6-2014

Quasi Real-Time Summarization for Consumer Videos

Bin Zhao
Carnegie Mellon University, binzhao@andrew.cmu.edu

Eric P. Xing
Carnegie Mellon University, epxing@cs.cmu.edu

Published In
Proceedings of Computer Vision and Pattern Recognition (CVPR 2014), 2513-2520.

Quasi Real-Time Summarization for Consumer Videos

Bin Zhao Eric P. Xing
School of Computer Science, Carnegie Mellon University
{binzhao,epxing}@cs.cmu.edu

Abstract

With the widespread availability of video cameras, we are facing an ever-growing enormous collection of unedited and unstructured video data. Due to lack of an automatic way to generate summaries from this large collection of consumer videos, they can be tedious and time consuming to index or search. In this work, we propose online video highlighting*, a principled way of generating short video summarizing the most important and interesting contents of an unedited and unstructured video, costly both time-wise and financially for manual processing. Specifically, our method learns a dictionary from given video using* group sparse coding*, and updates atoms in the dictionary on-the-fly. A summary video is then generated by combining segments that cannot be sparsely reconstructed using the learned dictionary. The online fashion of our proposed method enables it to process arbitrarily long videos and start generating summaries before seeing the end of the video. Moreover, the processing time required by our proposed method is close to the original video length, achieving quasi real-time summarization speed. Theoretical analysis, together with experimental results on more than 12 hours of surveillance and YouTube videos are provided, demonstrating the effectiveness of* online video highlighting.

1. Introduction

With the widespread availability, to both consumers and organizations, of low-cost devices capable of high-volume video recording, such as digital cameras on mobile phones, tablets, and soon, wearable gadgets such as glasses and watches; and various surveillance cameras and monitoring devices all over the world and in space, we are inundated with billion hours of video footage every day potentially containing events, people, and objects of context-dependent and time-space-sensitive interests. However, even to the creators/owners of such data, let alone all the people who are granted access for various purposes, the contents in all these videos remain *dark matter* in the data universe, because watching these recorded footage in real-time, or even playing at 2x or 4x speed is hardly possible and enjoyable. It is no surprise that with this increasing body of video data, which are largely left unedited and unstructured, all information therein are like trees falling in the forest — they are nearly impossible to access unless already been seen and indexed, an undertaking too tedious and time consuming for human, but an ideal challenge for machine intelligence. In this paper, we refer to those unstructured and unedited videos as *consumer videos*, in contrast to movies, news or sports videos which are often edited by human or having special structure (such as shot, scene, etc.).

Specifically, we attempt to develop a method that offers the following function and alike: *"I only have 1 minute for this hour-long video, tell me where/what to watch"*. That is, it automatically compiles the most salient and informative portion of the video for users, by automatically scanning through video stream, in an online fashion, to remove repetitive and uninteresting contents. Our method differs from some previous attempts to video summarization that eliminate completely the time axis, and show a synopsis of the video by collecting a few key frames which are selected either arbitrarily, or according to some importance criteria [33, 11, 15]. Such key frame representation loses the dynamic aspect of video and is uninteresting to watch. More importantly, taking merely frames as unit of content in a video prevents many important information such as suspicious behaviors to be recognized automatically by a machine, therefore compromises the quality of the summary. On the other hand, the summary generated by our proposed method is a short video itself, revealing the essence of the original video, just like a "trailer".

We propose *onLIne VidEo highLIGHTing (LiveLight)*, a principled way of online generation of a short video summarizing the most important and interesting contents of a potentially very long video. Specifically, *LiveLight* scans through the video stream, divided into a collection of video segments temporally. After processing the first few segments, it starts to build its own dictionary, which will be kept updated and refined later. Given a new video segment, *LiveLight* attempts to employ its current version of dictionary to sparsely reconstruct this previously unseen segment,

using *group sparse coding* [3]. A small reconstruction error of the new video segment reflects that its content is already well represented in the current dictionary, further suggesting video segments containing similar contents have been observed in early part of the video. Hence, this segment is excluded from the summary, and the algorithm moves on to next segment. On the other hand, if the new video segment cannot be sparsely reconstructed, i.e., a high reconstruction error is suffered, indicating unseen contents from previous video data, our method incorporates this video segment into the summary, and updates the dictionary according to the newly included video data. This process continues until the end of the video is reached. In summary, our method sequentially scans the video stream once, learns a dictionary to summarize contents seen in the video and updates it after encountering video data that could not be explained using current dictionary. A summary video is then constructed as a combination of two groups of video segments: (1) the first few segments used to learn initial dictionary, capturing background and early contents of the video; (2) video segments causing dictionary update, suggesting unseen and interesting contents. Moreover, as the entire process is carried out online, *LiveLight* could handle hours or even endless video data, ubiquitous in consumer videos.

1.1. Related Works

Previous research on video summarization has mainly focused on edited videos, e.g., movies, news, and sports, which are highly structured [21, 30]. For example, a movie could be naturally divided into scenes, each formed by one or more shots taken place at the same site, and each shot is further composed of frames with smooth and continuous motions. However, consumer videos lack such structure, often rendering previous research not directly applicable.

Key frame based methods compose video summary as a collection of salient images (key frames) picked from the original video. Various strategies haven been studied, including shot boundary detection [9], color histogram [33], motion stability [33], clustering [13], curve splitting [7], and frame self-expressiveness [11]. However, isolated and uncorrelated still images, without smooth temporal continuation, are not best suited to help viewer understand the original video. Moreover, [15] proposes a saliency based method, which trains a linear regression model to predict importance score for each frame in egocentric videos [15]. However, special features designed in [15] limit its applicability only to videos generated by wearable cameras. Besides picking frames from the original video, methods creating new image not present in the original video have also been studied [1, 5, 17, 24, 26, 27], where a panoramic image is generated from a few consecutive frames having some important content. However, the number of consecutive frames from original video used to construct such panoramic image is limited by occlusion between objects from different frames. Consequently, these approaches generally assume short clips with few objects. Finally, summaries composed by a collection of video segments, have been studied for structured videos. Specifically, [23] use scene boundary detection, dialogue analysis, and color histogram to produce trailer for a feature film. [2] and [16] extract important segments from sports and news programs utilizing special characteristics of these videos, including fixed scene structures, dominant locations, and backgrounds. Moreover, [29] and [28] utilize closed caption and speech recognition to transform video summarization into a text summarization problem and generate summaries using natural language processing techniques. However, the large body of consumer videos usually have no such special structure, nor audio information at all.

Sparse coding [22] has led to state-of-the-art results in several vision tasks such as image denoising and restoration [10, 20], as well as classification [32, 31] and anomaly detection [34]. Moreover, brute-force deployment of sparse coding for video summarization, using the entire video as dictionary, and selecting key frames based on zero patterns of the coding vector, has recently been attempted [11, 6] on short videos (less than few minutes long). We will discuss and compare against such method later in this paper.

1.2. Summary of Contributions

To conclude the introduction, we summarize our main contributions as follows. (1) We propose a principled way of generating short summary video of a potentially very long video, summarizing its most important and interesting contents while eliminating repetitive events, enabling viewer to understand the video without watching the entire sequence. (2) We propose an online dictionary update method, enabling our method to generate summaries on-the-fly. (3) We provide theoretical analysis of the proposed method, guaranteeing convergence of the online dictionary update and generalization ability to unseen video segments. (4) We demonstrate the effectiveness of *LiveLight* on real-world data, including both surveillance videos and YouTube videos, achieving quasi real-time speed on all tested videos.

2. Online Video Highlighting

Given an unedited and unstructured consumer video, *online video highlighting* starts with temporal segmentation, breaking original video into segments. Such temporal segmentation should ensure minimum variation, and consistency of objects, view and dynamics within each segment. Unlike structured videos, where shot boundary detection could be employed for temporal segmentation, most consumer videos do not even have such shot boundary, but instead with continuous camera movement. Therefore, we choose to evenly divide the original video into segments,

each with a constant length of 50 frames. Such short temporal length ensures the consistency within each segment. These video segments are the base units in *LiveLight*, in the sense that a few selected ones will compose the final summary video. A key component in *LiveLight* is *dictionary*, which summarizes the contents of seen video. Specifically, a dictionary is initially learned using video segments at the beginning of the input video, with *group sparse coding*. After dictionary initialization, *LiveLight* scans through the rest video segments following temporal order, and attempts to reconstruct each video segment using the learned dictionary. Those video segments with reconstruction error higher than certain threshold are considered to contain interesting contents unprecedented in previous video, and are included into the summary video. Moreover, the dictionary is updated accordingly to incorporate the newly observed video contents, such that similar video segments seen later will suffer much smaller reconstruction error. On the other hand, those video segments that could be well reconstructed using the current dictionary is excluded from the summary, as small reconstruction error suggests its content is already well represented in the current dictionary, further indicating video segments containing similar contents have been observed in early part of the video. Hence, the dictionary represents the knowledge about previously seen video contents, and is updated in an online fashion to incorporate newly observed contents. Algorithm 1 provides the work flow of *LiveLight*, where $\mathcal{X}_0 = \{\mathbf{X}_1, \ldots, \mathbf{X}_m\}$ is used to learn initial dictionary with $m \ll K$, and ϵ_0 is a pre-set threshold parameter controlling length of the summary video.

Algorithm 1 Online Video Highlighting (LiveLight)

input Video $\mathcal{X}$ composed of temporal segments $\{\mathbf{X}_1, \ldots, \mathbf{X}_K\}$
output Short video $\mathcal{Z}$ summarizing most important and interesting contents of $\mathcal{X}$

1: Learn initial dictionary $\mathbf{D}$ using $\mathcal{X}_0 = \{\mathbf{X}_1, \ldots, \mathbf{X}_m\}$ via group sparse coding and initialize $\mathcal{Z} = \mathcal{X}_0$
2: **for all** Video segments $\mathbf{X}_k \in \{\mathbf{X}_{m+1}, \ldots, \mathbf{X}_K\}$ **do**
3: Reconstruct video segment $\mathbf{X}_k$ using current dictionary $\mathbf{D}$ and compute reconstruction error ϵ_k
4: **if** ϵ_k is larger than pre-set threshold ϵ_0 **then**
5: Update dictionary $\mathbf{D}$ using $\mathbf{X}_k$ and incorporate $\mathbf{X}_k$ into summary video $\mathcal{Z} = \mathcal{Z} \cup \mathbf{X}_k$
6: **end if**
7: **end for**

2.1. Video Segment Reconstruction

The basic idea for our approach is to represent the knowledge of previously observed video segments using the learned dictionary $\mathbf{D}$, whose columns (a.k.a. atoms) are bases for reconstructing future video segments. Given learned dictionary $\mathbf{D}$ (details of learning initial dictionary will be provided later in this section), *LiveLight* attempts to sparsely reconstruct query video segment using its atoms. Specifically, sparse reconstruction indicates both small reconstruction error and small footprint on the dictionary, i.e., using as few atoms from the dictionary as possible. Consequently, video summarization is formulated as a sparse coding problem, seeking linear decomposition of data using a few elements from a dictionary learned in online fashion.

We start with discussion of feature representation for video data. Specifically, we adopt the representation based on spatio-temporal cuboids [14, 8, 12], to detect salient points within the video and describe the local spatio-temporal patch around the detected interest points. Different from optical flow, this feature representation only describes spatio-temporal salient regions, instead of the entire frame. On the other hand, spatio-temporal cuboids are less affected by occlusion, a key difficulty in tracking trajectory based representations. Specifically, we adopt the spatio-temporal interest points detected using the method in [8], and describe each detected interest point with histogram of gradient (HoG) and histogram of optical flow (HoF). The feature representation for each detected interest point is then obtained by concatenating the HoG feature vector and HoF feature vector. Finally, each video segment is represented as a collection of feature vectors, corresponding to detected interest points, i.e., $\mathbf{X}_k = \{\mathbf{x}_1, \ldots, \mathbf{x}_{n_k}\}$, where n_k is the number of interest points detected in video segment $\mathbf{X}_k$.

Different from conventional settings of sparse coding, where input signal is a vector, the input signal in our problem is a video segment, represented as a group of vectors $\mathbf{X}_k = \{\mathbf{x}_1, \ldots, \mathbf{x}_{n_k}\}$. Therefore, our goal is to effectively encode groups of instances in terms of a set of dictionary atoms $\mathbf{D} = \{\mathbf{d}_j\}_{j=1}^{|\mathbf{D}|}$, where $|\mathbf{D}|$ is the size of the dictionary, i.e., number of atoms in $\mathbf{D}$. Specifically, given learned dictionary $\mathbf{D}$, *LiveLight* seeks sparse reconstruction of the query segment $\mathbf{X}$, as follows

$$\min_{\mathbf{A}} \frac{1}{2} \frac{1}{|\mathbf{X}|} \sum_{\mathbf{x}_i \in \mathbf{X}} \left\| \mathbf{x}_i - \sum_{j=1}^{|\mathbf{D}|} \boldsymbol{\alpha}_j^i \mathbf{d}_j \right\|_2^2 + \lambda \sum_{j=1}^{|\mathbf{D}|} \|\boldsymbol{\alpha}_j\|_2 \quad (1)$$

where $\mathbf{A} = \{\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^{|\mathbf{X}|}\}$, $\boldsymbol{\alpha}^i \in \mathbb{R}^{|\mathbf{D}|}$ is the reconstruction vector for interest point $\mathbf{x}_i \in \mathbf{X}$, and $|\mathbf{X}|$ is the number of interest points detected within video segment $\mathbf{X}$. The first term in (1) is reconstruction cost. If video segments similar to $\mathbf{X}$ have been observed before, this term should be small, due to the assumption that the learned dictionary represents knowledge in the previously seen video data. The second term is the group sparsity regularization. Since dictionary $\mathbf{D}$ is learned to sparsely reconstruct previously seen video segments, if $\mathbf{X}$ contains no interesting or unseen contents, it should also be sparsely reconstructible using few atoms in $\mathbf{D}$. On the other hand, if contents in $\mathbf{X}$ have never been observed in previous video segments, although it is possible that a fairly small reconstruction cost could be achieved, we

would expect using a large amount of video fragments for this reconstruction, resulting in dense reconstruction weight vectors. Moreover, the special mixed ℓ_1/ℓ_2 norm of $\mathbf{A}$ used in the second term regularizes the number of dictionary atoms used to reconstruct the entire video segment $\mathbf{X}$. This is more preferable over conventional ℓ_1 regularization, as a simple ℓ_1 regularizer only ensures sparse weight vector for each interest point $\mathbf{x}_i \in \mathbf{X}$, but it is highly possible that different interest points will have very different footprint on the dictionary, i.e., using very different atoms for sparse reconstruction. Consequently, reconstruction for the video segment $\mathbf{X}$ could still involve large number of atoms in $\mathbf{D}$. On the other hand, the ℓ_1/ℓ_2 regularizer ensures a small footprint of the entire video segment $\mathbf{X}$, as all interest points within segment $\mathbf{X}$ are regularized to use the same group of atoms for reconstruction. Moreover, the tradeoff between accurate reconstruction and compact encoding is controlled by regularization parameter λ. Finally, we denote the value of (1) with optimal reconstruction matrix $\mathbf{A}$ as ϵ, which is used in Algorithm 1 to decide if segment $\mathbf{X}$ should be incorporated into the summary video.

Consequently, *LiveLight* encapsulates the following intuitions for what we would think of a video summary. Given a dictionary optimized to sparsely reconstruct previously seen video contents, a new segment exhibiting similar contents seen in previous video data should be reconstructible from a small number of such atoms. On the other hand, a video segment unveiling contents never seen before is either not reconstructible from the dictionary of previous video segments with small error, or, even if it is reconstructible, it would necessarily build on a combination of a large number of atoms in the dictionary. Crucial to this technique, is the ability to learn a good dictionary of atoms representing contents seen in previous video segments, and being able to update the dictionary online to adapt to changing content of the video, which we discuss in detail later in this section.

2.1.1 Optimization

To find the optimal reconstruction vectors $\{\boldsymbol{\alpha}^i\}$ for interest points in $\mathbf{X}$, we need to solve problem (1). We employ *alternating direction method of multipliers (ADMM)* [4] to carry out such optimization, due to its efficiency. Specifically, *ADMM* consists of the following iterations:

$$\forall i\colon\ \boldsymbol{\alpha}^i_{k+1} = \left[\frac{\mathbf{D}^\top\mathbf{D}}{|\mathbf{X}|} + \rho\mathbf{I}\right]^{-1}\left[\frac{\mathbf{D}^\top\mathbf{x}_i}{|\mathbf{X}|} + \rho(\mathbf{z}^i_k - \mathbf{u}^i_k)\right] \quad (2)$$

$$\forall j\colon\ \mathbf{z}_{j,k+1} = \mathcal{S}_{\lambda/\rho}(\boldsymbol{\alpha}_{j,k+1} + \mathbf{u}_{j,k}) \quad (3)$$

$$\forall i\colon\ \mathbf{u}^i_{k+1} = \mathbf{u}^i_k + \boldsymbol{\alpha}^i_{k+1} - \mathbf{z}^i_{k+1} \quad (4)$$

and alternates among the above updates until convergence, where $\rho > 0$ is called the penalty parameter, $\mathbf{I} \in \mathbb{R}^{|\mathbf{D}|\times|\mathbf{D}|}$ is the identity matrix, $\mathcal{S}$ is the soft-thresholding operator defined as $\mathcal{S}_\kappa(a) = \max\{(1-\kappa/|a|), 0\}a$, $\mathbf{A} = \{\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^{|\mathbf{X}|}\}$, $\mathbf{Z} = \{\mathbf{z}^1, \ldots, \mathbf{z}^{|\mathbf{X}|}\} = \{\mathbf{z}_1^\top, \ldots, \mathbf{z}_{|\mathbf{D}|}^\top\}^\top$, $\mathbf{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^{|\mathbf{X}|}\}$, index i runs through $\{1, \ldots, |\mathbf{X}|\}$, index j runs through $\{1, \ldots, |\mathbf{D}|\}$ and k is the iteration counter. Both $\mathbf{Z}$ update (3) and $\mathbf{U}$ update (4) are trivial to compute. The $\mathbf{A}$ update in (2) can be accelerated via techniques such as warm start, caching factorization and fast matrix inversion as discussed in [4].

2.1.2 Learning Initial Dictionary

In this section, we discuss how to learn an initial dictionary, necessary to launch the *LiveLight* algorithm. Specifically, we would like a learning method that facilitates both induction of new dictionary atoms and removal of dictionary atoms with low predictive power. To achieve this goal, we again apply ℓ_1/ℓ_2 regularization, but this time to dictionary atoms. The idea for this regularization is that uninformative dictionary atoms will be regularized towards $\mathbf{0}$, effectively removing it from the dictionary. Given first few video segments $\mathcal{X}_0 = \{\mathbf{X}_1, \ldots, \mathbf{X}_m\}$, we formulate learning optimal initial dictionary as follows

$$\min_{\mathbf{D}, \{\mathbf{A}_1, \ldots, \mathbf{A}_m\}} \frac{1}{m} \sum_{\mathbf{X}_k \in \mathcal{X}_0} J(\mathbf{X}_k, \mathbf{A}_k, \mathbf{D}) + \gamma \sum_{j=1}^{|\mathbf{D}|} \|\mathbf{d}_j\|_2 \quad (5)$$

where $J(\mathbf{X}_k, \mathbf{A}_k, \mathbf{D})$ is the objective function in (1), and γ balances sparse reconstruction quality and dictionary size. Though non-convex to $\mathbf{D}$ and $\{\mathbf{A}_1, \ldots, \mathbf{A}_m\}$ jointly, (5) is convex w.r.t. $\{\mathbf{A}_1, \ldots, \mathbf{A}_m\}$ when $\mathbf{D}$ is fixed, and also convex w.r.t. $\mathbf{D}$ with fixed $\{\mathbf{A}_1, \ldots, \mathbf{A}_m\}$. A natural solution is to alternate between these two variables, optimizing one while clamping the other. Specifically, with fixed dictionary $\mathbf{D}$, each $\mathbf{A}_k \in \{\mathbf{A}_1, \ldots, \mathbf{A}_m\}$ can be optimized individually, using optimization method described in the previous section. On the other hand, with fixed $\{\mathbf{A}_1, \ldots, \mathbf{A}_m\}$, optimizing dictionary $\mathbf{D}$ can be similarly solved via *ADMM*. Due to limit of space, we omit details for this optimization.

2.2. Online Dictionary Update

As *LiveLight* scans through the video, segments that cannot be sparsely reconstructed using current dictionary, indicating unseen and interesting contents, are incorporated into the summary video. However, all following occurrences of similar contents appearing in later video segments, should ideally be excluded. Consequently, it is crucial to update the dictionary such that those video segments already included in the summary video should no longer result in large reconstruction error. Assume the current version of summary is $\mathcal{Z}_t$, composed of t video segments $\{\mathbf{X}_k\}_{k=1}^t$, then the optimal dictionary is the solution of the following problem

$$\min_{\mathbf{D}} f(\mathbf{D}) = \min_{\mathbf{A}_1, \ldots, \mathbf{A}_t} \frac{1}{t} \sum_{\mathbf{X}_k \in \mathcal{Z}_t} J(\mathbf{X}_k, \mathbf{A}_k, \mathbf{D}) + \gamma \sum_{j=1}^{|\mathbf{D}|} \|\mathbf{d}_j\|_2 \quad (6)$$

where we need to store feature representations $\{\mathbf{X}_k\}_{k=1}^t$ for all t segments in $\mathcal{Z}_t$. This might not cause problem for short videos, however, for hours of videos, especially surveillance videos running endlessly, storing these feature representations requires huge space. Moreover, solving the above optimization problem from scratch using alternating optimization for each dictionary update, is extremely time consuming, and would hinder the algorithm from applicable to real world consumer videos. Therefore, *LiveLight* employs online learning for approximate and efficient dictionary update [19]. Specifically, instead of optimizing dictionary $\mathbf{D}$ and reconstruction coefficients $\{\mathbf{A}_1, \ldots, \mathbf{A}_t\}$ simultaneously, *LiveLight* aggregates the past information computed during the previous steps of the algorithm, namely the reconstruction coefficients $\{\hat{\mathbf{A}}_1, \ldots, \hat{\mathbf{A}}_t\}$ computed using previous versions of dictionary, and only optimizes $\mathbf{D}$ in problem (6). Therefore, the online dictionary update seeks to solve the following approximate optimization problem

$$\min_{\mathbf{D}} \hat{f}(\mathbf{D}) = \frac{1}{t} \sum_{\mathbf{X}_k \in \mathcal{Z}_t} J(\mathbf{X}_k, \hat{\mathbf{A}}_k, \mathbf{D}) + \gamma \sum_{j=1}^{|\mathbf{D}|} \|\mathbf{d}_j\|_2 \quad (7)$$

It is easy to see that $\hat{f}(\mathbf{D})$ upper bounds $f(\mathbf{D})$ in problem (6). Moreover, theoretical analysis shown in the next section guarantees that $\hat{f}(\mathbf{D})$ and $f(\mathbf{D})$ converges to the same limit and consequently $\hat{f}(\mathbf{D})$ acts as a surrogate for $f(\mathbf{D})$. Moreover, it is easy to show that problem (7) could be equivalently reformulated as follows

$$\min_{\mathbf{D}} \frac{1}{2t} Tr(\mathbf{D}^\top \mathbf{D} \mathbf{P}_t) - \frac{1}{t} Tr(\mathbf{D}^\top \mathbf{Q}_t) + \gamma \sum_{j=1}^{|\mathbf{D}|} \|\mathbf{d}_j\|_2 \quad (8)$$

where $Tr(\cdot)$ is matrix trace, $\mathbf{P}_t$ and $\mathbf{Q}_t$ are defined as

$$\mathbf{P}_t = \sum_{k=1}^{t} \sum_{\alpha^i \in \mathbf{A}_k} \boldsymbol{\alpha}^i {\boldsymbol{\alpha}^i}^\top, \ \mathbf{Q}_t = \sum_{k=1}^{t} \sum_{\alpha^i \in \mathbf{A}_k} \mathbf{x}_i {\boldsymbol{\alpha}^i}^\top \quad (9)$$

Therefore, there is no need to store $\{\hat{\mathbf{A}}_k\}_{k=1}^t$ or $\{\mathbf{X}_k\}_{k=1}^t$, as all necessary information is stored in $\mathbf{P}_t$ and $\mathbf{Q}_t$. Finally, problem (8) could be efficiently solved using *ADMM*.

2.3. Importance of Dictionary

Very recently, there have been attempts employing the idea of sparse reconstruction for video summarization [11, 6]. However, those approaches use the entire video itself as basis for reconstruction, instead of learning and updating a dictionary as concise summary of video contents. Using the entire video as reconstruction basis [11, 6], significantly increases the complexity of optimization and computational time, as shown later in the experiments, the approach in [6] takes nearly 10 times more CPU time than *LiveLight* on the same videos. Such heavy computational footprint hinders those approaches from being applied in temporally long consumer videos (actually, [11, 6] only used videos with at most several minutes in their empirical study). Moreover, [11, 6] have to see the entire video before starting to generate summary, eliminating the possibility of real-time summarization. On the other hand, the dictionary learned and updated in *LiveLight* concisely summarizes the contents of seen video, significantly reduces computational cost, and captures any concept drift in video streams.

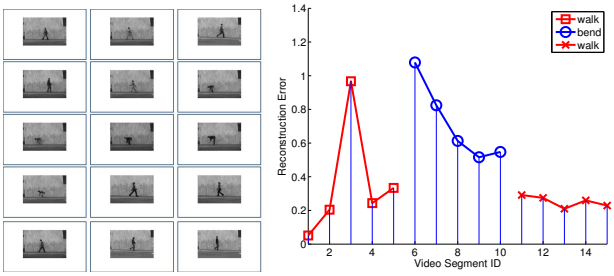

Figure 1. (Left) 15 video segments; (Right) reconstruction error for each video segment.

2.4. Sanity Check

We use synthetic video to perform sanity check on *LiveLight*. Specifically, we use two types of video sequences from *Weizmann* action recognition data [12], i.e., *walk* and *bend*. The synthetic video is constructed by combining 5 *walk* sequences, followed by 5 *bend* sequences, and 5 more *walk* sequences. Details of this synthetic video are shown in Figure 1. *LiveLight* learns initial dictionary using the first *walk* sequence, and carries out reconstruction and online dictionary update on the rest 14 video sequences. There are 2 clear peaks in Figure 1, corresponding to the third *walk* sequence, which is the first occurrence of walking from left to right (the first and second sequences are both walking from right to left), and the first *bend* sequence. Moreover, the reconstruction error for the fourth *walk* sequence, which also shows walking from left to right, is significantly smaller then the third *walk* sequence, indicating the dictionary has learned the contents of walking to the right, through online dictionary update. Finally, the last 5 *walk* sequences all result in small reconstruction errors, even after *LiveLight* has just observed 5 *bend* sequences, showing that the dictionary retains its knowledge about *walk*.

3. Theoretical Analysis

We first study the convergence property of online dictionary update. Specifically, we have the following theorem,

Theorem 1 *Denote the sequence of dictionaries learned in LiveLight as* $\{\mathbf{D}_t\}$*, where* $\mathbf{D}_1$ *is the initial dictionary. Then* $\hat{f}(\mathbf{D})$*, defined in (7), is the surrogate function of* $f(\mathbf{D})$*, defined in (6), satisfying*

(1) $f(\mathbf{D}) - \hat{f}(\mathbf{D})$ *converges to 0 almost surely;*

(2) $\mathbf{D}_t$ *obtained by optimizing* $\hat{f}$ *is asymptotically close to the set of stationary points of (6) with probability 1*

Theorem 1 guarantees that $\hat{f}(\mathbf{D})$ could be used as a proper surrogate for $f(\mathbf{D})$, such that we could optimize (7) to obtain the optimal dictionary efficiently, instead of solving the much more time-consuming optimization problem (6).

Next, we study the generalization ability of *LiveLight* on unseen video segments. Specifically, as *LiveLight* scans through the video sequence, the dictionary is learned and updated only using video segments seen so far. Consequently, the dictionary is optimized to sparsely reconstruct contents in seen video segments. It is crucial for *LiveLight* to also be able to sparsely reconstruct unseen video segments, composed of contents similar to video segments seen before. This property is called generalization ability in statistical machine learning terminology. Specifically,

Theorem 2 *Assume data points* $\mathbf{X}$ *(i.e., video segments) are generated from unknown probability distribution* $\mathcal{P}$*. Given* t *observations* $\{\mathbf{X}_1, \ldots, \mathbf{X}_t\}$*, for any dictionary* $\mathbf{D}$*, and any fixed* $\delta > 0$*, with probability at least* $1 - \delta$

$$\mathbb{E}_{\mathbf{X}\sim\mathcal{P}} J^*(\mathbf{X}, \mathbf{D}) - \frac{1}{t}\sum_{k=1}^{t} J^*(\mathbf{X}_k, \mathbf{D}) \leq \epsilon(t, \delta) \qquad (10)$$

where $J^*(\mathbf{X}, \mathbf{D}) = \min_{\mathbf{A}} J(\mathbf{X}, \mathbf{A}, \mathbf{D})$ *is the minimal reconstruction error for* $\mathbf{X}$ *using dictionary* $\mathbf{D}$*, as defined in (1), and* $\epsilon(t, \delta) = o(\ln t/\sqrt{t})$ *is a small constant that decreases as* t *increases.*

The above theorem is true for any dictionary $\mathbf{D}$, and obviously also true for the dictionary learned in *LiveLight*. Therefore, Theorem 2 guarantees that if dictionary $\mathbf{D}$ has small reconstruction error on previously seen video segments, it will also result in small reconstruction error for unseen video segments with similar contents.

4. Experiments

We test the performance of *LiveLight* on more than 12 hours of consumer videos, including both YouTube videos and surveillance videos. The 20 videos in our data set span a wide variety of scenarios: indoor and outdoor, moving camera and still camera, with and without camera zoom in/out, with different categories of targets (human, vehicles, planes, animals etc.) and covers a wide variety of activities and environmental conditions. Details about the data set are provided in Table 1.

4.1. Experiment Design and Evaluation

We compare *LiveLight* with several other methods, including evenly spaced segments, K-means clustering [6] using the same features as our method, and *DSVS* algorithm proposed in [6], state-of-the-art method for video summarization. It is shown in [6] that *DSVS* already beats color-histogram based method [25] and motion-based method [18]. Parameters for various algorithms are set such that the length of generated summary videos are the same as ground truth video. For *LiveLight*, we fix the number of atoms in dictionary to 200, though better performance is possible with fine tuning of parameters.

For each video in our data set, three judges selected segments from original video to compose their preferred version of summary video. The final ground truth is then constructed by pooling together those segments selected by at least two judges. Following [6], to quantitatively determine the overlap between algorithm generated summary and ground truth, both video segment content and time differences are considered. Specifically, two video segments

Video	Time	Frames	CamMo	Zoom
CarRace	34.00	61133	Yes	No
FirefighterSave	21.53	38714	Yes	Yes
MonsterTruck	50.85	91430	Yes	No
StockCar	38.91	69963	Yes	No
SpeedBoat	64.03	115249	Yes	Yes
DogSwimming	20.08	36106	No	No
PetEvent	33.63	60472	Yes	Yes
HorseTraining	20.67	37171	Yes	No
ShowJumping	20.73	31087	Yes	Yes
Snorkeling	21.92	39463	Yes	No
DisneyParade	29.10	50694	Yes	No
ShamuShow	21.08	37845	Yes	No
BoatTour	28.36	51043	Yes	Yes
AirShow	12.04	21626	Yes	Yes
PolicePullOver	46.58	83761	Yes	No
SubwayExit	43.27	64902	No	No
SubwayEntrance	96.17	144249	No	No
Mall-1	55.44	83156	No	No
Mall-2	39.98	59969	No	No
Mall-3	60.35	90525	No	No

Table 1. Data set details. The first 15 videos are downloaded from YouTube, and the last 5 videos are from surveillance cameras. Video length (Time) is measured in minutes. *CamMo* stands for camera motion, and *Zoom* means camera zoom in/out.

	T	LL(%)	ES(%)	CL(%)	DSVS(%)
CarRace	60.7	**59.80**	40.53	44.98	58.04
FirefighterSave	59.4	**66.84**	38.22	52.53	55.83
MonsterTruck	61.6	**54.38**	42.05	49.35	46.36
StockCar	60.2	**61.30**	35.55	40.03	48.36
SpeedBoat	59.3	75.55	45.03	59.53	**77.07**
DogSwimming	62.2	**79.10**	42.93	54.34	64.57
PetEvent	61.9	**58.32**	36.51	54.12	43.93
HorseTraining	60.3	**66.17**	54.06	53.90	60.63
ShowJumping	60.6	**46.86**	39.93	37.13	44.22
Snorkeling	60.4	60.10	37.58	**62.09**	56.83
DisneyParade	58.5	**67.69**	39.49	62.56	67.17
ShamuShow	59.0	**61.36**	34.75	21.02	48.75
BoatTour	59.9	**59.28**	33.89	40.57	50.93
AirShow	36.8	**85.33**	49.46	74.46	72.29
PolicePullOver	61.7	**91.41**	30.63	60.45	76.75
SubwayExit	89.1	**91.87**	21.55	47.70	85.03
SubwayEntrance	92.1	**80.56**	42.56	43.65	73.85
Mall-1	89.7	**94.98**	37.46	57.08	85.40
Mall-2	88.4	**96.83**	43.67	61.99	87.72
Mall-3	92.0	**88.26**	38.70	58.59	81.99
Average	-	**72.30**	39.23	51.80	64.29

Table 2. T is the length (seconds) of summary video. LL: LiveLight; ES: evenly spaced segments; CL: K-Means Clustering; DSVS: sparse reconstruction using original video as basis [6].

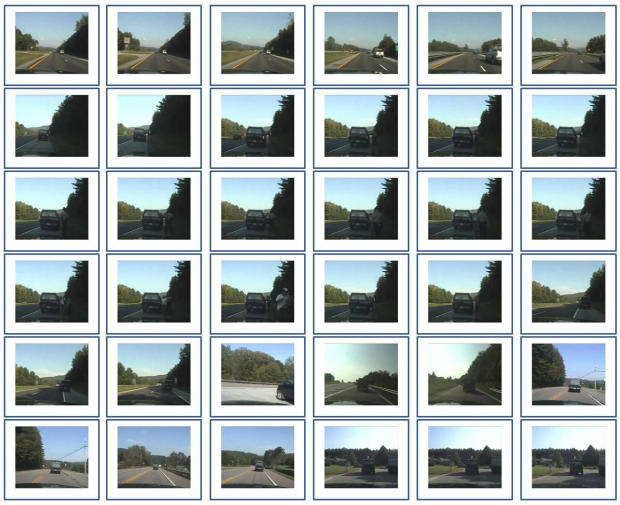

Figure 2. (Best viewed in color and zoom-in.) Some frames of the summary video generated by *LiveLight* for a YouTube video showing police pulling over a black SUV and making arrest (frames are organized from left to right, then top to bottom in temporal order). From the summary video, we could see the following storyline of the video: (1) Police car travels on the highway; (2) Police car pulls over black SUV; (3) Police officer talks to passenger in the SUV; (4) Two police officers walk up to the SUV, and open the passenger side door of the SUV; (5) Police officer makes arrest of a man in white shirt; (6) Police officer talks to passenger in the SUV again; (7) Both police car and black SUV pull into highway traffic; (8) Police car follows black SUV off the highway; (9) Both vehicles travel in local traffic; (10) Black SUV pulls into local community.

must occur within a short period of time (two seconds in our experiments), and must be similar in scene content and motion pattern to be considered equivalent. Final accuracy is computed as the ratio of segments in algorithm generated summary video that overlaps with ground truth.

4.2. Results

According to the quantitative comparison provided in Table 2, we have following observations: (1) *LiveLight* achieves highest accuracy on 18 out of 20 videos, and in most cases beats competing algorithms with a significant margin; (2) On the 5 surveillance videos, both *LiveLight* and *DSVS* outperform other two algorithms, showing the advantage of sparse reconstruction based methods on summarizing surveillance videos; (3) Averaged across 20 videos, *LiveLight* outperforms the state-of-the-art summarization method *DSVS* by 8%, revealing the advantage of *LiveLight*.

Besides quantitative measures, we also show the automatically generated summary ("trailer") for YouTube video *PolicePullOver* (more summary videos are provided in supplementary material). As shown in Figure 2, the summary video captures the entire story line of this near hour long video, achieving more than 40 times compression in time without losing semantic understandability of the summary video. Moreover, the background in this video involves various cars passing in both directions, and it is interesting that *LiveLight* is not affected by this background motion.

4.3. Time Complexity

LiveLight is implemented using MATLAB 7.12 on a 3.40 GHZ Intel Core i7 PC with 16.0 GB main memory. Table 3 compares the processing time of various algorithms, with the following observations: (1) The last column under *LiveLight* shows the ratio between its computational time and video length. For all videos, this ratio is less than 2, and for 6 videos even less than 1. Thus, with MATLAB implementation on a conventional PC, *LiveLight* already achieves near real-time speed, further revealing its promise in real world video analysis applications; (2) *LiveLight* is nearly 10 times faster than *DSVS*, revealing the advantage of learning and updating dictionary in an online fashion, instead of using original video as basis for sparse reconstruction.

5. Conclusions

In this paper, we propose *LiveLight* to generate short video summarizing the most important and interesting contents, of a potentially very long video. *LiveLight* enables viewer to understand the video without watching the entire sequence. Theoretical analysis is provided, focusing on online dictionary update convergence, and generalization ability to unseen video segments. Experiments on real world surveillance videos and YouTube videos demonstrate the effectiveness and efficiency of *LiveLight*. The fact that *LiveLight* is quasi real-time on all tested videos shows its promise on summarizing the huge body of consumer videos.

Acknowledgements

This research is supported by Google, NSF IIS-0713379, ONR N000140910758, and AFOSR FA9550010247.

References

[1] A. Aner and J. Kender. Video summaries through mosaic-based shot and scene clustering. In *ECCV*, 2002. 2

[2] N. Babaguchi. Towards abstracting sports video by highlights. In *ICME*, 2000. 2

[3] S. Bengio, F. Pereira, Y. Singer, and D. Strelow. Group sparse coding. In *NIPS*, 2009. 2

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, 2011. 4

[5] Y. Caspi, A. Axelrod, Y. Matsushita, and A. Gamliel. Dynamic stills and clip trailers. *Vis. Comput.*, 22(9):642–652, 2006. 2

[6] Y. Cong, J. Yuan, and J. Luo. Towards scalable summarization of consumer videos via sparse dictionary selection. *TMM*, 14(1):66–75, 2012. 2, 5, 6

[7] D. DeMenthon, V. Kobla, and D. Doermann. Video summarization by curve simplification. In *ACM MM*, 1998. 2

[8] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS*, 2005. 3

	Video	LiveLight				EvenlySpaced		Clustering		DSVS	
	T_{video}	T_1	T_2	T_{total}	Ratio	T_{total}	Ratio	T_{total}	Ratio	T_{total}	Ratio
CarRace	34.00	42.73	10.18	52.91	**1.56**	0.52	0.02	114.03	3.35	636.7	18.73
FirefighterSave	21.53	27.26	8.54	35.80	**1.66**	0.51	0.02	78.92	3.67	434.8	20.20
MonsterTruck	50.85	60.23	11.60	71.83	**1.41**	0.59	0.01	162.23	3.19	783.3	15.40
StockCar	38.91	51.95	10.08	62.03	**1.59**	0.50	0.01	150.59	3.87	672.2	17.27
SpeedBoat	64.03	39.65	9.31	48.96	**0.76**	0.54	0.01	102.56	1.60	460.2	7.19
DogSwimming	20.08	5.16	6.93	12.09	**0.60**	0.53	0.03	33.46	1.67	174.7	8.70
PetEvent	33.63	34.27	8.24	42.51	**1.26**	0.60	0.02	101.77	3.03	516.2	15.35
HorseTraining	20.67	9.82	7.64	17.46	**0.84**	0.56	0.03	34.84	1.69	271.8	13.15
ShowJumping	20.73	23.33	7.57	30.90	**1.49**	0.55	0.03	79.72	3.85	247.2	11.93
Snorkeling	21.92	28.30	9.13	37.43	**1.71**	0.54	0.03	88.19	4.02	536.6	24.48
DisneyParade	29.10	37.53	9.48	47.01	**1.62**	0.49	0.02	123.96	4.26	535.9	18.42
ShamuShow	21.08	17.91	7.83	25.74	**1.22**	0.55	0.03	51.46	2.44	399.0	18.93
BoatTour	28.36	25.56	9.13	34.69	**1.22**	0.50	0.02	80.48	2.84	379.8	13.39
AirShow	12.04	8.37	7.24	15.61	**1.30**	0.31	0.03	42.34	3.52	225.0	18.68
PolicePullOver	46.58	16.07	8.33	24.40	**0.52**	0.56	0.01	53.24	1.14	493.3	10.59
SubwayExit	43.27	14.46	6.67	21.13	**0.49**	0.75	0.02	148.8	3.44	558.6	12.91
SubwayEntrance	96.17	41.30	9.55	50.85	**0.53**	0.84	0.01	304.15	3.16	1299.1	13.51
Mall-1	55.44	51.02	9.93	60.95	**1.10**	0.82	0.02	139.23	2.51	646.4	11.66
Mall-2	39.98	33.49	8.95	42.44	**1.06**	0.80	0.02	94.75	2.37	436.5	10.92
Mall-3	60.35	58.02	10.31	68.33	**1.13**	0.78	0.01	137.20	2.27	698.4	11.57

Table 3. Processing time of *LiveLight* and competing algorithms (all time shown is in minutes). T_{video} is the length of original video. T_1 is the time spent on generating feature representations in *LiveLight*, and T_2 is the combined time spent on learning initial dictionary, video segment reconstruction and online dictionary update. $T_{total} = T_1 + T_2$ is the total processing time of *LiveLight*, and $Ratio = T_{total}/T_{video}$ for all algorithms.

[9] F. Dufaux. Key frame selection to represent a video. In *ICIP*, 2000. 2

[10] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *TIP*, 54(12):3736–3745, 2006. 2

[11] E. Elhamifar, G. Sapiro, and R. Vidal. See all by looking at a few: Sparse modeling for finding representative objects. In *CVPR*, 2012. 1, 2, 5

[12] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. *PAMI*, 29(12):2247–2253, 2007. 3, 5

[13] A. Hanjalic and H. Zhang. An integrated scheme for automated video abstraction based on unsupervised cluster-validity analysis. *IEEE TSCVT*, 9:1280–1289, 1999. 2

[14] I. Laptev. On space-time interest points. *IJCV*, 64:107–123, 2005. 3

[15] Y. Lee, J. Ghosh, and K. Grauman. Discovering important people and objects for egocentric video summarization. In *CVPR*, 2012. 1, 2

[16] B. Li and I. Sezan. Event detection and summarization in american football broadcast video. In *SPIE Storage ad Retrieval for Media Databases*, 2002. 2

[17] D. Liu, G. Hua, , and T. Chen. A hierarchical visual model for video object summarization. *PAMI*, 2009. 2

[18] J. Luo, C. Papin, and K. Costello. Towards extracting semantically meaningful key frames from personal video clips: from humans to computers. *TCSVT*, 19(2):289–301, 2009. 6

[19] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *JMLR*, 11:19–60, 2010. 5

[20] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *ICCV*, 2009. 2

[21] J. Oh, Q. Wen, J. lee, and S. Hwang. Video abstraction. *Video Data Mangement and Information Retrieval*, pages 321–346, 2004. 2

[22] B. Olshausen and D. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37:3311–3325, 1997. 2

[23] S. Pfeiffer, R. Lienhart, S. Fischer, and W. Effelsberg. Abstracting digital movies automatically. *Journal of Visual Communication and Image Representation*, 7(4):345–353, 1996. 2

[24] Y. Pritch, A. Rav-Acha, A. Gutman, and S. Peleg. Webcam synopsis: Peeking around the world. In *ICCV*, 2007. 2

[25] Z. Rasheed and M. Shah. Detection and representation of scenes in videos. *TMM*, 7(6):1097–1105, 2005. 6

[26] A. Rav-Acha, Y. Pritch, and S. Peleg. Making a long video short. In *CVPR*, 2006. 2

[27] D. Simakov, Y. Caspi, E. Shechtman, and M. Irani. Summarizing visual data using bidirectional similarity. In *CVPR*, 2008. 2

[28] M. Smith and T. Kanade. Video skimming and characterization through the combination of image and language understanding. In *CVPR*, 1997. 2

[29] C. Taskiran, A. Amir, D. Ponceleon, and E. Delp. Automated video summarization using speech transcripts. In *SPIE*, 2002. 2

[30] B. Truong and S. Venkatesh. Video abstraction: A systematic review and classification. *ACM Trans. Multimedia Comput. Commun. Appl.*, 3(1), 2007. 2

[31] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *PAMI*, 31(2):210–227, 2009. 2

[32] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009. 2

[33] H. Zhang. An integrated system for content-based video retrieval and browsing. *Pattern Recognition*, 30(4):643–658, 1997. 1, 2

[34] B. Zhao, L. Fei-Fei, and E. Xing. Online detection of unusual events in videos via dynamic sparse coding. In *CVPR*, 2011. 2