Compressed Domain Techniques to Support Information Retrieval Applications for Broadcast News Videos

Tarun Jain C.V. Jawahar
Center for Visual Information Technology
International Institute of Information Technology
Hyderabad, India.

***Abstract*—Huge amount of video data gets generated every day which needs to be efficiently processed and effectively presented to the users for retrieval of relevant information. The focus of this paper is primarily to explore techniques which can support IR applications on large scale video databases. Almost 25 times faster approximation of a widely used color representation is proposed by using information from the MPEG compressed domain. A scheme is presented for real time matching of videos in online video feeds. LSH based indexing is performed for handling efficient near neighbor queries and support fast clustering. A video clip can be searched in more than 100 hours of video data in few seconds. The techniques are verified on large video data with the help of a specially designed experimental setup. The proposed techniques are shown to be significantly better than several existing and state of the art algorithms.**

I. Introduction

Large video repositories are getting into the mainstream with the advent of information technology. High speed Internet and commoditization of storage hardware catalyzes this trend. The increasing number of news channels results in an abundance of data which makes it difficult for the users to visualize and comprehend the available information. In India, a typical cable or satellite TV distribution network daily delivers approximately 200 hours of news in English, Hindi and regional languages. Users can never afford to view this entire news broadcast. They are interested to (a) Avoid watching irrelevant videos like advertisements and other repetitive content like channel logos, program lead-in and lead-outs. (b) Search for videos similar to a particular news story. (c) Watch the most important news stories over a duration, say a day or week. (d) Keep track of the developments in a particular news story.

In this paper, we focus on presenting techniques which can effectively support applications like above. A spotting technique has been proposed to address (a) which enables real time recognition of query clips in online video feeds. An LSH based indexing scheme is presented which supports near neighbor queries at interactive speeds. This forms a core component for applications like (b). (c) and (d) can be solved by exploiting the occurrence of common video clips in different news broadcasts across channels and time. Relationships can be established between news stories by clustering such visually similar clips which leads to importance ranking of news stories based on cluster size. Developments to news stories can be tracked by analyzing the growth of clusters and addition of new videos to them. Significant gains in computation of the clusters can be achieved by using the LSH index. An overview is provided in Figure 1.

Video content is often represented using key frames which reduces the problem of video retrieval to CBIR. Early systems based on this approach include QBIC [1] and VisualSeek [2]. The general solution for content based image and video retrieval has traditionally been to build feature descriptions of the content and then defining distance measures between these features for matching. Features can typically be grouped into two categories - low level and high level. Low level features are compact, mathematical representations of the physical properties of video data like color atmosphere, texture, shape, edge information and motion [3]. Popular high level representations attempt at representing video content in terms of objects, faces, activities, etc. There exist specialized techniques developed for addressing specific categories like sports videos. However, they are not necessarily effective for general and diverse content like broadcast news. A lot of work has been done on hierarchical representation of structure for easy browsing of video content. The main objective of these methods is to accelerate the process of manual browsing of videos and thus are suitable only for access of limited video content (say a movie or a specific news broadcast).

Several algorithms represent video as a sequence of feature vectors. Each feature vector represents a frame or a group of frames. The matching is performed by determining the alignment score between the two sequences by using sequential correlation [4] dynamic programming based techniques like Least Common Subsequence (LCS) and Approximate Subsequence Matching (ASM) with certain constraints [5], [6]. Such techniques lose on the computational efficiency as computing the correlation of features per frame gets expensive. These methods focus on very fine frame level matching and the efficiency of searching through large collections is neglected to some extent for the accuracy of matching. Moreover, these algorithms check for inherent temporal order of frames in the sequence representation. This further increases the order of matching. However video clips differing in only the temporal order of frames are not expected to be found in general broadcast videos. This motivates us to work with video *clips*

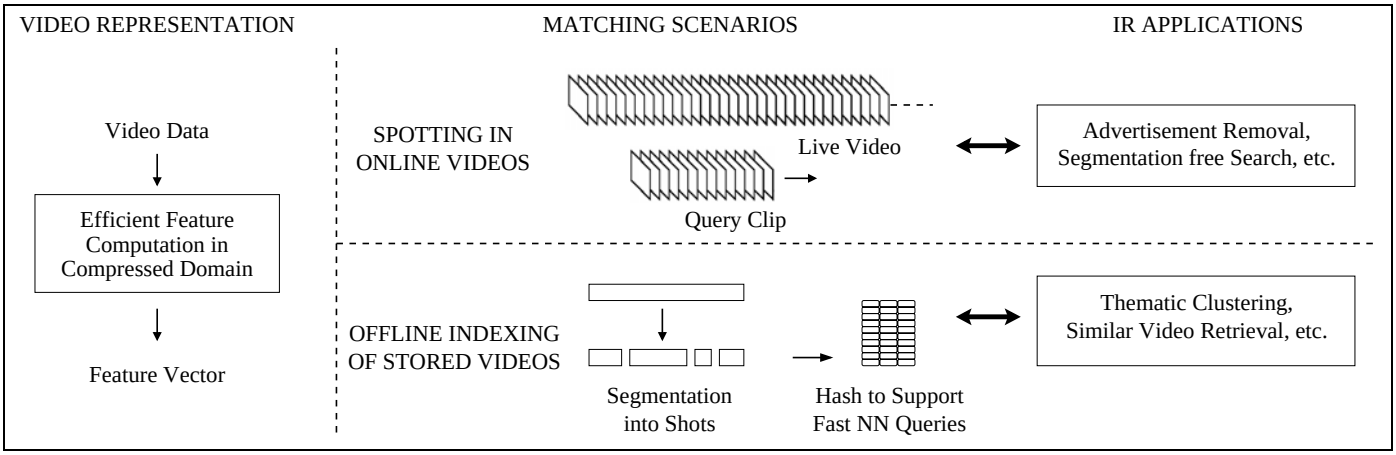


Fig. 1. An overview of the techniques presented in this paper.

as the fundamental unit and our features represent the entire clip.

In the context of applications for present times, the amount of video data does not remain constant anymore. The data keeps growing every day and that too at tremendous rates. Till recently, video matching papers had been concentrating on issues pertaining to better representation of video properties and higher matching accuracy. Emergence of large scale video repositories introduced a new range of issues concerning faster offline processing and efficient search. Accuracy can be compromised to certain extent for gains in speed. For applications that require very fine level accuracy, slower sophisticated algorithms can be applied on much smaller datasets obtained after first level of coarse pruning. We propose a significantly fast approximation of a popular color feature in the compressed domain. Our implementation is capable of indexing 140 hours of video data in a day. The LSH based indexing scheme can easily search through more than 100 hours of data in few seconds.

II. Efficient Indexing in Compressed Domain

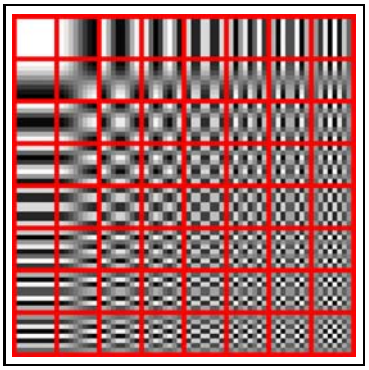
The first step in our processing is to segment the broadcast news videos into *shots*. A shot is defined as a visually continuous sequence of successive video frames taken from one camera. The segmentation of videos is performed by automatically detecting the shot boundaries. The algorithm uses block based feature difference with an adaptive threshold, dynamically calculated over successive frames. Each shot is then processed and features are extracted for a compact representation of the shot. The feature extraction process needs to be computationally fast for handling large amounts of data. Moreover in situations when online videos need to be processed in real time, fast feature extraction mechanisms are desired.

We work in the compressed domain for extracting features. Video data is almost always found in compressed formats for efficiency of storage and transmission. Feature extraction is much more efficient in the compressed domain than that in the uncompressed domain as the overhead of decoding is removed. Memory requirements are much less while processing. Some low level features can be determined from the information directly available in the compressed domain itself which removes the overhead to compute it again in the pixel domain. JPEG and MPEG have been the most explored formats for compressed domain processing. Figure 2 illustrates the nature of DCT blocks in a compressed image or video frame. DCT coefficients and block motion vectors are used for computation of various low level features. An overview of the popular features [7] is also provided in Figure 2.

A. Compressed Domain Color Atmosphere

Color (or intensity) information is among the maximum used in the content based image and video matching literature. It is usually measured by some sort of refined color histogram technique. One of the most widely used technique is the color coherence vector (CCV) [8]. It makes use of spatial coherence and is thus much more discriminative. Instead of counting only the number of pixels of a certain color, the CCV additionally distinguishes between coherent and incoherent pixels within each color class. This essentially aims at differentiating pixels belonging to the color classes associated with smooth and homogeneous regions from those associated with noisy or edgy regions. Though it provides significant advantage over basic color histogram in terms of performance, the computational complexity is a deterrent.

Typically color information is measured in the compressed domain using DC image sequence and represented with features like DC color histograms [7]. We propose an approximation of CCV by using information from MPEG compressed domain. This representation retains the benefits of CCV over a basic color histogram and is still a lot faster as it is directly computed from information available in the compressed domain. AC coefficients in the 8×8 3B DCT blocks are classified into frequency bands that roughly correspond to smooth areas, horizontal and vertical edges, and noisy areas [7]. DCT blocks corresponding to smooth regions have low frequency and have only very few initial AC coefficients as non-zero, ideally only the DC coefficient. Whereas, blocks corresponding to noisy regions have later AC coefficients as non-zero. This information is used to classify the DC value of



(i) A DCT Block

Video Property	Features	Information Used
Color Atmosphere	DC color histogram, DC YCbCr vector	DC coefficients from all 3 color channels
Texture & Edge	AC coefficient energy, DCT Block edge map	AC coefficients and relationships among them
Motion statistics	Motion Activity, Motion Histograms	Block Motion Vectors

(ii) Popular Compressed Domain Features [7]

Fig. 2. (i) shows an 8×8 DCT block. Each cell indicates the nature of the region that is represented by the particular DCT coefficient. (ii) provides a review of common compressed domain features.

Method	FPS	Amount of data that can be indexed each day
Pixel Domain CCV	4	6 hours
SIFT based frame representation [9], [10]	2	3 hours
Our Compressed Domain approximation of CCV	150	140 hours

TABLE I
COMPARISON OF OUR COMPRESSED DOMAIN FEATURE EXTRACTION WITH OTHER TECHNIQUES. ALL THE ABOVE NUMBERS ARE REPORTED ON VIDEOS OF RESOLUTION 320×240.

the corresponding block as coherent or incoherent. Two values are associated with each color bin j:

- α_j - The number of DC values corresponding to smooth DCT blocks of color class j
- β_j - The number of DC values corresponding to noisy or edgy DCT blocks of color class j

The feature vector $\mathcal{V}$ is thus defined as the vector

$$\mathcal{V} \;=\; \langle(\alpha_1, \beta 1), \ldots, (\alpha_n, \beta n)\rangle \tag{1}$$

normalized by the number of DCT blocks. n is the number of color classes. The popular distance measure to compare CCVs can directly be adapted for our feature:

$$Dist(\mathcal{V}^1, \mathcal{V}^2) \;=\; \sum_{j=1}^{n} \left(\frac{\left|\alpha_j^1 - \alpha_j^2\right|}{\alpha_j^1 + \alpha_j^2 + 1} + \frac{\left|\beta_j^1 - \beta_j^2\right|}{\beta_j^1 + \beta_j^2 + 1} \right) \tag{2}$$

The similarity measure between the two vectors is simply the inverse of the above equation,

$$Sim(\mathcal{V}^1, \mathcal{V}^2) \;=\; \frac{1}{Dist(\mathcal{V}^1, \mathcal{V}^2)} \tag{3}$$

Table I compares efficiency of our feature computation method with other well known standard color representations. We also compare with features used in latest video indexing schemes [9], [10]. It can be clearly seen from the numbers that our method of computing the approximate CCV is much superior to the standard pixel domain technique. In fact, it even turns out to be marginally faster than the computation of a basic color histogram in the pixel domain while still retaining the advantages of CCV to a large extent. Our version of the CCV is an approximation of the actual CCV and thus cannot be directly compared. We however conducted experiments which show that our approximation of the CCV is qualitatively good. Some of these are reported in Section III.

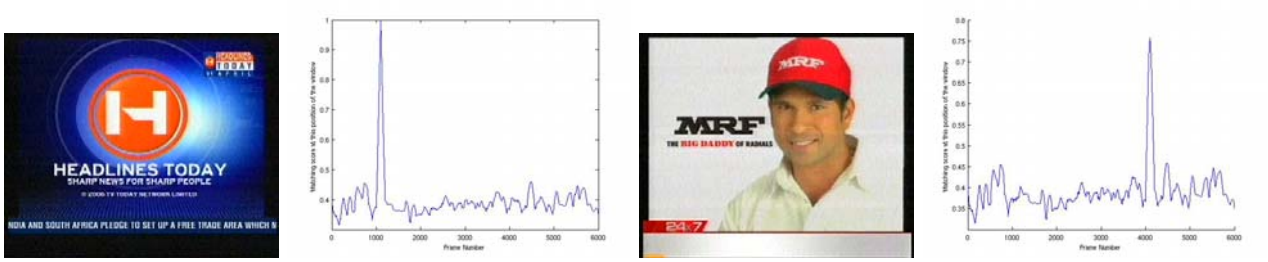
B. Spotting Clips in Online Videos

For applications like advertisement detection and removal from online video feeds, we use a segmentation free searching scheme which employ running window based comparison, on the lines of the one proposed in [11]. We make it further efficient by matching in the compressed domain. The feature vectors are computed incrementally to avoid redundant computation.

Let us denote the online video sequence by L and the small query video by Q, and the number of frames in them being $F(L)$ and $F(Q)$ respectively. Q is compared with successive subsequences of L, where each subsequence consists of frames i to $i + w$, where w is the width of the running window and i varies from 1 to $(F(L) - w)$. Any such subsequences is hereafter denoted as L_i. The width of the window, w is typically kept comparable to the number of frames in the query video, f_Q. Both the query clip as well as the subsequence are represented using vector described in equation 1. A single global vector is built by cumulating information from all the frames in the clip. The similarity score is computed at every shift (increment of i) and a peak in these sequences of scores indicate the position of a possible match. Few frames can be skipped at every increment for achieving further speed up without any noticeable degradation in the accuracy. In our implementation, we shift the window to the next I frame in the MPEG video.

$$\forall i \quad Sim(\mathcal{V}^Q, \mathcal{V}^{L_i})$$

For improving the efficiency, we update the feature vector $\mathcal{V}^{L_i}$ incrementally at each shift of i. We maintain a vector for each frame in L_i. These are denoted by $\{v_1, \ldots, v_w\}$ corresponding to the frames $\{f_i, \ldots, f_{i+w-1}\}$ belonging to the sequence L_i. These vectors are initialized from frames belonging to L_1. $\mathcal{V}^{L_i}$ is the cumulative vector for representing



(a) (b) (c) (d)

Fig. 3. Example of spotting advertisements in live online video. The similarity scores over a running window while matching in the compressed domain are plotted. A peak is observed when there is a match.

L_i which is essentially a summation of vectors $\{v_1, \ldots, v_w\}$,

$$\mathcal{V}^{L_i} = \sum_{k=1}^{w} v_k$$

After each increment of i, the new $\mathcal{V}^{L_i}$ is computed from $\mathcal{V}^{L_{i-1}}$ by subtracting the contribution of the outgoing frame and adding the contribution due to the newly added frame into the sequence.

$$\begin{aligned} t_{out} &= v_{i\%w} \\ v_{i\%w} &= v(f_{i+w}) \\ \mathcal{V}^{L_i} &= \mathcal{V}^{L_{i-1}} - t_{out} + v_{i\%w} \end{aligned}$$

where $v(f_{i+w})$ is the vector representation for the frame f_{i+w} and t_{out} is a temporary vector.

Figure 3 shows the results for the spotting scheme. The similarity score increases forming a distinct peak at the position of occurrence of the query video. Such peaks are detected using standard adaptive threshold mechanisms.

C. LSH based indexing for Efficient Near Neighbor Queries and Fast Clustering

The need for an efficient indexing mechanism for searching through video data is quite obvious. The naive approach of sequential search is typically too inefficient to handle large databases. The spatial data structures which have been extensively used are helpful in the context, however suffer from the so-called "curse of dimensionality" in handling high dimensional data points. As the number of dimensions grow sufficiently large, most algorithms and techniques perform not significantly better than a simple brute force linear search.

Locality Sensitive Hashing (LSH) is known to handle high dimensional data with greater efficiency. It is a relatively new scheme for approximate similarity search based on hashing [12]. Its developed for efficiently answering approximate near neighbor queries. The key idea is to hash the data points using several hash functions so as to ensure that, for each function, the probability of collision is much higher for objects which are close to each other than for those which are far apart. Then, one can determine near neighbors by hashing the query point and retrieving elements stored in buckets containing that point. The potential applications for earlier versions of LSH were narrow in scope as LSH functions were suggested for data with solely binary features. The most recent work [12] is extended to use data in Euclidean space.

A video shot is represented by a feature vector. Feature vectors corresponding to all the shots in the video collection are hashed using LSH. As LSH is designed for euclidean distance computation, we separate the coherent and incoherent parts of the feature vector and concatenate the two to form a single feature vector $\mathcal{V} = \langle \alpha_1, \ldots, \alpha_n, \beta 1, \ldots, \beta n \rangle$ to make it suitable for euclidean distance computation. A random set of vectors α are generated from a p-stable distribution of the same dimension as the feature vectors. These act as hash functions and map the d dimensional feature vector onto a real line. The real line is divided into equi-width segments of appropriate size r and vectors are assigned hash values based on which segment they project onto. Formally, the hash function is given by

$$h_{\mathbf{a},b}(v) = \left\lfloor \frac{\mathbf{a}.\mathbf{v} + b}{r} \right\rfloor \quad (4)$$

where b is a real number chosen randomly from the range $[0, r]$. If appropriate r is chosen, then intuitively two points which are nearby in the space will hash in the same bin. To avoid boundary effects, several hash functions are used. The time complexity for querying every hash table is constant and this returns a set of candidate points. Further pruning can be performed on this set by explicitly computing the distance.

Clustering is made a lot faster by employing the near neighbor queries from LSH based index. In typical clustering, $O(n^2)$ comparisons are required to arrive at clusters as each point needs to be compared with all the other points. We directly query for near neighbors of a query point from the hash which provides significant computational gains for clustering.

III. Experimental Results on a Large Scale Video Indexing System

A. The System used for Experiments

We have built a system [13] that is specifically designed for the information retrieval applications on broadcast news videos. The system is directly useful to an end user for easy access to the news stories of interest. It also acts as a platform for convenient deployment and experimentation of various video analysis and indexing techniques on real data, and on a large scale. The system is built upon a layered architecture with certain software design choices that makes

(a) Query (b) Result 1 (c) Result 2 (d) Result 3 (e) Result 4

(f) Query (g) Result 1 (h) Result 2 (i) Result 3 (j) Result 4

Fig. 4. Top retrieved results from near neighbor queries performed on the LSH index are shown.

the system highly scalable, extensible and modular. Lack of a specialized platform for the deployment and experimentation often becomes a roadblock in the development of robust and scalable algorithms. Our system works as an end to end framework for the researchers wherein they can plug in their specific algorithms and get to test them with real data and other supporting mechanisms.

The system has been in use for 20 months and and has consistently been evolving to come to its current form. Almost 50 hours of broadcast news video data from 8 different channels in 3 different languages (English, Hindi and Telugu) gets recorded and processed each day by the system. Almost 70Tb of videos have been processed in all by the system till date. There are 1000 valid users which have password protected access to the processed content over our university's LAN. The video is encoded in MPEG-1 format which is kept as the standard format throughout the system. Two processing servers with an AMD Athlon 3200+ processor and 1G of RAM each are in use. 1500GB of total storage spread over 5 storage nodes is in use.

The captured broadcasts are automatically segmented into shots and stories by analyzing the structure. All the video data gets processed and features gets indexed along with the source specific metadata. Broadly, following sets of operations are supported on the input videos:

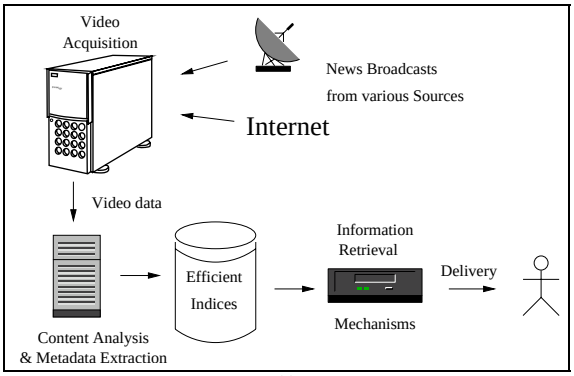
1) *Video Processing:* The standard pre-processing and video processing operations such as noise removal, shot detection and advertisement removal.
2) *Content Extraction:* Various feature extraction and content analysis algorithms are supported. These include key frames, color histograms, motion vectors and face videos.
3) *Indexing:* Efficient indexing schemes are supported to build index structures of all the metadata extracted. There can be multiple indices for different metadata information. These indices are used by the IR and Delivery modules for providing quick content based access to the users looking for some specific information.

The system supports various types of metadata for the videos:

1) Source information about the videos like channel, broadcast date and time, resolution, frame rate and language.
2) Low level features like color, motion, compressed domain features and audio stream information.
3) Object level annotation information like locations, faces, activities and events.
4) Direct and indirect user feedback for the videos.

Various IR modules deliver the processed content to the users over web. Users can see the latest news stories on the top and can browse through the archives too. A user friendly interface enables easy browsing for the users. An effective and intuitive way of visualizing the videos is designed such that the user gets a feel of the content without actually needing to stream the videos and see them. Users are typically presented news broadcasts automatically segmented into individual stories. These stories are of few minutes duration each. Browsing this by playing the video requires significant time. We extract the critical frames called key frames and form a visual summary which is presented in an innovative manner along with the metadata as show in Figure 6. It allows users to know the relevant meta information (eg. language, channel, time of news, importance, etc.) associated with each story. The key frames from the story are arranged left to right and move in a slide show. Any keyframe zooms out on mouse hover by the user. This enables the user to study the content in detail.

Figure 4 show results of qualitative experiments performed on the videos indexed by the system. Two examples of near neighbor queries executed on the data indexed by LSH are shown. Clips are matched even after differences in the appearance due to the channel logos, text captions, minor pose and illumination changes, etc. While simple features like color will not do matching at a very fine level (differentiation between cricket videos from two different matches), it can still be useful in retrieving related clips from all the news videos of a day or week which is enough for the kind of applications we are targeting. Near neighbor queries executed on an LSH index of around 100 hours of video data typically takes a few



(i)

No. of different channels being captured	8
Number of news broadcasts recorded and processed each day	100
Approximate amount of Video Data processed till date	70 Tb
Amount of video data present in the system at any moment	1.5Tb

(ii)

Fig. 5. An overview of the experimental system. Figure on the left depicts the coarse structure. Table on the right provides some quick statistics about the system.

Fig. 6. Screen Shot of the UI element designed for presenting a news story

seconds.

This shows that the proposed scheme is suitable both in terms of efficiency and matching performance for building information retrieval applications discussed in Section I.

IV. Conclusions and Future Work

We have presented an efficient method for approximate computation of a popular color feature in compressed domain. The superiority of the method in terms of speed over state of the art methods is clearly depicted through experiments. Fast matching techniques for two probable scenarios in information retrieval applications are proposed. The techniques are shown to be achieving significant gains in computational time over existing methods.

The immediate extension of the work is applying these techniques for clustering visually similar video clips. Analysis mechanisms will be established for these clusters for addressing various applications described in Section I. Later, hyper links can be established between related news stories. We presently detect all the faces appearing in the news videos. These can be clustered to establish a list of most cited personalities in news over a duration. We capture the direct and indirect user feedback as well as the user activity log which can be used for automatic personalization and recommendation based on user's history. The visualization of videos can extend in future to the use of mosaics and 3D walkthroughs.

References

[1] Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, David Steele, and Peter Yanker, "Query by image and video content: The QBIC system," *Computer*, vol. 28, no. 9, pp. 23–32, September 1995.

[2] John R. Smith and Shih-Fu Chang, "Visualseek: A fully automated content-based image query system," in *ACM Multimedia*, 1996, pp. 87–98.

[3] Rainer Lienhart, Wolfgang Effelsberg, and Ramesh Jain, "VisualGREP: A systematic method to compare and retrieve video sequences.," *Multimedia Tools and Applications*, vol. 10, no. 1, pp. 47–72, 1999.

[4] Arun Hampapur and R. Bolle, "Comparison of sequence matching techniques for video copy detection," in *Conference on Storage and Retrieval for Media Databases*, 2002, pp. 194–201.

[5] Nicholas Diakopoulos and Stephan Volmer, "Temporally tolerant video matching," in *Proc. of the ACM SIGIR Workshop on Multimedia Information Retrieval*, Toronto, Canada, August 2003.

[6] Yap-Peng Tan, Sanjeev R. Kulkarni, and Peter J. Ramadge, "A framework for measuring video similarity and its application to video query by example.," in *ICIP (2)*, 1999, pp. 106–110.

[7] H. Wang, A. Divakaran, A. Vetro, S.F. Chang, and H. Sun, "Survey of compressed-domain features used in audio-visual indexing and analysis," *Journal of Visual Communication and Image Representation*, vol. 14, no. 2, pp. 150–183, June 2003.

[8] Greg Pass, Ramin Zabih, and Justin Miller, "Comparing images using color coherence vectors," in *MULTIMEDIA '96: Proceedings of the fourth ACM international conference on Multimedia*, New York, NY, USA, 1996, pp. 65–73, ACM Press.

[9] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proceedings of the International Conference on Computer Vision*, Oct. 2003, vol. 2, pp. 1470–1477.

[10] O. Chum, J. Philbin, M. Isard, and A. Zisserman, "Scalable near identical image and shot detection," in *Proceedings of the International Conference on Image and Video Retrieval*, 2007.

[11] Kunio Kashino, Takayuki Kurozumi, and Hiroshi Murase, "A quick search method for audio and video signals based on histogram pruning," *IEEE Transactions on Multimedia*, vol. 5, no. 3, pp. 348–357, September 2003.

[12] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, New York, NY, USA, 2004, pp. 253–262, ACM Press.

[13] Tarun Jain, Sai Ram Kunala, Ravi Kishore Kandala, and C.V. Jawahar, "A system for information retrieval applications on broadcast news videos," in *Proceedings of the International Symposium on Data, Information and Knowledge Spectrum*, 2007.