On Fuzzy Clustering and Content Based Access to Networked Video Databases

Anupam Joshi[1], Sansanee Auephanwiriyakul[1], and Raghu Krishnapuram[2]

[1] *CECS Department, University of Missouri, Columbia, MO-65211*

[2] *Dept. of Math and Computer Science, Colorado School of Mines, Golden, CO-80401*

joshi@cecs.missouri.edu, sansanee@ece.missouri.edu, rkrishna@mines.edu

Abstract

Video Databases and video on demand represent an important application of the evolving Global Information Infrastructure. However, video querying involves a lot of user interaction and feedback based query refinement, which can generate large traffic volumes on the network if full video segments are sent. To aid in efficient Video browsing, search and retrieval across the network, we need to find good compact representations for long video sequences. Representative Frames (Rframes) provide such a representation. Extant algorithms use use scene change detection to segment video into shots and pick Rframes. However, scene change detection techniques fail badly in presence of gradual scene changes which are quite prevalent in most videos. We present another way of finding Rframe using fuzzy clustering without dealing with any scene change detection algorithms. Fuzzy clusters provide a more natural approach to this problem since membership of a frame in some particular scene is not binary. This allows us to handle gradual scene changes. We report on our approach, present preliminary experimental results, and discuss ongoing work.

1. Introduction

A significant and ever increasing portion of the total information being created today has an audiovisual format. Given the vast amount of this information, attempts to organize and store it in the form of video databases are gaining prominence. While video databases share several characteristics with traditional databases, they also raise several unique and novel avenues of research[5]. The fundamental reason for this is the nature of video data, where the semantics are inherent in, and intermingled with, the data itself. This is in contrast to the traditional systems, where the data and semantics are clearly distinguishable. For example, numbers in a traditional database mean almost nothing in of themselves, and acquire meaning only when we know what fields in a table they represent. On the other hand, a frame or a video sequence stored in a database carries meaning in of itself.

One of the most important challenges that arises in video databases is providing content-based query and retrieval. Such queries represent the common manner in which humans would interact with video databases - for example, asking to find *all scenes where Humphry Bogart and Ingrid Bergmann are together in a cafe*. A further complication arises because the interpretation of content is not just based on the data, but also on the person querying. So a query of the form *show clips of ski resorts* will likely result in different sequences for a person living in the US (maybe clips about Aspen) and Switzerland. In almost all applications, accessing video data based on its content is more natural and germane to humans.

In most realistic scenarios, video databases will be accessed across the network, and in an interactive manner. Bolle *et. al.* [4] have proposed a sequential model for such access that contains four steps. These are Navigating (where the user provides broad categorical information about the clips s/he wants such as news, sports etc.), Searching (where the user narrows down the candidate video pool using principally text based attributes such as keywords, date, creator etc.), Browsing (where the user refines his search by browsing through the candidate video streams), and finally viewing. Given the high bandwidth needed for video data, it is evident that sending complete video streams from the server during the browsing phase is not a good solution. Thus the problem of efficiently representing the visual content of a video stream, in the sense of providing maximal semantic information while using minimum bandwidth, becomes extremely important for networked video databases. There is a need to augment any video data model with a compact representation that can be presented to the user during the browsing mode. A full video stream is sent across network only when the user explicitly wants it at the end of the querying process. Note that this is not the same as compressing the video stream, which is done by most video formats anyway. Nor is it simply a matter of creating a metadata framework for the content. Rather, the aim is to obtain some compact representation of the given video stream. One such representation is the *Representative Frames*, or RFrames. The effort here is to identify some key image frames that can visually summarize the video stream to the user.

The visual representation of logical video segments is closely related to problem of video Scene Change Detection (SCD) which has been extensively studied in literature [10]. The basic idea of SCD algorithms is to measure inter-frame differences in terms of visual features, and declare a scene change once the difference exceeds a certain threshold. These features include luminance, color, spatial correlation, object shape, motion contained in the video image, or DC coefficients in case of compressed video data. The frames with little significant difference form a scene or a shot, and can be represented by any single frame from set, which we call a representative frame (Rframe). For example, if a logical video segment is $v_{10\text{-}100}$, and the Rframe set from the whole video is $\{v_1, v_{40}, v_{75}, v_{120}...\}$, then $\{v_{40}, v_{75}\}$ can be used to visually represent the segment.

Scene change in a video sequence can be either abrupt or gradual. Abrupt scene changes result from editing "cuts". Gradual scene changes result from chromatic edits, spatial edits, and combined edits [7]. They include special effects like zoom, camera pan, dissolve, and fade in/out. Gradual scene changes generally are more difficult to detect than abrupt scene changes and may cause the majority of scene detection algorithms to fail under certain circumstances. Consequently, algorithms that select Rframes based on SCD fail as well. Jiang [9] discuss problems of current SCD algorithms.

The rest of the paper is organized as follows. In section 2, we review existing work in this area. We next outline the algorithms and features that we used. In section 4, we present the results of our experiments and future directions are discussed in section 5.

2. Related Work

There have been a few studies on finding the "key frames" in a video segment. For example, Armen et al. [2] compared the similarity of frames between shots using a combination of the shape of object (measured by the gray level moments) and the color histogram. Tonomura and Akutsu et al. [1, 8, 14, 15] computed the indexes for every three frames in order to achieve real-time video indexing. However, using these will lead to a large number of Rframes, which defeats the purpose. Yeung and Liu [16] found the Rframe for each shot using dissimilarity index over some error threshold. They count the first frame as being representative and then skip the next *n* frames whose dissimilarity is below a threshold. The first frame whose dissimilarity exceeds the threshold is made a representative frame, and the process repeated. This scheme is highly susceptible to noisy frames though. Note that most existing methods such as these are still using SCD in one form or another.

In this paper, we use fuzzy clustering to group similar frames together. We do not explicitly use any scene change detection algorithms. The usage of fuzzy techniques is particularly appropriate here because frames can be "similar" to one another (and hence belong to scenes) to different extents. Traditional approaches which insist on a frame belonging to one group only fail, as has been noted earlier, when confronted with gradual scene changes. Using soft decision criteria allows us to overcome this weakness of existing approaches.

3. Features and Algorithms

Traditional video databases use image properties of each frame such as color histogram, Chi-square histogram etc to compute measures of dissimilarity between frames. In our experiments, we use two properties described in literature [14]. One is a simple per pixel measure. Suppose the size of image frame is $J \times K$. The pixel values of the image frame $f(j, k)$, $j = 1, 2,..., J$ and $k = 1, 2, ..., K$ can be viewed as a feature vector of dimension $J \times K$ representing this frame. The other uses row-column projection. The projection for *n*th row is $l_n^r = \sum_{j=1}^{J} f(j,n)$ and the projection for *m*th column is $l_m^c = \sum_{k=1}^{K} f(m,k)$. The projection for each row and column give a feature vector of dimension $J + K$.

The features are obtained from the luminance (Y) and the two chrominance (U, V) channels in an MPEG stream [10]. We used just the luminance, and also a combination of luminance and chrominance images extracted from compressed MPEG as features. The image size from MPEG is 160×128 and 352×240 in the videos we considered. We donwsampled these by a factor of 4 and 8, reducing them to 40×32 and 44×30 in size. This is similar to the DC component of the MPEG stream. Note that this choice of features requires uncompressing the video stream. In ongoing work, we are investigating the use of features such as DC component and motion vectors that can be obtained directly from the MPEG data.

Fuzzy set theory (and clustering) has been extensively used in pattern recognition and computer vision. A classic text in this area is due to Bezdek[3], while Klir[17] presents more recent and comprehensive information. Also germane to our effort are the work of Frigui and Krishnapuram [6], who combined fuzzy clustering and robust statistical estimators in order to find the number of cluster and obtain the prototype parameter from noisy data. Krishnapuram and Keller [12] introduced possibilistic clustering which was different from ordinary fuzzy clustering in that the sum of memberships of a data point across clusters does not need to be 1.

We now describe briefly the algorithms used in our present system, starting with some notation. Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ be a set of vectors, where each vector is *p*-dimensional. Let $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_c\}$ represent a *C*-tuple of prototypes, each of which is the center of a cluster. Further, u_{ij} is the membership value of vector $\mathbf{x}_j$ in cluster. The constraint for the membership (except in the case of possibilistic C means) is:

$$\sum_{i=1}^{C} u_{ik} = 1, \quad 1 \le k \le n \qquad (1)$$

$$0 < \sum_{k=1}^{n} u_{ik} < n, \quad 1 \le i \le c$$

Given these features, we can obviously define a Euclidean distance between the vectors. We also define L_1 like distances better suited for image similarity. In case of pixel based features, we use a correlation between vector and center defined as:

$$d(\mathbf{x}_m, \mathbf{c}_n) = \sum_{j=1}^{J}\sum_{k=1}^{K} \left| f^m(j,k) - f^n(j,k) \right| \qquad (2)$$

For the row-col projections, the sum of absolute difference of projection distance normalized to [0, 1] , as described in equation (3), is used.

$$d_{ij} = \frac{1}{255 \times (J+K)} \left(\frac{1}{J} \sum_{n=1}^{K} \left| l^r(i)_n - l^r(j)_n \right| + \frac{1}{K} \sum_{m=1}^{J} \left| l^c(i)_m - l^c(j)_m \right| \right) \qquad (3)$$

3.1 Hard C-Median Algorithm

In this algorithm[3], $u_{ij} = 1$ if vector x_j is assigned to cluster i , and 0 otherwise. Each vector is assigned to the cluster with the "nearest" prototype.
The objective function for this is:

$$J = \sum_{i=1}^{C}\sum_{j=1}^{N} u_{ij}\, d(\mathbf{x}_j, \mathbf{c}_i)$$

The distance here is 1-norm distance as equation (2) and (3). The minimization of this objective function with respect to the prototype centers on each dimension *k* of each cluster *i* will give

$$\sum_{j=1}^{N} u_{ij} \operatorname{sgn}(\mathbf{x}_j(k) - \mathbf{c}_i(k)) = 0 \qquad (4)$$

Since we are using an L_1 distance, the update of the centers will be done using the median along each dimension, rather than the (more typical) mean. The algorithm for Hard C-Median thus becomes

Fix number of the clusters C;
Initial the center using FCM
***Do**{*
Assign vector to the closest center;
On each dimension find median for each cluster;
*} **Until** (center stabilize)*

3.2 Fuzzy C-Median and Fuzzy C-Means(FCM)

This algorithm was first introduced by Bezdek [3], and is one of the classic fuzzy clustering algorithms. The objective function for FCM is:

$$J = \sum_{i=1}^{C}\sum_{j=1}^{N} \left(u_{ij}\right)^m d^2(\mathbf{x}_j, \mathbf{c}_i) \qquad (5)$$

where $u_{ij} \in [0,1]$ for all *i* and obeys the constraint defined in equation (1). The minimization of the objective with respect to membership values leads to

$$u_{ij} = 1 \Big/ \sum_{k=1}^{C} \left(d^2(\mathbf{x}_j, \mathbf{c}_i) \Big/ d^2(\mathbf{x}_j, \mathbf{c}_k) \right)^{1/(m-1)} \qquad (6)$$

The minimization of the objective function with respect to the center of each cluster will gives us:

$$\mathbf{c}_i = \sum_{j=1}^{N} \left(u_{ij}\right)^m \mathbf{x}_j \Big/ \sum_{j=1}^{N} \left(u_{ij}\right)^m \qquad (7)$$

In above equation $m \in [1, \infty)$ is the fuzzifier. The algorithm is thus

Fix number of the clusters C;
Fix the fuzzifier m;
***Do** {*
Update member ship using equation (6)
Update center using equation (7)
*} **Until** (center stabilize)*

When the 1-norm distances defined in (2) and (3) are used, we once again need to update our centers using the median instead of the mean. The minimization of the objective function with respect to membership will remain as in (6). The minimization of the objective function with respect to center on each dimension of each cluster will give us the fuzzy median that is the root of this equation [11] for dimension *k.* The algorithms for Fuzzy C-Median is

Fix number of the clusters C;
Initial the center using FCM
***Do**{*
Update membership using equation (6)
with 1-norm distance;
On each dimension find fuzzy median
for each cluster;
*} **Until** (center stabilize)*

3.3 Possibilistic C-mean (PCM)

This algorithm was introduced by Krishnapuram and Keller, and defines the objective function as [12]:
The constrain for membership is as following:

$$0 < \sum_{k=1}^{n} u_{ik} < n, \quad 1 \le i \le c$$

$$J = \sum_{i=1}^{C}\sum_{j=1}^{N}\left(u_{ij}\right)^m d^2\left(\mathbf{x}_j, \mathbf{c}_i\right) + \sum_{i=1}^{n}\eta_i \sum_{j=1}^{n}\left(1-u_{ij}\right)^m$$

η_i determine the relative degree to which the second term in the objective function is important compared with the first and is defined as

$$\eta_i = \frac{\sum_{j=1}^{N}\left(u_{ij}\right)^m d^2\left(\mathbf{x}_j, \mathbf{c}_i\right)}{\sum_{j=1}^{N}\left(u_{ij}\right)^m} \tag{8}$$

The minimization of the objective function with respect to membership gives us

$$u_{ij} = \frac{1}{1+\left(\dfrac{d^2\left(\mathbf{x}_j, \mathbf{c}_i\right)}{\eta_i}\right)^{\frac{1}{m-1}}} \tag{9}$$

The algorithm for PCM thus becomes

Fix number of the clusters C;
Initial the center using FCM
Estimate η_i using equation (8);
***Do**{*
Update membership using equation (9);
Update center using equation (7);
*} **Until** (center stabilize)*

4. Experimental Results

In this section, we present the result of our shot clustering experiments. We run the algorithms described earlier on several video segments, but only show the results of two representative samples here due to space constraints. They are the short videos of an Eric Clapton concert (which involves gradual scene changes due to mixes) and a flower garden, which involves a slow camera movement from left to right. The first sequence comes from http://proximity.cs.purdue.edu: 7000/mpeg/clapton.mpg and the latter comes from ftp://mm-ftp.cs.berkeley.edu/pub/multimedia/mpeg/ movies. These (and our other experimental data) may be viewed at http://meru.cecs.missouri.edu/~sansanee/mpeg to better appreciate the results presented here.

We use FCM and Fuzzy C-Median with the fuzzifier equal to 1.5. For Hard C-Median, we initialize the centers with the results of FCM with m equal to 2 or 1.5. For PCM we initialize the centers with the result of FCM with m = 1.5 and use m = 1.3 when running PCM. We let humans observe these video segments and estimate the number of clusters. We use that number to be our initial approximation. We show the results when using luminance features only because no substantial improvement was observed when using a combination of luminance and chrominance as features.

The actual prototype center and the nearest neighbor of the prototype for each cluster found in the two videos are shown in Figures 1 and 2 respectively. From these figures it appears that the algorithms find reasonable representative frames for video sequence even in the presence of gradual scene changes. This is a significant improvement over extent techniques. The prototypes from FCM are not substantially different from Fuzzy C-Median and Hard C-Median using correlation distance. Also, the nearest neighbor for each cluster center from these three algorithms comes from the same shot, either the same frame or the adjacent frame. The results from using PCM are different from the other algorithms on most video sequences that we tried it on. It seems to us that PCM tends to put all shots that represent a gradual scene change into the same group. We are investigating to see if this is an artifact of the specific videos we used, or some choice of parameters.

We observe that the result from using the projection of row-col gives results similar to when all pixels are used as features. The projection of row-col is clearly more efficient because it needs less computational time and space. The algorithms run quite fast, and can be used in near real time mode.

We have a choice on whether to use the actual center prototype as the Rframe, or actual frame nearest to it. Humans can make sense from the actual prototype only if the diversity of the frames within the cluster is small.

The nearest neighbor therefore generally serves as a more meaningful Rframe for human vision.

5. Ongoing Work

In this paper, we present fuzzy shot clustering used as a tool to find representative frames for video sequences. By not using scene change detection in the process, we avoid the problems faced by many existing algorithms. The use of fuzzy membership also allows us to handle video sequences with gradual scene changes. We have tested several different feature choices, distance measures, and algorithms. Our initial results are very encouraging and show good performance in shot clustering that conforms to human judgment.

In our ongoing effort, we are expanding this work along several dimensions. First, we are investigating better algorithms. We are experimenting with algorithms that automatically estimate the number of clusters. We are also interested in robustifying these methods, as well as producing hierarchical approaches. Hierarchical clusters will allow us to provide users with "drill down" abilities which are important for video browsing. Second, we are investigating the use of features that do not require uncompressing the video streams. These include the DC Component, and the motion vectors. The development of MPEG4 will also allow us to group data together as visual objects, and look for changes in these groups of features. Moreover, we are investigating the use of a combination of visual and audio features to do multimodal clustering. Finally, at present we have a simple web based front end to the system that uses HTTP/1.1 content negotiation to return the Rframe or the actual video. We are creating a Java based front-end tool that will mimic the actual browsing and content based retrieval process. This tool will be integrated with the VideoText Querying and Video model that has recently been proposed by our colleagues[8].

6. References

[1] A.Akutsu and Y.Tonomura, "Video tomography: An efficient method for camerawork motion vectors, *In Proceedings Second Annual ACM Multimedia Conference*, October 1994. Association of Computing Machinery.

[2] A.H.F.Arman, et al, "Content-based browsing of video", *In Proceedings Second Annual ACM Multimedia Conference*, October, 1994.

[3] J.C.Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.

[4] R.M.Bolle, Boon-Lock Yeo and M.M.Yeung, "Video Query: Beyond the keywords", *IBM Research Report*, October 1996.

[5] A.K.Elmagarmid, H.Jiang, A.A.Helal, A.Joshi and M.Ahmed, *Video Database Systems: Issues, products and Applications*, Kluwer Academic Publishers, London, 1997.

[6] Hichem Frigui and Raghu Krishnapuram, "A Robust algorithm for automatic extraction of an unknown number of clusters from noisy data", *Pattern Recognition Letters*, 1996.

[7]. A.Hampapur, R.Jain and T.Weymouth, "digital video indexing in multimedia systems", *In Proceedings of the Workshop on Indexing and Reuse in Multimedia Systems*, August 1994.

[8] H.Hashimoto, A.Akutsu, Y.Tonumura and Y.Ohba, "Video indexing using motion vectors", *In Proceedings of SPIE: Visual Communications and Image Processing 92*, November 1992.

[8] H. Jiang, A. K. Elmagarmid, "Extract Visual Content Representation in Video Databases ", *Accepted by the International Conference on Imaging Science, Systems, and Technology (CISST'97)*, 1997

[10] H.Jiang, A.Helal, A.K.Elmagarmid and A.Joshi, "Scene change detection techniques for video database systems", *ACM Multimedia Information Systems*, 1996.

[11] P.R.Kersten, "Fuzzy Order Statistics and Its Application to Fuzzy Clustering", *Naval Air Warfare Center Weapons divion Report*, August 1995.

[12] R.Krishnapuram and J.M.Keller, "A Possibility Approach to Clustering", *IEEE Transactions of Fuzzy Systems*, vol. 1, No. 2 pp. 98-110, May 1993.

[13] A.M. Tekalp, *Digital Video Processing*, Prentice Hall, New Jersey, 1995

[14] Y.Tonomura and A. Akutsu, "A structured video handling technique for multimedia systems", *IEICE Trans. Inf. & Sys.*, E78-D(6), pp. 764-777, June 1994.

[15] Y.Tonomura, A.Akutsu, Y.Taniguchi and G. Suzuk, "structured video computing", *IEEE Multimedia*, vol 1 No 3, pp. 34-43, 1994.

[16] M.M.Yeung and B.Liu, "Efficient Matching and Clustering of Video Shots", *To be appear in the international conference on image processing*, October 1995.

[17] G. Klir and Bo Yuan, Fuzzy Sets and Fuzzy Logic, Prentice Hall, 1995

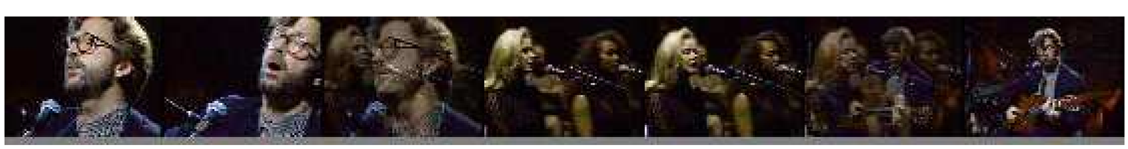
(a)

(b)

(c)

(d)

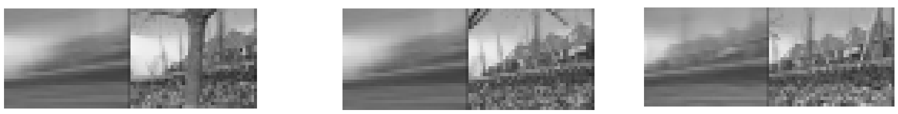
(e)



(f)

(g)

Fig. 1. Center and nearest neighbor luminance image of clapton.mpg. (a) From FCM (left-actual center, right-nearest neighbor). (b) Nearest neighbor of FCM (c) Nearest Neighbor of Fuzzy C-Median using correlation (left-actual center, right-nearest neighbor). (d) Nearest neighbor of Fuzzy C-Median using the sum of absolute difference of projection only nearest neighbor (e) Nearest neighbor of Hard C-Median using correlation (left-actual center, right-nearest neighbor). (f) Nearest neighbor of Hard C-Median using the sum of absolute difference of projection only nearest neighbor (g) Nearest neighbor of PCM (left-actual center, right-nearest neighbor.



(a)

(b)

(c)

(d)

(e)

(f)

(g)

Fig. 2. Center and nearest neighbor luminance image of flower.mpg. (a) From FCM (left-actual center, right-nearest neighbor). (b) Nearest neighbor of FCM (c) Nearest Neighbor of Fuzzy C-Median using correlation (left-actual center, right-nearest neighbor). (d) Nearest neighbor of Fuzzy C-Median using the sum of absolute difference of projection only nearest neighbor (e) Nearest neighbor of Hard C-Median using correlation (left-actual center, right-nearest neighbor). (f) Nearest neighbor of Hard C-Median using the sum of absolute difference of projection only nearest neighbor (g) Nearest neighbor of PCM (left-actual center, right-nearest neighbor.