Multi-Video Browsing and Summarization

Kevin Dale[1] Eli Shechtman[2] Shai Avidan[3] Hanspeter Pfister[1]

[1]Harvard University [2]Adobe Systems, Inc. [3]Tel-Aviv University

{kdale,pfister}@seas.harvard.edu elishe@adobe.com avidan@eng.tau.ac.il

Abstract

We propose a method for browsing multiple videos with a common theme, such as the result of a search query on a video sharing website, or videos of an event covered by multiple cameras. Given the collection of videos we first align each video with all others. This pairwise video alignment forms the basis of a novel browsing interface, termed the Browsing Companion. It is used to play a primary video and, in addition as thumbnails, other video clips that are temporally synchronized with it. The user can, at any time, click on one of the thumbnails to make it the primary. We also show that video alignment can be used for other applications such as automatic highlight detection and multi-video summarization.

1. Introduction

We are drowning in video. The amount of online video is growing at such a rapid pace that browsing and searching it is becoming an increasingly frustrating task. At the same time we are able to easily capture videos with cell phones and other portable devices any time we like. What is required are new approaches to summarize video in intelligent ways for faster browsing.

Here we focus on the case of a small collection of multiple videos with a common theme. This case arises naturally in a number of scenarios–results from a search query on YouTube, an event covered by multiple cameras, a collection of clips from a single camera or smart phone taken at an event, a set of episodes of a TV show, or multiple takes of the same shot on a film set.

In a pre-processing stage we align all pairs of videos to each other. This is essentially a search operation that suggests, for every frame in one video (the *primary video*), similar frames in each of the other videos (the *secondary videos*). At run-time, the user watches a primary video in a novel user interface we term the Browsing Companion, that also displays thumbnails from these aligned secondaries (see Fig. 1). As a result, the user has access to similar content at all times, and can switch videos instantaneously, e.g., to get a better angle of a particular event or to watch similar events.

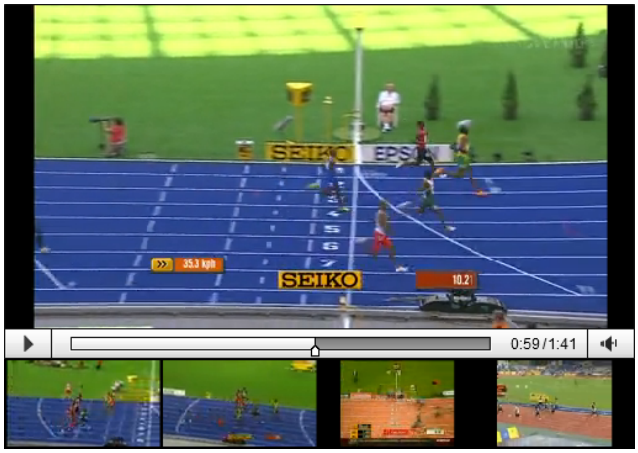


Figure 1: *The Browsing Companion with a number of videos from a YouTube search query of "Usain Bolt". As the user is watching the race coming to the finish line, she can easily switch to finishes in other videos.*

In this way, the Browsing Companion exposes similarities across videos in a simple way that allows users to quickly and efficiently find content of interest, and by parallelizing, to some degree, what would otherwise be a tedious, serial process of exploring the collection, video by video. We do not expect users to absorb all video streams simultaneously. Rather, dynamic secondary thumbnails provide constantly-updated links to content similar to the primary, so that, when something of interest is found in the primary window via typical video navigation controls, the user can jump to related content from a secondary window. Put another way, with the effort to find something of interest in a single video, the Browser can locate multiple instances across multiple videos, automatically and instantaneously.

For example, by finding one monologue in a collection of Conan OBrien episodes, the user has immediate access to other monologues and can navigate freely among them. With the same setup, a different user can immediately hone in on interview footage, and another, on musical performances. Alternatively, consider Usain Bolt, who famously slowed down right before the finish line during the 100m final at the 2008 Olympics. Has he slowed down like this

in the past? Instead of searching through each YouTube clip, one at a time, a user can find multiple race finishes simultaneously, and even scrub across the finish line with the playhead slider.

Video alignment provides useful information for other applications. The key observation we make is that now it is possible to determine, for each frame, how it relates to the rest of the *entire video collection*, not just its own or the secondary video. This is an important distinction, because now we can ask questions such as: "Find me a highlight that is unique within each video, but common across videos", or "Generate a summary of the *entire* video set".

Our methods for browsing and summarization are intended to augment large-scale video search. For example, consider YouTube's search, or its recommendation system, which provides easy access to videos that are potentially similar to the current video. Here similarity is only exploited at the granularity of the video, and the user still must scan each query result or related video in turn to find relevant content. Alternatively, consider Video Google [Sivic and Zisserman 2003], a method for video search which first requires shot boundary detection to split the video(s) into discrete clips, as well as an explicit example query from the user, and involves static image previews of search results. As such, it's not well-suited to browsing unedited video with no shot boundaries or exploratory browsing when the contents of the collection are unknown and a query can't be formulated a priori.

Contributions. Novel aspects of the Browsing Companion include (1) concurrent preview of temporally aligned videos, which exposes similarities across multiple videos in a simple way; (2) continuous query-by-example, whereby the current frame of the primary video serves as an ever-updating search query, and the aligned secondaries, corresponding real-time results; and (3) dynamic hyperlinks for video, where these clickable secondary videos provide links among related video content analagous to hyperlinks in text documents. Additionally, alignment allows for new modes of video summarization, particularly (4) our 'unique-within, common-across' video highlight criteria.

2. Related Work

Video search has been actively investigated in recent years. For example, in Video Google [19] the video is first indexed and then the user can pick an object/face and get pointers to other locations in the video with the same object. This is suitable for well structured objects (with many features) but is not ideal for general scenes. The addition of temporal information was later suggested by [10], but with an eye towards action recognition.

We, on the other hand, consider search in the context of browsing. Much existing work on video browsing is limited to domain-specific applications [23] or focuses on browsing static keyframes extracted after explicit shot boundary detection [2, 13]. Our approach is broadly applicable to many different types of video, and we do not assume that the video is organized in shots, nor do we attempt to detect shot boundaries.

There are two prevailing approaches to *video summarization*: key-frame abstraction and video skimming. In key-frame abstraction, the video is reduced to a collection of key frames that capture the most "important" aspect of the video [8, 9, 1]. Video skimming, in its different forms, creates a single or multiple short video clips out of the original video using multiple cues such as video, audio, and captions [22, 15]. Related methods for adaptive fast-forward [14, 6] adjust the video's framerate adaptively based on content for quick previewing. Other work uses explicit shot boundary detection as a basis for generating user-controlled skimming summaries [7]. Summarization may involve additional highlight detection, which is available in several commercial products. A leading approach in this field is to extract visual and possibly audio cues, and to apply an HMM to detect sports highlights [26] or unusual events [27]. Many of these existing summarization methods apply straightforwardly to the multi-video case. However, as we will show, multi-video analysis with our all-pairs alignment method reveals information about a collection that cannot be extracted from single-video methods alone.

There has also been some work on spatially and temporally *aligning two videos*, both for sequences captured simultaneously [5, 24] and for those captured at different times [17, 25]. Others have utilized pairwise video alignment inspired by methods in genomics for duplicate video detection [4]. These methods are concerned with two videos of the same scene or action, or two possible duplicates, and do not consider the case of multiple heterogeneous videos of similar content that must be aligned, as we do here. Our alignment is most similar to the use of Dynamic Time Warping (DTW) for speech alignment in audio [16].

There has also been much work in *video analytics* on analysis, detection and retrieval in large video collections. The focus in this body of work is exemplified by the TRECVID Video Retrieval Evaluation program [21]. Our work differs from the TRECVID program and from similar work in video analytics in that we do not assume a predefined set of topics to learn or search for, or a query video per-se.

3. Multi-Video Browsing

Multi-Video Browsing consists of three ingredients. A method for temporally synchronizing a pair of video sequences, a frame similarity measure, and a browsing interface that makes use of the alignment information. Fig. 1 gives an example of the Browsing Companion interface. The main window plays the video selected by the user, the thumbnails play footage from other videos that are synchro-

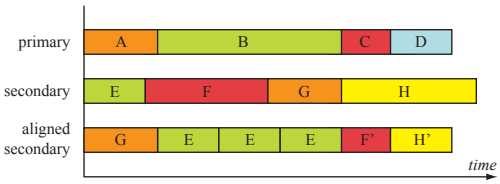


Figure 2: *Overview of video alignment. Given a primary and secondary video, the process generates a concatenation of clips from the secondary video that, when played alongside the primary, shows similar content. Here color encodes appearance. Clips in the aligned secondary can be reordered (G), duplicated (E), and abbreviated (F') with respect to the original secondary. The output may also contain a best guess (H') when there are no good matches for portions of the primary (D).*

nized with the primary video. The user can make any of the secondaries the primary with a click of the mouse, or set the audio focus to any secondary window to preview that clip with sound. The key to the interface is that the secondaries are properly synchronized to the primary through our video alignment process.

Video alignment. An HMM is a statistical model of a Markov process with unobserved state. The unobserved variables are related to their neighbors with a transition probability matrix, and to a sequence of observed variables. In the context of video alignment, we model the output video frames as unobserved random variables that are sampled from a long input video with some transition probabilities to jump from one frame to another. Intuitively this means that the frame will have high probability to follow its natural consecutive neighbor from the long video, but also a small probability for an abrupt transition to a different frame (a "cut").

Output video frames should also be similar to the frames of a reference video (the observed variables) to obtain good alignment. HMMs can be used both for learning the model parameters and for doing inference on new examples. For video alignment we fix the transition matrix and the similarities to the reference sequence to obtain some desired alignment properties. We perform inference using the Viterbi algorithm, a variant of dynamic programming, to find the best alignment between the two video sequences (see Fig. 2).

Formally, let $X = X_1, \cdots, X_n$ and $Y = Y_1, \cdots, Y_m$ denote two video sequences (primary X and secondary Y, respectively) that we wish to align. We will treat Y as the states of the HMM and X as the observations. An alignment of X and Y is a sequence of states (elements of Y), $S = S_1, \cdots, S_n$. Our goal is to find the sequence S^* that best explains the observed sequence X:

$$S^* = \arg\max_S \Pr(S_1) \prod_{t=2}^{n} \Pr(S_t|S_{t-1}) \prod_{t=1}^{n} \Pr(X_t|S_t). \tag{1}$$

We define observation probabilities based on frame similarity, where $\Pr(X_t|S_t) \propto f(X_t, S_t)$ for some similarity function f. We discuss frame similarity measures in more detail below . Initial probabilities $\Pr(S_t)$ are uniform. Transition probabilities are controlled by a single parameter s such that the probability of frame Y_i following Y_{i-1} is s times as likely as any other frame. The normalized transition probabilities are given as

$$\Pr(S_t|S_{t-1}) = \begin{cases} \frac{s}{s+m-1} & (S_t - S_{t-1}) \bmod m = 1, \\ \frac{1}{s+m-1} & \text{otherwise} \end{cases} \tag{2}$$

for s in $[1, \infty)$.

Intuitively, the parameter s provides control over the number of cuts in the output; a cut occurs when two consecutive frames in the output were taken from non-consecutive positions in the input secondary. When $s = 1$, the output sequence is determined solely by frame similarity, and the result is often choppy and unpleasant to watch. When $s = \infty$, the result is a continuous loop of consecutive frames, starting at some offset; this results in a poor alignment unless the original timing of the two videos matches very closely. A value inbetween these two extremes produces a good alignment, balancing frame similarity and temporal continuity. A value of $s = 6$ works well for a variety of data sets and is used on all examples in the paper. However the appropriate value for s can vary depending on the structure of the video; for example, for highly edited video with frequent cuts, a lower, more permissive value of s may produce better results. While the video alignment process operates on frames, the result can be seen as a concatenation of *clips* from the secondary video, with cuts occurring at clip boundaries (see Fig. 2). For efficiency, we compute the alignment at every k^{th} frame instead of every frame; $k = 10$ for the examples shown here.

We define the quality of an alignment as the average frame similarity score between frames in the input sequence X and the corresponding frames in the aligned result S. This score is then used to rank the order in which we display the secondary videos in the Browsing Companion.

Frame similarity measure. The HMM combines a frame-to-frame similarity measure with temporal continuity. We use a combination of three frame-to-frame similarity measures:

1. *Bag-of-words*: Each frame is represented as a histogram of quantized SIFT [12] features, or *visual words*. We extract SIFT features over a regular grid and quantize to a vocabulary of visual words. We compute vocabularies for each collection using median shift clustering [18] with a radius $r = 0.5$; for the data sets here, this generates vocabularies of approximately 5000 visual words.

2. *Spatial pyramid matching*: We also use a small visual word vocabulary and spatial pyramid matching [11]. As opposed to the bag-of-words model, this approach retains

coarse spatial information about the visual words found in the image. We use a 50-word vocabulary and a 4-level square grid, with levels of width 1, 2, 4, and 8.

3. *Color histogram*: Finally, we compute a coarse $4 \times 10 \times 10$ L*a*b* color histogram for each frame.

For each of the three measures, frame similarity is determined by cosine similarity between histograms. We use a convex combination of these three scores as our final frame similarity measure. We found that weights of $\{0.4, 0.4, 0.2\}$ for bag-of-words, spatial pyramid, and color histogram similarities, respectively, perform well across a variety of different data sets; we use these weights for all results in the paper.

There are clearly many other frame similarity measures, such as motion cues, face detection, or sound that could be used. However, in this work we focused solely on image appearance based measures and leave multi-modal frame similarity features for future research.

4. Automatic Multi-Video Summaries

In addition to user-assisted browsing, video alignment also provides useful information to automatically create a summary of the video collection subject to some relevance score. For example, the user might want to create a summary video that is representative of the entire video collection, or he might want to generate a highlight summary that covers just the highlights. In fact, as we will show later, the user can also guide the process by supplying key-frames that guide the system how to create the summary.

In all cases, we first define a relevance measure for each frame that relates the frame to the entire video set, and then use HMM to generate a summary video (composed of several clips) that maximize (or minimize) this measure.

Relevance measures. We illustrate these relevance measures on the problem of automatic highlight detection. One approach would be to say that unique frames are highlights. Therefore, all we have to do is find unique frames (or clips) within each video as is done, for example, by [3]. But the highlights generated this way are with respect to each video, and *not* with respect to the entire video collection. Trying to overcome this limitation by concatenating all input videos into one long video and then detecting highlights might produce undesirable results. For example, in the case of the Usain Bolt data set, the race is clearly the highlight, but it appears once in every video and hence will appear multiple times in a concatenated video and therefor will not be selected as a highlight.

We propose an alternative definition: *A highlight is footage that is unique within each video, but common across all videos* (see Fig. 4). This way we obtain highlights that are determined by the *entire* video collection.

Our goal now is to define a highlight score (h-score) that measures how good is a particular frame as a highlight. And since the highlight must be unique within its own video and common across multiple videos we need to define two additional scores that will help us compute the h-score: A representative score (r-score) that measures how common is a frame across the video set, and a uniqueness score (u-score) that measures how unique is a frame within its own video.

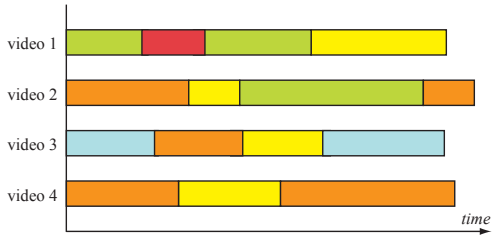


Figure 4: *Overview of highlight detection. Highlights, by definition, are rare events. However, in this example if we consider video 1 alone, there is no clear way to distinguish between the red and yellow clips as the best highlight. Only by considering the collection of videos can we robustly identify relevant unique clips; these highlight clips are rare within each video but common across all videos. In this example, the yellow content occurs only once in any video, but it occurs in all videos. In this way, multi-video analysis exposes information not available to existing single-video summarization methods.*

Representative score. The representative score (r-score) measures how well a particular frame represents the video collection. But, what does it mean for a frame to represent the video collection well? Intuitively, we would like that this frame of a particular video will be similar to frames in other videos. If a frame has a high r-score then we say that it is a representative frame (because we found good matching frames for it in other videos).

Formally, let X_t^i denote frame t in the i^{th} video and $S_t^{i,j}$, frame t in the output of the alignment between primary i and secondary j. The r-score for frame t in video i is:

$$r_t^i = \frac{1}{v-1} \sum_{j \in [1,v] \setminus i} f(X_t^i, S_t^{i,j}). \tag{3}$$

This is simply the per-frame average appearance similarity score between frames in video i and corresponding frames from all other videos in the set.

Uniqueness score. Our score for uniqueness follows common methods for identifying the uniqueness of nearest neighbors as the ratio of the distances between a query point and its first and second nearest neighbors [12]. A ratio of 1 occurs when the two neighbors are equidistant from the query point. When the second nearest neighbor is much further away, this ratio becomes very small, indicating the uniqueness of the first neighbor with respect to the rest of the data set.

In our case, the alignment performed in the previous

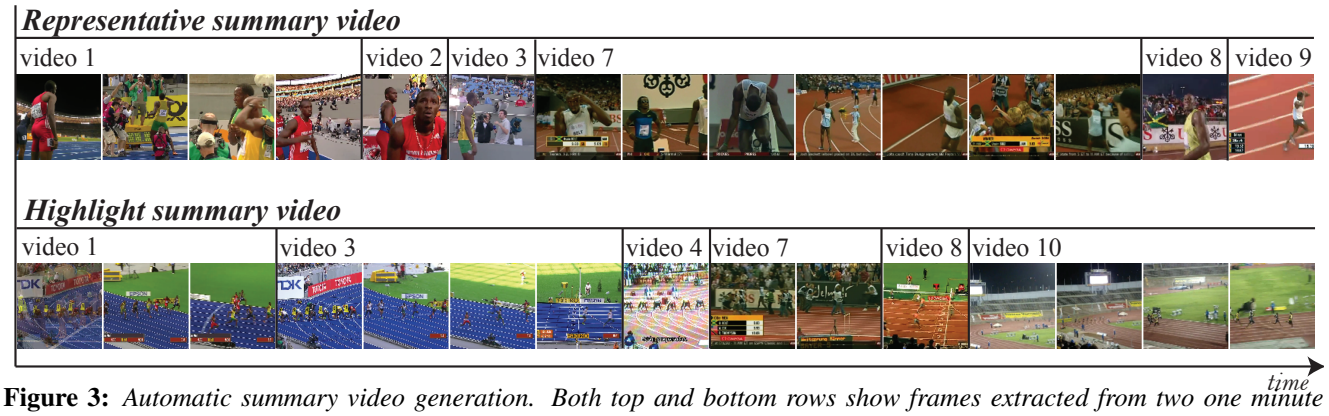


Figure 3: *Automatic summary video generation. Both top and bottom rows show frames extracted from two one minute summary videos automatically generated from the Bolt data set using different scores. In each example, the labels above indicate the corresponding frames' source video. (Top) Optimizing the representative score produces a result that is self-consistent and captures Usain Bolt in different settings. These frames accurately describe the most common content of the data set. (Bottom) Optimizing the highlight score finds footage that is unique within each video, but common across all videos. These results are also consistent with one another and depict highlight race footage. While both groups are self-consistent, there are no hard constraints that require them to be so, e.g., by encouraging high similarity scores among frames. This is a property of the data set that is illuminated by our approach.*

section for browsing gives, for every primary frame, a best match, or nearest neighbor, in each of the secondary videos. To compute a second nearest neighbor, or second best match, we run the alignment procedure a second time, ignoring the best match from the previous alignment.

Specifically, we run the video alignment procedure. This breaks the secondary video into clips that are organized to best explain the primary video. This gives us a match between each frame in the primary and a frame in the secondary. And since each frame in the secondary belongs to a clip, induced by the alignment, we now have an assignment of each frame in the primary to a clip in the secondary. Now we repeat the alignment procedure again, only this time for each frame in the primary we disallow *all* frames in the clip (in the secondary) associated with it. This forces the algorithm on the next iteration to find the second best match in the secondary video. If we had instead disallowed only single frames instead of entire clips, the second best match would likely be one frame before or after the initial match.

To compute a robust uniqueness score for frame t in video j, we first find the primary video and corresponding frame within that video (i^*, τ^*) to which frame (j, t) is aligned with the highest similarity score,

$$i^*, \tau^* = \underset{i,\tau | S_\tau^{i,j} = X_t^j}{\arg\max} \; f(X_\tau^i, S_\tau^{i,j}) \tag{4}$$

While we're interested in the degree of uniqueness of a frame with respect to its source video, which is a property that is independent of the rest of the data set, we still need a degree of robustness to ensure that the frame (j, t) has at least some relevance to the rest of the data set. A high value for the $\max$ of Eq. 4 indicates that frame (j, t) occurs at least once elsewhere in the collection.

Now we use (i^*, τ^*) as a query frame and compute the ratio of best and second best matches. Let S and S' denote our initial and alternate alignments. The uniqueness score u_j^t of frame t in video j is simply the ratio of similarity scores between (i^*, τ^*) and (j, t), and between (i^*, τ^*) its second best match in S':

$$u_t^j = \frac{f(X_{\tau^*}^{i^*}, S_{\tau^*}^{i^*,j})}{f(X_{\tau^*}^{i^*}, S_{\tau^*}^{\prime i^*,j})}. \tag{5}$$

Highlight score. The highlight score (h-score) measures how unique is a frame within its own video (u-score) and how representative it is, that is how common it is across videos (r-score). We found that taking a linear combination of the r-score and u-score gives good results. Specifically, we define the h-score h_t^i, as

$$h_t^i = \alpha u_t^i + (1 - \alpha) r_t^i \tag{6}$$

While the parameter α allows fine control over the relative contributions of the two scores, we found a value of $\alpha = 0.5$ to work consistently well.

Generating Video Summaries. Once the relevance measure has been established, we apply the Viterbi algorithm to an unnormalized HMM to obtain a video summary of a user-specified length. The Viterbi algorithm, which is a variant of Dynamic Programming, generates a video (that might consist of a sequence of clips) that maximizes the relevance score we selected. Without it, we would select the frames with the highest score, but such a video will have many cuts that will lead to a very choppy and unpleasant viewing experience. The Viterbi algorithm, on the other hand, will properly trade smoothness (i.e., minimizing number of cuts in the output video) against maximizing the relevance score.

Figure 5: *User-guided summaries. The original data consists of an* 18 *minute video of a museum tour, where each exhibit was visited multiple times. Here the user can quickly generate a summary by selecting a number of key frames (*4 *in this case) and their duration (top). The system, in response, will find the best set of clips that match the requested keyframes and their durations. Observe that the system can choose multiple clips to fit a particular keyframe, or use one clip to match two consecutive keyframes. The output of this process is a* 80 *second summary of the original* 18 *minute video, generated according to the user specification (extracted frames, bottom). The user can quickly and easily change parameters to generate alternative summaries.*

In a recent work [20], unnormalized HMMs have been successfully used for classification; the authors acknowledge that unnormalized HMMs can handle a larger class of problems than standard HMMs. We take advantage of that here for generating a summary, expressing the framework as a variation of the HMM for video alignment.

Here we treat all frames in the collection as hidden states $Z_1, \cdots, Z_m$, where m is the total number of frames in all videos. The output of the algorithm is a sequence of states, $S = S_1, \cdots, S_n$, for a user-specified output length n. Since there is no primary video to act as a guide, unary terms corresponding to observation probabilities are simply defined by the score $g(S_t)$, where g can be any of the relvance measures defined above. We have transition affinities (unnormalized probabilities) defined similarly to transition probabilities in Eq. 2 to discourage cuts by penalizing transitions to non-consecutive frames, specifically, for affinities a,

$$a(S_t, S_{t-1}) = \begin{cases} 0 & S_t \leq S_{t-1} \\ s & (S_t - S_{t-1}) = 1, \\ & S_t, S_{t-1} \text{ are from same video} \\ 1 & \text{otherwise.} \end{cases} \tag{7}$$

Here we've added a monotonicity constraint that enforces chronological order, which requires that the order of the clips in the summary video respects their original order in the input. Without it, the resulting video would just contain the best frame(s) repeated for the duration of the output video. Transitions from one frame to another that occurred earlier in the input are disallowed; for such a transition the affinity is zero. Transition affinities a are defined with the same parameter s from Section 3 that controls the frequency of cuts in the final result. The objective function we maximize here is given as

$$S^* = \arg\max_S \prod_{t=1}^{n} g(S_t) \prod_{t=1}^{n} a(S_t, S_{t-1}), \tag{8}$$

for user-specified output video length n.

Eq. 8 can be solved efficiently with the Viterbi algorithm. Since we have a lot more frames to select from here than in alignment, we subsample at a larger interval k than we use for alignment. This allows us to efficiently summarize large collections, with a tradeoff in accuracy (e.g., we may miss very short highlights). The video summary will consist of a number of clips, taken from different input videos, that give the best trade-off between maximizing relevance score and minimizing the number of cuts, and it can quickly inform the user about the contents of the data set. Additionally, video summaries can be used as a jump off point for exploring the collection inside the browsing companion.

5. User-Guided Summaries

We can use similar methods to generate semi-automatic, i.e., user-guided summaries. This alternative approach gives the user some degree of control over the output. For example, this is a convenient tool to quickly create a short summary of a long family video. The user simply selects a couple of key frames and their duration, which are treated as a primary "video". Then we can align the original, long video to this primary video to obtain a user-assisted summary.

In this application, both the observations and hidden states $Z_1, \cdots, Z_m$ correspond to frames in the data set. The algorithm takes as input a sequence of (keyframe, duration) pairs. A sequence of observations is first constructed from these pairs. For example, if the input was $(3, 2), (4, 1), (1, 2)$, the observed sequence would be Z_3, Z_3, Z_4, Z_1, Z_1. Each (keyframe, duration) pair produces a clip, or a number of clips, of the given duration that will match the keyframe.

Let $X = X_1, \cdots, X_n$ denote such a user-specificed observed sequence of length n. Unary terms are defined by appearance similarity between the corresponding keyframe and output frame, $f(X_t, S_t)$. The transition affinities are defined similarly to those in Eq. 7 with one minor change– at keyframe segment boundaries, we relax the monotonicity constraint. This allows control over if, and when, certain

data set	description	#	hh:mm:ss
bolt	from YouTube, query "Usain Bolt"	11	00:30:15
conan	*The Tonight Show with Conan O'Brien*	10	07:13:51
runway	*Project Runway: Season Five*	28	10:46:07
chopper	*American Chopper: Season Six*	15	06:30:33
hawaii	travel videos on Hawaii	8	01:58:15
museum	home video of a museum tour	1	00:18:27

Table 1: *Description, number of videos, and total duration for data sets used in the paper.*

content is repeated, since chronological order will be enforced within segments but not across them. Affinities are given as:

$$a(S_t, S_{t-1}) = \begin{cases} 0 & S_t \leq S_{t-1}, \\ & t-1 \text{ not last in segment} \\ s & (S_t - S_{t-1}) = 1, \\ & S_t, S_{t-1} \text{ from same video} \\ 1 & \text{otherwise} \end{cases} \quad (9)$$

6. Discussion

Implementation details. Each pairwise alignment of a length n primary and length m secondary with the Viterbi algorithm is $O(nm^2)$; in our implementation, this is as little as 5 seconds for a pair from the Bolt data set, and as much as 80 minutes for two full Conan episodes. As a result, it takes 15 minutes to align all the Bolt videos, and 120 hours for the Conan data. We use a cluster to parallelize the process. Descriptor generation, done once per video, is also performed on the cluster. The Browsing Companion is implemented in Adobe Flash; videos are assembled and streamed interactively from a RTMP server based on a sequence of secondary clip requests from the client.

Relevance scores are simple and very quick to generate. Generating a video summary takes $O(mn^2)$, where m is the number of frames in the source video or video collection, and n is the output video length. For the video summary examples shown here (Fig. 3) we sub-sample at $k = 10$. Generating a 1-minute video summary for the Bolt set takes less than two minutes, and less than a minute for the Museum data. Generating the summary does not require as much precision as alignment in terms of placing cuts for the amateur user, nor does it require sub-second accuracy, so a larger k can easily be used to generate good summaries for larger data sets.

Results. Table 1 lists the data sets used in the paper, which vary significantly in source, content, quality, and size. In all cases, we've used the same frame descriptors and system parameters. Alignment examples can be seen in Figs. 6 and 1. Please see the companion video for screen captures of the Browser. Fig. 3 shows a result for automatic summary generation on the Bolt data set using both the representative and highlight score. Fig. 5 shows the result of our user-assisted video summarization. For small input data or for larger inputs sufficiently sub-sampled, our method can be used to quickly generate user-guided summaries as part of an iterative editing process.

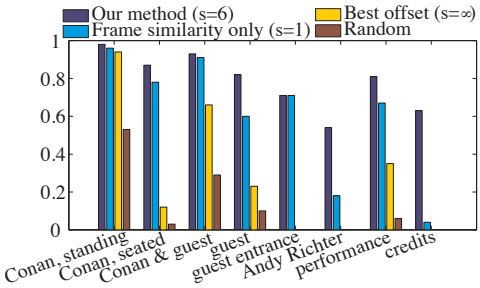


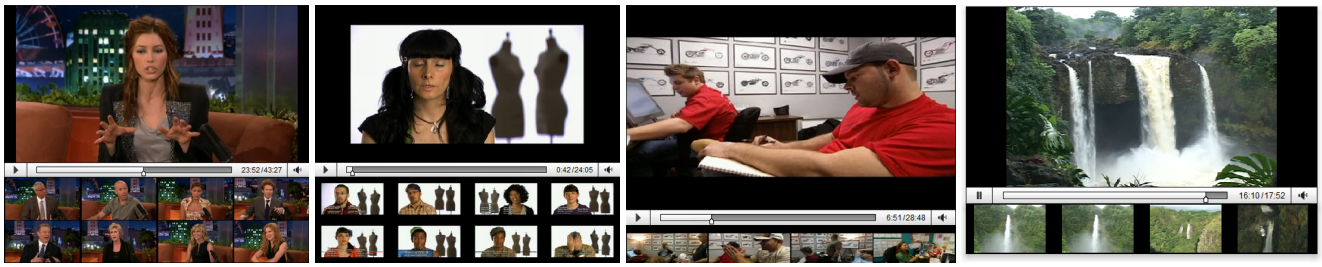
Figure 7: *Classification rates on the Conan O'Brien data set using our method (purple), frame similarity only (teal), best offset (yellow), and random assignment (brown) for alignment. As can be seen, our method consistently outperforms the other methods.*

Alignment validation. In a simple experiment, we first extracted frames from the Conan data set at 10 second intervals, then used Amazon's Mechanical Turk to obtain ground truth class labels for all 2606 frames from among 8 classes. We then performed leave-one-out cross validation, comparing known assignments for each frame against the consensus classification among the $(v-1)$ aligned secondary frames, where consensus is taken to be the mode.

We repeated the experiment with four different alignment methods, including random frame matching, alignment with frame similarity only ($s = 1$), alignment by finding the single best offset ($s = \infty$), and our method ($s = 6$). Our method offers a nice trade-off between frame similarity and continuity, where it produces far fewer cuts than frame similarity alone, while matching content significantly better than a simple offset (Fig. 7).

Some observations are in order. First, our method performs the best, even better than the frame-similarity-only measure. This is because our method benefits from larger temporal support for identifying longer segments that leads to more robust matching. However, this temporal support respects transitions; we get smoothing from the HMM but are still able to transition with precision in the alignment (as is demonstrated in the accompanying video). Of particular interest is the "credits" class; portions of the opening credits vary from episode to episode, depending on, e.g., who the guests are that night, but a lot of the credits stay the same; this was enough for our method to correctly classify a much larger percentage of the frames.

Limitations. Ours is a simple appearance model based on similar-scene matching over the entire frame, using descriptors that have been successfully applied to classification and similar-scene matching. With this model, we focus primarily on data sets with strong spatial structure, where the foreground is highly correlated with the background and camera view. This is a result of the similarity measure used in our implementation and not of the browser or summa-



(a) conan (b) runway (c) chopper (d) hawaii

Figure 6: *Some alignment examples. Please see the accompanying video for more results.*

rization method, as this measure works best for whole-scene matching, not foreground matching alone. Our approach is orthogonal to choice of similarity measure, and in a production setting, a user could simply specify which measure to use for a particular task, choosing to browse or summarize based on similarity of foreground object, face, or action, or background, or a combination thereof.

When browsing with a given primary video, irrelevant content in secondary windows can appear for a number of reasons. This will occur when the appearance model fails to give high similarity scores to relevant content in other videos, or when very short segments of the primary are ignored in favor of smoothness in the HMM. It will also occur when there is no similar content in the collection. Additionally, we only show a fixed set of aligned secondary videos for a given primary; when there is locally relevant content in a lower-ranked secondary video, it will not be shown.

Acknowledgements. The authors thank Lior Shapira for providing source code and acknowledge support by the National Science Foundation under Grant No. PHY-0835713.

References

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

[2] F. Arman, R. Depommier, A. Hsu, and M.-Y. Chiu. Content-based browsing of video sequences. In *Proc. ACM Multimedia*, 1994. 2

[3] O. Boiman and M. Irani. Detecting irregularities in images and in video. *IJCV*, 74(1):17–31, 2007. 4

[4] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. The video genome, March 2010. arXiv:1003.5320v1. 2

[5] Y. Caspi and M. Irani. Spatio-temporal alignment of sequences. *PAMI*, 24(11):1409–1424, 2002. 2

[6] K.-Y. Cheng, S.-J. Luo, B.-Y. Chen, and H.-H. Chu. Smartplayer: User-centric video fast-forwarding. In *Proc. CHI*, 2009. 2

[7] S. H. Cooray, H. Bredin, L.-Q. Xu, and N. E. O'Connor. An interactive and multi-level framework for summarising user generated videos. In *Proc. ACM Multimedia*, 2009. 2

[8] F. Daufaux. Key frame selection to represent a video. In *Proc. ICIP*, 2000. 2

[9] D. DeMenthon, V. Kobla, and D. Doermann. Video summarization by curve simplification. *Proc. ACM Multimedia*, 1998. 2

[10] I. Laptev and P. Perez. Retrieving actions in movies. In *Proc. ICCV*, 2007. 2

[11] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. CVPR*, 2006. 3

[12] D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. ICCV*, 1999. 3, 4

[13] X. Mu. A content-based video browsing system based on visual neighbor similarity. In *Proc. JCDL*, 2006. 2

[14] N. Petrovic, N. Jojic, and T. Huang. Adaptive video fast forward. *Multimedia Tools and Applications*, 2005. 2

[15] Y. Pritch, A. Rav-Acha, and S. Peleg. Non-chronological video synopsis and indexing. *PAMI*, 30(11):1971–1984, 2008. 2

[16] L. Rabiner and B.-H. Juang. *Fundamentals of speech recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993. 2

[17] P. Sand and S. Teller. Video matching. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 22(3), 2004. 2

[18] L. Shapira, S. Avidan, and A. Shamir. Mode-detection via median-shift. In *Proc. ICCV*, 2009. 3

[19] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, 2003. 2

[20] A. Sloin and D. Burshtein. Support vector machine training for improved hidden markov modeling. *IEEE Trans. Signal Processing*, 56(1):172–188, 2008. 6

[21] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and TRECVID. In *Proc. MIR*, 2006. 2

[22] M. A. Smith and T. Kanade. Video skimming and characterization through the combination of image and language understanding techniques. In *Proc. CVPR*, 2001. 2

[23] D. Tjondronegoro, Y.-P. P. Chen, and B. Pham. Content-based video indexing for sports applications using integrated multi-modal approach. In *Proc. ACM Multimedia*, 2005. 2

[24] T. Tuytelaars and L. V. Gool. Synchronizing video sequences. In *Proc. CVPR*, 2004. 2

[25] Y. Ukrainitz and M. Irani. Aligning sequences and actions by maximizing spacetime correlations. In *Proc. ECCV*, 2006. 2

[26] J. Wang, C. Xu, E. Chng, K. Wah, and Q. Tian. Automatic replay generation for soccer video broadcasting. In *Proc. ACM Multimedia*, 2004. 2

[27] D. Zhang, D. Gatica-Perez, S. Bengio, and I. McCowan. Semi-supervised adapted HMMs for unusual event detection. In *Proc. CVPR*, 2005. 2