Hierarchical Summarization for Easy Video Applications

Abstract

With growing use of videos, demand on retrieval video applications has become intense. Most existing methods that analyze the semantics of a video build specific models; for example, ones that aim at event detection, or targeted video albumization. These might be called as application specific works and useful in their own right. In this paper, however, we propose a video abstraction framework that unifies the creation of various applications, rather than the application itself.

Specifically, we present a dictionary summarization of a video that provides abstractions at various hierarchical levels such as pixels, frames, shots, and the complete video. We illustrate the usability of our model with four different "apps".

1 Introduction

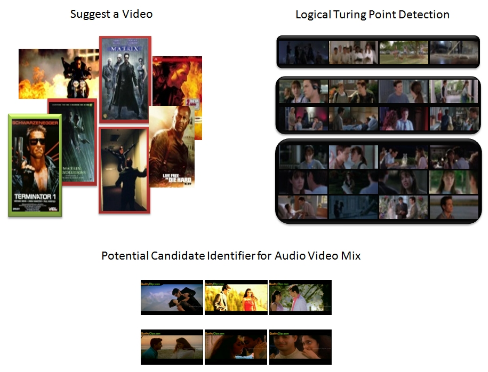
Say you want to find an action scene of your favorite hero. Or want to watch a romantic movie with a happy ending. Or say you saw a interesting trailer, and want to watch related movies. Finding these videos in the ocean of videos available has become noticeably difficult, and requires a trustworthy friend or editor. Can we quickly design computer "apps" that act like this friend?

This work presents an abstraction for making this happen. We present a model which summarizes the video in terms of dictionaries at different hierarchical levels — pixel, frame, and shot. This makes it easier for creating applications that summarizes videos, and address complex queries like the ones listed above.

The abstraction leads to a toolkit that we use to create several different applications demonstrated in Figure 1. In the "Suggest a Video" application, from a set of action movies, three *Matrix* sequels were given as input; the movie *Terminator* was found to be the closest suggested match. In the "Story Detection" application, the movie *A Walk to Remember* is segmented into three parts; user experience suggests that these parts correspond to prominent changes in the twist and plot of the movie. In the "Audio Video Mix" application, given a song with a video as a backdrop, the application finds another video with a similar song and video; the application thus can be used to generate a "remix" for the input song. This application illustrates the ability of the data representation to find a video which closely matches both content and tempo.

1.1 Related Work

In creating dictionaries, soft quantization [1] accounting distance from a number of codewords is considered for classifying scenes; Fisher Vector is then used in classification [3, 2] leading to significant improvement over Bag Of word methods.



Suggest a Video
Logical Turing Point Detection
Potential Candidate Identifier for Audio Video Mix

Fig. 1. Retrieval applications designed using our *Hierarchical Dictionary Summarization* method. "Suggest a Video" suggests *Terminator* for *Matrix* sequels. "Story Detection" segments a movie into logical turning points in the plot of the movie *A Walk to Remember*. "Audio Video Mix" generates a "remix" song from a given set of music videos.

Similar to [3, 2], we use local descriptors and form visual dictionaries. However unlike [3, 2], we preserve more information instead of extracting only specific information. In addition to building dictionary at pixel level, we extend this dictionary to frame and shot level, forming a hierarchical dictionary. Having similarity information available at various granularities is the key to creating applications that need features at the level desired.

1.2 Contributions

In this paper, we propose a Hierarchical Dictionary Model (termed H-Video) to make the task of creating application easier. Our method learns semantic dictionaries at three different levels — pixel patches, frames, and shots. Video is represented in the form of learned dictionary units that reveal semantic similarity and video structure. The main intention of this model is to provide these semantic dictionaries, so that comparison of video units at different levels in the same video and different videos becomes easier.

The benefits of H-Video include the following

(i) The model advocates run-time leveraging of prior offline processing time. As a consequence, applications run fast.

(ii) The model is built in an unsupervised fashion. As no application specific assumption is made, many retrieval applications can use this model and its features. This can potentially save enormous amount of computation time spent in learning.

(iii) The model represents learned information using a hierarchical dictionary. This allows video to be represented as indexes to elements in the video. This makes it easier for the developer of a new retrieval applications as similarity information is available as a one dimensional array. In other words, our model doesn't demand deep video understanding

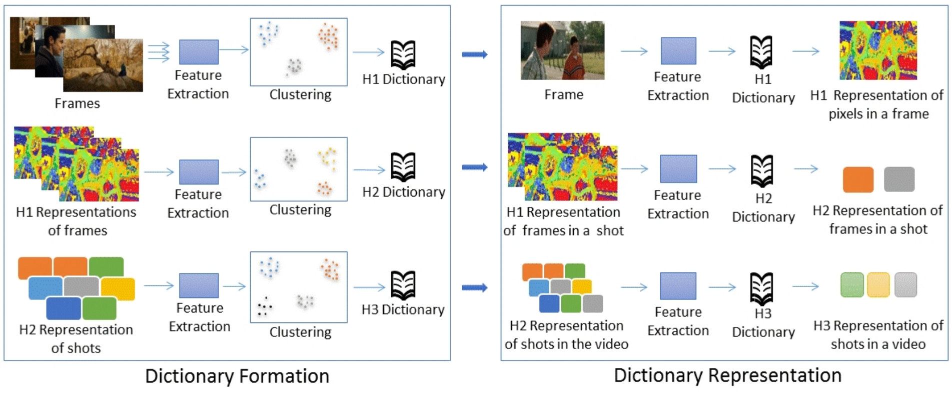

Frames
Feature Extraction
Clustering
H1 Dictionary
H1 Representations of frames
Feature Extraction
Clustering
H2 Dictionary
H2 Representation of shots
Feature Extraction
Clustering
H3 Dictionary
Dictionary Formation
Frame
Feature Extraction
H1 Dictionary
H1 Representation of pixels in a frame
H1 Representation of frames in a shot
Feature Extraction
H2 Dictionary
H2 Representation of frames in a shot
H2 Representation of shots in the video
Feature Extraction
H3 Dictionary
H3 Representation of shots in a video
Dictionary Representation

Fig. 2. Illustration of the H-Video Abstraction

background from application developers.

(iv) We have illustrated our model through several applications. Figure 1 illustrates these applications.

2 Methodology

Overview: Our model first extracts local descriptors like colour and edge from pixels patches. (Color and edge descriptors are simply examples.) We then build a dictionary, termed H1 dictionary, out of these features. At this point, the video could be, in principle, be represented in terms of this H1 dictionary. We refer each frame of this representation as an H1 frame. We then extract slightly higher level features such as the histogram of the H1 dictionary units, the number of regions from these H1 frames and so on, and form a new H2 dictionary. The H2 dictionary is yet another representation of the video and captures the type of objects and their distribution in the scene; in this way, it captures the nature of the scene. The video could also be represented using this H2 dictionary. We refer each shot in this representation as an H2 shot. Further, we extract features based on the histogram and position of H2 dictionary units and build yet another structure, the H3 dictionary. This dictionary represents the type of shots occurring in the video. The video is now represented in terms of this H3 dictionary to form H3 video.

Details: For forming the H1 dictionary, at each pixel, the nearby 8×8 window of pixels are considered. We extract local descriptors from the moving window using pyramidal Gaussian features, neighborhood layout features, and edge filters.

As an example, we use the list of filters listed in Fig. 3, which we have found to be effective in capturing the colour & shape information. A different set of features like SIFT or HOG can also be used based on the requirement of the application.

These features are extracted from the complete video. We use principal component analysis to determine the number of clusters. The top principal component is chosen based on the allowed error. We then use k-means to obtain the H1 dictionary. However as clustering very long videos could be time consuming, one alternative is to do this process in various stages. For example, we first form H1 dictionary for each frame, combine dictionaries from a sequence of frames, and do clustering to form a more representative dictionary. Several dictionaries from temporal stages in the video are then combined to form a global H1 dictionary. This step-by-step approach of building dictionary makes this easy to compute and scalable.

HG1	HG2	SG1	SG2	*IG1*	IG2	ILg1	ILg2	IStx	ISty	ILx	ILy	IC	HN	SN	I→	I↗	I↑	I↖

H – Hue	G – Gaussian Filter	1 – Filter with Sigma = 1
S – Saturation	Lg – LoG Filter	2 – Filter with Sigma = 2
I – Intensity	St – Short Edge Kernel	x – Horizontal Direction
	L – Long Edge Kernel	y – Vertical Direction
	N – Neighbourhood Color Layout	→ – 0^o line
	Ln – Lines using Hough Transform	↗ – 45^o line
		↑ – 90^o line
		↖ – 135^o line

Fig. 3. Pixel-Level Feature Extraction Filters

Once the global H1 dictionary is available we process the video again to remove duplicates, or near duplicates to form a less redundant dictionary-based representation. The complete video is then represented using these dictionary units (left hand side of Fig. 2). Each frame in this representation is referred as an H1 frame. *H1 frames may also be thought of as a segmentation, but in addition to the segmentation, the dictionary units have the information about the nature of the objects.*

H2 Formation: From each H1 frame, conglomerate features like a histogram, the number of regions and the distance between them are captured. A subset of these features can also be chosen based on the application of these features. Similar to H1 dictionary formation, these features are clustered to form an H2 dictionary. We use the step-by-step dictionary building approach, wherein first dictionaries for a sequence of frames is built and they are clustered again to form the global dictionary.

Using the H2 dictionary, we represent the video in terms of H2 units. We refer each shot in this representation as an H2 shot. The change in the H2 units correspond to dynamically changing shots, whereas more or less similar H2 units corresponds to relatively static content. Representing video as a one dimensional array of dictionary units makes the comparison of video elements easier. *H2 units capture higher level details and are simpler for comparison purposes. Generally information needed by similarity application will be captured at this level.*

H3 Formation: Features capturing distribution of H2 units like histogram, and distance between blocks are considered. These extracted features are clustered to form H3 dictionary, wherein first the dictionary is computed at shot level, and the dictionary units are again clustered to form global H3 dictionary. Using this dictionary, the video is represented as a one dimensional sequence of H3 units, where each unit corresponds to shot. *This representation will be typically useful in a applications involving a huge collection of videos; applications can quickly narrow down to the correct interest group. This level will also be useful in segmentation and classification applications. For example while classifying shots of lecture video into professor, student and board slides, this level will be of helpful.*

Some applications require videos to be compared at multiple dictionary levels. For this purpose, we cluster H1 clusters of more than one input videos in the database together, and construct a H1 dictionary at the database level, rather than an individual video. Using this notion of global H1 dictionary, the H2 units and H3 units are generated again. (Input videos from the database are randomly sampled.)

3 Experiments

In this section, we first evaluate the individual performance of the constituents of H-Video model itself. Next, in creating dictionaries, we compare the usage of popular features such as SIFT. Finally the performance of applications that may use more than one of the three levels, H1, H2, or H3, are evaluated.

Data: We have collected around 100 movie trailer from `youtube`, twenty five full length movie films, and a dozen music video clips.

Computational Time: Our implementation is in Matlab. Typically a full-length movie takes around two hours for feature extraction. Building local dictionaries for a movie takes around 10 hours. Building the global dictionary, which is extracted from local dictionary of multiple videos takes around 6 hours. Note that building local dictionaries and a global dictionary (left hand side of Fig. 2) are one time jobs. Once these are built, the dictionaries are directly used to create the right hand side of Fig. 2. In other words, relevant model building typically takes two hours which is no different from the average runtime of the video itself. Once the model is constructed, each access operation typically takes only around 10 seconds per video.

3.1 Individual Evaluation of H1 and H2

For illustrating the effectiveness of H1 dictionary, we considered the classification problem and collected videos from the category "car", "cricket" and "news anchor". We have collected 6 videos from each category summing up to total of 18 videos. We computed H1 dictionary for each of these videos, and formed a global H1 dictionary for the given dataset and represented all videos in terms of this global dictionary. For testing purposes we randomly selected two videos from each category for training data and remaining as testing set and test set against one of the three categories.

The recall and precision of classification using only H1 is provided in Table 1.

Table 1. Classification using only H1. With limited information, we are able to glean classification hints.

Category	Precision	Recall
Car	1.00	0.75
Cricket	0.67	1.00
News Anchor	1.00	0.75

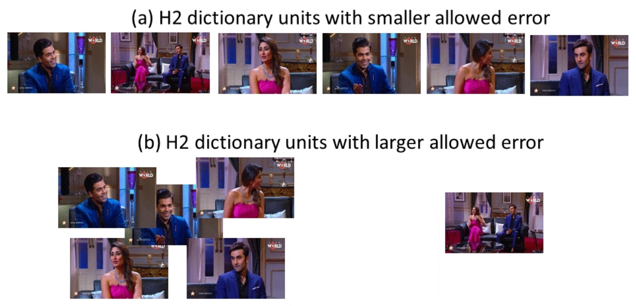

(a) H2 dictionary units with smaller allowed error
(b) H2 dictionary units with larger allowed error

Fig. 4. Classification using only H2. With only limited information, and with smaller allowed error, fine details were captured. With larger allowed error, broad categories were captured.

To evaluate the effectiveness of only H2 dictionary units, we have built our model on a TV interview show; these had scenes of only individuals, as well as people groups. When we built H2 dictionaries with allowed error as "top principal component / 1000", we got six categories capturing different scenes, people and their positions. As we relaxed the allowed error, it resulted in two categories between individual scene and group of people. This result is presented in Fig. 4. Hence applications can tune allowed error parameters to suit their requirement.

3.2 Evaluation of Alternate Features

In this section we evaluate popular features like SURF, SIFT and contrast with the color & edge features used in this paper. Given any feature set, one may do a "direct comparison" (which will take longer time), or do our proposed H-Video model-based comparison (which will take far lesser time). This experiment is performed on the "Suggest-a-video" problem using only trailers of movies as the database. The result is presented in Table 2.

When the H-Video model is used, we use the H2 as the basis of comparison. We observe that the use of the hierarchical model helped improving the accuracy for SIFT and color & edge features; the accuracy was almost the same when using SURF features. In producing these statistics, for the ground truth we have used information available on `imdb.com`. One problem in using `imdb.com` is that the truth is limited to the top twelve only. We therefore have

Table 2. Video suggestion using popular features

Methods	Direct Comparison	H-Model	Percentage Improvement
SURF	54%	54%	0%
SIFT	29%	53%	83%
Color, Edge	48%	59%	23%

added transpose and transitive relationships as well. (In transpose relationships, if a movie A is deemed to be related to B, we mark B as related to A. In transitivity, if a movie A is related to B, and B is related to C, then we mark A as related to both B and C. The transitive closure is maintained via recursion.)

3.3 Evaluation of Video Classification

We have considered three genres for video classification. We took the category annotation of 100 movie **trailers** and for each category considered 30% of the data for training and remaining as testing set. We have build the H-video for these videos, extracted H2 and H3 representations, and classified using the random forest model. Example output is shown in Fig. 5.

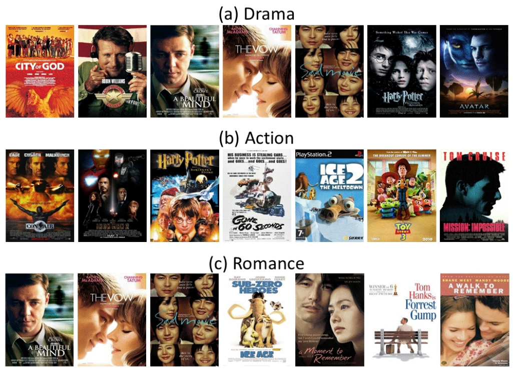


Fig. 5. Sample Result of classifying movie **trailers** for categories Drama, Action and Romance. In most of the cases, our model has classified movie trailers correctly.

3.4 Evaluation of Logical Turning Point Detection

We define Logical Turning Point in the video as a point where the characteristics of objects in the video change drastically. Detecting such places helps in summarizing the video effectively.

We consider shots within a moving window of 5 shots. We compute the semantic overlap of H1 and H2 units between the shots. When the amount of overlap between shots is low in a specific window of shots, we detect that as a logical turning point.

Typically drama movies have three logical parts. First the characters are introduced, then they get together and then a punchline is presented towards the end. Considering this as ground truth, we validated the detected logical turning points. The logical turning points were detected with precision of 1.0 and recall of 0.75.

3.5 Evaluation of Potential Remix Candidates

Remix is the process of generating a new video from existing videos by either changing audio or video. Remixes are typically performed on video songs for a variation on the entertainment needs. The remix candidates need to have similar phase in the change of the scene to generate a pleasing output.

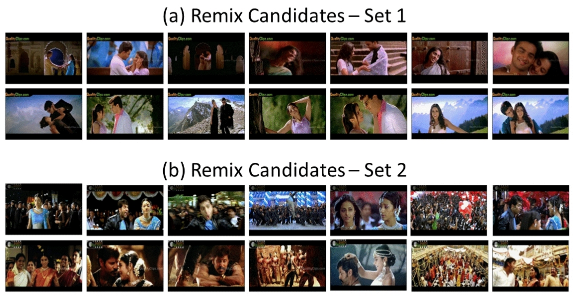


Fig. 6. Sample frames from identified remix candidates is presented. In each sets, top row correspond to a original video song and the second row corresponds to the remix candidate. The content and sequencing of the first and second rows match suggesting the effectiveness of our method.

We use cross correlation of H2 units to identify that they have same phase of change. Once the closest candidate is identified, we replace the remake candidate's audio with the original video's audio.

We have conducted experiments on 20 song clips, where the aim is to find best remix candidates. Our algorithm found two pairs of songs which are best candidates for remix in the given set. Sample frames from matched video are presented in Fig. 6.

4 Conclusion

In traditional video retrieval systems, relevant features are extracted from the video and applications are built using the extracted features. For multimedia database retrieval systems, there are typically plethora of applications that would be required for satisfying different user needs. A unified model which uses fundamental notions of similarity would therefore be valuable to reduce computation time for *building* applications.

In this paper, we have proposed a novel model called *H-Video*, which provides the semantic information needed by retrieval applications. In summary, both creation (programmer time) and runtime (computer time) of the resulting applications are reduced. First, our model provides semantic information of video in a simple way, so that it is easy for programmers. Second, due to our suggested pre-processing of long video data, runtime is reduced. We have built four applications as examples to demonstrate our model.

5 References

[1] J. C. Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. Smeulders. Kernel codebooks for scene categorization. In *ECCV*, pages 696–709, 2008.

[2] H. Jegou, M. Douze, C. Schmid, and P. Perez. Aggregating local descriptors into a compact image representation. In *CVPR*, pages 3304–3311, 2010.

[3] C. Sun and R. Nevatia. Large-scale web video event classification by use of fisher vectors. In *WACV*, pages 15–22, 2013.