Content Based Video Retrieval

CSE 509 Digital Video Processing, Fall-2015

Jenil Jain
Arizona State University
jpjain@asu.edu

Jayaram Theegala
Arizona State University
jtheegala@asu.edu

Abstract—**In this work a content based video retrieval system is implemented using the low level features of the video such as motion vectors. A dataset containing set of videos is passed to the training module which first generates the motion vectors using appropriate window size. These generated motion vectors are clustered with a predetermined number of clusters. This trained dataset can now be used for retrieving relevant videos for a given input video. An average motion vector is calculated for each clustered segments and the input video which are then compared using Euclidean distance and Cosine distance. The segment with the least distance in comparison to the input implies they are most similar. Finally, all the calculated comparisons are used to create the ranking system which uses a Heap to store the retrieved videos.**

Key terms—**Content-based video retrieval; motion estimation; low level feature extraction.**

I. Introduction

With ever growing digital data, retrieval of relevant videos has become key for deep learning and building insights. Although text based retrieval techniques are widely used in the internet search engines owing to their computational efficiency and low retrieval time, they are not sufficient for all the applications. This technique can also be prone to errors if human intervention is needed for building the annotations for the videos. Content based retrieval can be done at two levels: (1) Low-level based, in which the low level features such as color, shape, texture, power spectrum, pitch or motion vectors are used to retrieve the relevant videos. (2) High-Level based retrieval pertains to semantics of the video. The high level feature that interests us can be anything ranging from scoring a goal in a soccer game to car accident retrieval in traffic videos.

II. Background

One of the most important part of CBVR is video segmentation. In this project we used K-Means clustering technique for creating the video clip segments from all the videos in the dataset. K-Means starts with random seeds whose number is equal to the given number of clusters. Then it iteratively moves the seeds to the centroids of the clusters calculated during that iteration. This way all the data points closer to one seed forms one cluster. After the video segmentation to compare the input motion vector with dataset motion vector we used Euclidean and Cosine distance. Euclidean distance is the distance between points. If the motion vectors of the input and the dataset videos has similar magnitudes, then they are closer in the Euclidean space. It is also means that they have similar amount of motion at the given position. We also used Cosine distance between the input and dataset motion vectors. It is expected to perform better as it also accounts for the direction of the vectors and not just magnitude.

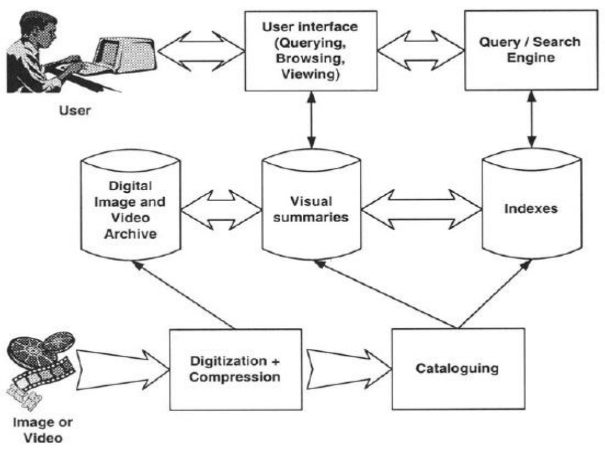

User
User interface (Querying, Browsing, Viewing)
Query / Search Engine
Digital Image and Video Archive
Visual summaries
Indexes
Image or Video
Digitization + Compression
Cataloguing

Fig 1. Block Diagram of CBVR System [5]

The above diagram shows the various segments that involved in the CBVR system. Active research has been going in every aspect of the system. In [1] Motion estimation, color, edge, texture have been used to identify the key frames. The identified distinct features are used to rank the output videos retrieved. [2] An efficient clustering method is implemented for generating a shot cluster tree structure which facilitates querying the video based on either content or by example. It uses visual similarity and time adjacency in forming the clusters. [3] In this paper a Feature selection technique for CBVR is implemented. The salient object within video sequence are extracted through segmentation. Then during retrieval the spatio-temporal properties are compared using Hausdorff distance matching.

III. Proposed Framework

The low level feature we used for our implementation was motion vectors. Detecting motion along a particular region of the video can give many details about the content. Motion can be estimated using many algorithms. The limitations on this algorithm is the computation time and power. For calculating pixel by pixel motion vector of two 480 x 640 frames using

Lucas-Kanade algorithm took us 6 secs. So for a 5 minute video with framerate 30, we have 9000 frames and it will take 54000 sec, i.e. 15 hours! Even saving the motion vectors in mat files will take memory in order of Gigabytes. MATLAB is not capable of handling files that large. So the motion vectors were calculated for consecutive frames at an interval of 5. This reduced the overall time as well as space. As Lucas-Kanade is slow, a better and fast motion estimation algorithm developed by Ce Liu [4] is used. It still takes 1 sec for a motion vector, but comparatively it's fast.

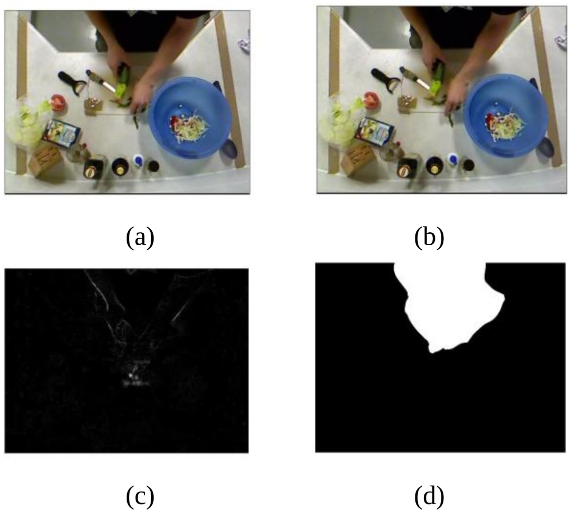
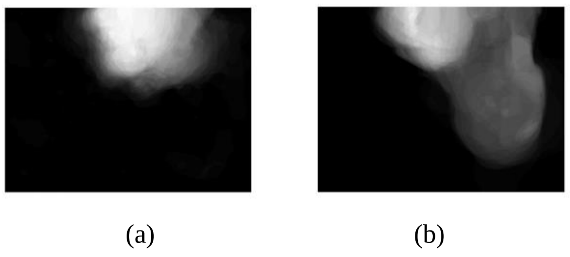


Fig 2. a frame 1, 1.b frame 2. 1.c difference between them, 1.d motion segmentation along the difference.

The analysis has been carried with only one activity in the input video. Thus motion for that activity will be along some particular region. Segmenting the image based on motion vector using thresholding technique will give us clear picture of what kind of activity is done in which region. For example in the 50 salad dataset, cutting of ingredients is done at the center while mixing is done at the right side for most of the videos. Thus for cutting motion vector will be along the center of the image and for mixing the ingredients, motion vector will be along center as well as right. Similarly preparing the dressing will have motion along bottom as well as center. Thus different activities will have different regions of motion vector. Since it is just low level feature extraction, we cannot differentiate cutting of tomato and cutting of cucumber.



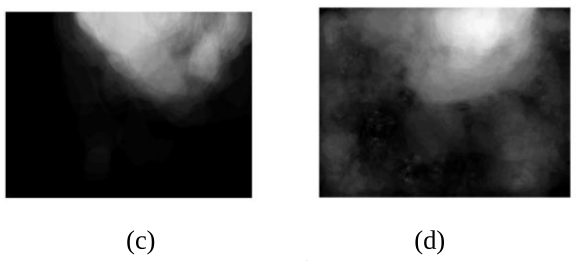


Fig 3. a Averaged motion vector for a cutting clip, 2.b averaged motion vector for a mixing ingredients clip, 2.c average motion vector for preparing a dressing, 2.d average motion vector for some random activity. For cutting motion vectors are concentrated mainly at the center. For mixing it is concentrated along center as well as right side and for dressing you can see some motion along down side.

IV. Implementation

After extracting the motion vectors and segmenting the image, we classified the frames of the videos in the dataset based on the motion. The classification method we used was K-means clustering. K-Means clustering aims to partition n observation into k clusters in which each observation belongs to the clusters with the nearest mean.

K-means in MATLAB take a matrix of n x p where n is the number of observations and p is the number of variables. So in our case n is the number of motion segmented frames and p is the number of pixels in the frame. Since we just need to know in which region the motion has occurred, we divide the whole frame into W x W grid. A grid represent the average value of the pixels in it. So if the motion occurred in that region the average value of the region will be higher than the region with no motion. Hence comparing the regions, we can deduce which activity has occurred. We choose value of W to be 80. Dividing the image into grid also reduce the variable for the k-means.

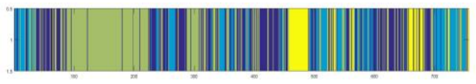


Fig 4. a video clustered using k-means. Region with green uniform area is the cutting of vegetables, region with yellow is the dressing preparation, region with blue region is the mixing ingredients and purple are the one with random activity. There are blue stripes between regions of yellow and green, that is because of adding the ingredients into the bowl.

After clustering the frames in the video, we need to separate the video into sequences of similar k-mean ID. The region with uniform color represents a single activity. Splitting the video into sequences is a onetime process and we can save this information for future iterations. Hence we don't have to repeat this procedure every time a video is added as input. Comparing this sequences with the input video can show how similar they are. For similarity measure we can use either Euclidean distance or the cosine distance. Rankings can be assigned to these clips based on any of the above measurements.

For ranking the clips we used a simple Heap data structure to store all the clip's information in ascending order of their respective Euclidean or cosine distance from the input clip.

From each video in the dataset, 3 top clips are picked by extracting min from the heap. Once all the videos are processed, top 10 out of the total clips are selected and displayed in a list.

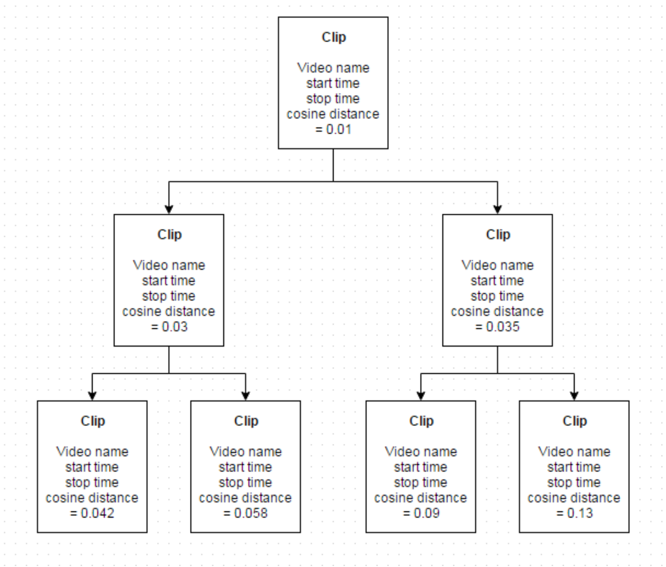
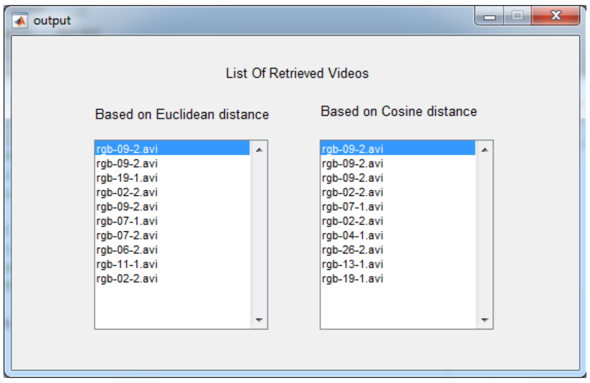


Fig 5. A MinHeap storing all the clips information based on their respective Euclidean or cosine distance from the input clip.

V. Results

Dataset of 31 videos was selected and motion estimation and segmentation was performed first. These motion vectors were saved in mat files, so that time can be saved for future retrieval. Dividing the segmented frames, performing k-means clustering and splitting the video into sequences of similar frames, takes around 2 min per video. Again this time can be saved by storing all these information. So whenever new videos are added into the dataset, the old videos are not processed again.

When an input video is selected, all the clips from all the videos in the dataset are compared based on Euclidean and cosine distance. Calculating these distances don't take much time. The ranking list generated by both this parameter contains same clips which are similar to the input videos. But the accuracy of cosine distance is better than Euclidean distance. Out of 10 videos retrieved based on cosine distance, 7 to 8 videos have motion occurring in same regions as in input videos. While video retrieved based on Euclidean distance also have similar videos, but compared to cosine, the number of similar videos are less.



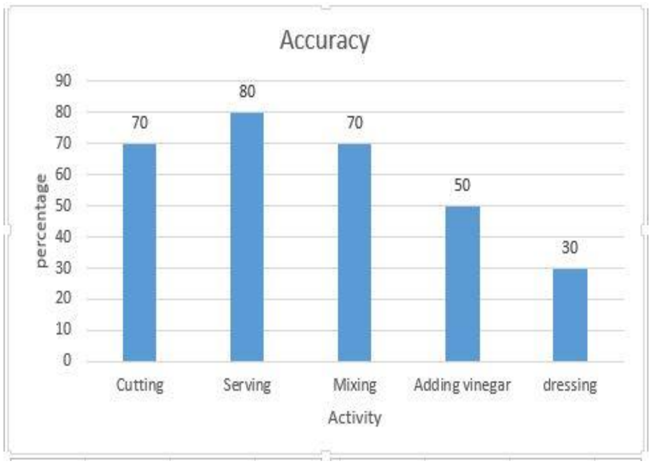
List Of Retrieved Videos

Based on Euclidean distance	Based on Cosine distance
rgb-09-2.avi	rgb-09-2.avi
rgb-09-2.avi	rgb-09-2.avi
rgb-19-1.avi	rgb-09-2.avi
rgb-02-2.avi	rgb-02-2.avi
rgb-09-2.avi	rgb-07-1.avi
rgb-07-1.avi	rgb-02-2.avi
rgb-07-2.avi	rgb-04-1.avi
rgb-06-2.avi	rgb-26-2.avi
rgb-11-1.avi	rgb-13-1.avi
rgb-02-2.avi	rgb-19-1.avi

Fig 5. Comparing both the list we can see number of similar clips is high. And those clips are also similar to the input. But if we compare the non-similar ones, Cosine videos are more accurate than Euclidean.

The feature we used for comparing and retrieving similar videos is the motion vector, which is a low level feature. Hence given an input video of cutting a tomato can retrieve videos with cutting of lettuce and cucumber. To differentiate between them, we need some high level features. The retrieved videos and the input video may have different activity present and their annotations may be different, but the overall activity present in both the videos will have motion along same regions. For example, for mixing the ingredients we can get videos with serving salad or placing ingredients in the bowl.

The Accuracy has been measured for various activities. The retrieved videos are videos are verified by human intervention. For instance, if the input video of cutting tomato is given any cutting video retrieved is a success and failure if otherwise. Based on the top 10 results retrieved for various activities the accuracy percentage is plotted against various activities. We can see Dressing accuracy is lowest as its motion vectors were close to mixing and serving most of the time.



VI. Conclusion

Content based Video Retrieval system using Motion Estimation and Clustering has been implemented. The system also has a loosely coupled user interface which allows to provide input video and watch the retrieved outputs. Motion vectors have been used for estimating a high level activity and also to rank the videos in the dataset with respect to a given input video. As our next work we will be using color features to detect cutting of different items and also improve the accuracy.

.

REFERENCES

[1] P.Geetha, Vasumathi Narayanan, "An Effective Video Search Re-Ranking for Content Based Video Retrieval"

[2] WEN-GANG CHENG, DE XU, "CONTENT-BASED VIDEO RETRIEVAL USING THE SHOT CLUSTER TREE"

[3] Sameh Megrhi, Wided Souidene, "Spatio-Temporal Salient Feature Extraction For Perceptual Content Based Video Retrieval"

[4] Ce Liu, https://people.csail.mit.edu/celiu/OpticalFlow/

[5] B.V.Patel, B.B.Meshram, "Content Based Video Retrieval"