KEY FRAME EXTRACTION FROM CONSUMER VIDEOS USING SPARSE REPRESENTATION

Mrityunjay Kumar, Alexander C. Loui

Corporate Research & Engineering
Eastman Kodak Company
Rochester, NY, USA
Email: {mrityunjay.kumar, alexander.loui}@kodak.com

ABSTRACT

Key frame extraction algorithms select a subset of the most informative frames from videos. Key frame extraction finds applications in several broad areas of video processing research such as video summarization, creating "chapter titles" in DVDs, video indexing, and prints from video. In this paper, a sparse representation based method to extract key frames from unstructured consumer videos is presented. In the proposed approach, video frames are projected to a low dimensional random feature space and theory from sparse signal representation is used to analyze the spatio-temporal information of the video data and generate key frames. The proposed approach is computationally efficient and does not require shot(s) detection, segmentation, or semantic understanding. A comparison of the results obtained by this method with the ground truth agreed by multiple judges and another approach based on camera operator's intent clearly indicates the feasibility of the proposed approach.

Index Terms— Key frame extraction, sparse representation, compressed sensing, consumer videos, video analysis.

1. INTRODUCTION

Key frame extraction algorithms select a subset of the most informative frames from videos [1–3]. Key frame extraction finds applications in several broad areas of video processing research such as video summarization, creating "chapter titles" in DVDs, video indexing, and prints from video. Key frame extraction is an active research area, and many approaches for extracting key frames from videos have been proposed. Conventional key frame extraction approaches can be loosely divided into two groups: (i) shot-based, and (ii) segment-based. In shot-based key frame extraction, the shots of the original video are first detected, and then one or more key frames are extracted for each shot [4]. In segment-based key frame extraction approaches, a video is segmented into higher-level video components, where each segment or component could be a scene, an event, a set of one or more shots, or even the entire video sequence. Representative frame(s) from each segment are then selected as the key frames [2].

Existing key frame approaches, shot-based as well as segment-based, are usually suitable for structured videos such as news and sports videos. However, they are sub-optimal for consumer videos as these videos are typically captured in an unconstrained environment and record extremely diverse contents. Moreover, consumer videos often lack a pre-imposed structure, which makes it even more challenging to detect shots or segment such videos for key frame extraction [5, 6].

Because of the enormous number of pixels present in videos, as a rule of thumb, first, the feature vectors are computed from video frames and then these features are processed to extract key frames. Due to diverse nature and lack of structure in the consumer videos, it is challenging to identify (set of) features that represent the entire consumer video space adequately. Therefore, a key frame extraction framework in which feature selection is less critical is highly desirable for consumer videos. Furthermore, computational efficiency is also crucial as key frames extraction may just be an intermediate step for other applications of interest such as print from video, video retrieval, etc.

Recently, feature selection has been shown to be less critical within the sparse signal representation framework for various computer vision tasks [7]. Motivated by this, a sparse representation based framework for extracting key frames from consumer videos is presented in this paper. In the proposed approach, video frames are projected to a low dimensional random feature space and theory from sparse signal representation is exploited in the random feature space to analyze the spatio-temporal information of the video data and generate key frames.

This paper is organized as follows. Section 2 reviews sparse signal representation. In Section 3, the proposed key frame extraction algorithm is described, while Section 4 summarizes the overall algorithm. Section 5 presents benchmarking results comparing to ground truth data. Finally, concluding remarks are given in Section 6

2. A REVIEW OF SPARSE SIGNAL REPRESENTATION

Sparse signal representation has received tremendous research attention in recent years and has been applied to many image processing and computer vision applications [8, 9]. In sparse signal representation, a signal is represented as a linear combination of a few columns (commonly referred to as atoms) of a given overcomplete dictionary. For a signal $f \in \mathbb{R}^p$ and a dictionary $\Theta \in \mathbb{R}^{p \times q}, p < q$, the sparse representation of f can be stated as

$$\min_{\alpha} \|\alpha\|_0 \quad \text{such that} \quad f = \Theta\alpha, \tag{1}$$

where $\|\alpha\|_0$ is the ℓ_0-norm (number of non-zero elements) of the sparse coefficient vector $\alpha \in \mathbb{R}^q$ and typically $\|\alpha\|_0 \ll q$.

To use sparse signal representation in applications, first the dictionary Θ is fixed and then efficient solvers are used to estimate α from eq. (1). However, solving eq. (1) is NP-hard, and thus approximate solutions (greedy algorithms, ℓ_1-norm based approximations, etc.) are considered in practice to estimate α [10].

As explained in the next section, the proposed approach builds on the theory presented in this section to extract key frames from consumer videos.

3. PROPOSED APPROACH

Let $y_i \in \mathbb{R}^n$, $1 \leq i \leq N$, be the vector representation of the i^{th} frame with pixels arranged in lexicographic order and N be the total number of frames of the input video. The random projection of y_i, denoted as $f_i \in \mathbb{R}^m$ ($m \ll n, m < N$), is computed as given below:

$$f_i = \Phi y_i \tag{2}$$

where $\Phi \in \mathbb{R}^{m \times n}$ is a random projection matrix. Each entry of the random projection matrix Φ is selected by sampling the discrete uniform distribution $U(-1, 1)$ independently. Note that Φ does not depend on the input video, hence, it can be pre-computed and stored for subsequent processing of the input videos. Therefore, computation of f_i, which acts as a feature of the i^{th} frame, is just a matrix multiplication operation and is computationally much more efficient than other more traditional features [11] such as SIFT, GIST, color histogram, etc.

Each f_i estimated by solving eq. (2) is further represented as a sparse linear combination of the columns of the corresponding over-complete dictionary $\Theta_i \in \mathbb{R}^{m \times N}$ as shown below

$$f_i = \Theta_i \alpha_i \tag{3}$$

where $\alpha_i \in \mathbb{R}^N$ is a sparse vector having k_i non-zero elements such that $k_i \ll N$. For simplicity, dictionary Θ_i is constructed by arranging (in temporal order) all the f_is but the i^{th} one as columns of Θ_i, *i.e.*, $\Theta_i = \left[f_1, \cdots, f_{i-1}, \underline{0}, f_{i+1}, f_N\right]$ and $\underline{0}$ is a zero vector. Furthermore, each column of Θ_i is normalized to one, *i.e.*, $f_i^T f_i = 1$, where "T" denotes the matrix transpose operation. Note that the proposed approach is generic in nature and, therefore, other dictionary learning algorithms [10] can be exploited to construct Θ_i. However, discussion of alternate ways to construct Θ_i is outside the scope of this paper and is not discussed here.

In the proposed approach, α_i is assumed to be a non-negative vector, *i.e.*, $\alpha_i^j \geq 0$, where α_i^j is the j^{th} component of the sparse vector α_i and $1 \leq j \leq N$. Furthermore, α_i is estimated from eq. (3) by solving an optimization problem of the form [12]

$$\min \|\Theta_i \alpha_i - f_i\|_2^2 + \lambda \sum_{j=1}^{N} \alpha_i^j \tag{4}$$

$$\text{subject to } \alpha_i^j \geq 0, \quad 1 \leq j \leq N \tag{5}$$

where $\| \bullet \|_2$ represents ℓ_2-norm and $\lambda (\geq 0)$ is the regularization parameter.

The overcompleteness or redundancy of the matrix Θ_i, $1 \leq i \leq N$ is absolutely critical to ensure the validity of the proposed approach, which in turns requires m to be selected such that $m < N$. The duration and the frame rate of the input video can be used as guidelines to determine m appropriately. For example, $n = 5 *$ frame rate can be used as a ballpark estimate as videos smaller than 5 sec are too small and key frame extraction is usually not needed to browse such videos.

After estimating α_i, $1 \leq i \leq N$, a clustering algorithm is applied on α_i to estimate the key frames. This is explained in details next.

In order to estimate the key frames using α_i, first a symmetric matrix B is computed as shown below

$$B = (W + W^T)/2 \tag{6}$$

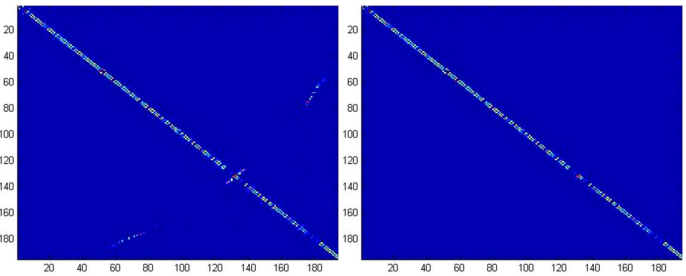
where $B, W \in \mathbb{R}^{N \times N}$, and $W(i, j) = \alpha_i^j$. Furthermore, the diagonal elements of B are set to zero for ease of clustering. Note that there is an intriguing relationship between sparse representation and data clustering, therefore, the former has been exploited for data clustering for various image processing and computer vision applications [10, 13].

To make sure that the frame at the i^{th} time instant is influenced only by the temporally nearby frames, B is further modified as

$$B^*(i, j) = \exp^{-\gamma |i-j|^2} B(i, j), \;\; 1 \leq i, j \leq N \tag{7}$$

where γ is a constant. The effectiveness of the temporal constraint on B, *i.e.*, estimation of B^*, is illustrated in Fig. 1. Figure 1(a) is B (*i.e.*, $\gamma = 0$ in eq. (7)) estimated from the input video shown in Fig. 3(b) in Section 5. Figure 1(b) is the corresponding B^*, derived from B by setting $\gamma = 0.08$ in eq. (7). Clearly, B^* reflects a better temporal topology of the video and is more appropriate for subsequent clustering purpose.

To extract key frames from the input video, a normalized clustering [14] is applied on the B^*. The number of clusters is set equal to the desired number of key frames. Then, within each cluster, the video frames are sorted in temporal order and the middle frame from each cluster is selected as a key frame.



(a) (b)

Fig. 1. Temporal constraint (a) B (b) $B^*(\gamma = 0.08)$.

4. OVERALL APPROACH

The flowchart of the proposed key frame extraction algorithm is shown in Fig. 2. The source video and the desired number of key frames, denoted as π in Fig. 2, are the input parameters of the proposed algorithm. The number of key frames is a function of the overall representativeness of the video and the application at hand. In this paper, the number of the key frames corresponding to the source video is assumed to be known. The random projection matrix Φ, required to compute $f_i (1 \leq i \leq N)$, is pre-computed only once by sampling $U(-1, 1)$ as discussed in Section 3. The number of columns of the random projection matrix is set to the size of the video frame resolution. For example, for a VGA resolution video, $n = 640 \times 480 = 307200$. The guideline suggested in Section 3 is used to select m. Selection of m is elaborated more in Section 5 with respect to specific videos selected to validate the proposed approach.

In order to apply the proposed approach to the source/input video, each frame of the source video is first converted into luminance frame by taking a linear combination of the red, green, and

blue color channels of the corresponding frame and y_i is constructed by lexicographically arranging the pixels of the i^{th} luminance frame. The rest of the steps to extract key frames from the source video are straightforward and are summarized as Step 3-6 in Fig. 2.

- **Input:** Source video, number of key frames(= π)
- **Output:** π number of key frames of the source video
  - **STEP 1:** Load random projection matrix Φ in the memory
  - **STEP 2:** Compute $y_i, 1 \leq i \leq N$
  - **STEP 3:** Compute f_i using eq. (2)
  - **STEP 4:** Estimate sparse vectors α_i
  - **STEP 5:** Compute B^* and apply normalized clustering on B^*, where number of clusters = π
  - **STEP 6:** For each cluster
    - Arrange video frames in temporal order
    - Select middle frame as a key frame

Fig. 2. Flowchart of the proposed key frame extraction method

5. RESULTS

The results of the proposed key frame extraction algorithm are presented in this section. A database having 100 video clips selected from a large database of over 3000 video clips [6] was used to validate the proposed approach. These 100 video clips were captured using Kodak EasyShare C360 and V550 zoom digital cameras, with frame size 640×480 and frame rates ranging from 24 to 30 fps. These video clips were intended to capture a wide range of events typically encountered in consumer video clips such as outdoor activities, natural sceneries, trips, sports, weddings, etc.

The proposed method was compared with the ground truth agreed by multiple human judges [5] and another approach based on camera operator's intent [6]. To establish the ground truth, three human judges were asked to independently browse the video clips and provide the key frames. Photographers who actually captured the videos were not selected as the judges. The key frames estimated by the three judges were reviewed in a group session with a fourth judge (arbitrator) to derive final key frames for each of the 100 video clips. The number of key frames was determined by the human judges based on the representativeness and quality of the corresponding video clips.

To compare the proposed approach against an automatic algorithm, a key frame extraction algorithm based on camera operator's intent [5] was selected as another benchmark algorithm. This algorithm will henceforth in this paper be referred to as camera motion-based key frame extraction (MKFE) algorithm. MKFE was designed explicitly to deal with consumer video clips. This algorithm first segments an input video clip into homogeneous parts based on major types of camera motion, e.g., pan, zoom, pause, and steady, and dedicated rules are used to extract key frames from each segment.

Due to page constraints, only three sets of results are presented in this paper. For each video clip in the database, the number of key frames determined by the judges was considered as the ground truth number of key frames for the corresponding video. Therefore, both MKFE and the proposed approach were executed to generate exactly the same number of key frames as those determined by the judges.

For the proposed approach, the salient steps summarized in Fig. 2 were applied to the video clips to extract the key frames. As explained in Section 4, $y_i (1 \leq i \leq N)$ was constructed by lexicographically arranging the pixels of the i^{th} luminance frame. Also, the random projection matrix Φ was pre-computed only once and the same Φ was applied to all of the videos used in the experiment. The number of columns of the random projection matrix Φ was set to the size of VGA resolution, *i.e.*, $n = 640 \times 480 = 307200$. All of the videos used to validate the proposed approach were more than 5 sec long and their frame rates ranged between 24-30 fps. Therefore, following the guidelines explained in Section 3, the value of m was set to 100 ($\cong 4 \times 24$). This choice of m not only projected each frame to a significantly low-dimensional (random) feature space but also made the dictionaries Θ_i $(1 \leq i \leq N)$ overcomplete ($m < N$). The value of γ in eq. (7) was tuned manually for each video clip to optimize the accuracy of the key frame extraction.

Comparisons of the proposed approach (third row) with the ground truth determined by multiple judges (first row) and the MKFE algorithm (second row) are provided in Fig. 3(a)-(c). Key frames are numbered from left to right in each row for reference. The input video corresponding to the key frames in Fig. 3(a) was 22.63 sec long and was captured at 24 fps. The video depicted moving landscape and had a significant amount of perspective and brightness change. The ground truth number of key frames was 5 and the value of γ was 0.02. Visually the proposed approach appears to have an advantage over the MKFE algorithm. For example, the proposed approach captured the first and the fourth key frames of the ground truth successfully, which MKFE failed to detect. In addition to that, the proposed approach detected the second key frame partially but MKFE missed it completely. Furthermore, the proposed approach did not require any video segmentation or semantic understanding.

The videos shown in Fig. 3(b) and (c) were 8.13 and 25.88 sec long, respectively, and were captured at 24 fps. Both videos were taken outdoors with a significant amount of change in perspective and brightness. The ground truth number of key frames were 4 and 6, respectively, whereas, γ values for these videos were 0.08 and 0.008, respectively. For video in Fig. 3(b), both MKFE and the proposed algorithms appear to be comparable with the ground truth. In the last case (Fig. 3(c)), a visual comparison indicates that the second key frame detected by both MKFE and the proposed algorithms is redundant. Also, the proposed algorithm and MKFE missed the second and the third key frame of the ground truth, respectively. Except for that, both algorithms performed similarly to the ground truth.

The proposed algorithm was implemented in Matlab. On a 2.66 GHz Intel Machine, with 3.0 GB RAM, random projection (*i.e.*, computation of f_is) of the video shown in Fig. 3(c) took 81.32 sec (*i.e.*, 0.131 sec/frame), which is much lower than the time it usually takes to compute features that are more commonly used for computer vision tasks [11] such as SIFT, GIST, color histogram, etc. Furthermore, sparse coding, clustering, and key frames selection took an additional 35.25 sec. Similarly, it took total of 103.04, and 33.45 sec (including random projection, sparse coding, clustering, and key frames selection) to extract key frames from the videos shown in Fig. 3(a), and (b), respectively. Due to the unavailability of the MKFE code (only final key frame results were available), we are unable to report its computational time. However, to the best of our knowledge, it is computationally slower as compared to the proposed method due to various camera motion detection steps.

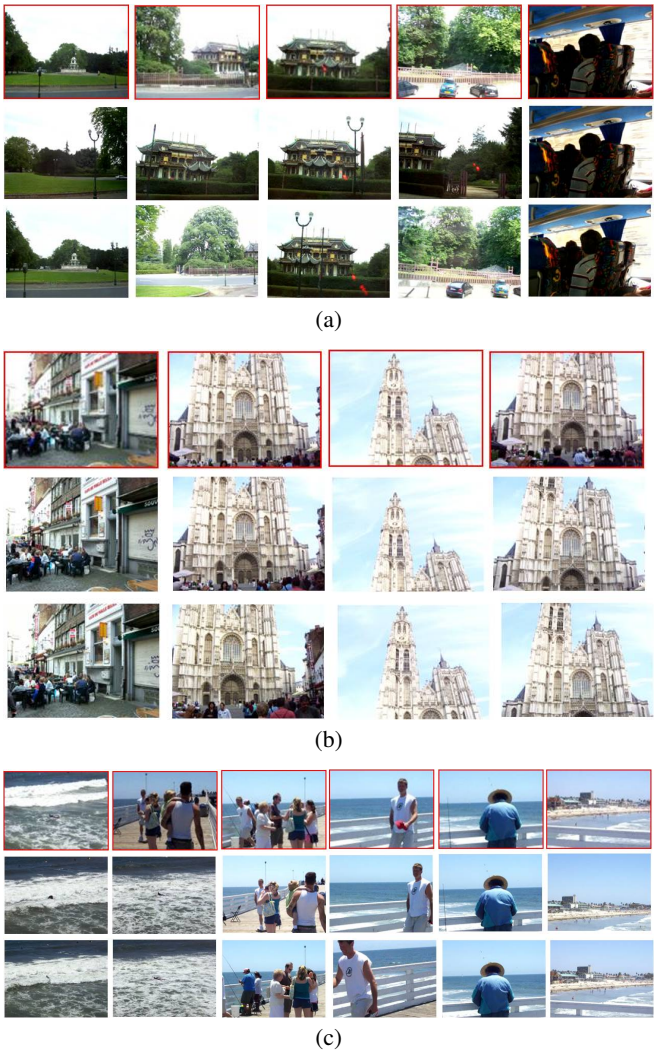


Fig. 3. Key frames extracted using ground truth (first row), MKFE (second row), and the proposed approach (third row) are compared. Videos (a) "BusTour", (b) "OrnateChurch", (c) "SoloSurfer".

6. CONCLUSIONS AND FUTURE WORK

In this paper, a sparse representation based framework for extracting key frames from consumer videos is proposed. In the proposed approach, video frames are projected to a low dimension feature space using a random projection matrix and sparse representation is exploited in the random feature space to analyze the spatio-temporal information of the video data and generate key frames. In contrast to shot- and segment-based key frame extraction algorithms, the proposed approach does not require shot(s) detection, segmentation, or semantic understanding and is computationally efficient. Extensive experimental results clearly indicate the feasibility of the proposed approach. Future work will focus on exploring more sophisticated approaches for designing the dictionary for the proposed framework. A study of the feasibility of using more conventional features (SIFT, GIST, color histogram, etc.) within the proposed framework and quantitative evaluation of the quality of the extracted key frames will also be carried out in the future. Also, the proposed approach will be validated on larger scale video datasets.

7. REFERENCES

[1] B. T. Truong and S. Venkatesh, "Video abstraction: a systematic review and classification," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 3, no. 1, pp. 3–es, Feb. 2007.

[2] Z. Rasheed and M. Shah, "Detection and representation of scenes in videos," *IEEE Trans. Multimedia*, vol. 7, no. 6, pp. 1097–1105, Dec. 2005.

[3] N. Dimitrova, T. McGee, and H. Elenbaas, "Video keyframe extraction and filtering: a keyframe is not a keyframe to everyone," in *Proc. CIKM*, Mar. 1997, pp. 113–120.

[4] S. Uchihashi and J. Foote, "Summarizing video using a shot importance measure and a frame-packing algorithm," in *IEEE ICASSP*, 1999, vol. 6, pp. 3041–3044.

[5] K. Costello and J. Luo, "First-and third-party ground truth for key frame extraction from consumer video clips," in *Proc. SPIE 6492*, 64921N (2007).

[6] J. Luo, C. Papin, and K. Costello, "Towards extracting semantically meaningful key frames from personal video clips: from humans to computers," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 2, pp. 289–301, Feb. 2009.

[7] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, Feb. 2009.

[8] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan, "Sparse representation for computer vision and pattern recognition," *Proc. IEEE*, vol. 98, no. 6, pp. 1031–1044, Jun. 2010.

[9] M. Elad, M. A. T. Figueiredo, and Y. Ma, "On the role of sparse and redundant representations in image processing," *Proc. IEEE*, vol. 98, no. 6, pp. 972–982, Jun. 2010.

[10] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: an algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[11] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "SUN database: large-scale scene recognition from abbey to zoo," in *IEEE CVPR*, 2010, pp. 3485–3492.

[12] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An interior-point method for large-scale ℓ_1-regularized least squares," *IEEE J. Sel. Topics Signal Process.*, vol. 1, no. 4, pp. 606–617, Dec. 2007.

[13] B. Cheng, J. Yang, S. Yan, Y. Fu, and T. S. Huang, "Learning with ℓ^1-graph for image analysis," *IEEE Trans. Image Process.*, vol. 19, no. 4, pp. 858–866, Apr. 2010.

[14] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.