

CONTENT BASED VIDEO RETRIEVAL

CS674/EE512: COURSE PROJECT – FINAL REPORT (MAY 2014)

Syed Muhammad Abbas
syed.abbas@lums.edu.pk

Hamza Anwar
14100048@lums.edu.pk

ABSTRACT

Context based video retrieval has a wide spread application across different domain. As videos in databases are increasing in millions, timely response is the real challenge today. In this paper, we present a novel technique of context based video retrieval for large databases. In our approach, key frames are used to generate the feature vectors for context based search. We have also compared different context based video retrieval approaches in order to depict the efficiency of the state of the art approach. Different approaches include CBIR, CBVR, near duplication video retrieval and semantic video retrieval. Our experiment results shows that near duplication and semantic retrieval approaches outperforms others. Finally, we analyze future research directions.

Index Terms— CBVR, semantics, kmeans, learning

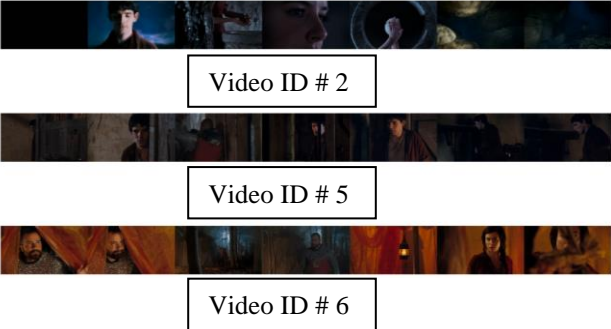
1. INTRODUCTION

Efficient video retrieval was not a problem few years back. But as the World Wide Web multiplies into a gigantic data giant, video retrieval problem started emerging. There were other factors too that increased the need of efficient video retrieval. As time went, more and faster machines came into being, video resolutions increased to high definition (HD) which made this problem even more complex. Today, video has become a major part of many information such as news, sports, entertainment etc. And people prefer to watch them in high definition too. We can well imagine that YouTube has a video database of over 100 million videos. There are others video databases too which also possess millions of videos. So it is very important that we should have a very efficient algorithm that can search us out relevant videos. Otherwise making this huge database is not wise if we do not get the video what we want in a acceptable time.

A lot of work has been done in improving the video retrieval and still it's an open problem as there are a lot to improve according to this modern world. In this paper, we present a novel approach of context based video retrieval. In this approach, video is being retrieved based upon the context present in the video. This is a general approach used in majority of video retrieval systems.

In the video retrieval approach used, we extract the frames from a video, use histograms to make a feature vector out of it. This feature vector made against an image frame is then combined with others to make a feature vector for the whole video. If we have features vectors against many videos, we used indexing to reduce the search time for a particular video's feature vector. Another novel technique is used to cluster the shots (few frames in video) after they have been annotated by the user during learning phase, against a keyword. This help us to efficiently retrieve the frames in a particular video based upon the context we are searching.

Query keyword: 'night'. Result of shots found:



(Dataset: 3 manual & 5 automatically labelled videos)

2. RELATED WORK

Before discussing different techniques of video retrieval, we should answer the question that: why video retrieval is so important? The rapid growth of web-based videos has really made the video databases in millions of gigabytes. Now it's impossible to search a required video while browsing all of them. So just like images, there should be a smart system of searching among all those videos provided it should fulfill two important aspects i.e. accuracy and time response.

A lot of work has been done on video based retrieval but still this techniques is not as mature as it is in case of images and text searches. Most of the techniques tries to handle different kinds of transforms in order to increase the accuracy of the retrieval but taking care of different kinds of

transforms really make them resource hungry and slow responsive. This thing multiplies many times in case of web videos as there are hundreds of millions of them. Normally the video searching algorithms are categorized into three classes: clip-level search, shot-level search and frame-level search. In clip level search, small video clip is used to search. A lot of work has been done on it [1][2][3]. But the leading work has been done by Karpenko et. Al [4]. They have used the technique of tiny images on videos and have produced sound results both in terms of accuracy and time response. They have highly compressed the videos into tiny ones but keeping the overall visual appearance as it. There are a lot of features in any given video frame by frame. To generalize them into fewer number of features, they have used clustering for every video. To make the system more responsive, they further classify the tiny videos into different classes. Experiments show that these techniques perform quite well but there still is a tradeoff between the accuracy and the response time.

In order to increase the response time, a lot of work has also been done on improving indexing approaches [5][6][7]. Kennedy et.al. approach takes initial search results from established video search methods (which typically are conservative in usage of concept detectors) and mines these results to discover and leverage co-occurrence patterns with detection results for hundreds of other concepts, thereby refining and re-ranking the initial video search result. After testing their algorithm on different datasets, they showed that their algorithm has improved the response time and accuracy up to 15 percent. Hu et.al. [8] has presented a survey regarding video retrieval techniques. Discussing all of them would not be possible here, but they focused on the performance analysis of different techniques with the aspects of video structure analysis, key frame extraction accuracy, scene segmentation, static key frame features, motion features, object features, video data mining and video annotations. In the end, they have also discussed the future directions of video retrieval. According to them though a lot of work has been done in video retrieval but there are still open challenges in Motion Feature Analysis, Hierarchical Analysis of videos, video in dices, fusion of multi models, video indexing, human-computer interface and distributed network video retrieval.

Lastly Cai et. al. proposed a nice and efficient approach of video retrieval. They name it near-duplicate video retrieval approach. In this approach first they have extracted key frames and then from that key frames they have generated signatures against each video. Then the signatures are indexed for fast retrieval. Experiment results show that with the compact video representation and efficient indexing, they have achieved an acceptable query response time for million-scale dataset.

3. CBIR VS. CBVR

In this task, we first load the image to search in the video. Then divide the image into 3x3 matrix. Then for each segmented image, we make a feature vector against it. For generation of feature vector, we first convert the image into three color channels i.e. red, green and blue. Then for each of the color channel separated, we calculate histograms for each. Then each histogram calculated is concatenated into a single feature vector. The total length of feature vector is 3x256x9 which is 6912.

For a video, according to the requirement it uses clip-level search for video retrieval. First it loads a video, and processes only first 3 minutes of video. It takes frames one after every 15 seconds. Once the frame has been taken, it generates the feature vector against it. The procedure of making a feature vector is same as mentioned above for an image to be searched in videos. The only difference is that for a given video the feature vector will be of length 82944.

Once we have both the feature vectors, we see the similarity among then frame by frame. For similarity, we have used Euclidean distance.

ALGORITHM

Parameters: *a, image, videos, frame, VideoFeatures*

```

1: image ← read query image.
2: GenerateFeatureVector(image)
   1: separate RGB channels
   2: normalize the channels into 256 steps.
   3: divide the image into 9 splits.
   4: against each split generate a feature vector by
      generating histogram.
   5: combine all to get final feature vector.
3: videos = get videos from file to search the query image.
4: for each video do
   1: Get the uniform frames from the video.
   2: for each frame do
      1: GenerateFeatureVector(frame)
      3: VideoFeatures ← all frames features in a form of
         single vector.
5: frame ← get one frame from VideoFeatures
6: for each VideoFeatures do
   1: EuclideanDistance (queryImageFeatureVector,
      VideoFeatures)
   2: if Euclidean Distance < 20000
      query image matched [output]
   3: else
      frame ← next frame from VideoFeature.
7: a ← output the video ID and the frame number matched.
8: return a

```

4. VIDEO RETRIEVAL VIS NEAR DUPLICATE DETECTION

The goal of this task is to search a video based upon another video. Though at the grass root level, we are dealing with the frames but now there sequence is also important for matching. In this task, we take the video to be searched in a video database, then extract the frames out of it. For the evaluation purpose, we are taking uniform frames which is constant for all the videos. Then we read video by video, extract its frames with the same method as mentioned above.

Once we have the frames, we make the feature vector against each image frame and then by combining we get the overall feature vector for the whole video. For making a feature vector, we used the method mentioned in task 1.

Once we have the feature vectors, we take the first frames of loaded video from the database and compare it with the first frame of the query video. If it matches, we take the next frame of both the videos and see whether they match or not. If yes then again repeat the process unless or until the query video ends. If we have a mismatch before the query video ends, then it's a mismatch video. Here the assumption is that the query video may be full or a subset of any video but should completely match a certain video.

ALGORITHM

Parameters: a , $queryVideo$, $refVideo$, $frame$, $VideoFeatures$

```
1:  $queryVideo \leftarrow$  read query video.
2:  $refVideo[] \leftarrow$  get videos from database
3: VideoRetrieval ( $queryVideo$ ,  $refVideo[]$ )
  1:  $queryFrames[] \leftarrow queryVideo$ 
  2: for each  $refVideo[]$  do
    1:  $refFrames[] \leftarrow refVideo$ 
    2: GenerateFeatureVector ( $refFrames[]$ )
      1: separate RGB channels
      2: normalize the channels into 256 steps.
      3: divide the image into 9 splits.
      4: against each split generate a feature vector by
      generating histogram.
      5:  $refFeatureVector \leftarrow$  Combine all to get final
      feature vector.
    3:  $qFeatureVector \leftarrow$ 
    GenerateFeatureVector( $queryFrames[]$ )
    4: while 1: $refFeatureVector$  do
    5: while 1:  $qFeatureVector$  do
      1: EuclideanDistance ( $refFeatureVector$ ,
       $qFeatureVector$ )
      2: if EuclideanDistance < 20000
      3: increment  $qFeatureVector$ ,  $refFeatureVector$ 
      4: else
      5: increment  $refFeatureVector$ 
      6: if  $qFeatureVector$  equal total frame length
```

```
7:  $a \leftarrow$  Video matched
7: return  $a$ ;
```

5. SEMANTIC VIDEO RETRIEVAL

The goal of semantic video retrieval was to label videos with certain attached concepts, and query based on those concepts for fast searching.

Our approach in this regard was to create an initial dataset with manually labelled shots in some videos, then use a clustering algorithm (e.g. K-Means [1]) for each concept. Once learning is complete, then we do automatic labelling of many more videos from the dataset and keep updating the learned clustering means. For any query, searching is fast because we just have to find concepts relating to query word and show all shots with maximal confidence for those concepts.

PARAMETERS

1: 'Annos'

(Hierarchical data structure that stores keywords, IDs, start-frame, end-frame, and feature matrix for each shot in dataset)

- $Annos\{vid\}(sho).keywords :=$ cell array containing all keywords for a shot with ID 'sho', from a video in the dataset named 'vid'.
- $Annos\{vid\}(sho).vidID :=$ global path to 'vid'.
- $Annos\{vid\}(sho).start :=$ starting frame # of shot 'sho' in video 'vid'.
- $Annos\{vid\}(sho).end :=$ ending frame # of shot 'sho' in video 'vid'.
- $Annos\{vid\}(sho).featureMat :=$ a feature matrix of size $(n,10)$, where,
 - $n = num_of_frames_in_shot * \#_of_16x16_blocks_in_each_frame_of_shot$
 - each row of this 'featureMat' has 10 DCT components of the corresponding block in corresponding frame.

2: 'keyMeans'

(Stores distribution parameters for all concepts)

- $keyMeans(i).word :=$ the concept name e.g. mountain, etc. of i^{th} concept.
- $keyMeans(i).means :=$ the learned distribution means i.e. a matrix of size $(4,10)$ where 4 is number of clusters and is 10 dimensional for 10 DCT coefficients.

3: Uniform sampling parameters

(In every 5 minutes interval of a video, take 8 uniformly sampled frames out of first 60 seconds, to get one shot in the video)

ALGORITHM: MANUAL SEMANTIC LABELLING AND LEARNING (TASK 1)

- 1: **load** last '*Annos.mat*' and '*keyMeans.mat*'
- 2: **for each** video v **to be manually labelled**
- 3: **for each** shot s **in** v
- 4: $s \leftarrow$ sample uniformly
- 5: **for each** frame f **in** s
- 6: $Annos\{v\}(s).featureMat \leftarrow$ features(f)
- 7: show(some frames f from s)
- 8: assign keywords \leftarrow user input
- 9: perform **K-Means** clustering with 4 clusters
 - a. **for each** keyword k
 - b. **search** relevant *featureMat* in *Annos*
 - c. **append** to make $(n, 10)$ matrix
 - d. $keyMeans(k).means :=$ **learn** using *kmeans* to get $(4, 10)$ mean matrix
- 10: **return** *keyMeans* & *Annos*

ALGORITHM: AUTOMATIC LABELLING (TASK 2)

- 1: **input** video v to be labelled
- 2: **repeat** algorithm for Task 1 to get *featureMat* of all shots s out of v
- 3: **learn** using *kmeans* to get v 's own $(4, 10)$ mean matrix
- 4: **check similarity** with mean matrices of all keywords
- 5: **assign** closest matched keyword to all shots s

ALGORITHM: QUERY PHASE (TASK 3)

- 1: **query** $q \leftarrow$ user input
- 2: **search** for all shots s having the input keyword q
- 3: **return** *frameID* and *videoID* of all found shots

6. EXPERIMENTS

6.1 SETUP: SUMMARY OF VIDEO CORPUS

The database we used was a series of videos from popular TV Series ('Big Bang Theory', 'Merlin' and 'Sons of Anarchy'). We have tested our algorithms for 4 – 5 videos. The dataset covers a range of types of content in videos, ranging from dark to bright colors, outdoor, humans, greenery, mountains, roads, etc.

6.2 DISCUSSION ON RESULTS FROM METHOD 1

For a given image, after being searched against a video, the result shows the frame matched for a particular video.

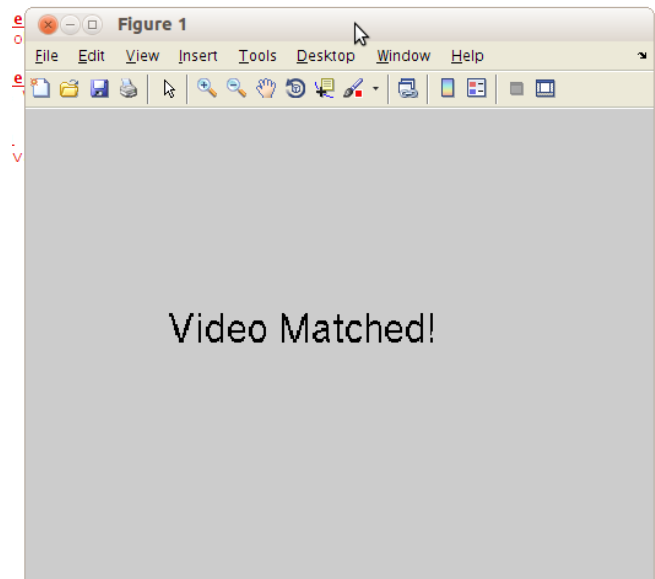
From experiments, it is evident that the Euclidean distance can never be zero as there is always some differences. So this function uses a threshold value of 20000. Any frame whose difference is less than the threshold is declared to be similar. Note that matching is never 100 %.



6.3 DISCUSSION ON RESULTS FROM METHOD 2

Since our method works on the principle that we give it two video (one from user query and other from database), it generates there feature vectors and compares those online, hence our output is in the form of either matched or not matched.

So for a video not in the dataset, we gave it as a query, and searched for videos similar, and for one of those (in our search), we found a close match. The distance function we are using is the same as in task 1 i.e. Euclidean distance.



6.4 DISCUSSION ON RESULTS FROM METHOD 3

Some of the results from the query phase are shown here. Here, one thing to note is that as we keep increasing our database by automatic semantic labelling, our *kmeans* learned distributions for each concept also is updated, this makes our system better with time.

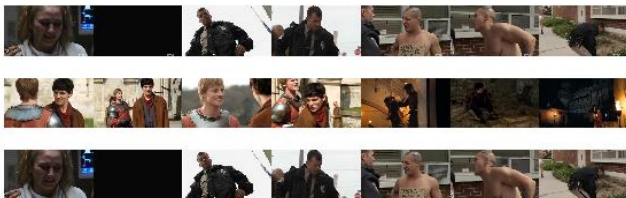
In following results, the dataset we used was of around 8

videos out of which 3 were initially manually marked and 5 were automatically marked. In each video we have around 9-12 shots with each shot of approximately 8-9 frames. So, dataset is good enough for learning the following list of eight concepts:

traffic	greenery	indoor
mountain	night	outdoor
man	woman	

One more thing in this regard is that we are using SAD as a measure of similarity between the learned distributions at the time of learning and classifying. And the value of SAD corresponds to the *confidence* of a particular video containing a particular concept.

Query: 'outdoor'

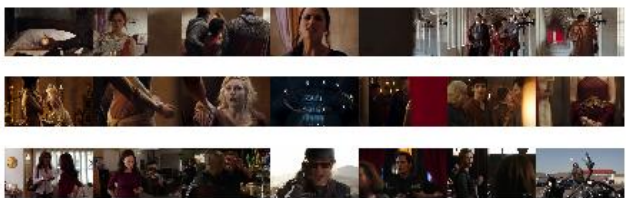


```

>> Query
Enter the next keyword (if no more press q)outdoor
Enter the next keyword (if no more press q)q
Match # 2
Frame: 64736.2648 to 66174.8262
Video ID: D:\Personal\LUMS\Spring14\DIP\Project\dataset\Sons.of.Anarchy.S01E03
Keyword query: outdoor
*****
Match # 3
Frame: 22501 to 24001
Video ID: D:\Personal\LUMS\Spring14\DIP\Project\dataset\Merlin S01E01 (BHARRU)
Keyword query: outdoor
*****
Match # 4
Frame: 64736.2648 to 66174.8262
Video ID: D:\Personal\LUMS\Spring14\DIP\Project\dataset\Sons.of.Anarchy.S01E03
Keyword query: outdoor
*****

```

Query: 'woman'




```

>> Query
Enter the next keyword (if no more press q)woman
Enter the next keyword (if no more press q)q
Match # 1
Frame: 30001 to 31501
Video ID: D:\Personal\LUMS\Spring14\DIP\Project\dataset\Merlin S01E03 (BHARRU)
Keyword query: woman
*****
Match # 2
Frame: 52501 to 54001
Video ID: D:\Personal\LUMS\Spring14\DIP\Project\dataset\Merlin S01E01 (BHARRU)
Keyword query: woman
*****
Match # 3
Frame: 50350.6504 to 51789.2118
Video ID: D:\Personal\LUMS\Spring14\DIP\Project\dataset\Sons.of.Anarchy.S01E04
Keyword query: woman
*****

```

Query: 'man'



```

Original Window
>> Query
Enter the next keyword (if no more press q)man
Enter the next keyword (if no more press q)q
Match # 1
Frame: 7501 to 9001
Video ID: D:\Personal\LUMS\Spring14\DIP\Project\dataset\Merlin S01E03
Keyword query: man
*****
Match # 2
Frame: 37501 to 39001
Video ID: D:\Personal\LUMS\Spring14\DIP\Project\dataset\Merlin S01E03
Keyword query: man
*****
Match # 3
Frame: 30001 to 31501
Video ID: D:\Personal\LUMS\Spring14\DIP\Project\dataset\Merlin S01E01
Keyword query: man
*****

```

6.4 COMPARISON B/W METHODS

The last method of semantic labelling is fastest and the best method in terms of query response time. But, it has limitations that for a small dataset it doesn't give very nice results. The other two methods we employed were good too but their performance is same for larger and smaller datasets. Also note that, our approach in method 3 of using *kmeans* clustering can be improved by training better classifiers.

Another improvement, that can drastically change the performance of method 3, would be to not use uniform sampling but using an intensity-maximizing technique to gather shots in videos. One more thing that can be easily done to increase dataset is to incorporate multiple keywords for the same shot. Lastly, we do employ a confidence metric for sorting our videos, which shows that our approach has the information on which shot of certain video has how much concept.

The retrieval time for semantic labelling is around 2 seconds, and processing one video for automatic labelling its shots takes around 40 seconds (video of length 45 minutes). This time also includes the learning time for *kmeans*. Retrieval times for the first and second approach is larger, and so is the duration for building of feature vector database quite long even if we employ 5:8 quantization (i.e. 8 bits are mapped to 5 bit numbers).

7. CONCLUSIONS

In the end, we have demonstrated that our method of semantic labelling and near duplicate detection gives nice

video retrieving results. The uniqueness of our approach lies in the fact that we train a simple classifier. Although *kmeans* has its limitations of outlier problem, etc., but because of a smaller and cleaner dataset our approach and proof of robustness of the algorithms stands out.

8. REFERENCES

- [1] Wu, Xiao, Alexander G. Hauptmann, and Chong-Wah Ngo. "Practical elimination of near-duplicates from web video search." Proceedings of the 15th international conference on Multimedia. ACM, 2007.
- [2] Arman, Farshid, Arding Hsu, and Ming-Yee Chiu. "Image processing on compressed data for large video databases." Proceedings of the first ACM international conference on Multimedia. ACM, 1993.
- [3] Chang, Shih-Fu, et al. "A fully automated content-based video search engine supporting spatiotemporal queries." Circuits and Systems for Video Technology, IEEE Transactions on 8.5 (1998): 602-615.
- [4] Karpenko, Alexandre, and Parham Aarabi. "Tiny videos: a large data set for nonparametric video retrieval and frame classification." Pattern Analysis and Machine Intelligence, IEEE Transactions on 33.3 (2011): 618-630.
- [5] Kennedy, Lyndon S., and Shih-Fu Chang. "A reranking approach for context-based concept fusion in video indexing and retrieval." Proceedings of the 6th ACM international conference on Image and video retrieval. ACM, 2007.
- [6] Irani, Michal, et al. "Efficient representations of video sequences and their applications." Signal Processing: Image Communication 8.4 (1996): 327-351.
- [7] Miller, John David. "Video indexing protocol." U.S. Patent No. 6,064,438. 16 May 2000.
- [8] Hu, Weiming, et al. "A survey on visual content-based video indexing and retrieval." Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 41.6 (2011): 797-819.
- [9] Cai, Yang, et al. "Million-scale near-duplicate video retrieval system." Proceedings of the 19th ACM international conference on Multimedia. ACM, 2011.