then, for $l \in [2, N]$ and $i \in [1, N - l + 1]$, the algorithm computes $a(\sigma_{i,i+l-1})$ as

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

The first two terms of (10) are the sums of the entries belonging to the sub-matrices $D_{i,i+l-2}$ and $D_{i+1,i+l-1}$. Since these two sub-matrices share the entries in $D_{i,i+l-2}$, then $a(\sigma_{i,i+l-2})$ is counted twice in the sum $a(\sigma_{i,i+l-2}) + a(\sigma_{i+1,i+l-1})$, hence $a(\sigma_{i+1,i+l-2})$ 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 $d_{i+1,i+l-1}$ are the only two entries in $D_{i,i+l-1}$ that are not considered in $a(\sigma_{i,i+l-2}) + a(\sigma_{i+1,i+l-1}) - a(\sigma_{i+1,i+l-2})$. 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

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

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

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

$$\sim O\left(\sum_{i=1}^{N} \sum_{k=M_{1,i}}^{m_{K,i}} N \frac{k - 1}{k}i^2 + Nk^2 - 2Nki - 2Nk + 2Ni$$

$$- ki^2 - k - k^3 - 2ki^2 - 2k^2 - 2ki - \frac{k - 1}{k}i^3 - k^2i + 2ki^2 + 2ki - 2i^2\right). \quad (11)$$

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

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

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

\[
\begin{bmatrix}
0 & 2 & 1 & 2 & 1 & 1 \\
2 & 0 & 2 & 2 & 1 & 0 \\
1 & 2 & 0 & 0 & 0 & 2 \\
2 & 2 & 0 & 0 & 0 & 2 \\
1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 2 & 2 & 0 & 0 \\
\end{bmatrix}
\]
A New Heuristic for the Automatic Scene Detection Problem

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

By recalling that

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

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

\[
O\left( \sum_{i=1}^{N-K} \frac{K(K + 2N + 1)}{2} i^2 + (N - 3i) \frac{K(K + 1)(2K + 1)}{6} - Ki^3 - \frac{K^2(K + 1)^2}{4} + \sum_{i=N-K+1}^{N} \frac{(N - i + 1)(3N - i + 2)}{2} i^2 + (N - 3i)(N - i + 1)(N - i + 2)(2N - 2i + 3)
\]

\[
- i^3 (N - i + 1) - (N - i + 2)^2 \right)
\]

\[
\sim O\left( \sum_{i=1}^{N-K} (K^2 i^2 + NK i^2 + K^3 N - K^3 i - Ki^3 - K^4) + \sum_{i=N-K+1}^{N} ((N^2 - i^2) i^2 + N^4 - N^3 i + N^2 i^2 - Ni^3 + i^4) \right). \tag{13}
\]

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

\[
O(K^2(N-K)^2 + NK(N-K)^2 + K^3 N - K^3(N-K) - K(N-K)^3 - K^4 + KN^4) \sim O(KN^4).
\]

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. [17]'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 \(I = (S, D, K)\) of the ASDP, a subset of shots \([i, j] \subseteq S\) and a positive integer \(k\) such that \(1 \leq k \leq \min\{K, j - i + 1\}\), we denote by \(I_{i,j}^k := ([i, j], D, K, k)\) a sub-instance of \(I\) 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 \(I_{i,j}^k = I_{1,N}^k = I\), i.e., the sub-instance \(I_{i,j}^k\) coincides with the given input instance \(I\) to solve. We denote by \(\tilde{I}_{i,j}^k\) the partition of the feasible sub-instance \(I_{i,j}^k\) with (locally) minimum cost \(z(\tilde{I}_{i,j}^k)\), obtained by recursively splitting \(I_{i,j}^k\) into the two feasible sub-instances \(I_{i,v}^h, I_{v+1,j}^{k-h}\) 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 \(I_{i,j}^k\) into \(I_{i,v}^h\) and \(I_{v+1,j}^{k-h}\), for each \(v \in [i, j - 1]\) and \(h \in [1, k - 1]\), (ii) finding the (locally) minimum cost partitions \(\tilde{I}_{i,v}^h\) and \(\tilde{I}_{v+1,j}^{k-h}\) for \(I_{i,v}^h\) and \(I_{v+1,j}^{k-h}\) respectively, and finally (iii) choosing a partition \(\tilde{I}_{i,j}^k\) for \(I_{i,j}^k\) that can be written as \(\tilde{I}_{i,j}^k = \tilde{I}_{i,v}^h \cup \tilde{I}_{v+1,j}^{k-h}\) and that (locally) minimizes the cost function \(z(\tilde{I}_{i,j}^k)\). 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 \(I = ([1, 12], D, 7)\) of the ASDP, for some distance matrix \(D\). Suppose that the solution to \(I\) can be obtained by splitting \(I\) as \(I_{1,6}^4 \cup I_{7,12}^3\) and by computing the partition \(P_{1,12}^7\) as the union of the
partitions $P_{1,6}^4$ and $P_{7,12}^3$, respectively. Finally, suppose that $z(P_{7,12}^3) = 11 / 12$, and that $D_{1,6} = [d_{q,r}], q, r \in [1,6]$ be as follows

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

It is easy to see that the optimal partition

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

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

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

and observe that it is locally suboptimal for $I_{1,6}^4$, as characterized by a cost $z(\hat{P}_{1,6}^4) = 2 / 12 > z(\hat{P}_{1,6}^4)$. However, the union of $\hat{P}_{1,6}^4$ with $\hat{P}_{7,12}^3$ gives rise to the partition $\hat{P}_{1,12}^7 = \hat{P}_{1,6}^4 \cup \hat{P}_{7,12}^3$ with cost $z(\hat{P}_{1,12}^7) = (2 + 11) / (12 + 12) = 13 / 24 < z(\hat{P}_{1,12}^7)$. 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. [17]'s splitting strategy. Specifically, note that Rotman et al. [17]'s solution space, consisting of the set of partitions that can be written as $P_{i,N}^k = P_{i,j}^k \cup P_{j+1,N}^{k-h}$, is contained in the solution space constituted by the partitions that can be written as $P_{i,N}^k = P_{i,j}^h \cup P_{j+1,N}^{k-h}$. Hence, provided that both $k > 2$ and $1 < h < k$, 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 $S$.

**Example 3.** Consider the instance $I = ([1, 8], D, 4)$ and the sub-instance $I_{2,7}^3$. It is easy to see that the union of $\hat{P}_{2,7}^3$, (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 $I_{2,7}^3$ is infeasible.

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

$$
k \in \kappa_{i,j} := \lfloor \max\{K - N + j - i + 1, 1\}, \min\{K - \min\{i - 1, 1\}, \min\{N - j, 1\}, j - i + 1\} \rfloor
$$

$$
= \min\{K - \min\{i - 1, 1\}, \min\{N - j, 1\}, j - i + 1, K - 2, K - i, j - i + 1\}.
$$

(14)

In particular, because the shots in $[i, j]$ must be partitioned in $k$ scenes and the shots in $S \setminus [i, j]$ 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 $S \setminus [i, j]$ 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 $I_{i,j}^k$ is feasible when the indices $i, j, and k$ satisfy (14).

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

$$
v \in [i, j - 1],
$$

(15a)

D. Catanzaro et al.: Preprint submitted to Elsevier
we obtain:

\[ v, h \]

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

\[ h \in h_{i,j,v} := \max \{ k - i + 1 - N + v, 1, k - K + 1 + \min \{ N - j, 1 \}, k - j + v \}, \]

\[ \min \{ k - K + N - j + v, k - 1, K - \min \{ i - 1, 1 \} - 1, v - i + 1 \} \].

\[ (15b) \]

**Proof.** Since \( I_{i,j}^k \) is feasible, then \( h \in K_{i,j} \), and \( k \leq j - 1 \) + 1. Hence, there exists a pair \((v, h)\) such that \( i \leq v \leq j - 1 \) for \( h \in [1, k] \). Moreover, each of such pairs \((v, h)\) must satisfy the properties \( h \leq v - i + 1 \) and \( k - h \leq j - v \), because both \( I_{i,v}^h \) and \( I_{v+1,j}^{h-k} \) are feasible, i.e., \( h \in K_{i,v} \), and \( k - h \in K_{v+1,j} \). 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 \( I_{i,v}^h \) and \( I_{v+1,j}^{h-k} \) if and only if \( i \leq v \leq j - 1 \). This proves (15a). By applying (14) to \( I_{i,v}^h \) and \( I_{v+1,j}^{h-k} \), we obtain:

\[
\begin{aligned}
    &h \geq \max \{ K - N + v - i + 1, 1 \} \\
    &h \leq \min \{ K - \min \{ i - 1, 1 \} - 1, v - i + 1 \} \\
    &k - h \geq K - N + j - v \\
    &k - h \geq 1 \\
    &k - h \leq K - 1 - \min \{ N - j, 1 \} \\
    &k - h \leq j - v.
\end{aligned}
\]

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 \( I_{i,j}^k = I_{i,v}^h \cup I_{v+1,j}^{h-k} \) of \( I \) as \( P_{i,v}^h \cup P_{v+1,j}^{h-k} \) with \((v, h)\) satisfying (15). This notation will prove useful to outline

---

**Algorithm 2: Recursive-Solver**

**Input:** A feasible sub-instance \((i, j), D, K, k\), and matrix \(B\).

**Output:** The values \( q_{i,j}^k, i_{i,j}^k, \) and \( \hat{P}_{i,j}^k \).

**Internal Data:** \( q' \in \mathbb{R}_+ \), \( y' \in \mathbb{Z}_+ \), and set \( Q\).

**Internal Functions:** \( q_{p,q}^r \in \mathbb{R}_+ \), \( P_{p,q}^r \in \mathbb{R}_+ \). ∀ \( p, q, r : i \leq p \leq q \leq r \in [1, k] \) (global scope).

1 // Returns the solution to processed subproblems
2 if \( q_{i,j}^k < \infty \) then
3 return \( q_{i,j}^k, i_{i,j}^k, \hat{P}_{i,j}^k \);
4 // Recursion base case
5 if \( k == 1 \) then
6 \( q_{i,j}^1 \leftarrow a(i_{i,j}) \);
7 \( \gamma_{i,j}^1 \leftarrow (i - j + 1)^2 \);
8 \( \hat{P}^1_{i,j} \leftarrow [i, j] \);
9 else
10 // Recursion step case
11 \( q' \leftarrow \infty \), \( \gamma' \leftarrow 1 \);
12 foreach \( v \in [i, j - 1] \) do
13 \( q' \leftarrow \infty \), \( \gamma' \leftarrow 1 \);
14 foreach \( h \) satisfying (15b) do
15 \( q_{i,j,v}^h \leftarrow \text{Recursive-Solver}([[i, v], D, K, h], B);\)
16 \( q_{i,j,v}^{h-k} \leftarrow \text{Recursive-Solver}([[v+1, j], D, K, k-h], B);\)
17 \( q' \leftarrow q_{i,j,v}^h + q_{v+1,j}^{h-k};\)
18 \( \gamma' \leftarrow \gamma_{i,j,v}^h + \gamma_{v+1,j}^{h-k};\)
19 \( Q \leftarrow \hat{P}_{i,j}^k \cup \hat{P}_{v+1,j}^{k-h} \).
20 return \( q_{i,j,v}^{k-h}, \gamma_{i,j,v}^{k-h}, \hat{P}_{i,j}^k \).

\[ h \in h_{i,j,v} := [\max \{ k - i + 1 - N + v, 1, k - K + 1 + \min \{ N - j, 1 \}, k - j + v \}, \]

\[ \min \{ k - K + N - j + v, k - 1, K - \min \{ i - 1, 1 \} - 1, v - i + 1 \} \].

\[ (15b) \]
A New Heuristic for the Automatic Scene Detection Problem

Algorithm 3: Heuristic-Partitioner

Input : Instance $\mathcal{I} = ([1, N], D, K)$, matrix $B$.
Output : $z_{1,N}^K, \tilde{P}_{1,N}^K$.

Internal Functions : $\varphi_{i,j}, \gamma_{i,j}, \tilde{P}_{i,j}^k, i \in [1, N], j \in [i, N], k \in [1, K]$ (global scope).

1. $\varphi_{i,j}^k \leftarrow \infty$ for $i = 1, \ldots, N - 1$, $j = 2, \ldots, N$, $k = 1, \ldots, N$: // Initialization
2. $\varphi_{1,N}^K, \gamma_{1,N}^K, \tilde{P}_{1,N}^K \leftarrow$ Recursive-Solver($\mathcal{I}, B$): // Main
3. return $\varphi_{1,N}^K / \gamma_{1,N}^K, \tilde{P}_{1,N}^K$.

the new heuristic for the ASDP shown in Algorithm 2. Specifically, given a partition $P_{i,j}^k$ for $\mathcal{I}_{i,j}^k$, we denote

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

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

and we rewrite $z(P_{i,j}^k)$ as

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

We also denote by

$$
\varphi_{i,j}^k := \varphi(\tilde{P}_{i,j}^k), \text{ and } \gamma_{i,j}^k := \gamma(\tilde{P}_{i,j}^k),
$$

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

$$
\Lambda_{i,j}^k = \begin{cases}
\Omega_{i,j,v}^{k,h} : (v, h) \text{ satisfies (15)} & \text{if } k \geq 2 \\
\varphi_{i,j}^1, \gamma_{i,j}^1 & \text{if } k = 1
\end{cases} \tag{16}
$$

with

$$
\Omega_{i,j,v}^{k,h} = \frac{\varphi_{i,b}^h + \varphi_{v+1,j}^{k-h}}{\gamma_{i,b}^h + \gamma_{v+1,j}^{k-h}}. \tag{17}
$$

Observe that, by definition, $\varphi_{i,j}^k$ and $\gamma_{i,j}^k$ are such that

$$
\min \Lambda_{i,j}^k = \frac{\varphi_{i,j}^1}{\gamma_{i,j}^1}. \tag{18}
$$

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 $\tilde{P}_{i,j}^k$ for the instance $\mathcal{I}_{i,j}^k$. 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 $z_{1,N}^K$ and $\tilde{P}_{1,N}^K$. We observe that both Recursive-Solver and Heuristic-Partitioner treat $\varphi_{i,j}^k, \gamma_{i,j}^k, \tilde{P}_{i,j}^k$ as tensors in order to store the solutions to the sub-
instances of \( I \) already processed. In particular, Heuristic-Partitioner first initializes \( \varphi^k_{i,j} \) for all \( i, j \in S, i \leq j, k \in [1, K] \), at line 1, then it solves \( I = I^K_{1,N} \), by calling Recursive-Solver\((I^K_{1,N}, \mathbf{B})\) at line 2, and finally returns the computed values of \( z^K_{1,N} \) and \( P^K_{1,N} \). We observe that \( \varphi^k_{i,j}, \gamma^k_{i,j}, \) and \( \tilde{P}^k_{i,j} \) have global scope so that their values can be directly accessed. Recursive-Solver first sets \( \varphi^k_{i,j} = \infty \) for all \( i, j \in S, i \leq j, k \in [1, K] \). The same operation is done also for \( \gamma^k_{i,j} \) and \( \tilde{P}^k_{i,j} \). This convention is used to indicate that the feasible sub-instance \( I^k_{i,j} \) 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 \( \tilde{P}^k_{i,j} \) in correspondence to a given input feasible sub-instance \( I^k_{i,j} \) of \( I \). 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 \( I^k_{i,j}, k \geq 2 \), Recursive-Solver computes \( \Omega^{k,h}_{v,i,j} \) for each pair \((v, h)\) satisfying (15), by recursively calling itself on \( I^h_{i,v} \) and \( I^h_{i,v} \), so as to determine \( \min\{\Lambda^k_{i,j}\} \). When the recursion unfolds, the same sub-instances may arise multiple times while splitting distinct \( I^k_{i,j} \). 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 \( I^k_{i,j} \) has already been solved, and its computed solution can be immediately returned. Lines 4–8 compute the only possible solution of \( I^1_{i,j} \). Lines 9–22 tackle the problem of solving \( I^k_{i,j} \) when \( k \geq 2 \) 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 \( I^h_{i,v} \) and \( I^{k-h}_{v+1,i,j} \) 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 \( \varphi^{h}_{i,v} + \varphi^{k-h}_{v+1,i,j} \) and \( \gamma^{h}_{i,v} + \gamma^{k-h}_{v+1,i,j} \) to \( \varphi' / \gamma' \), lines 17 and 18 assign the values \( \varphi^{h}_{i,v} + \varphi^{k-h}_{v+1,i,j} + \gamma^{h}_{i,v} + \gamma^{k-h}_{v+1,i,j} \) to \( \varphi' / \gamma' \), respectively, so that \( \varphi' / \gamma' = (\varphi^{h}_{i,v} + \varphi^{k-h}_{v+1,i,j}) / (\gamma^{h}_{i,v} + \gamma^{k-h}_{v+1,i,j}) \). Moreover, line 19 saves the associated partition \( \tilde{P}^h_{i,v} \cup P^{k-h}_{v+1,i,j} \) by assigning it to the local variable \( Q \). At the end of the algorithm, \( z^k_{i,j} \) is the value given by equation (18), associated with the partition \( \tilde{P}^k_{i,j} \). At line 23, the algorithm finally returns \( \varphi^k_{i,j}, \gamma^k_{i,j}, \) and the computed partition \( \tilde{P}^k_{i,j} \). 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 \( \text{Recursive-Solver}(I^K_{1,N}) \) 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 \( I^k_{i,j} \). It is easy to see that this number is

\[
O\left(\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k \in \mathbb{K}_{i,j}} |\Lambda^k_{i,j}| \right).
\]

(19)

By Proposition 1, (19) is equal to

\[
O\left(\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k \in \mathbb{K}_{i,j}} (j-i-2)(k-1) \right).
\]

By observing that \( |\mathbb{K}_{i,j}| \sim O(\min\{N-K,K\}) \), 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

\[
O\left(\max\{\min\{N-K,K\}^2 N^3, \min\{N-K,K\}^3 N^2\} \right).
\]

(20)

Because \( K \leq N \), (20) further reduces to

\[
O(1) = O(\min\{N-K,K\}^2 N^3)
\]

which concludes the proof. \( \square \)

D. Catanzaro et al.: Preprint submitted to Elsevier
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. [17]. 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 [4], middle frame selection, and visual features extraction with an Inception-v3 neural network [21] 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 Subsection 4.1, we discuss the details of the algorithms’ implementations. In Subsection 4.2, we describe the datasets included in the set of instances used for the experimental tests. Finally, in Subsection 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 gigabytes of RAM. We implemented Rotman et al.’s algorithm accordingly to the information provided in [17]. 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 $C^k_i(e)$, $X^k_i(e)$, $A^k_i(e)$, and $T^k_i(e)$ 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 $K \approx N / 2$, 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 $q_{i,j}^k$, $r_{i,j}^l$, and $\tilde{P}_{i,j}^k$ allow to set the space complexity of Recursive-Solver to $O(KN^2)$. Rotman et al.’s algorithm implements $C^k_i(e)$, $X^k_i(e)$, and $A^k_i(e)$ 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 Subsection 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 $P \in P$ that minimizes the additive cost function

$$H(P) = \sum_{\sigma \in P} a(\sigma),$$

(21)
D. Catanzaro et al.: Preprint submitted to Elsevier

A New Heuristic for the Automatic Scene Detection Problem

proposed by Rotman et al. [15]. 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 \( \zeta(P) \). 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 \( \phi^h_{i,v} \) and \( \phi^{k-h}_{v+1,j} \) with \( \phi' \) 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. [17, 18], 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. [2] to evaluate their scene detection algorithm. The BBC dataset, introduced by Baraldi et al. [3], 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 subset 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/hai/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 Figure 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 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 Subsection 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 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) [20] 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
Figure 4: Box-and-whiskers plot of the CPU times (expressed in seconds) obtained by HP, RT, and RH when solving the instances $I_1$.

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 [17]. 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 [17]. 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.

Acknowledgment
The first author acknowledge support from the Université Catholique de Louvain via the Fonds Spéciaux de Recherche (FSR) 2017-2021 and the Fondation Louvain 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.
A New Heuristic for the Automatic Scene Detection Problem

<table>
<thead>
<tr>
<th>Instance</th>
<th>N</th>
<th>K</th>
<th>Time (sec)</th>
<th>Δ_{HP,R}</th>
<th>Cost function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000 Days (cut)</td>
<td>204</td>
<td>15</td>
<td>224.4706</td>
<td>4582.7344</td>
<td>5194.4145</td>
</tr>
<tr>
<td>Big Buck Bunny</td>
<td>146</td>
<td>15</td>
<td>75.2031</td>
<td>1166.9219</td>
<td>1017.8210</td>
</tr>
<tr>
<td>Boy Who Never Slept (cut)</td>
<td>193</td>
<td>15</td>
<td>188.1875</td>
<td>3659.8351</td>
<td>3978.9020</td>
</tr>
<tr>
<td>CH7 (cut)</td>
<td>276</td>
<td>10</td>
<td>237.4844</td>
<td>9382.3037</td>
<td>11597.1574</td>
</tr>
<tr>
<td>Cosmos Laundromat - First Cycle</td>
<td>113</td>
<td>7</td>
<td>5.6406</td>
<td>151.6250</td>
<td>69.5630</td>
</tr>
<tr>
<td>Elephants Dream</td>
<td>142</td>
<td>9</td>
<td>22.0000</td>
<td>557.1406</td>
<td>393.8910</td>
</tr>
<tr>
<td>Fires Beneath Water (cut)</td>
<td>143</td>
<td>15</td>
<td>71.6719</td>
<td>1078.0625</td>
<td>915.5120</td>
</tr>
<tr>
<td>Honey (cut)</td>
<td>303</td>
<td>10</td>
<td>306.0000</td>
<td>13718.0156</td>
<td>18240.5745</td>
</tr>
<tr>
<td>Jathia’s Wager</td>
<td>181</td>
<td>15</td>
<td>150.9686</td>
<td>2819.5215</td>
<td>2931.1050</td>
</tr>
<tr>
<td>La Chute d’une Plume</td>
<td>83</td>
<td>11</td>
<td>6.3750</td>
<td>81.7500</td>
<td>44.2190</td>
</tr>
<tr>
<td>Lord Mea (cut)</td>
<td>103</td>
<td>15</td>
<td>23.9687</td>
<td>281.1562</td>
<td>183.5780</td>
</tr>
<tr>
<td>Meridian</td>
<td>66</td>
<td>9</td>
<td>2.0000</td>
<td>25.1250</td>
<td>10.8210</td>
</tr>
<tr>
<td>Oceania (cut)</td>
<td>114</td>
<td>20</td>
<td>60.3438</td>
<td>586.1250</td>
<td>381.3300</td>
</tr>
<tr>
<td>Pentagon (cut)</td>
<td>263</td>
<td>20</td>
<td>903.0781</td>
<td>17905.1563</td>
<td>26797.1501</td>
</tr>
<tr>
<td>Route 66 (cut)</td>
<td>279</td>
<td>15</td>
<td>593.2500</td>
<td>16456.9688</td>
<td>25337.4079</td>
</tr>
<tr>
<td>Seven Dead Men (cut)</td>
<td>111</td>
<td>20</td>
<td>53.8906</td>
<td>524.6663</td>
<td>336.2300</td>
</tr>
<tr>
<td>Sintel</td>
<td>154</td>
<td>8</td>
<td>21.4687</td>
<td>645.2187</td>
<td>443.4400</td>
</tr>
<tr>
<td>Sita Sings the Blues (cut)</td>
<td>275</td>
<td>10</td>
<td>229.3125</td>
<td>9284.6361</td>
<td>11842.6030</td>
</tr>
<tr>
<td>Star Wreck (cut)</td>
<td>237</td>
<td>15</td>
<td>345.0781</td>
<td>8045.5689</td>
<td>11263.1990</td>
</tr>
<tr>
<td>Tears of Steel</td>
<td>158</td>
<td>11</td>
<td>51.4219</td>
<td>19198.9734</td>
<td>28776.2360</td>
</tr>
<tr>
<td>Valkaama (cut)</td>
<td>275</td>
<td>18</td>
<td>828.1875</td>
<td>19198.9734</td>
<td>28776.2360</td>
</tr>
<tr>
<td>Ra101</td>
<td>72</td>
<td>12</td>
<td>2.2500</td>
<td>21.9467</td>
<td>80.4694</td>
</tr>
<tr>
<td>Ra102</td>
<td>57</td>
<td>12</td>
<td>2.2500</td>
<td>21.9467</td>
<td>80.4694</td>
</tr>
<tr>
<td>Ra103</td>
<td>107</td>
<td>16</td>
<td>31.3906</td>
<td>359.2500</td>
<td>246.0781</td>
</tr>
<tr>
<td>Ra104</td>
<td>143</td>
<td>22</td>
<td>153.3125</td>
<td>1654.1719</td>
<td>1354.4062</td>
</tr>
<tr>
<td>Ra105</td>
<td>66</td>
<td>13</td>
<td>4.2344</td>
<td>39.7500</td>
<td>18.0781</td>
</tr>
<tr>
<td>Ra106</td>
<td>59</td>
<td>5</td>
<td>0.2969</td>
<td>6.6719</td>
<td>1.0000</td>
</tr>
<tr>
<td>Ra107</td>
<td>116</td>
<td>9</td>
<td>11.6250</td>
<td>247.0313</td>
<td>171.0781</td>
</tr>
<tr>
<td>Ra108</td>
<td>205</td>
<td>12</td>
<td>137.5938</td>
<td>3632.6094</td>
<td>3979.9375</td>
</tr>
<tr>
<td>Ra109</td>
<td>106</td>
<td>14</td>
<td>22.9375</td>
<td>300.0469</td>
<td>206.0312</td>
</tr>
<tr>
<td>Ra110</td>
<td>100</td>
<td>16</td>
<td>24.9388</td>
<td>272.6904</td>
<td>174.6875</td>
</tr>
<tr>
<td>RCC08</td>
<td>242</td>
<td>29</td>
<td>1412.6890</td>
<td>19016.8960</td>
<td>24232.4531</td>
</tr>
</tbody>
</table>

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

References

A New Heuristic for the Automatic Scene Detection Problem

<table>
<thead>
<tr>
<th>Instance</th>
<th>N</th>
<th>K</th>
<th>Time (sec)</th>
<th>Cost function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000 Days</td>
<td>329</td>
<td>22</td>
<td>2190.4531</td>
<td>15.7323</td>
</tr>
<tr>
<td>Fires Beneath Water</td>
<td>358</td>
<td>62</td>
<td>19860.2090</td>
<td>15.4998</td>
</tr>
<tr>
<td>Honey</td>
<td>371</td>
<td>20</td>
<td>2620.5625</td>
<td>16.1890</td>
</tr>
<tr>
<td>Lord Meia</td>
<td>325</td>
<td>27</td>
<td>3166.1719</td>
<td>15.5539</td>
</tr>
<tr>
<td>Oceania</td>
<td>279</td>
<td>31</td>
<td>2545.3438</td>
<td>13.8514</td>
</tr>
<tr>
<td>Pentagon</td>
<td>333</td>
<td>31</td>
<td>4544.3281</td>
<td>15.3077</td>
</tr>
<tr>
<td>Seven Dead Men</td>
<td>164</td>
<td>34</td>
<td>507.8125</td>
<td>11.9474</td>
</tr>
<tr>
<td>BBC01</td>
<td>458</td>
<td>23</td>
<td>6742.3125</td>
<td>17.3984</td>
</tr>
<tr>
<td>BBC02</td>
<td>382</td>
<td>36</td>
<td>9232.1094</td>
<td>16.3077</td>
</tr>
<tr>
<td>BBC03</td>
<td>412</td>
<td>33</td>
<td>10051.9688</td>
<td>15.9327</td>
</tr>
<tr>
<td>BBC04</td>
<td>461</td>
<td>30</td>
<td>11854.6105</td>
<td>16.2738</td>
</tr>
<tr>
<td>BBC05</td>
<td>436</td>
<td>25</td>
<td>6903.8490</td>
<td>16.4621</td>
</tr>
<tr>
<td>BBC06</td>
<td>506</td>
<td>33</td>
<td>19177.9616</td>
<td>17.4224</td>
</tr>
<tr>
<td>BBC07</td>
<td>549</td>
<td>37</td>
<td>31293.5912</td>
<td>16.3247</td>
</tr>
<tr>
<td>BBC09</td>
<td>348</td>
<td>33</td>
<td>5808.5000</td>
<td>15.1449</td>
</tr>
<tr>
<td>BBC10</td>
<td>346</td>
<td>22</td>
<td>2565.9531</td>
<td>17.2508</td>
</tr>
<tr>
<td>BBC11</td>
<td>509</td>
<td>26</td>
<td>12312.1875</td>
<td>17.1233</td>
</tr>
</tbody>
</table>

Table 2
Results obtained by HP and AHP when solving the instances in $I_2$. for movie scene detection. IEEE Transactions on Cybernetics 48, 836–847.

<table>
<thead>
<tr>
<th>Instance</th>
<th>HP</th>
<th>RT/RH</th>
<th>AHP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000 Days</td>
<td>0.3860</td>
<td>-</td>
<td>0.4742</td>
</tr>
<tr>
<td>1000 Days (cut)</td>
<td>0.3627</td>
<td>0.3529</td>
<td>0.4069</td>
</tr>
<tr>
<td>Big Buck Bunny</td>
<td>0.4110</td>
<td>0.3973</td>
<td>0.4247</td>
</tr>
<tr>
<td>BBC01</td>
<td>0.3777</td>
<td>-</td>
<td>0.4105</td>
</tr>
<tr>
<td>BBC02</td>
<td>0.4319</td>
<td>-</td>
<td>0.4084</td>
</tr>
<tr>
<td>BBC03</td>
<td>0.3495</td>
<td>-</td>
<td>0.3738</td>
</tr>
<tr>
<td>BBC04</td>
<td>0.4534</td>
<td>-</td>
<td>0.4230</td>
</tr>
<tr>
<td>BBC05</td>
<td>0.4726</td>
<td>-</td>
<td>0.4656</td>
</tr>
<tr>
<td>BBC06</td>
<td>0.4506</td>
<td>-</td>
<td>0.4269</td>
</tr>
<tr>
<td>BBC07</td>
<td>0.3752</td>
<td>-</td>
<td>0.4463</td>
</tr>
<tr>
<td>BBC08</td>
<td>0.4380</td>
<td>0.4339</td>
<td>0.4421</td>
</tr>
<tr>
<td>BBC09</td>
<td>0.3994</td>
<td>-</td>
<td>0.4511</td>
</tr>
<tr>
<td>BBC10</td>
<td>0.4017</td>
<td>-</td>
<td>0.4682</td>
</tr>
<tr>
<td>BBC11</td>
<td>0.3831</td>
<td>-</td>
<td>0.4676</td>
</tr>
<tr>
<td>Boy Who Never Slept</td>
<td>0.3973</td>
<td>-</td>
<td>0.4587</td>
</tr>
<tr>
<td>Boy Who Never Slept (cut)</td>
<td>0.2953</td>
<td>0.4249</td>
<td>0.4508</td>
</tr>
<tr>
<td>CH7 (cut)</td>
<td>0.1522</td>
<td>0.1775</td>
<td>0.1993</td>
</tr>
<tr>
<td>Cosmos Laundromat - First Cycle</td>
<td>0.3540</td>
<td>0.4159</td>
<td>0.5044</td>
</tr>
<tr>
<td>Elephants Dream</td>
<td>0.3592</td>
<td>0.3732</td>
<td>0.4225</td>
</tr>
<tr>
<td>Fires Beneath Water</td>
<td>0.4106</td>
<td>-</td>
<td>0.4246</td>
</tr>
<tr>
<td>Fires Beneath Water (cut)</td>
<td>0.3916</td>
<td>0.4196</td>
<td>0.4196</td>
</tr>
<tr>
<td>Honey</td>
<td>0.4378</td>
<td>-</td>
<td>0.6270</td>
</tr>
<tr>
<td>Honey (cut)</td>
<td>0.2838</td>
<td>0.3465</td>
<td>0.4158</td>
</tr>
<tr>
<td>Jathia’s Wager</td>
<td>0.3757</td>
<td>0.3812</td>
<td>0.4254</td>
</tr>
<tr>
<td>La Chute d’une Plume</td>
<td>0.5663</td>
<td>0.6386</td>
<td>0.6627</td>
</tr>
<tr>
<td>Lord Meia</td>
<td>0.4031</td>
<td>-</td>
<td>0.5231</td>
</tr>
<tr>
<td>Lord Meia (cut)</td>
<td>0.4757</td>
<td>0.5243</td>
<td>0.5728</td>
</tr>
<tr>
<td>Meridian</td>
<td>0.4242</td>
<td>0.5000</td>
<td>0.5303</td>
</tr>
<tr>
<td>Oceania</td>
<td>0.3692</td>
<td>-</td>
<td>0.4516</td>
</tr>
<tr>
<td>Oceania (cut)</td>
<td>0.2807</td>
<td>0.3421</td>
<td>0.3772</td>
</tr>
<tr>
<td>Pentagon</td>
<td>0.3964</td>
<td>-</td>
<td>0.4865</td>
</tr>
<tr>
<td>Pentagon (cut)</td>
<td>0.4259</td>
<td>0.4867</td>
<td>0.5285</td>
</tr>
<tr>
<td>Rai01</td>
<td>0.4825</td>
<td>0.4912</td>
<td>0.4474</td>
</tr>
<tr>
<td>Rai02</td>
<td>0.5000</td>
<td>0.4828</td>
<td>0.4828</td>
</tr>
<tr>
<td>Rai03</td>
<td>0.1869</td>
<td>0.1869</td>
<td>0.2617</td>
</tr>
<tr>
<td>Rai04</td>
<td>0.5503</td>
<td>0.6040</td>
<td>0.6040</td>
</tr>
<tr>
<td>Rai05</td>
<td>0.7714</td>
<td>0.7143</td>
<td>0.7714</td>
</tr>
<tr>
<td>Rai06</td>
<td>0.1186</td>
<td>0.1864</td>
<td>0.3051</td>
</tr>
<tr>
<td>Rai07</td>
<td>0.3621</td>
<td>0.3966</td>
<td>0.3707</td>
</tr>
<tr>
<td>Rai08</td>
<td>0.3659</td>
<td>0.3415</td>
<td>0.3951</td>
</tr>
<tr>
<td>Rai09</td>
<td>0.2736</td>
<td>0.2453</td>
<td>0.1981</td>
</tr>
<tr>
<td>Rai10</td>
<td>0.2745</td>
<td>0.3333</td>
<td>0.5784</td>
</tr>
<tr>
<td>Route 66 (cut)</td>
<td>0.3692</td>
<td>0.4444</td>
<td>0.4731</td>
</tr>
<tr>
<td>Seven Dead Men</td>
<td>0.3598</td>
<td>-</td>
<td>0.4024</td>
</tr>
<tr>
<td>Seven Dead Men (cut)</td>
<td>0.2883</td>
<td>0.3423</td>
<td>0.3604</td>
</tr>
<tr>
<td>Sintel</td>
<td>0.4156</td>
<td>0.4351</td>
<td>0.4675</td>
</tr>
<tr>
<td>Sita Sings the Blues (cut)</td>
<td>0.2836</td>
<td>0.2800</td>
<td>0.3055</td>
</tr>
<tr>
<td>Star Wreck (cut)</td>
<td>0.3882</td>
<td>0.3966</td>
<td>0.4430</td>
</tr>
<tr>
<td>Tears of Steel</td>
<td>0.4051</td>
<td>0.4620</td>
<td>0.4747</td>
</tr>
<tr>
<td>Valkaama (cut)</td>
<td>0.1600</td>
<td>0.2400</td>
<td>0.2727</td>
</tr>
</tbody>
</table>

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

D. Catanzaro et al.: Preprint submitted to Elsevier  Page 19 of 18