Electronic Research Archive

http://www.aimspress.com/journal/era

ERA, 31(10): 5976–5995.
DOI: 10.3934/era.2023304
Received: 02 June 2023
Revised: 15 August 2023
Accepted: 22 August 2023
Published: 05 September 2023

Research article

Multi-features combinatorial optimization for keyframe extraction

Lei Ma[1,2], Weiyu Wang[1,2], Yaozong Zhang[1,2,*], Yu Shi[1,2], Zhenghua Huang[1,2] and Hanyu Hong[1,2]

[1] School of Electrical and Information Engineering, Wuhan Institute of Technology, Wuhan 430205, China

[2] Hubei Key Laboratory of Optical Information and Pattern Recognition, Wuhan 430205, China

* **Correspondence:** Email: zhangyaozong@wit.edu.cn.

Abstract: Recent advancements in network and multimedia technologies have facilitated the distribution and sharing of digital videos over the Internet. These long videos contain very complex contents. Additionally, it is very challenging to use as few frames as possible to cover the video contents without missing too much information. There are at least two ways to describe these complex videos contents with minimal frames: the keyframes extracted from the video or the video summary. The former lays stress on covering the whole video contents as much as possible. The latter emphasizes covering the video contents of interest. As a consequence, keyframes are widely used in many areas such as video segmentation and object tracking. In this paper, we propose a keyframe extraction method based on multiple features via a novel combinatorial optimization algorithm. The key frame extraction is modeled as a combinatorial optimization problem. A fast dynamic programming algorithm based on a forward non-overlapping transfer matrix in polynomial time and a 0-1 integer linear programming algorithm based on an overlapping matrix is proposed to solve our maximization problem. In order to quantitatively evaluate our approach, a long video dataset named 'Animal world' is self-constructed, and the segmentation evaluation criterions are introduced. A good result is achieved on 'Animal world' dataset and a public available Keyframe-Sydney KFSYD dataset [1].

Keywords: key frame extraction; combinatorial optimization; multi-features guidance

1. Introduction

Keyframe extraction algorithms select the most representative frames with a minimal level of redundancy from the videos [1]. Keyframe extraction plays an important role in several broad areas of video processing research such as real-time camera tracking [2], human shape and pose tracking [3], video annotation [4], video summarization [5–10], creating chapter titles in DVDs, video editing, 3D model reconstruction [11], human action recognition [12], and video retrieval and browsing [13]. The

key challenges in keyframe extraction are twofold: *i*) covering the representative content of the video and *ii*) reducing redundancies between neighboring keyframes. Recent deep learning-based methods mainly extract relevant semantic features [14–16], while ignoring the whole image content, which is a key factor for keyframe extraction.

Keyframe extraction has been tackled from various perspectives [17–19]. Below, we review the most representative works in three common approaches: *i*) shot detection or video segmentation, *ii*) clustering methods, and *iii*) the optimization problem, and differentiate our work from these previous works.

i) Shot detection or video segmentation: A shot is defined as an unbroken sequence of frames recorded from a single camera, which forms the building block of a video. During the shot-based keyframe extraction, the shot boundaries of the original video are first detected, and then one or more keyframes are extracted from each shot. The purpose of shot boundaries detection is to segment the video stream into multiple shots [20, 21]. For example, a pixel-wise difference based metric with a threshold is used to detect the shot boundaries of the video presented in [20]. [21] uses *k-means* clustering on a two dimensional feature extracted from both histogram and spatial difference metrics, followed by a heuristic elimination process to detect the shot boundaries. After the shots are segmented, keyframes can be extracted from each shot based on it's contents [1, 22]. A segment is defined as homogenous sequence of frames in the visual content domain. During the segment-based keyframe extraction approaches, a video is segmented into higher-level video components, where each component could be either a scene, an event, a set of shots, or even the entire video sequence. Then, each segment can be described by one or more keyframe(s) [23]. Han et al. [8] found key segments by agglomerative clustering followed by a variant of the 0-1 knapsack problem. Omidyeganeh et al. [24] employed generalized Gaussian density parameters of wavelet transform subbands along with the Kullback-Leibler distance measurement to address the keyframe extraction problem from each video segment. The performance of the keyframe extraction based on either the shot detection or the video segmentation approach relies on the accuracy of either shot detection or video segmentation.

ii) Clustering methods: The clustering methods [25–28] take all the frames of a shot together and classify them according to their content similarity. Then, the keyframes are determined as the representative frames of a cluster. A similarity-driven cluster merging method with a threshold parameter which controls the density of classification has been achieved in keyframe extraction [25]. In the approach presented in [26], the authors applied multiple partitional clustering to the whole video sequence in order to remove the visual-content redundancy among video frames. The keyframes were selected as centroids of the optimal clusters obtained by an unsupervised procedure based on a cluster-validity analysis. In [27], a hierarchical clustering was introduced when clustering the video, and the temporal constraints was used to filter out unsuitable clusters. A representative frame was selected from each cluster. Authors in [28] proposed a spectral clustering method for extracting keyframes based on a sparse representation. The disadvantage of clustering methods is that the temporal information of a video sequence is omitted. An inherent problem in this approach is the selection of appropriate thresholds. Although adaptive clustering methods can manipulate the threshold to produce a pre-designed number of keyframes, this iterative searching process makes these computational methods expensive. Compared to both the video segmentation and the shot detection approach, the above approach requires a proper visual content analysis and the selection of appropriate features. On the other hand, our method exploits the combinations of multiple clustering features without regard to the video contents, and can thus be easily applied to any complex video content. Many times, clustering can supply different contents of

video segments. The longest non-overlapping segments can cover the contents of the video as much as possible. In addition, a combinatorial optimization problem without thresholds or a pre-designed number of keyframes was solved by either a quick dynamic programming [29] or a 0–1 integer linear programming approach [30].

iii) Optimization problem: The one main optimization problem is the set cover problem. Set cover problem: Given a universe $U = \{e_1, e_2, ..., e_n\}$ of n elements, a collection of subsets of U, $\mathcal{S} = \{S_1, S_2, ..., S_k\}$, and a cost function $c : \mathcal{S} \longrightarrow \mathbf{Q}^+$, and find a minimum cost subcollection of $\mathcal{S}$ that covers all elements of U.

The minimum set cover problem can be formulated as the following integer linear program:

$$\begin{aligned} &\min c = \mathbf{w}^T\mathbf{y} \\ &s.t.\ \mathbf{y}^T\mathbf{1}\{e_i \in \mathcal{S}\} \geq 1, \forall i \in \{1, 2, ..., n\}, \\ &\mathbf{y} \in \{0, 1\}^k. \end{aligned} \tag{1.1}$$

where $\mathbf{w}$ is the weight vector and $\mathbf{y}$ is a binary vector. $y_i = 1$ denotes that S_i is selected and 0 is otherwise. $\mathbf{1}\{e_i \in \mathcal{S}\}$ is a indicator vector of k dimensions. If e_i is in S_j, then the j^{th} element in $\mathbf{1}\{e_i \in \mathcal{S}\}$ is 1 and 0 otherwise.

It is an NP-complete problem [31]. The set cover problem has been widely applied in saliency detection [32] and keyframe extraction [1, 33]. In [32], the social group candidates selection was formulated as the set cover problem, which was represented as the form of a quadratic integer programming, and the optimization was solved with a branch and bound method. In [1], the global keypoint pool is formed by matching keypoints, and the keypoints of the keyframes should cover the global keypoint pool as much as possible. It was formulated as a variation of the set cover problem, and a greedy algorithm was adopted to approximately tackle this issue. The set cover problem was also employed in [33], in which the suboptimal solution for the minimum covering problem was found using parts of the Quine-McCluskey algorithm. The disadvantage of this approach is obvious, since it can not obtain the optimal solution, but rather the suboptimal solution. The most closest to our approach is the combinatorial optimization problem. Combinatorial optimization is a topic that consists of finding an optimal object from a finite set of objects [34]. Chang et al. [33] constructed a tree-structured keyframe hierarchy in which a video segment for a given degree of fidelity was represented with a compact set of keyframes, and used a depth-first search scheme with pruning to enable an efficient content-based retrieval. The energy minimization or maximization based methods [35] extract keyframes by solving a rate-constrained problem, [36] extracting a small number of keyframes within a shot by maximizing the divergence between video objects in a feature space. This approach comes with two advantages: it's very intuitive and it can reach an optimal solution. The main contributions can be summarized as follows:

- We propose a keyframe extraction method based on multiple features via a novel combinatorial optimization algorithm.
- The proposed method achieves a promising performance on the Animal-world and Keyframe-Sydney (KFSYD) datasets. Extensive qualitative and quantitative experiments demonstrate the proposed method's effectiveness.

The rest of the paper is organized as follows. Section 2 introduces the proposed method, including features extraction, candidate video segments generation and key segments selection. In Section 3, the optimization methods are presented, including the dynamic programming approach and a 0–1 integer

linear programming approach. In Section 4, some related analyses and discussions are presented. Experimental results are shown in Section 5 to verify the proposed approach. Finally, concluding remarks are given in Section 6.

2. Proposed method

Figure 1 shows the main framework of the proposed keyframe extraction algorithm. Our algorithm includes three stages, (i.e., features extraction, candidate video segments generation, and key segments selection). The details will be introduced as follows.

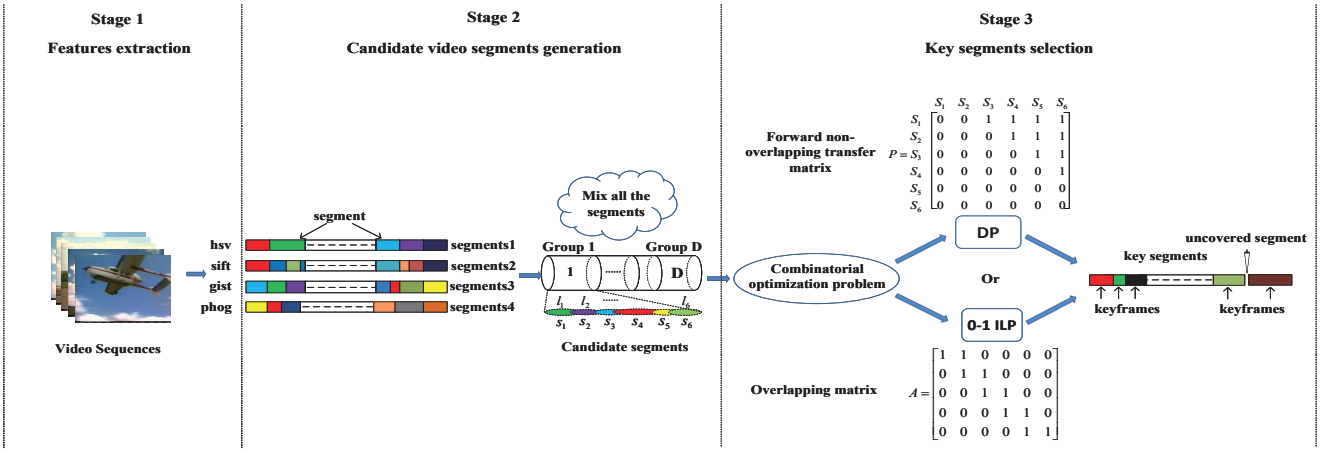


Figure 1. Overview of our approach on KFSYD dataset.

2.1. Features extraction

There is a strong relationship among neighbouring frames. Meanwhile, there may exist a blurring in the video caused by either the quick movement of objects in the shot or the jitter of the camera lens. This will interfere with the shot detection or the segmentation of video sequence. Illumination variation is another challenging point. Errors in key frames selection can arise by solely using the color or texture features to cluster the video. Intuitively, the more features there are in the video, the more detail will be presented. In our problem, we use the following features to cluster the video sequence: $f_1 = HSV\ histogram$, $f_2 = SIFT$ [37], $f_3 = GIST$ [38], and $f_4 = PHOG$ [39]. These features are usually used in keyframe extraction. These low-level visual features describe the image from various aspects such as the color, shape and context, which are in favour of accurately and comprehensively representing the video's contents.

2.2. Candidate video segments generation

The candidate video segments generation includes the following three steps, in the given order.

- Step 1: For each feature, the k-means clustering method is used to obtain the initial video segments by clustering the video sequence.
- Step 2: The initial video segments are further pruned to remove some invalid segments.
- Step 3: The ultimate candidate video segments are produced by mixing the video segments generated by all the features in Step 2.

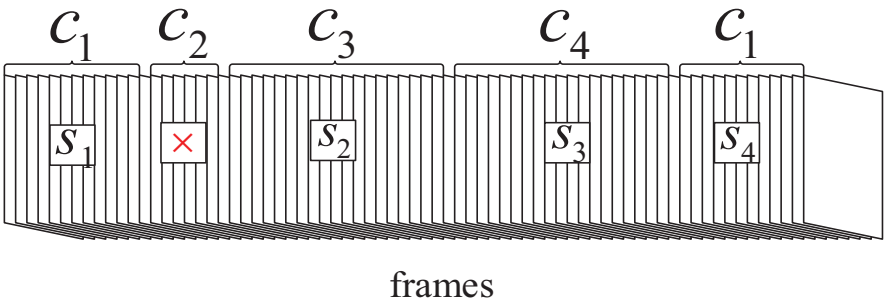


Figure 2. An illustration of generating candidate segments in Steps 1 and 2.

For example, as shown in Figure 2, the video sequence is clustered into four clusters with the *GIST* feature according to Step 1. We can obtain five initial video segments, where the first video segment and the last video segment belong to the same cluster c_1. In Step 2, the second video segment , which belongs to cluster c_2, is pruned, since the frame length is less than 10. Such a segment is unlikely to compose a complete shot. The left video segments are labeled as $s_1 \sim s_4$. In Step 3, we start to mix the segments set $\{s_n^v\}_{v=1,n=1}^{V,N_v}$, where V refers to the number of features, and N_v denotes the number of segments generated by v-th feature in Step 2. There may exist such a case that more than two segments in $\{s_n^v\}_{v=1,n=1}^{V,N_v}$ have the same beginning and ending times. Then, only one of these reduplicated segments will ultimately remained in the segments set by random.

The remaining video segments are called candidate segments. These candidate segments are sorted by the beginning time, followed by being divided into D groups that don't overlap in time. The whole process of generating candidate video segments is similar to the Stage 2 of Figure 1.

2.3. Key segments selection

The candidate segments in the d^{th} group are denoted as a set $S_d = \{s_j\}$, $j \in \{1 \dots n\}$, where n is the number of candidate segments in the group. The segments selected from the candidate segments are called key segments.

Combinatorial optimization problem: The target of our problem is to find the optimal subset S_d^* that covers all the video sequences as possible. S_d^* has the property that there is no overlap in time for any segment that are selected from the d^{th} group. S_d^* is the set of *key segments* that we seek. Then, the problem then becomes how to find the longest non-overlapping segments in each group. For the d^{th} group, it can be formulated as follows:

$$\begin{aligned} &\max_{k,q_i} \sum_{i=1}^{k} l_{q_i} \\ &s.t.\ l_{s_{q_i} \cap s_{q_j}} = 0,\ \forall i \neq j \in \{1, ..., k\},\ q_i, q_j \in \{1, ..., n\}, \\ &\quad s_{q_i} \in S_d,\ \forall i \in \{1, ..., k\},\ q_i \in \{1, ..., n\}. \end{aligned} \tag{2.1}$$

where n is the number of selected candidate segments in the d^{th} group and q_i is the index of q_i^{th} segment in this group. The i^{th} element l_i in column vector $\mathcal{L} = (l_1, \dots, l_n)$ denotes the number of frames of i^{th} candidate segment. The k denotes the number of *key segments* selected from the current group. The first constraint denotes that the selected segment s_{q_i} and the selected segment s_{q_j} don't overlap.

3. Our maximization algorithm

3.1. Definition of symbols

A forward non-overlapping transfer matrix is defined to represent a directed graph $G = (V, E)$ with a temporal non-overlapping relationship, as shown in Figure 3, where the segments correspond to the nodes $V = \{v_1, \cdots, v_6\}$, and a directed edge e_{ij} is connected between node i and j with non-overlapping segments, where the corresponding segment s_i is in front of segment s_j. The forward non-overlapping transfer matrix can be formulated in the $n \times n$ upper triangular matrix, where n is the number of nodes in the graph:

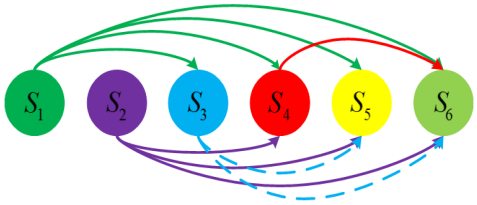


Figure 3. A directed graph with temporal non-overlapping relationship. The node in the graph denotes a video segment in one group, the edge in the graph denotes that the connected two video segments don't overlap.

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & & & \\ & 0 & 0 & P_{i,j} & \\ & & \ddots & \ddots & \\ & \mathbf{0} & & 0 & 0 \\ & & & & 0 \end{bmatrix}$$

where $P_{ij} = 1, i \neq j$ denotes that the i^{th} and j^{th} segment has no overlap, and 0 if j^{th} segment is in front of the i^{th} segment or there is no overlap between them. For any $i \in \{1, 2, ...n-1\}$, $P_{i,i+1} = 0$, (i.e., the elements in the diagonal right above the main diagonal are all zeros). It is because the i^{th} segment and the $i + 1^{th}$ segment have overlapped with each other; otherwise, these two segments belong to different groups. Additionally, the element P_{ij} in matrix $\mathbf{P}$ means that there exists a path of length 1 from the node i to the node j if $P_{ij} = 1$ and no path if $P_{ij} = 0$, according to [40]. In the same way, the element P^k_{ij} in the power matrix $\mathbf{P}^k$ means that there exist P^k_{ij} path(s) of length k from the node i to the node j. The transfer matrix serves as the building block throughout our following algorithm.

We define a selection matrix $\mathbf{A}^k$, where k is the number of key segments. The elements of $\mathbf{A}^k$ are the total lengths of the key segments. We set $\mathbf{P}^0 = \mathbf{I}$, where $\mathbf{I}$ is an identity matrix. An AND operator for non negative integer vectors is defined as $\mathbf{c} = \mathbf{a} \circ \mathbf{b}$, where $\mathbf{c} = logical(\mathbf{a})\&logical(\mathbf{b})$, $logical(\bullet)$ denotes that the vector is an element-wise binarization operation and & is a logical AND operator.

3.2. *Dynamic programming algorithm (DP)*

As mentioned previously, the candidate segments in each group can be represented as a graph. The difficulties in solving Eq (1.1) include the following: the variable is binary and the number of key segments is not given in advance and has to been optimized. To deal these problems, we propose the following dynamic programming based on the graph. For d^{th} subgraph, our dynamic programming algorithm is described as follows:

- Step 1: Construct a transfer matrix $\mathbf{P}$ and initialize a selection matrix $\mathbf{A}^k = \mathbf{l} \cdot \mathbf{1}^T, k = 1$, where $\mathbf{l}$ is a column vector of length.
- Step 2: Determine whether $\mathbf{P}^k$ is all zeros matrix. If not, determine which elements are activated in $\mathbf{A}^k(i,:)$ in the i^{th} row based on the activated indicator vector $\mathbf{a} = \mathbf{P}^{k-1}(i,:) \circ \mathbf{P}(:,j)$, where $\mathbf{P}^{k-1}(i,:), \mathbf{P}(:,j)$ denote the elements in the i^{th} row of $\mathbf{P}^{k-1}$ and in the j^{th} column of $\mathbf{P}$, respectively. Then, the maximum value $\mathbf{A}^k(i,r)$ in $\mathbf{A}^k(i,:)$ and the corresponding r^{th} column segment l_r are added to $\mathbf{A}^{k+1}(i,j)$. If so, go to Step 4.
- Step 3: $k = k + 1$, and go to Step 2.
- Step 4: Return the maximum value in $\{\mathbf{A}^1, \cdots, \mathbf{A}^k\}$ and its corresponding key segments.

Figure 4 displays a toy example and is a subgraph in graph G according to our dynamic programming. The forward non-overlapping transfer matrix and initial selection matrix are represented as follows:

$$\mathbf{P} = \begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{array} \begin{array}{c} \begin{array}{cccccc} s_1 & s_2 & s_3 & s_4 & s_5 & s_6 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

$$\mathbf{A}^1 = \begin{bmatrix} \boxed{l_1} & l_1 & l_1 & l_1 & l_1 & l_1 \\ l_2 & \boxed{l_2} & l_2 & l_2 & l_2 & l_2 \\ l_3 & l_3 & \boxed{l_3} & l_3 & l_3 & l_3 \\ l_4 & l_4 & l_4 & \boxed{l_4} & l_4 & l_4 \\ l_5 & l_5 & l_5 & l_5 & \boxed{l_5} & l_5 \\ l_6 & l_6 & l_6 & l_6 & l_6 & \boxed{l_6} \end{bmatrix}$$

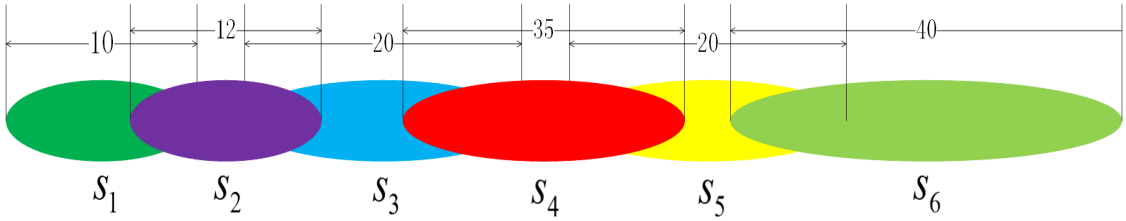


Figure 4. A toy candidate segments $s_1 \sim s_6$ in a group. The length of each segment is shown over the corresponding ellipse as above.

As mentioned earlier, the forward non-overlapping transfer matrix represents whether or not there exists a path of step length 1 from one node to subsequent node. In other words, $\mathbf{P}(i, j) = 1$ denotes that s_i and s_j do not overlap in time. The activated indicator vector $\mathbf{a} = \mathbf{P}^0(1, :) \circ \mathbf{P}(:, 3)$, $\mathbf{a} = [1\ 0\ 0\ 0\ 0\ 0]$ denotes that only one path $\mathbf{A}^1(1, 1)$ is activated. The segment s_3 is added to path $\mathbf{A}^2(1, 3) = \mathbf{A}^1(1, 1) + l_3$. In the same way, we can calculate $\mathbf{A}^2(i, j)$, where i, j are the index of the non zeros elements in $\mathbf{P}$. Since $\mathbf{A}^2$ and $\mathbf{P}$ have the same data structure, the other elements in $\mathbf{A}^2$ are all zeros. The elements with a box around them are activated as shown in $\mathbf{A}^1$ and $\mathbf{A}^2$:

$$\mathbf{A}^2 = \begin{bmatrix} 0 & 0 & \boxed{l_1 + l_3} & \boxed{l_1 + l_4} & l_1 + l_5 & l_1 + l_6 \\ 0 & 0 & 0 & \boxed{l_2 + l_4} & l_2 + l_5 & l_2 + l_6 \\ 0 & 0 & 0 & 0 & l_3 + l_5 & l_3 + l_6 \\ 0 & 0 & 0 & 0 & 0 & l_4 + l_6 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

where $\mathbf{A}^2$ is a upper triangular matrix, and $A^2_{i,j} \neq 0$ denotes that two candidate segments are selected.

$$\mathbf{P}^2 = \begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{array} \begin{array}{c} \begin{array}{cccccc} s_1 & s_2 & s_3 & s_4 & s_5 & s_6 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

As for $P^2_{1,6} = 2$, there exist two paths of step length 2, while only the longest one is selected and stored in $A^3_{1,6}$. The activation indicator vector $\mathbf{a} = \mathbf{P}(1, :) \circ \mathbf{P}(:, 6)$, $\mathbf{a} = [0\ 0\ 1\ 1\ 0\ 0]$ denotes that $A^2_{1,3}, A^2_{1,4}$ are both activated as in the first row of $\mathbf{A}^2$ with a box around them. The largest value between the activation paths $A^2_{1,3}, A^2_{1,4}$ is transferred to $A^3_{1,6}$. As $A^2_{1,3} = l_1 + l_3, A^2_{1,3} = 30, A^2_{1,4} = l_1 + l_4, A^2_{1,4} = 45$, $A^2_{1,4} > A^2_{1,3}$; then, the segment s_6 is added to path $A^2_{1,4}$, $A^3_{1,6} = A^2_{1,4} + l_6$. If $\mathbf{a} = [0\ 0\ 0\ 0\ 0\ 0]$, the corresponding $A^3_{i,j} = 0$. The whole $\mathbf{A}^3$ is shown as follows:

$$\mathbf{A}^3 = \begin{bmatrix} 0 & 0 & 0 & 0 & l_1 + l_3 + l_5 & l_1 + l_4 + l_6 \\ 0 & 0 & 0 & 0 & 0 & l_2 + l_4 + l_6 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

When $\mathbf{P}^3 = 0$, then $\mathbf{A}^4 = 0$. There is no path of step length 3. Then, the algorithm stops. The maximum value in $\{\mathbf{A}^1, \mathbf{A}^2, \mathbf{A}^3\}$ is $A^3_{2,6} = l_2 + l_4 + l_6$, and the corresponding key segments are s_2, s_4, s_6 in the subgraph.

Complexity analysis: Our algorithm is parallelizable and it's very easy to predict the iteration step k subjects to $\mathbf{P}^k = 0$ for each subgraph. In addition, the transfer matrix $\mathbf{P}$ is zero on the first diagonal

above the main diagonal and below it for a subgraph of any, which will facilitate the acceleration of calculating $\mathbf{P}^n, \{n = 2, 3, \ldots, k\}$. It is very easy to verify that $\mathbf{P}^n$ has all zeros on the $(2n-1)^{th}$ diagonal above and below the main diagonal. Since $\mathbf{A}^{n+1}$ and $\mathbf{P}^n$ have the same data structure, $\mathbf{A}^{n+1}$ can be calculated quickly. The computation complexity is $C(\frac{m^3}{6} - \frac{7m^2}{24} - \frac{m}{12})$ if m is even, or $C(\frac{m^3}{2} - \frac{m^2}{8} - \frac{m}{2} + \frac{1}{8})$ if m is odd, where m is the maximum number of nodes in all subgraphs. Once the longest key segments were found, the frames which are closest to the corresponding feature centroid of key segments are selected as keyframes.

3.3. 0–1 integer linear programming algorithm (0–1 ILP)

The pairwise overlapping relationship of segments are represented as an overlapping matrix $\mathbf{A} \in \{0, 1\}^{m \times n}$, where m is the number of pairwise overlapping segments, and n is the number of segments. If the i^{th} segment and the $j^{th}(j > i)$ problem overlap in time, then $\mathbf{A}_{i,i} = 1, \mathbf{A}_{i,j} = 1$. In addition, $\mathbf{A} \cdot \mathbf{1} = \mathbf{2}$, i.e., only one pairwise overlapping segments are presented in each row of matrix $\mathbf{A}$, where $\mathbf{1}$ is a column vector of all ones, and $\mathbf{2}$ are a column vector of all twos. For example, take the toy game in Figure 2. The overlapping matrix can be represented as follows:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

We define a selection indicator vector $\mathbf{y} \in \{0, 1\}^n$, i.e., $y_i = 1$ denotes that the i^{th} segment is selected, and 0 otherwise. Equation (1.1) can be represented as a form of 0-1 integer linear programming.

$$\begin{aligned} &\max \mathcal{L}^T \mathbf{y} \\ &s.t.\ \mathbf{A}\mathbf{y} \leq \mathbf{1}, \\ &\mathbf{y} \in \{0, 1\}^6 \end{aligned} \tag{3.1}$$

The inequality constraint means that, at most, one segment can be selected from the pairwise overlapping segments. The embedded matlab function *bintprog* is used to optimize the aforementioned problem. Once the optimal key segments are found, the frames which are closest to the corresponding feature centroids of key segments are selected as the keyframes.

4. Analyses and discussions

4.1. Relation to set cover problem and shortest path problem

The main differences between our problem and the set cover problem, as well as the shortest path problem, are described as follows.

There are four main differences between our combinational problem and the set cover problem: 1) the subsets in $\mathcal{S}$ have no order in the set cover problem, but they are sorted in time order in our problem; 2) All the elements in the universe should be covered in set cover problem, but they should be covered as much as possible in our problem; 3) The subsets should not overlap with each other in our problem, but

it has no such constraint in the set cover problem; and 4) The set cover problem is a minimal problem, but ours is a maximal problem.

Shortest path problem: In graph theory, the shortest path problem is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized. The main difference between our problem and the shortest path problem is that the start node and the end node are not given in our problem, which should be inferred by our proposed combinatorial optimization.

4.2. The comparisons between DP and 0–1 ILP

The 0–1 inter linear program (ILP) approach is very time consuming when the dimensions of the constraint matrix are very large, but it is very easy to implement. It takes about 10 minutes to select key segments from one groups candidate segments on the Animal-world dataset with the 0–1 ILP approach, while the DP approach take less than 20 s to extract the same length of key segments. To obtain the longest key segments, the combination of key segments is not unique. The dynamic program (DP) approach searches optimal key segments from a smaller combination of candidate segments to a larger combination of candidate segments. In this process, it selects the minimal combination of candidate segments but with the longest sequences coverage. However, the 0–1 ILP approach selects more key segments than the dynamic approach with the same length of sequence coverage. The average gap between neighboring keyframes extracted by the DP approach is larger than the 0–1 ILP approach. The larger average gap between neighboring keyframes can reduce the redundancies between neighboring keyframes. The longer key segments can cover the representative content of the video as much as possible. In the experimental part, the dynamic approach is utilized to extract the key segments.

4.3. Specify the number of keyframes

When the number of keyframes is specified as k, all of the video segments will serve as a whole. In our DP approach or the 0–1 ILP approach, the forward non-overlapping transfer matrix will include the whole video segments. For the DP approach, the element $P_{i,i+1} = 1$, if i^{th} and $i + 1^{th}$ segments are in different groups. The selection matrix $\mathbf{A}^k$ can be obtained after running the dynamic programming algorithm. The combinatorial segments, which correspond to the maximum value in $\mathbf{A}^k$, are the desired key segments. For the 0–1 ILP approach, a similar overlapping matrix is constructed. The new constraint $\mathbf{y}^T\mathbf{1} = k$ means that the k segments are selected as key segments.

5. Experiment

To evaluate the performance of the proposed keyframes extraction algorithm, two types of experiments were performed. The first one compared the performance of video segmentation by giving the groundtruth of keyframes on the Animal-world dataset. The second one was a quantitative experiment evaluated on the KFSYD dataset [1].

5.1. Databases and experimental settings

Databases: 1) The Animal-world dataset is a long video sequence of 8000 frames constructed by ourselves. The resolution of the frames is 352×288. It contains six different types of objects including

leopard, rhinoceros, hyena, zebra, elephant, and human. We chose 36 shots, which included 4213 frames of the Animal-world dataset to evaluate our experiment. The same object class in the same frames were given different labels according to the order of their emergence. The groundtruth were annotated frame by frame, which took almost two months to complete this task. There existed frequent shot changes, visual transitions (such as dissolves), occlusions and complex scenes. 2) The KFSYD dataset [1] was constructed from the open video project (http://www.open-video.org) for quantitative evaluation. It consists of 10 video shots across several genres. The resolution of frames is 352×240. The ground-truth keyframes of the videos were manually selected by three students with video processing backgrounds. The main differences between these two datasets are listed in Table 1.

Table 1. The comparisons between Animal-world and KFSYD dataset.

	# of Videos	# of frames	# of shots per video	given ground truth keyframes	evaluation
Animal-world	1	4213	36	no	The quality of video segmentation
KFSYD	10	95 ~ 352	1	yes	The quality of keyframes extraction

Experimental settings: For feature $f1$, 16 histogram bins were used to represent the features in each color channel. For feature $f2$, SIFT features were computed on a uniform grid and a k-means cluster method was used to obtain 300 centroids for the bag-of-words [41] representation. The whole video sequence with different features were all clustered into C clusters by k-means clustering. In our experiment, the HSV (Hue, Saturation, Value) histogram had 32 bins, and the SIFT (scale-invariant feature transform) descriptor [37], which was quantized into visual words had 300 dimensions, the GIST [38] had 512 dimensions, and the PHOG (pyramid histogram of oriented gradients) [39] had 628 dimensions. All the experiments were conducted on an Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz and implemented with the MATLAB programming software.

5.2. Comparison of results from video segmentation on Animal-world dataset

By giving the groundtruth of keyframes extracted by [42, 43], we used the publicly available code [42]* to segment the whole video. The algorithm can detect the shot change and determine the pixel-level label of each frame by either transferring the groundtruth of the nearest left keyframe or the nearest right keyframe based on propagation of errors.

In our approach, the k-means parameter C was experimentally set to 50 for each feature. The ultimate 52 key segments produced by our dynamic programming approach covered 4197 frames. The mean keyframe interval between neighbourhood keyframes was 84.1. For keyframe extraction approach in [42], a cost matrix was calculated with accumulative errors from the pixel flow propagation. A dynamic programming algorithm was used to extract the keyframes based on the cost matrix. The number of keyframes was set to 52. The Markov Random Field (MRF) model was utilized to achieve the video segmentation. For the keyframe extraction approach in [43], the keyframe extraction was formulated as a conditional k keyframe selection problem. A dynamic programming algorithm was proposed to search for the optimal k keyframes. The number of keyframes was also set to 52. By giving

*http://vision.cs.utexas.edu/projects/active_frame_selection/

the groundtruth of keyframes extracted by three different algorithms, a similar MRF-based approach was used to segment the video.

The comparison between an algorithm's output and the groundtruth was performed on four evaluation metrics (i.e., average precision (AP), average recall (AR), average F-score (AF) and average intersection-over-union (AIoU)).

$$\begin{aligned} AP &= \frac{1}{N}\sum_{i=1}^{N}\frac{1}{n_i}\sum_{k=1}^{n_i}\sum_{\hat{k}=1}^{m_i} Pre_i^{k\hat{k}}\sigma(y_i^k = g_i^{\hat{k}}), \\ Pre_i^{k\hat{k}} &= \frac{|R_i^k \cap G_i^{\hat{k}}|}{|R_i^k|} \end{aligned} \tag{5.1}$$

$$\begin{aligned} AR &= \frac{1}{N}\sum_{i=1}^{N}\frac{1}{m_i}\sum_{k=1}^{n_i}\sum_{\hat{k}=1}^{m_i} Rec_i^{k\hat{k}}\sigma(y_i^j = g_i^{\hat{k}}), \\ Rec_i^{k\hat{k}} &= \frac{|R_i^k \cap G_i^{\hat{k}}|}{|G_i^{\hat{k}}|} \end{aligned} \tag{5.2}$$

$$\begin{aligned} AF &= \frac{1}{N}\sum_{i=1}^{N}\frac{1}{N_i}\sum_{k=1}^{n_i}\sum_{\hat{k}=1}^{m_i}\frac{1.3Pre_i^{k\hat{k}} \times Rec_i^{k\hat{k}}}{0.3Pre_i^{k\hat{k}} + Rec_i^{k\hat{k}}}\sigma(y_i^k = g_i^{\hat{k}}), \\ N_i &= n_i + m_i - \sum_{k=1}^{n_i}\sum_{\hat{k}=1}^{m_i}\sigma(y_i^k = g_i^{\hat{k}}) \end{aligned} \tag{5.3}$$

$$AIoU = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{N_i}\sum_{k=1}^{n_i}\sum_{\hat{k}=1}^{m_i}\frac{|R_i^k \cap G_i^{\hat{k}}|}{|R_i^k \cup G_i^{\hat{k}}|}\sigma(y_i^k = g_i^{\hat{k}}) \tag{5.4}$$

where R_i^k is the k^{th} region of semantic segmentation in the i^{th} frame and y_i^k is the label of it, $G_i^{\hat{k}}$ is the $\hat{k}^{th}$ groundtruth semantic region of the i^{th} frame and $g_i^{\hat{k}}$ is the groundtruth label of it, n_i is the number of segmentation objects including background in the i^{th} frame, m_i is the number of groundtruth objects including background in the i^{th} frame and σ is a indicator function.

The results of the video segmentation are shown in Figure 5. As illustrated, our algorithm includes the 0–1 integer linear programming approach (ILP) and the dynamic programming approach (DP-1) yields a much better result. The higher value of AF and $AIoU$, the more representative of the keyframes. The illumination variation, blurring, and occlusion in the *Animal-world* dataset have a big impact in extracting keyframes based on the label propagation errors approach and the appearance transfer [42]. Though the algorithm can detect the shot change, it may fail in some cases mentioned above.

The video segmentation performance can be further improved by merging the uncovered frames, as shown in Figure 1, into key segments and transferring the appearance model based on flows between neighboring frames within key segments in our dynamic programming approach, as shown in ILP-2 and DP-2. The video segmentation is restricted within each segments. According to the segmentation method proposed in [42] Section 3.1, the unary terms include two terms (i.e., a unary potential based on an appearance model and a unary potential based on transferred label). The binary term is based on both the appearance and motion similarity of neighbouring pixels within each frame. Then, an MRF method is used to infer the labels of each pixels. As shown in Figure 5, the promotion of the segmentation performance is very obvious.

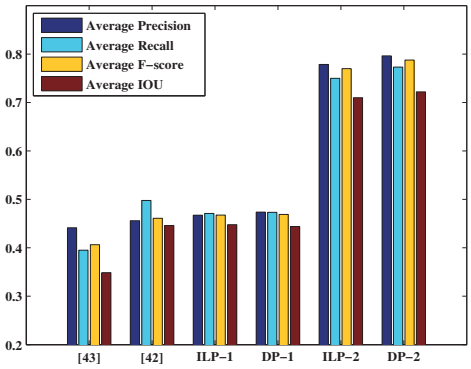


Figure 5. Comparison of segmentation performance. ILP-1 is our 0–1 integer linear programming approach. DP-1 is our dynamic programming approach. Compared with the original ILP-1 and DP-1, a minor modification is made on video segmentation process for ILP-2 and DP-2. The number of keyframes are 52. The same MRF segmentation propagation proposed in [42] is used to propagate the groundtruth of keyframes extracted by different algorithms including [42, 43], ILP1 and DP1, but except ILP-2 and DP-2.

Execution times: The computational cost of our approach is largely affected by the multiple feature extraction and clustering. It takes about 20 minutes to complete the feature extraction and clustering. However, the dynamic programming optimization process takes less than 30 seconds, while the 0–1 integer linear programming approach takes more than 20 minutes in key segments selection. [42] takes about 40 hours in computing its cost matrix. The keyframe selection process takes about five minutes. [43] has a execution times of 90 minutes.

5.3. Quantitative evaluation on KFSYD dataset

A keyframe is considered a true keyframe if it is no more than 15 frames apart from a ground-truth keyframe in the KFSYD dataset. A groundtruth keyframe will be matched with, at most, one keyframe. The matching procedure can be achieved by the hungarian algorithm [44].

To evaluate the quality of keyframes, we used three evaluations metrics (i.e., the average precision (AP) metric, the average recall (AR), and the average F-score (AF) metric), which are defined as follows:

$$AP = \frac{1}{30}\sum_{i=1}^{10}\sum_{j=1}^{3}\frac{M_{ij}}{B_i} \tag{5.5}$$

$$AR = \frac{1}{30}\sum_{i=1}^{10}\sum_{j=1}^{3}\frac{M_{ij}}{N_{ij}} \tag{5.6}$$

$$AF = \frac{1}{15}\sum_{i=1}^{10}\sum_{j=1}^{3}\frac{M_{ij}}{B_i + N_{ij}} \tag{5.7}$$

where M_{ij} denotes the number of keyframes extracted from the i^{th} video, which are matched with the groundtruth keyframes selected by the j^{th} student from the i^{th} video, B_i is the number of keyframes

extracted from the i^{th} video by the proposed approach, and N_{ij} is the number of the groundtruth keyframes selected by the j^{th} student from the i^{th} video.

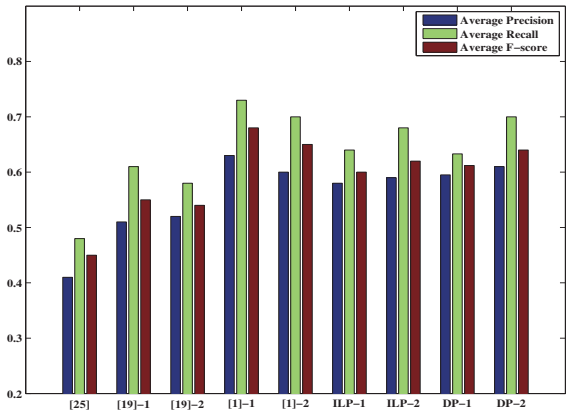
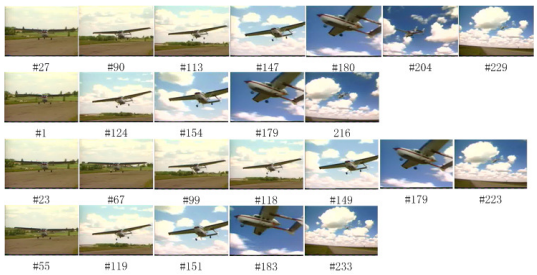


Figure 6. Quantitative evaluation on KFSYD dataset [1] in terms of average precision, average recall, average F-score. Our approach is compared against five state-of-art approaches: clustering [25], iso-content distance [19]-1, iso-content distortion [19]-2, KBKS [1]-1, KBKS-fast [1]-2. ILP-1(2) is 0–1 integer linear programming approach. DP-1(2) is dynamic programming approach. In ILP-1 or DP-1, the frames closest to the feature centroids of the key segments are selected as keyframes. Compared with ILP-1 and DP-1, the beginning frame of the first *key segment* and the ending frame of the last *key segment* are selected as the keyframes in ILP-2 and DP-2.

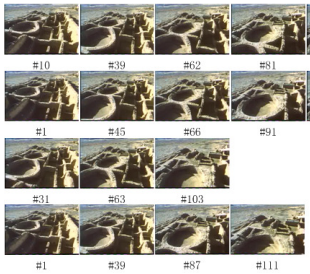
In our approach, the number of feature clustering are all set to $C = 5$. As illustrated in Figure 6, our two approaches achieve an improved performance with regards to the state of the art clustering approach [25] and the method in [19]. A clustering-based method is unlikely to select either the first or the last frames as a keyframe, but our approach still gives as good of a result as in ILP-1 and DP-1. As shown in Figure 6, the performance of the proposed method is lower than existing the KBKS and KBKS-fast methods. The main reason is that the proposed clustering-based method is unlikely to select either the first or the last frames as keyframe. If the beginning frame of the first key segment and the ending frame of the last key segment are selected as the keyframes, just like the method in [42], our approach can achieve a further improvement of the experimental performance in the ILP-2 or DP-2. These results show that the keyframes extracted with our approach are efficient. The visualization result of the DP-1 are shown in Figure 7. Our method can cover the primary contents of the video. As shown in Figure 7(a), our method can capture not only the posture change of an airplane, but also the shape change of the clouds. The keyframes extracted by our method have few redundancies. As shown in Figure 7(i), there are few similarities of visual content among them.

Execution times: For a video shot with 300 frames, KBKS [1]-1 needs roughly 150 seconds and KBKS [1]-2 needs about 15 seconds. The clustering method [25] needs about 13 seconds. The iso-content distance [19]-1 needs about 15 seconds and the iso-content distance [19]-2 needs about 16 seconds. The running time of our dynamic programming approach is roughly 250 seconds, where the feature extraction consumes a great deal of time. The running time of our 0–1 integer linear programming approach is roughly 300 seconds.

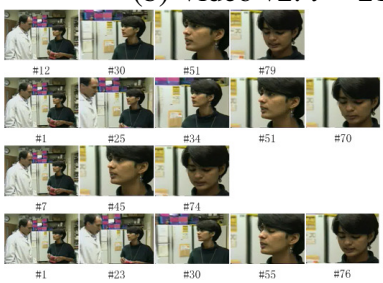


(a) Video v1. l = 238

(b) Video v2. l = 219



(c) Video v3. l = 112

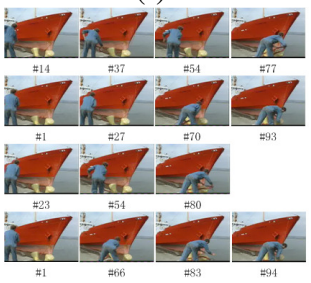


(d) Video v4. l = 97

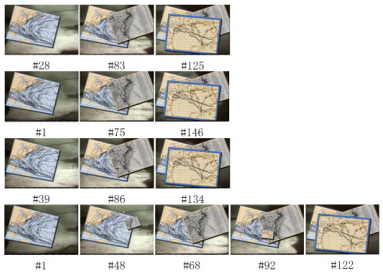
(e) Video v5. l = 201

(f) Video v6. l = 352

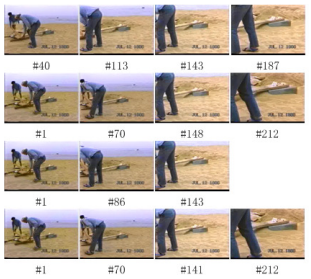
(g) Video v7. l = 202



(h) Video v8. l = 95



(i) Video v9. l = 147



(j) Video v10. l = 213

Figure 7. Keyframes extraction results on KFSYD dataset. Row 1: keyframes extraction results by the proposed method. The results in rows 2–4 are ground-truth keyframes of the videos. Numbers bellow are the keyframe indices. l is the total length of the video.

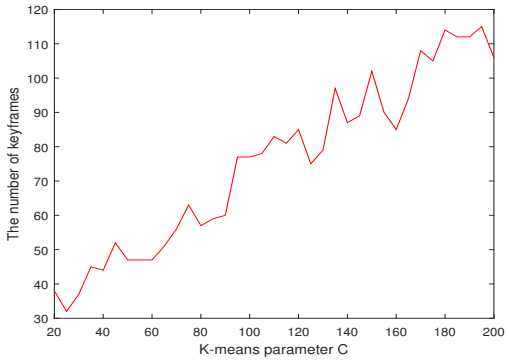


Figure 8. The influence of k-means parameter C in the number of keyframes. C is set from 20 to 200.

5.4. Discussion

In order to explore the influence of the k-means parameter C in the number of keyframes, the experiments are conducted on the Animal-world dataset by varying C at five intervals from 20 to 200. As shown in Figure 7, the parameter C does influence the number of keyframes. As C increases, the video will be separated into more segments, and the number of candidate segments also increases. The final number of keyframes becomes larger. When C is more than 50, the number of keyframes is far less than the parameter C as C increases. It shows that the relation between them is not very strong.

The main advantages of our proposed method are listed as follows: 1) From the result on the Animal-world dataset, our proposed method is adaptive to long video content, including complex video; 2) Since our proposed method is based on multiple features clustering, it is also easy to integrate other image descriptors; 3) The number of keyframes can be determined by semiautomatic, which has some influence from the parameter C, or specified by the users; and 4) Our proposed method is easy to implement and can obtain an optimal solution. However, every coin has two sides. Our proposed method is no exception. The main disadvantages or limitations are given as follows: (i) The procedures for feature extraction and clustering take too much time. It is not ideal for a real-time application. Through current popular hash technology [45], the large scale image clustering problem can be addressed very fast; (ii) The k-means parameter C has some influence on the number of keyframes. It requires human intervention to determine parameter C; and (iii) The motion information between frames or local descriptors within frames has not been taken into consideration. It is very valuable to mine these information and will require further investigation regarding integrating them into the keyframe extraction.

6. Conclusions

In this paper, we present a key frame extraction approach based on a combinatorial optimization problem. A novel dynamic programming approach and a 0–1 integer linear programming approach is developed to solve the combinatorial optimization problem. The experimental results on the Animal-world dataset and the KFSYD dataset demonstrated that the performance of the proposed approach is very promising. More experiments on other large and diverse datasets will be explored in future.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grants No.62201406, No.62171329 and the Startup Foundation of Wuhan Institute of Technology under Grant No.20QD16.

Conflict of interest

The authors declare there is no conflicts of interest.

References

1. G. Guan, Z. Wang, S. Lu, J. Deng, D. Feng, Keypoint-based keyframe selection, *IEEE Trans. Circuits Syst. Video Technol.*, **23** (2013), 729–734. https://doi.org/10.1007/978-1-4684-2001-2-9
2. Z. Dong, G. Zhang, J. Jia, H. Bao, Keyframe-based real-time camera tracking, in *IEEE International Conference on Computer Vision*, (2009), 1538–1545. https://doi.org/10.1109/ICCV.2009.5459273
3. C. Huang, E. Boyer, N. Navab, S. Ilic, Human shape and pose tracking using keyframes, in *IEEE Conference on Computer Vision and Pattern Recognition*, (2014), 3446–3453. https://doi.org/10.1109/CVPR.2014.440
4. C. Vondrick, D. Ramanan, Video annotation and tracking with active learning, in *Conference on Neural Information Processing Systems*, (2011), 28–36.
5. Z. Li, G. M. Schuster, A. K. Katsaggelos, MINMAX optimal video summarization, *IEEE Trans. Circuits Syst. Video Technol.*, **15** (2005), 1245–1256. https://doi.org/10.1109/TCSVT.2005.854230
6. C. Ngo, Y. Ma, H. Zhang, Video summarization and scene detection by graph modeling, *IEEE Trans. Circuits Syst. Video Technol.*, **15** (2005), 296–305. https://doi.org/10.1109/TCSVT.2004.841694
7. C. Ngo, T. Pong, H. Zhang, Motion-based video representation for scene change detection, *Int. J. Comput. Vision*, **50** (2002), 127–142. https://doi.org/https://doi.org/10.1023/A:1020341931699
8. B. Han, J. Hamm, J. Sim, Personalized video summarization with human in the loop, in *IEEE Workshop on Applications of Computer Vision*, (2011), 51–57. https://doi.org/10.1109/WACV.2011.5711483
9. K. Mahmoud, N. Ghanem, M. Ismail, Vgraph: An effective approach for generating static video summaries, in *IEEE International Conference on Computer Vision Workshops*, (2013), 811–818. https://doi.org/10.1109/ICCVW.2013.111
10. K. R. Raval, M. M. Goyani, A survey on event detection based video summarization for cricket, *Multimedia Tools Appl.*, **81** (2022), 29253–29281. https://doi.org/10.1007/s11042-022-12834-y
11. G. Wang, Y. Lu, L. Zhang, A. Alfarrarjeh, R. Zimmermann, S. H. Kim, et al., Active key frame selection for 3d model reconstruction from crowdsourced geo-tagged videos, in *IEEE International Conference on Multimedia and Expo*, (2014), 1–6. https://doi.org/10.1109/ICME.2014.6890253

12. L. Liu, L. Shao, P. Rockett, Boosted key-frame selection and correlated pyramidal motion-feature representation for human action recognition, *Pattern Recognit.*, **46** (2013), 1810–1818. https://doi.org/10.1016/j.patcog.2012.10.004

13. H. Zhang, J. Wu, D. Zhong, S. W. Smoliar, An integrated system for content-based video retrieval and browsing, *Pattern Recognit.*, **30** (1997), 643–658. https://doi.org/10.1016/S0031-3203(96)00109-4

14. R. S. Kiziltepe, J. Q. Gan, J. Escobar, A novel keyframe extraction method for video classification using deep neural networks, *Neural Comput. Appl.*, **2021** (2021), 1–12. https://doi.org/10.1007/s00521-021-06322-x

15. M. Jian, S. Zhang, L. Wu, S. Zhang, X. Wang, Y. He, Deep key frame extraction for sport training, *Neurocomputing*, **328** (2019), 147–156. https://doi.org/10.1016/j.neucom.2018.03.077

16. S. Jadon, M. Jasim, Unsupervised video summarization framework using keyframe extraction and video skimming, in *IEEE International Conference on Computing Communication and Automation (ICCCA)*, (2020), 140–145. https://doi.org/10.1109/ICCCA49541.2020.9250764

17. J. S. Boreczky, L. A. Rowe, Comparison of video shot boundary detection techniques, *J. Electron. Imaging*, **5** (1996), 122–128. https://doi.org/10.1117/12.238675

18. M. K. Mandal, F. M. Idris, S. Panchanathan, A critical evaluation of image and video indexing techniques in the compressed domain, *Image Vision Comput.*, **17** (1999), 513–529. https://doi.org/10.1016/S0262-8856(98)00143-7

19. C. Panagiotakis, A. Doulamis, G. Tziritas, Equivalent key frames selection based on iso-content principles, *IEEE Trans. Circuits Syst. Video Technol.*, **19** (2009), 447–451. https://doi.org/10.1109/TCSVT.2009.2013517

20. W. Chu, Y. Song, A. Jaimes, Video co-summarization: Video summarization by visual co-occurrence, in *IEEE Conference on Computer Vision and Pattern Recognition*, (2015), 3584–3592. https://doi.org/10.1109/CVPR.2015.7298981

21. M. Naphade, R. Mehrotra, A. Ferman, J. Warnick, T. Huang, A. Tekalp, A high-performance shot boundary detection algorithm using multiple cues, in *International Conference on Image Processing*, **1** (1998), 884–887. https://doi.org/10.1109/ICIP.1998.723662

22. V. Karasev, A. Ravichandran, S. Soatto, Active frame, location, and detector selection for automated and manual video annotation, in *IEEE Conference on Computer Vision and Pattern Recognition*, (2014), 2131–2138. https://doi.org/10.1109/CVPR.2014.273

23. M. Yeung, B. L. Yeo, Video visualization for compact presentation and fast browsing of pictorial content, *IEEE Trans. Circuits Syst. Video Technol.*, **7** (1997), 771–785. https://doi.org/10.1109/76.633496

24. M. Omidyeganeh, S. Ghaemmaghami, S. Shirmohammadi, Video keyframe analysis using a segment-based statistical metric in a visually sensitive parametric space, *IEEE Trans. Image Process.*, **20** (2011), 2730–2737. https://doi.org/10.1109/TIP.2011.2143421

25. Y. Zhuang, Y. Rui, T. Huang, S. Mehrotra, Adaptive key frame extraction using unsupervised clustering, in *International Conference on Image Processing*, **1** (1998), 866–870. https://doi.org/10.1109/ICIP.1998.723655

26. A. Hanjalic, H. Zhang, An integrated scheme for automated video abstraction based on unsupervised cluster-validity analysis, *IEEE Trans. Circuits Syst. Video Technol.*, **9** (1999), 1280–1289. https://doi.org/10.1109/76.809162
27. A. Girgensohn, J. Boreczky, Time-constrained keyframe selection technique, in *IEEE International Conference on Multimedia Computing and Systems*, **1** (1999), 756–761. https://doi.org/10.1109/MMCS.1999.779294
28. M. Kumar, A. Loui, Key frame extraction from consumer videos using sparse representation, in *IEEE International Conference on Image Processing*, (2011), 2437–2440. https://doi.org/10.1109/ICIP.2011.6116136
29. T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*, MIT Press, 2009.
30. G. L. Nemhauser, L. A. Wolsey, *Integer and Combinatorial Optimization*, Wiley, 1988. https://doi.org/10.1007/978-3-030-45771-6
31. R. M. Karp, *Reducibility among Combinatorial Problems*, Springer Berlin Heidelberg, 2010.
32. H. S. Park, J. Shi, Social saliency prediction, in *IEEE Conference on Computer Vision and Pattern Recognition*, (2015), 971–987. https://doi.org/10.1109/CVPR.2015.7299110
33. H. S. Chang, S. Sull, S. U. Lee, Efficient video indexing scheme for content-based retrieval, *IEEE Trans. Circuits Syst. Video Technol.*, **9** (1999), 1269–1279. https://doi.org/10.1109/76.809161
34. V. T. Paschos, A first course in combinatorial optimization, cambridge texts in applied mathematics, *Eur. J. Oper. Res.*, **168** (2006), 1042–1044. https://doi.org/10.1017/CBO9780511616655
35. H. Lee, S. Kim, Iterative key frame selection in the rate-constraint environment, *Signal Process. Image Commun.*, **18** (2003), 1–15. https://doi.org/10.1016/S0923-5965(02)00089-9
36. X. Liu, M. Song, L. Zhang, S. Wang, J. Bu, C. Chen, et al., Joint shot boundary detection and key frame extraction, in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, (2012), 2565–2568.
37. D. G. Lowe, Distinctive image features from scale-invariant keypoints, *Int. J. Comput. Vision*, **60** (2004), 91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94
38. A. T. Aude Oliva, Modeling the shape of the scene: A holistic representation of the spatial envelope, *Int. J. Comput. Vision*, **42** (2001), 145–175. https://doi.org/10.1023/A:1011139631724
39. N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (2005), 886–893. https://doi.org/10.1109/CVPR.2005.177
40. J. Bang-Jensen, G. Gutin, *Digraphs: Theory, Algorithms and Applications*, Springer, 2002. https://doi.org/10.1007/978-1-84800-998-1
41. F. F. Li, P. Perona, A bayesian hierarchical model for learning natural scene categories, in *IEEE Conference on Computer Vision and Pattern Recognition*, (2005), 524–531. https://doi.org/10.1109/CVPR.2005.16
42. S. Vijayanarasimhan, K. Grauman, Active frame selection for label propagation in videos, in *European Conference on Computer Vision*, (2012), 496–509. https://doi.org/10.1007/978-3-642-33715-4-36

43. T. Liu, J. R. Kender, Optimization algorithms for the selection of key frame sequences of variable length, in *European Conference on Computer Vision*, (2002), 403–417. https://doi.org/10.1007/3-540-47979-1-27

44. J. Munkres, Algorithms for the assignment and transportation problems, *J. Soc. Ind. Appl. Math.*, **5** (1957), 32–38. https://doi.org/10.1137/0105003

45. Y. Gong, M. Pawlowski, F. Yang, L. Brandy, L. Boundev, R. Fergus, Web scale photo hash clustering on a single machine, in *IEEE Conference on Computer Vision and Pattern Recognition*, (2015), 19–27. https://doi.org/10.1109/CVPR.2015.7298596

AIMS Press