LAMP-TR-003 November 1996
CFAR-TR-847
CS-TR-3732

Feature Normalization for Video Indexing and Retrieval

Vikrant Kobla, David Doermann, King-Ip (David) Lin, Christos Faloutsos

Language and Media Processing Labratory
Instititue for Advanced Computer Studies
College Park, MD 20742

Abstract

Fast and efficient storage, browsing, indexing, and retrieval of video is necessary for the development of various multimedia database applications. Given that video is typically stored efficiently in a compressed format, if we can analyze the compressed representation directly, we can avoid the costly overhead of decompressing and operating at the pixel level. Compressed domain parsing of video has been presented in earlier work where key frames are identified for shots, subshots, and scenes. In this paper, we describe key frame selection, feature extraction, indexing, and retrieval techniques that are directly applicable to MPEG-compressed video. We develop a frame-type independent representation of the various types of frames present in an MPEG video in which all frames can be considered equivalent. Features are derived from the available DCT, macroblock, and motion vector information and mapped to a low-dimensional space where they can be accessed using standard database techniques. The spatial information is used as primary index while the temporal information is used to enhance the robustness of the system during the retrieval process. The techniques presented enable fast archiving, indexing, and retrieval of video. Our operational prototype typically takes a fraction of a second to retrieve similar video scenes from our database, with over 95% success.

***The support of the LAMP Technical Report Series and the partial support of this research by the National Science Foundation under grant EIA0130422 and the Department of Defense under contract MDA9049-C6-1250 is gratefully acknowledged.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
NOV 1996		**00-11-1996 to 00-11-1996**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Feature Normalization for Video Indexing and Retrieval	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Language and Media Processing Laboratory,Institute for Advanced Computer Studies,University of Maryland,College Park,MD,20742-3275**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**39**	

1 Introduction

With the development of various multimedia compression standards and significant increases in desktop computer performance and storage, the widespread exchange of multimedia information is becoming a reality. Video is arguably the most popular means of communication and entertainment. With this popularity comes an increase in the volume of video and an increased need for the ability to automatically sift through and search for relevant material stored in large video databases (LVDBs). Even with increases in hardware capabilities which make video distribution possible, factors such as algorithm speed and storage costs are concerns that must still be addressed.

With this in mind, a first consideration should therefore be to attempt to increase speed when using existing compression standards. Performing analysis in the compressed domain reduces the amount of effort involved in decompression, and providing a means of abstracting the data keeps the storage costs of the resulting feature set low. Both of these problems are active areas of research.

A second consideration is that a user who is interested in searching for and retrieving video clips needs a way to interface with the database by formulating appropriate queries. These queries need to be appropriately translated into a form that can be used to search an index and retrieve the matching clips. A typical approach to indexing and archiving video for retrieval requires parsing the video, extracting key information from each clip (possibly a single frame), indexing the information, and providing a representation which allows accurate and efficient retrieval based on the user's request.

Traditional query-by-content algorithms operate on the principle that a query can be formulated which accurately describes features that can be extracted automatically by the system, such as color, texture and shape. In the case of video this approach must be augmented to deal with the additional temporal and spatio-temporal dimensions.

In our system, we address these issues by providing algorithms which perform these tasks on compressed video. In particular, we consider the problem of extracting indexable features from compressed MPEG frames, indexing the features for each clip and providing efficient query capabilities. We present techniques which provide a framework within which all types of MPEG frames can be considered equivalent.

In the next section, we provide some background on compressed domain video analysis, including a brief description of MPEG compression, and an overview of related work.

2 Background

The analysis of compressed video can proceed in one of two fundamental ways. The first is by decompressing some or all of the video and using the individual frames to gather information about various characteristics of the video such as content or motion, and extracting indexable features in the pixel domain. The second involves exploiting encoded information contained in the compressed representation without incurring the overhead of complete decompression.

The problem of video retrieval arises when a user or an application poses queries to a large database of video clips in some format, and a fast, efficient, and precise reply is required. If the query is an image or another video and the user requests that the system retrieve similar clips, it is called a "query-by-example". In this case, the main challenge in comparing clips or frames of a clip is providing a suitable definition of what it means for two clips to be similar. In the pixel domain, color-based similarity can be implemented using, for example, features extracted from color histograms, or a pixel by pixel comparison, though the latter is computationally expensive. Other methods of specifying queries may involve the user sketching the shapes that he/she is interested in, or providing textual queries to access annotations that were derived automatically or entered manually. Other features that can be used to define similarity which have proven useful in related domains include image

texture, object shape, and spatial relations between objects.

With an increasing amount of video available, and given the query challenges stated above, automated techniques for searching large video databases in a fast, efficient manner are necessary. Since decompressing video is very time consuming, we explore techniques for analyzing video using the information available in an MPEG-compressed video stream.

2.1 MPEG stream

The Moving Picture Expert Group (MPEG) standard for digital video is arguably the most widely accepted international video compression standard. The MPEG encoding algorithm [12] relies on two basic techniques: block-based motion compensation to capture temporal redundancy, and transform-domain-based compression to capture spatial redundancy. Motion compensation techniques are applied using both predictive and interpolative techniques. The prediction error signal is further compressed using spatial redundancy reduction techniques. The fact that temporal and spatial changes are fundamental for segmentation makes MPEG an ideal candidate for compressed domain analysis.

An MPEG stream [11] consists of three types of frames — I,P, and B frames — occurring in a repetitive pattern (called the IPB pattern). An I frame is an anchor frame that is simply a JPEG encoding [15] of its corresponding pixel image. A P frame is predicted from its preceding I or P frame, and a B frame is predicted from both its preceding and following I and/or P frames. I and P frames are collectively called reference frames since only these two types of frames are used during prediction and interpolation of other frames. Figure 1 shows an example of the predictive relationships. For most clips, such as the example shown in Figure 1, the IPB pattern is regular with the number of B frames between reference frames and the number of P frames between I frames being constant. Clips with irregular IPB patterns do not pose a problem to our system; in later sections we describe techniques to han-

dle such cases. All three types of frames are ultimately encoded using 2D Discrete Cosine Transform (DCT), quantization, and run-length coding. In the I frames, the DCT information is derived directly from the image samples, but in the P and B frames, the DCT information is derived from the residual error after prediction. The motion compensation information represented as vectors is also differentially coded. We use the term 'motion vector' to refer to the block-based motion compensation vector. Each frame in the clip is divided into blocks of 16×16 pixels called Macroblocks (MBs), and most of the low-level processing, including spatial and temporal redundancy reduction, is performed at this level of spatial resolution.

During the encoding process, a procedure is run on each macroblock in a P frame, to see if that macroblock can be predicted from its corresponding block in the previous I/P frame with a possible offset to compensate for motion. If it appears that little would be gained in predicting and encoding, then that block is not predicted but is intra-coded. This typically occurs if the current macroblock does not have much in common with the previous one. Every P frame therefore consists of intra-coded MBs and forward-predicted MBs. There are also skipped MBs which resemble forward-predicted MBs, since they are identical to some 16×16 block of the previous reference frame.

Similarly, for each macroblock in a B frame, a test is made to see if it can be predicted from both its previous and next I/P frames, its previous I/P frame alone, its next I/P frame alone, or if it cannot be predicted from any reference frame and thus must be intra-coded. An MB can also be skipped if it is identical to some part of the reference frames and the residual error becomes zero. Each skipped MB behaves identically to the previous non-skipped MB in the current slice of the current frame.[1] In other words, the first MB of a slice cannot be skipped, and if the MB type and motion vectors of this non-skipped MB are sufficient for any continuous string of MBs in the slice immediately following the non-skipped MB, those MBs are skipped. Each

[1] MBs in a single frame are grouped into slices of MBs, as a means of layering.

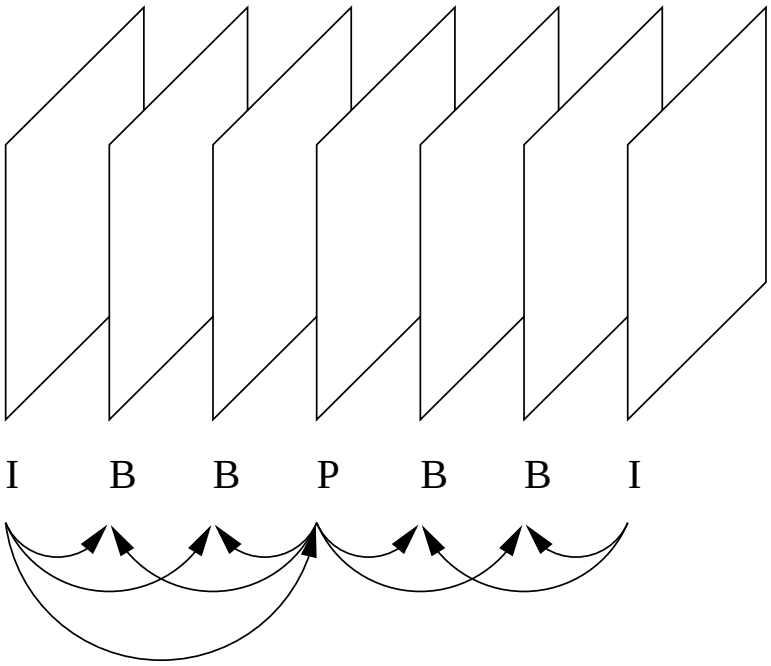


Figure 1: Predictive relationships between I, P and B frames

MB in a B frame can be of any one of these five types.

The MPEG standard does not specify which techniques are to be used for motion estimation and prediction during the encoding process. Thus, little can be assumed about the quality of the prediction obtained while encoding. Nevertheless, we assume that the prediction techniques are reasonable enough to yield reliable motion vector and macroblock data. The reader should refer to two articles by Le Gall [11, 12] for more information on MPEG.

The computationally expensive step in decompressing MPEG video is the inverse DCT (IDCT), which should be avoided if possible. The information available in the compressed representation without performing IDCT includes the type of each MB, the DCT coefficients of each MB, and the motion vector components for the forward, backward, and bidirectionally predicted MBs. The approach followed in this paper involves utilizing all three: the MB types, DCT coefficients, and motion vectors.

2.2 Related work

The video indexing and retrieval problem has been addressed by researchers in a number of ways. A survey of digital video parsing and indexing technologies, mainly in the pixel domain, is presented in the paper by Ahanger and Little [1], including a discussion of research trends in video indexing and requirements of future data delivery systems. Topics such as video data indexing, video data modeling, information extraction, and video scene segmentation are also presented.

In other work, Zhang et al. [17] describe techniques for use in the pixel domain for dealing with the representation of shot content, as well as content-based retrieval techniques using key frames and temporal properties of shots. They also present techniques for video parsing in the pixel domain followed by key frame extraction. The representation of shot content is based on several types of features, including color histogram and moment features, texture features, shape features, and edge features. Similar work can be found in [13] by Nagasaka and Tanaka who present pixel-domain techniques for performing full-video search for specified objects using features derived locally. Two recent papers by Flickner et al. [7] describe the QBIC system, which performs content-based retrieval based on color, shape, texture, and sketches in large image and video databases. Ardizzone et al. [2, 3] also deal with content-based video indexing based on motion, color, and texture and other global features. All the aforementioned papers present techniques in the pixel domain, but much less has been done in the compressed domain.

Idris and Panchanathan [8] propose an algorithm based on vector quantization (VQ) for indexing of video sequences in compressed form. During compression, the image is decomposed into vectors and mapped to a finite set of codewords and encoded using adaptive VQ. Each frame is represented by a set of labels and a codebook which are used to generate indices. A generic paper on compressed-domain video indexing techniques by Chang [5] describes some of the issues involved in addressing such a problem in the DCT, wavelet, and subband transform compressed domains.

2.3 Approach

Our approach to compressed domain indexing and retrieval can be split into three parts — segmentation, indexing, and query processing. First, video segmentation divides the incoming video into shots or scenes, and selects one or more key or representative frames for each shot. A shot in a video clip is defined as a maximal sequence of frames resulting from a continuous uninterrupted recording of video data. These shots may be further subdivided into scenes, if, for example, significant camera motion[2] is present. This step has been presented in our earlier work on video segmentation [9, 10] and is described briefly in Section 3, along with the key frame identification procedure.

Second, features are extracted from the key frames supplied by the segmentation process and used to create a database index (Section 4). The features used are derived from the DCT coefficients and the motion vector information available in the MPEG compressed video. Unfortunately, the dimensionality of each feature vector prohibits the implementation of standard database retrieval techniques, not to mention the tremendous overhead of storage per frame. Using a technique called *FastMap* [6], the dimensionality of the features can be reduced to a manageable level where they can be represented using standard database techniques (Section 5).

Finally, when need arises, the database is accessed using the features derived from a query clip as an index. When a query arrives, segmentation and key frame extraction is performed, if necessary, and features are extracted from the key frames of the query. These features are then used to index into the database to perform retrieval. By mapping the query to the same space as the stored key frames, standard similarity metrics such as Euclidean distance between a query frame and frames in the database can be used to retrieve the best matches. Experiments and results of query processing are discussed in Section 6.

[2]Throughout this paper, we assume that "camera motion" refers to operations like panning, tilting, and zooming a stationary camera and not to changes in the position of the camera.

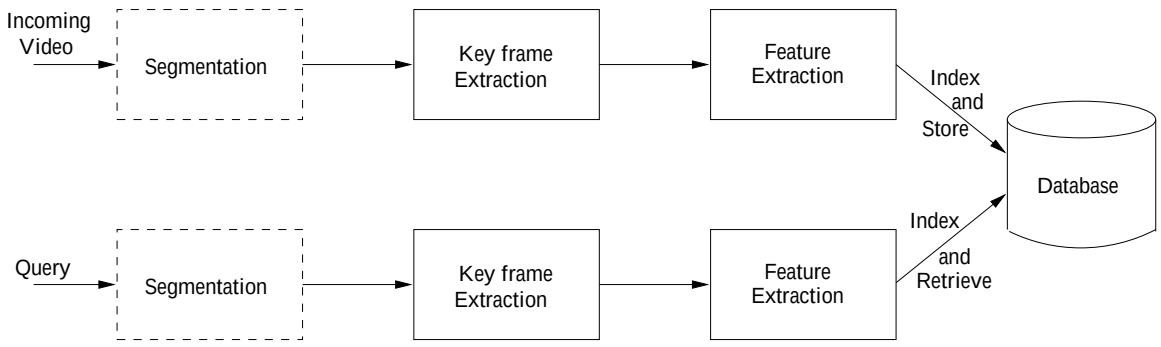


Figure 2: Flowchart of our approach

Figure 2 shows the flowchart of how the individual components of the system interact.

3 Video segmentation overview

Our approach to indexing involves extracting a set of key frames for the entire clip, such that as much of the content of the video is captured as possible, but at the same time, redundant frames are excluded. Two key frames of essentially the same content that are separated by other key frames are not considered redundant, as the physical and temporal structure of the video needs to be preserved.

The first step involves segmenting the video by identifying the frame(s) where a transition takes place from one shot to another. A change which occurs exactly between two frames is called a *cut* or a *break*, whereas transitions that occur gradually over several frames are called *fades*, *dissolves*, *wipes*, or *special effect edits*.

Our approach to segmentation analyzes the types of MBs that have been used to encode the P and B frames, and uses the counts of the different types of MBs to derive a metric that pinpoints where cuts or breaks occur. An analysis of the macroblock types alone does not always provide sufficient information to indicate that a shot change occurs between two frames. We have developed a DCT validation procedure that is used to confirm the existence of shot changes for which the macroblock information is found to be insufficient.

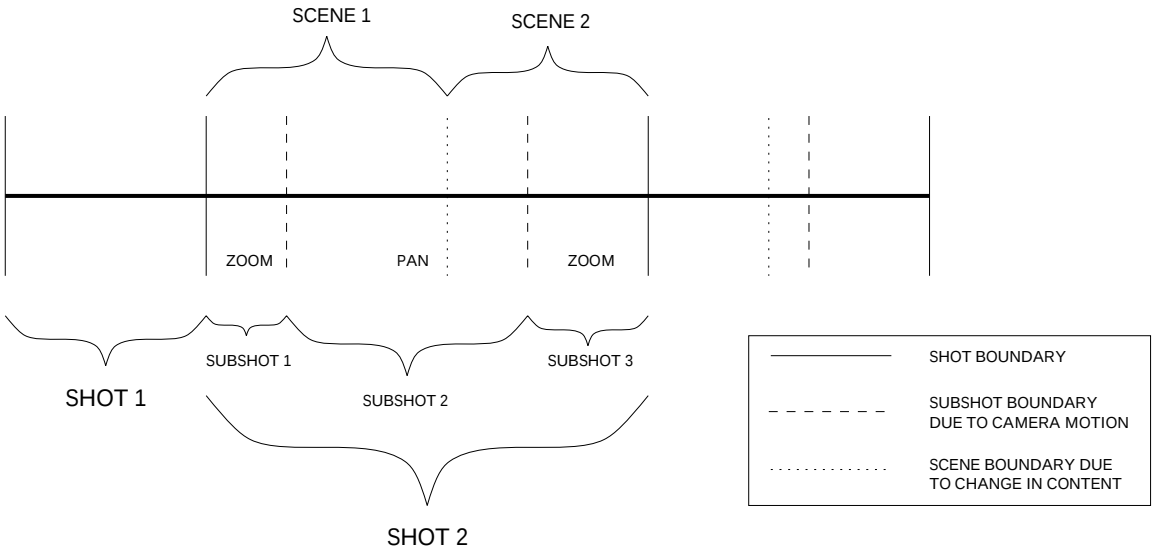


Figure 3: Diagram showing the shot subdivision concept.

If large camera motion is present in a single shot, then two frames that are spaced well apart in the shot may be quite dissimilar. Therefore, once video is segmented into shots, these shots are further segmented into "subshots" based on some attribute, that exists in common among the frames of the subshot. When shots are subdivided based on changes in content due to camera motion, these subdivisions are called "scenes" (Figure 3). Our approach to subdividing shots into scenes involves using the motion vectors encoded into the MPEG format to determine any type of camera motion that may be present, including zoom-in, zoom-out, pan left, pan right, tilt up, tilt down, or a combination of pan and tilt. We use the "flow" information (to be described in the next section) to derive a unique vector (called the dominant flow vector) that describes the relative displacement of the contents of the current frame with respect to the next frame. Starting from the first frame of the shot, by successively adding these dominant flow vectors, we can determine the displacement of each frame from the first frame. The magnitude of this total displacement vector gives an estimate of the magnitude of the perceptual translation caused by the motion of the camera. By comparing this magnitude with the dimensions of the frame, we determine whether the particular frame under consideration has undergone enough translation from the

first frame, and if so, the frame is tagged as the possible start of a new scene. If the camera motion involves a zoom operation, we tag the last frame of the zoom sequence as the possible start of a new scene. By comparing the DCT information of the tagged frame with the DCT information of the frame from which the total displacement vector is calculated, we can determine if the current frame is a true candidate for the start of a new scene.

The final step in the segmentation process involves identifying a key frame for each of the subshots or scenes. Different subshots that have been divided based on camera motion may or may not have similar content, or there may be different content in the same subshot; such subshots would not be the best candidates for key frame selection. Choosing key frames of scenes allows us to capture most of the content variations, due at least to camera motion, while at the same time excluding other key frames which may be redundant.

The reader can refer to our previous work [9, 10] for more information on segmentation of video.

3.1 Key Frame Identification

It is important that the choice of key frames be made carefully, since a key frame will represent an entire shot in all future applications. The key frame candidates should possess all the requisite information to enable features to be extracted, and also enable the features to capture as much of the content and other attributes of the scene as possible.

The ideal method of selecting key frames would be to compare each frame to every other frame in the scene and select the frame with the least difference from other frames in terms of a given similarity measure. Obviously, this requires extensive computation and is not practical for most applications. On the other hand, choosing the first frame seems to be the natural choice, as all the rest of the frames in the scene can be considered to be logical and continuous extensions of the first frame, but it

may not be the best match for all the frames in the scene. A third possible choice is the middle frame of the scene, as it might be expected to have the most similarity with all the other frames of the scene, although this is not guaranteed. In a more general framework, we would like to choose frames with the greatest content or index potential — for example, frames with text, or frames with a clear unoccluded object. Other factors that influence the choice include encoding patterns. For example, the frequency of I frames may affect the choice, as I frames represent the best candidates for key frame selection since they have the actual DCT coefficients which form the spatial component of the data. If I frames occur fairly frequently, then the first I frame can be chosen as the key frame. It is, however, possible to have an entire scene with no I frame, in which case, alternate measures are required.

We have found it is, in general, sufficient to select the first frame of each scene as a key frame. This is based in the observation that cinematographers attempt to "characterize" a shot with the first few frames, before beginning to track or zoom to a close-up. The practical reason for this choice will become clear in the next section, as we develop techniques to circumvent the problems due to encoding, and generate a framework where all frames can be considered equivalent.

In the final representation, the video is partitioned into a set of scenes which exhibit consistency in content, and each scene is represented by a key frame.

4 Feature Extraction

A difficulty with identifying key frames in video that has been compressed by a method like MPEG is that frames can be of different types, i.e., I, P, or B frames and can occur in a variety of patterns. An I frame contains DCT coefficients of actual pixel data, but has no motion vectors, whereas a P or B frame contains DCT coefficients generated from residual error data after prediction or interpolation from other reference frame(s), but has motion vectors relating the frame to its reference frame(s). Different MPEG clips may also have different patterns of I, P, and B

frame orderings. A problem then arises when we try to identify key frames and subsequent index information. Should we identify only certain types of frames, for example, I frames, as key frames, or do we desire the flexibility to choose any frame independently of the frame type? To avoid these problems, we desire a frame-type-independent representation, in which all the features that we extract for the indexing and retrieval phase are obtained independently of factors such as frame type.

In our indexing and retrieval phase, we use the DC[3] coefficients and the motion vectors of the key frames. The features that must be extracted from each key frame include the DC coefficients, which form the spatial component, and the motion vectors of the MBs, which form the temporal component. The next two subsections explain the techniques used to extract these features from each type of frame.

4.1 DCT estimation

DCT coefficients are readily accessible for I frames, but since P and B frames are represented by the residual error after prediction or interpolation, their DCT coefficients need to be estimated. To calculate the DCT coefficients of an MB in a P frame or B frame, the DCT coefficients of the 16×16 area of the reference frame that the current MB was predicted from need to be calculated. Let us call this area the reference MB (though it is not an actual MB). Since the DCT is a linear transform, the DCT coefficients of the reference MB in the reference frame can be calculated from the DCT coefficients of the four MBs that can overlap this reference MB, albeit with substantial computational expense. It is easy, however, to calculate reasonable approximations to the DC coefficients of an MB of a P or B frame. Techniques for doing this were suggested by Yeo and Liu [16] and also by Shen and Delp [14].

Figure 4 shows an MB in a P frame, MB_{Cur}, being predicted from a 16×16 area denoted by MB_{Ref}. While encoding the P frame, only the residual error of MB_{Cur}

[3]Of the 64 DCT coefficients, the coefficient with zero frequency in both dimensions is called the 'DC coefficient', while the remaining 63 are called the 'AC coefficients'.

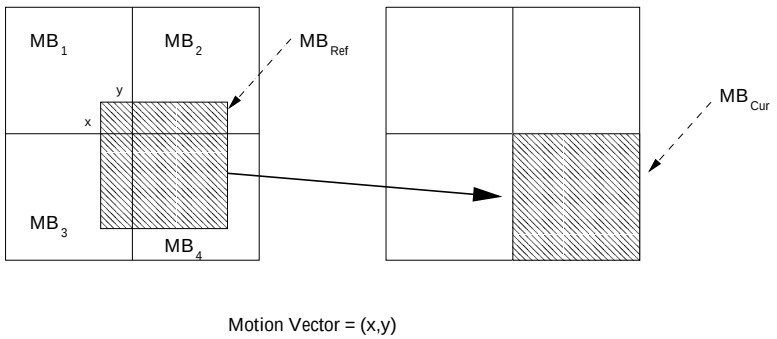


Figure 4: DC estimation: MB_{Cur} is predicted from MB_{Ref}. The motion vector is (x, y).

with respect to MB_{Ref} is stored. The DC coefficients of MB_{Ref} can be calculated from the DCT coefficients of four MBs — MB_1, MB_2, MB_3, and MB_4 (see [16] for the details of the calculation). To avoid expensive computation, the DC coefficient alone is approximated by a weighted sum of the DC coefficients of the four MBs, with the weights being the fractions of the areas of these MBs that overlap the reference MB, i.e.,

$$DC(\mathrm{MB}_{Ref}) = \sum_{i=1}^{4} w_i \times DC(\mathrm{MB}_i)$$

where w_i is given by the ratio of the area of the shaded region of MB_i to its total area.

Similarly, if an MB in a B frame is interpolated from two reference MBs, its DC coefficient is approximated by an average of the estimated DC coefficients of each of these two MBs.

4.2 Flow estimation

An MB can have zero, one, or two motion vectors depending on its frame type and whether the MB is intra-coded, forward- or backward-predicted, or bidirectionally-predicted, respectively. Moreover, the motion vectors of a given frame can be forward-predicted or backward-predicted with respect to a reference frame which may or may not occur adjacent to it. A problem occurs if, for example, we wish to compare an I

frame with no motion vectors to a B frame with primarily bidirectionally-predicted MBs, or even two B frames, one of which is primarily forward-predicted and the other primarily backward-predicted. We therefore require a more uniform set of motion vectors, independent of the frame type and the direction of prediction.

Our approach involves representing each motion vector as a backward-predicted vector with respect to the next frame, independent of frame type. The set of motion (or "flow") vectors for each frame then represents the direction of motion of each MB with respect to the next frame.

It should be noted that not all MBs will have this flow vector associated with them; but the number of such MBs is rarely large enough to affect our analysis. Across shot cuts or breaks, most of the MBs are not expected to have flow information.

The first step in deriving the flow is to analyze the frame-type pattern (i.e., the pattern of I, P, and B frames) in the MPEG stream. If the video is in XING format, i.e., it contains only I frames, then there exists no motion information and this analysis is not relevant.

For clips containing only I and P frames: If there are only P frames between I frames, and there are no B frames, then flow can be derived for each of the frames between two consecutive I frames, including the I frames themselves, except for the last P frame, for which we have no information about its relationship with the I frame that follows it.

The flow for an I frame that is followed by a P frame is the set of forward-predicted motion vectors of the P frame after inversion. Intuitively, if an MB in the P frame is displaced by a motion vector (x, y) with respect to an MB in the I frame, then it is logical to conjecture that the latter MB is displaced by a motion vector $(-x, -y)$ with respect to the MB in the P frame. The same reasoning is applied to the flow estimation of the MBs of a P frame that is followed by another P frame. The MPEG stream, however, does not contain any information relating a P frame to the I frame that follows it, unless B frames are present between them.

For clips containing B frames: Most MPEG streams contain B frames between consecutive reference frames. Let us consider two consecutive reference frames, R_i and R_j. Define *domain* $\mathcal{D}$ to be the set containing all B frames between the two consecutive reference frames R_i and R_j, and the frame R_j itself:

$$R_i \underbrace{B_1 B_2 \ldots B_n R_j}_{domain\ \mathcal{D}}$$

Let the B frames in the domain be denoted by $B_1, \ldots, B_n$, where n is the number of B frames between these two reference frames (typically, $n = 2$). The first step is to derive the flow between the first reference frame R_i and its next frame B_1 using the forward-predicted motion vectors of B_1. This case is similar to the I-P case discussed above. The inverses of the forward-predicted motion vectors form the flow vectors for the MBs of R_i. Similarly, using the backward motion vectors of frame B_n with respect to R_j, the flow is derived for frame B_n. There is no need to invert the motion vectors here, since the flow vectors essentially are backward-predicted vectors. Flow for R_j will be derived when R_j is analyzed with the reference frame following R_j.

We have not yet considered the case where an MB in B_1 does not have a forward-predicted motion vector with respect to R_i, or the case where an MB in B_n does not have a backward-predicted motion vector with respect to R_j. In the former case, we look at the next frames successively until we find a frame, say B_k, in which the corresponding MB has a valid forward-predicted motion vector, and we use the inverse of that vector. Since this vector is predicted from k frames earlier, we scale it down by a factor of k. If we are not able to find such a B frame, we tag that flow vector as undefined. Similarly, in the latter case, we look at the previous frames successively until we find a B frame with a valid backward-predicted motion vector, which is similarly scaled down by the number of frames over which it was predicted before being assigned.

The next step is to determine the flow between consecutive B frames in the domain. Obviously, there is no direct interaction between such consecutive B frames in the

MPEG stream, in contrast to the aforementioned flow derivation step involving a reference frame.

Flow between successive B frames is derived by analyzing corresponding MBs in those B frames and their motion vectors with respect to their reference frames. We want to find the vector from an MB in one B frame, say B_1, to the corresponding MB in the next B frame, say B_2. Since each MB in each B frame can be of one of three types[4], namely forward-predicted (F), backward-predicted (B), or bidirectionally-predicted (D), there exist nine possible combinations. We can represent these nine pairs by FF, FB, FD, BF, BB, BD, DF, DB, and DD. Each of these nine combinations is considered individually, and flow is estimated between them by analyzing each of the motion vectors with respect to the reference frame.

Let the forward-predicted motion vector of the current MB in B_1 be denoted by $B_1\vec{R}_i$, and let the forward-predicted motion vector of the corresponding MB in B_2 be denoted by $B_2\vec{R}_i$. If, we denote the flow between the MBs of the B frames by $B_1\vec{B}_2$, then we have the relationship

$$-B_2\vec{R}_i = -B_1\vec{R}_i + B_1\vec{B}_2$$

from which $B_1\vec{B}_2$ can be obtained easily. Since only the forward-predicted motion vectors $B_1\vec{R}_i$ and $B_2\vec{R}_i$ are required, this covers the cases of the MB pair having patterns FF, FD, DF, or DD.

Similarly, if the MB pair has pattern BB, BD, or DB, we can find $B_1\vec{B}_2$ by using the backward-predicted motion vectors of B_1 and B_2 with respect to R_j. Let the backward-predicted motion vector of the current MB in B_1 be denoted by $B_1\vec{R}_j$, and let the backward-predicted motion vector of the corresponding MB in B_2 be denoted by $B_2\vec{R}_j$. Then we have

$$B_1\vec{R}_j = B_1\vec{B}_2 + B_2\vec{R}_j$$

[4]Skipped MBs are also actually either forward, backward, or bidirectionally predicted. Intra-coded MBs have no motion vectors and are therefore not considered, and their flow vectors are undefined.

from which $B_1\vec{B}_2$ can be obtained easily.

The only remaining cases to be considered are FB and BF. Clearly, for the case of FB, the flow $B_1\vec{B}_2$ is undefined, because this pattern is an indication of the presence of a cut between the two B frames.

For the BF case, we first find the flow vector $R_i\vec{B}_1$ for the corresponding MB of R_i using the scale-down technique explained above. Then, using the forward-predicted motion vector of B_2, $B_2\vec{R}_i$, we calculate $B_1\vec{B}_2$ as

$$-B_2\vec{R}_i = R_i\vec{B}_1 + B_1\vec{B}_2$$

Similarly, we find the flow vector $B_2\vec{R}_j$ for the MB in B_2 using the scale-down technique. Using the backward-predicted motion vector of B_1, $B_1\vec{R}_j$, we then calculate $B_1\vec{B}_2$ as

$$B_1\vec{R}_j = B_1\vec{B}_2 + B_2\vec{R}_j$$

Since the vectors have been estimated with respect to the reference frames using the scale-down technique, we take the average of these two vectors to yield a better estimate of the actual $B_1\vec{B}_2$.

It should be mentioned that it may not always be appropriate to use the vectors of the same corresponding MBs over the B frames and the reference frames. Consider, for example, Figure 5. We wish to calculate the flow of $(B_1)_{l,m}$, where l and m denote the indices of the current MB in the array of MBs. The forward-predicted vector of B_2 is large enough that its reference 16×16 area is from another adjacent MB, with indices $l-1$, $m-1$. We then use the flow of $(R_i)_{l-1,m-1}$ instead of the flow of $(R_i)_{l,m}$, which would not be proper. We assume that the need to use vectors of alternate MBs arises only when vectors have been predicted over more than one frame, i.e., they are not predicted over adjacent frames. We assume that using corresponding MBs of B_1 and B_2 is sufficient.

Accuracy of computation: To evaluate the accuracy of the estimation, we must provide ground truth and compare them to the results from the flow estimation step.

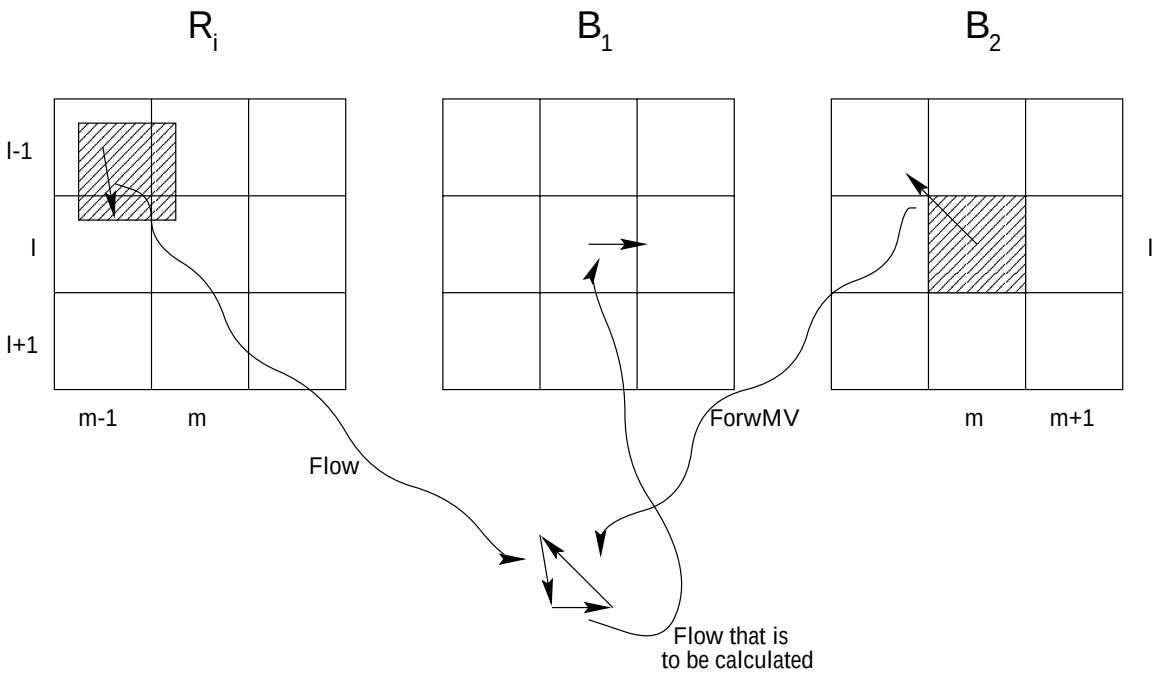


Figure 5: Flow estimation: The MB in B_2 is forward-predicted from an adjacent MB in R_i whose flow is used to calculate the flow of the middle MB in B_1.

Using the original uncompressed image frames, we encode the frames into MPEG files in which all B frames were replaced by P frames using a widely available MPEG encoder called `mpeg_encode` developed by the Plateau Multimedia Research Group at the University of California in Berkeley [4]. IBBPBB ordering, for example, then becomes IPPPPP ordering. Hence every frame is a reference frame and all P frames are predicted from their respective previous reference frames, I or P. The I frames, on the other hand, are not related to any previous frames, and therefore the last P frame occurring before an I frame has no flow information. Nonetheless, using the other frames, we are still able to obtain a good evaluation of the results of the flow estimation process.

We apply the flow estimation step to the files in IPPPPP format, and we compare the flow vectors of the frames of the two encodings in three ways. First, we quantize the vectors of the two encodings in the four principal directions and compare the directions of the corresponding MBs. Second, we compare the angles the

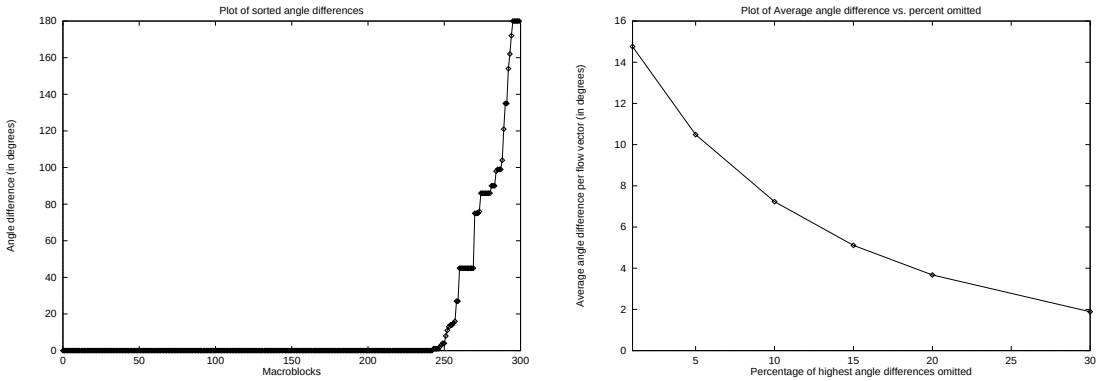
Table 1: Results of the flow estimation process.

Results of flow estimation		
	all frames	frames in valid motion sequences only
matching directions (%)	70 %	71 %
angle difference (deg)	5.1°	5.4°
pixel difference ratio	0.36	0.33

corresponding flow vectors make with the positive x-axis, and determine the average difference in angle between the vectors. Third, we determine the average magnitude of the vector difference of two corresponding vectors in pixel units. The ratio of this average difference vector magnitude to the average magnitude of the flow vectors gives a metric for our evaluations. Due to imperfections during encoding of the MPEG video, experimental results show that noise is frequently present in the motion vectors. Full search during the block matching phase of the encoding process is very time-consuming. Therefore, to exclude the noise, we discard the top 15% of the magnitudes and the angle differences, and only consider the remainder for evaluation. The results of the three experiments are summarized in the three rows of Table 1. The results in the first column are for all the frames of our test clips, whereas the results in the second column are for the frames that belong to sequences in motion classes such as zooms, pans, and tilts.

The results from the test involving only the frames in valid motion sequences (column 2) are marginally better than those from the test involving all the frames (column 1) because the flow vectors are more organized due to the distinct motion, and during encoding, the block matching usually is not very flexible. In uniformly textured backgrounds, or in frames with no motion or irregular motion, the flow can be predicted from different directions.

Figure 6(a) shows a plot of the flow vector angle differences between the IBBPBB and the IPPPPP encodings in a typical frame after sorting in ascending order of



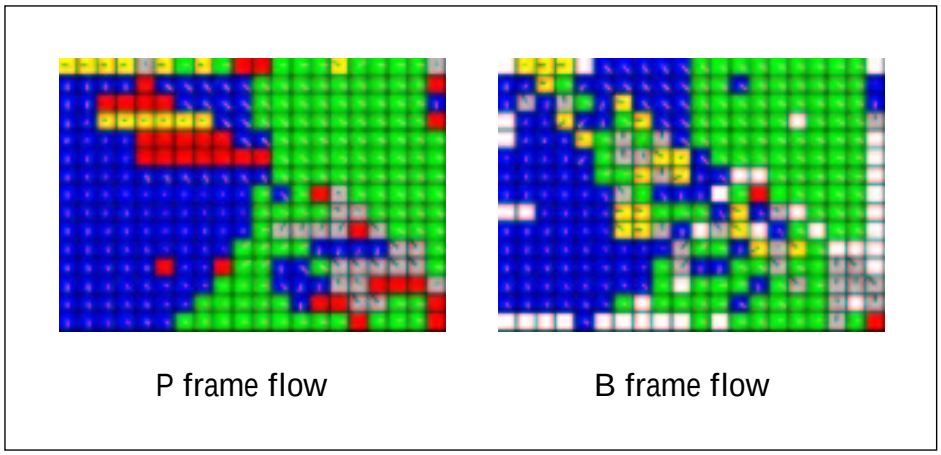
(a) (b)

Figure 6: Flow estimation plots : (a) A sorted plot of the flow vector angle differences between the two encodings of the same clip. (b) A plot showing the relationship between the average flow vector angle difference and the percent of highest angle differences omitted.

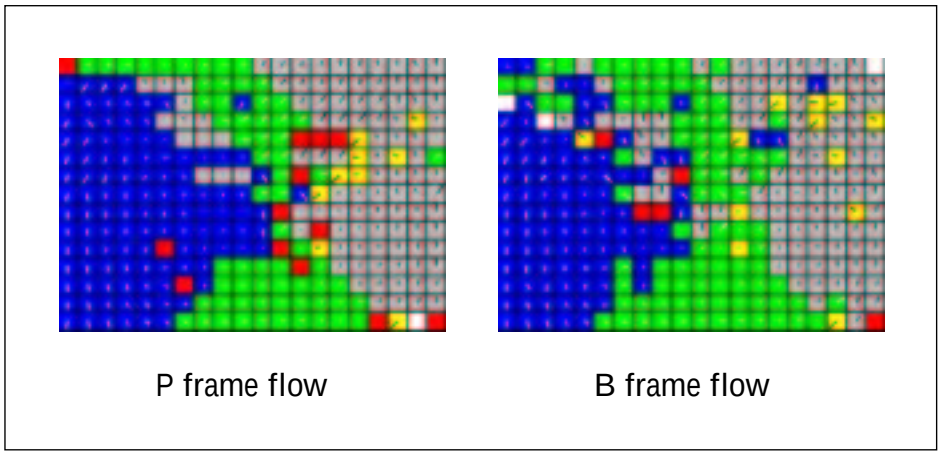
angle differences. Figure 6(b) shows how the average angle difference varies with the percentage of highest angle differences omitted from the calculation. Since the number of angle differences with large magnitudes is relatively low, the average angle difference drops rather quickly with increasing percentage of omitted angle differences.

Examples of the estimated flow vectors are shown in Figures 7(a) and (b). The first pair was taken from the beginning of a "pan left and tilt up" sequence. The second pair was taken from a sequence in which the camera is rotating in the clockwise direction, and is gently tilting up at the same time. For each pair, the MB image on the left was derived from the re-encoded IPPPPP format files, and its corresponding image on the right contains the estimated flow between two B frames from the IBBPBB encoded format files. The shade of the MB represents the direction of the flow vector; the ranges of directions for each shade are shown in Figure 7(c). For "zero" vectors, the shade shown in the center of the circle was used, and for "undefined" vectors, the shade WHITE was used.

We observe that by using the flow vectors of each frame, movements of objects can be sketched. For example, a group of adjacent MBs having similar flow vectors can be associated with a rigid object undergoing some motion. Once a frame is segmented



(a)



(b)

(c)

Figure 7: Examples of flow estimation: In (a) and (b), the images on the left are taken from the IPPPPP encoded files, and the corresponding images on the right show the estimated flow from the IBBPBB encoded files. In (c), the shade codes depend on the range of directions of the flow vectors shown on the circle. The shade at the center of the circle is used for "zero" vectors, and WHITE for "undefined" vectors.

into regions having different flow vectors, the individual segments can be displaced according to their flows, and these deformed frames can be used to generate more robust results during the retrieval phase. This is another reason we select the first frame as a key frame of a scene.

Using this frame-type-independent framework, we are able to consider videos that have differing IPB patterns as being equivalent, and we are not constrained to comparing videos that have the same patterns.

5 Video Indexing using FastMap

An important goal of our research is to be able to organize and retrieve video data that a user is interested in. Now that we have extracted a uniform representation to work on, in this section we explore techniques that can be used to organize and retrieve video clips in the compressed domain. In the next section, we provide experimental results to evaluate how effective these techniques are.

Given a set of video clips encoded in MPEG, we would like to index them to allow "querie-by-frame", or "query-by-sequence". These allow us to 'Retrieve the video clips in the database most similar to this query image', or 'Return the three video clips most similar to the given sequence of frames', respectively. As video clips have both a spatial dimension (represented by DCT coefficients) and a temporal dimension (represented by motion vectors), we propose to use both sets of information to determine the correct answer to the query. If the query consists of a single frame, of course, no temporal information can be derived to match with the motion information of the videos stored in the database. In our approach, the key frame of each scene serves as the index for that scene, with its DC coefficients representing the spatial dimension and its flow vectors representing its temporal dimension.

Using the DC coefficients of all the MBs of a frame alone leads to large feature vectors, and standard database techniques become impractical. Therefore, we use a technique called *FastMap* to reduce the size of the feature vectors to a manageable

level, and use these low-dimensional feature vectors for indexing. The primary advantage of *FastMap* is that it runs in time linear in the number of objects in the database.

5.1 Spatial or frame similarity

Spatial similarity between two frames implies that the frames have similar spatial properties such as luminance, chrominance, texture, and shape. Testing for this similarity involves comparing values that represent those properties. In the pixel domain, the color and luminance values are represented by the values associated with a pixel, but in the compressed domain, the 64 DCT coefficients together represent the values for an 8×8 block of pixels. The DC coefficient alone specifies the average intensity value for that block. Since we do not want to decompress individual frames, we use the DC coefficients of the luminance and chrominance components and compare them with the DC coefficients of the corresponding blocks of other frames to test for spatial similarity.

One approach to computing spatial similarity is to store all the DCT information for every frame of every clip, since the DCT information provide a reasonable abstraction of the spatial information. When a query frame arrives, we can compare it with all other available frames in order to determine the most similar one. However, this approach is not efficient in either time or space. Since most of the frames are similar to the frames adjacent to them, large amounts of redundancy exist. We would like to use properties of the video clips to search only in a small subset of the frames of a clip, and still generate robust matches.

As stated earlier, each clip can be divided up into shots and then into scenes, with each scene denoting a basic coherent sequence of similar frames. A key frame is chosen for each scene and that frame is used to represent that scene. The question is then how to determine similarity between frames. One approach is to compare the corresponding DCT coefficients of the frames directly, since in the compressed domain,

the DCT coefficients best represent the spatial information of the frames. A simpler approach is to treat each frame as a vector of DC coefficients alone (as opposed to all 64 DCT coefficients) and use the Euclidean distance between the vectors to determine the similarity of the frames. This is accomplished as follows.

In a video database, each key frame can be represented as a point in an N-dimensional Euclidean space, where N is the number of DC coefficients. For a 320×240 frame, there are 300 MBs, and if we use the six DC coefficients that exist in each MB, then N is 1800. Traditional multi-dimensional indexing techniques like R-trees tend to be very inefficient in such high-dimensional spaces. Thus we need to have a way of reducing the dimensionality of the points to a manageable level while maintaining the proximity of the points (and thus the similarity).

To achieve this, we use the *FastMap* algorithm described by Faloutsos and Lin [6]. *FastMap* takes as input a distance function between key frames and outputs a point in a low-dimensional space for every key frame in linear time. The main characteristic of *FastMap* is that the output points tend to approximate well the relative distance between the original key frames while keeping the number of dimensions to a manageable level.

The basic idea is that *FastMap* assumes the objects do indeed lie in a certain unknown, k-dimensional space. The goal is to recover the values of each dimension, given only the distances between the 'points'. This is achieved through the use of *projection*: we choose two objects O_a and O_b (referred to as 'pivot objects' from now on), and consider the 'line' that passes through them. We project all the points onto this line. Since we have the distance information between the points, we utilize the *cosine law* to recover the coordinates, as shown in Figure 8(a).

Given an object O_i to be projected, we consider the triangle formed by O_a, O_b, and O_i. The coordinate for object O_i is equal to the length of the line segment O_aE. Applying the cosine law gives

$$x_i = \frac{{d_{a,i}}^2 + {d_{a,b}}^2 - {d_{b,i}}^2}{2d_{a,b}} \tag{1}$$

where $d_{i,j}$ is the distance $\mathcal{D}(O_i, O_j)$ between two objects. Note that the computation of x_i requires only the distances between objects, which are given. Observe that, based on Equation 1, we can map objects into points on a line, preserving some of the distance information. Thus, we have solved the problem for $k = 1$.

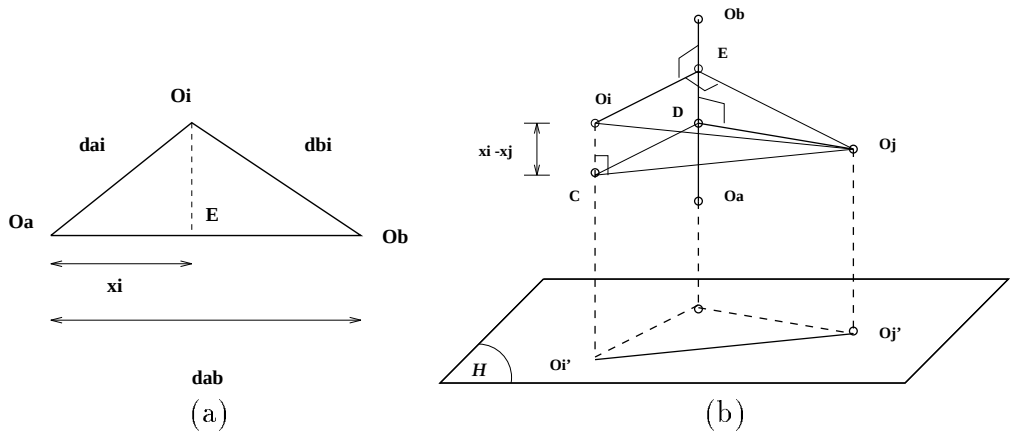


Figure 8: Illustration for *FastMap* : (a) The cosine law. (b) Projection on to the perpendicular hyperplane.

In order to map the objects into a multi-dimensional space, we need to extend the method so that more coordinates can be generated. Once again, we assume the points to be lying in a k-dimensional space (Figure 8(b)). After the first step of *FastMap*, we have found a line (O_a, O_b) on which we can project all the points. Consider a $k-1$-dimensional hyperplane H perpendicular to that line. If we project all the points onto this plane, the points lie in a $k-1$-dimensional space. Let O_i' stand for the projection of O_i (for $i = 1, \ldots, n$). Thus, the problem is transformed to one of finding the coordinates for objects on the hyperplane $\mathcal{H}$. This is the same as the original problem, with k decreased by 1. Once we obtain the projected distances between the points, we recursively apply this algorithm to generate the next coordinate.

To obtain the projected distances $\mathcal{D}'()$ between the points on the hyperplane $\mathcal{H}$, consider the triangle (C, O_i, O_j) in Figure 8(b). The Pythagorean theorem gives the following equation from which the projected distances are calculated:

$$(\mathcal{D}'(O_i', O_j'))^2 = (\mathcal{D}(O_i, O_j))^2 - (x_i - x_j)^2 \quad i, j = 1, \ldots, n \tag{2}$$

Although we indicated that we were projecting onto a $k-1$-dimensional space, note that initially k is arbitrary. We can repeat the above procedure to generate coordinates in any number of dimensions.

For each new axis/dimension, the basic steps of the algorithm are:

1. Pick two pivot objects (as far apart as possible).

2. Compute the projection of each object on the 'line' defined by the pivot objects.

Note that the second step takes linear time. In order for the algorithm to run fast, we need to ensure that the first step also takes linear time. Thus we need a heuristic that can select the pivots in linear time. We would like the pivots to be far apart, however, so that the objects will be more spread out along the projection axis. To avoid costly $O(n^2)$ algorithms, we use a linear time heuristic: starting with a point, pick the point that is farthest away from it. Then use this new point, and repeat this heuristic. Thus the complexity of *FastMap* is $O(n)$. The reader can refer to a paper on *FastMap*[6] for more information, including the pseudo-code of the algorithm.

In retrieval, it is necessary to map new objects (like queries) onto the space formed by *FastMap*. This can be done efficiently. Since we can retain the pivots picked by *FastMap*, we can calculate the projection of the new point onto each axis to obtain its coordinates.

Using *FastMap*, together with the Euclidean distance function, we can organize the frames in an efficient spatial data structure and retrieve nearest neighbors efficiently.

5.2 Temporal similarity

As stated earlier, MPEG streams provide motion information as part of the encoding. Many clips have shots of fairly similar content, e.g., conversational scenes. Key frames can be generated for the same content or action occurring at different points in a clip.

If the query comes in the form of a video clip, e.g., 'Retrieve video clips that might contain this short video scene', we can also compare the motion information. We can consider the corresponding macroblocks of the query key frame, and the set of key frame candidates short-listed by the technique presented in the previous section, to find the candidate key frame undergoing the most similar motion, and return that key frame as the best match.

Our focus is primarily on the direction of the motion and not the maginitude, which gives us a measure less susceptible to noise and minor changes. For each frame to be indexed, we identify the direction of the flow vectors and quantize them angularly into eight bins. We compare the flows of corresponding blocks and use the number of corresponding blocks that have the same flow direction as a second measure of similarity.

6 Experiments and Results

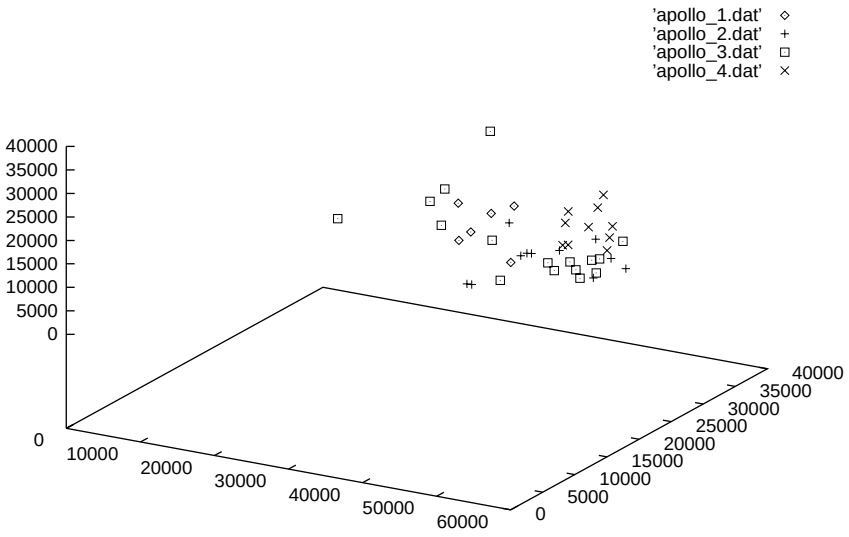
Our system combines both spatial and temporal similarity techniques to provide a simple and efficient method of indexing. First, we index into the key frames using the vectors generated by *FastMap* from the DC coefficients. These DC coefficients are readily available if the query frame is an I frame of a video clip, or a JPEG image. If it is a B or P frame, the coefficients are estimated using the technique referred to earlier, or if it is an image in a different format, it can be converted to a JPEG image. If a query consists of only one frame, we use the index of the *FastMap* vector to locate the key frame which is most similar to the query. We can also return the first few most similar frames and let the user browse the results. In the case of a short query sequence, we ask one query for each frame of the query sequence, and tabulate the votes to identify the winners. These key frames are treated as candidates, and we compare any motion information to modify their ranking.

For the experiments, we used a total of 30 videos containing approximately 15,000 frames digitized at frame rates varying from 5 to 30 frames per second (fps). A total

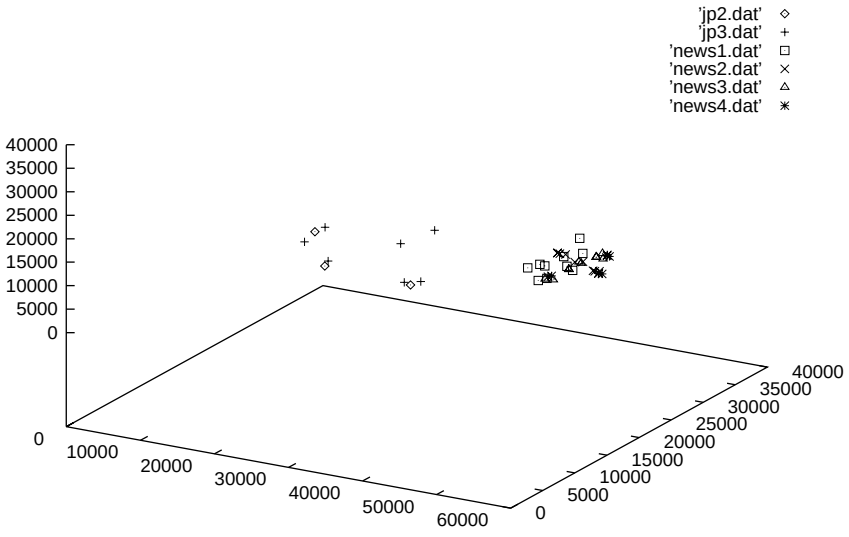
of 329 key frames were identified. For the experiments, we used the first I frame in a subshot or scene as a key frame, instead of the first frame regardless of the frame type, because the I frames do not have the estimated DC coefficients that P and B frames have. This enabled us to study the effectiveness of the method under ideal conditions instead of having noise or artifacts from the DC estimation hinder our evaluation. The clips can roughly be divided into five categories — sports clips, news clips, movie clips, commercial clips, and outdoor surveillance clips. The five groups of videos have different visual properties. For example, the sports clips have predominantly a green grassy field in the background and involve large amounts of motion, while the newscast clips feature people speaking in front of a (generally) dark background with very little motion. The surveillance videos were taken in bright daylight outdoors.

Clustering: Figure 9 shows the clustering of the key frames achieved in a *FastMap* space of three dimensions. The key frames in Figure 9(a) were taken from different clips of the same movie and cluster well. Figure 9(b) shows three distinct clusters of key frames, with the sparse clusters at the left and middle taken from one movie. The two clips that form those clusters are of the same two scenes with each scene forming one cluster. The large cluster at the right is composed of key frames taken from four news interview clips of essentially the same content. Figure 10(a) shows the grouping of the key frames of the surveillance shots, and Figure 10(b) shows primarily two clusters, with the smaller cluster containing key frames from a football game on natural grass, and the larger cluster containing key frames from another football game on astroturf and a baseball game. In Figures 9 and 10, the same scale is used for each of the three axes.

Indexing and Retrieval: Any newly developed technique is incomplete without a thorough testing of its reliability and validity based on quantitative results. First, a test needs to be performed in which, if a query that is clearly very similar to an existing index in the databse is posed to the system, the system finds the appropriate match. If

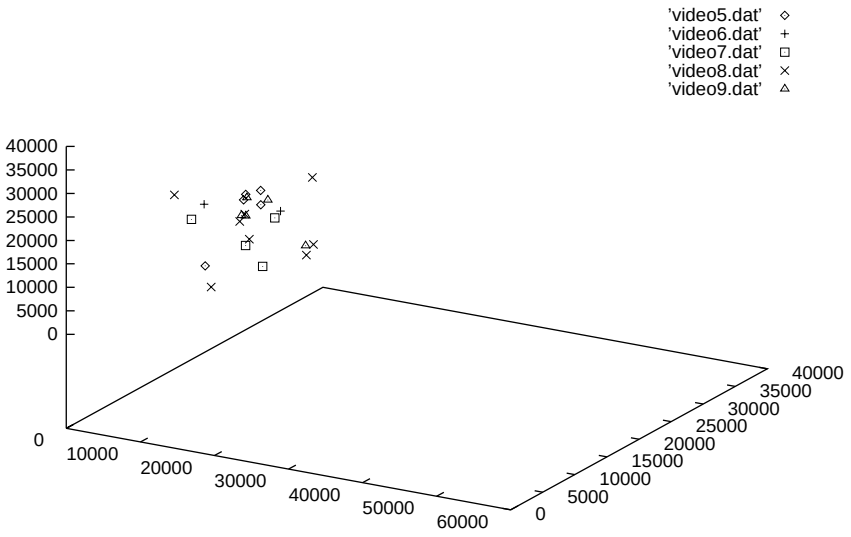


(a)

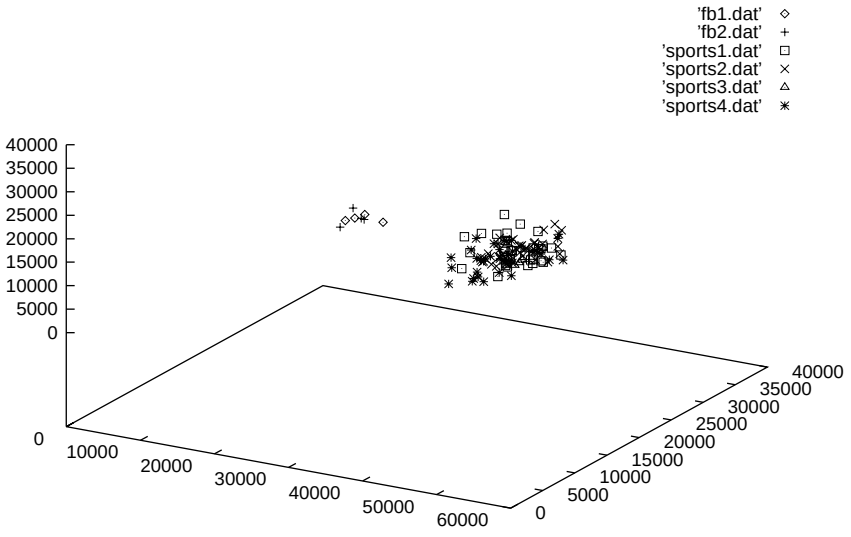


(b)

Figure 9: Key frame clusterings.



(a)



(b)

Figure 10: Key frame clusterings.

this appropriate match is known a priori, evaluating performance is straightforward. Second, if random queries consisting of similar and dissimilar queries are posed to the system, the system should find the best match. For this, we require a means of determining whether the system was able to retrieve the best possible match. That is, we require a means of determining whether the match obtained was the same as the match an *ideal* retrieval system would find. We use a retrieval system employing all the original features (all 1800 DC coefficients) and the Euclidean distance metric as the ideal retrieval system.

Two tests of indexing and retrieval were performed. First, a test was performed to see that if a query that is clearly very similar to an existing index in the database, is posed to the system, the system retrieves the appropriate match. This test consisted of 329 query sequences, one for each key frame in the database, formed by taking the six frames immediately following each key frame. In all the test clips, I frames occurred every six frames, so we used the 'flow' of the sixth offset frame for matching with the flow of the key frame. We used the pivots generated by *FastMap* to calculate the coordinates of the query frames and then found the nearest key frame point(s). A successful query should identify the key frame of the clip and retrieve the shot from which the query frame was taken. The two parameters that were varied were the number of dimensions *FastMap* produced, and the number of nearest neighbors retrieved. Tests were carried out for *FastMap* dimensions 4, 6, 8, 10, and 15. The number of nearest neighbors retrieved varied from 1 to 3 (ordered by distance to the query point).

The query results can be categorized into three types.

Queries that yielded the correct answer as one of the top choices (type A). By a correct answer, we mean that the retrieved key frame is the most recent key frame in the temporal ordering, i.e, it is the key frame of the scene in from which the query was taken.

Queries that yielded another key frame from the same clip (type B). In this case,

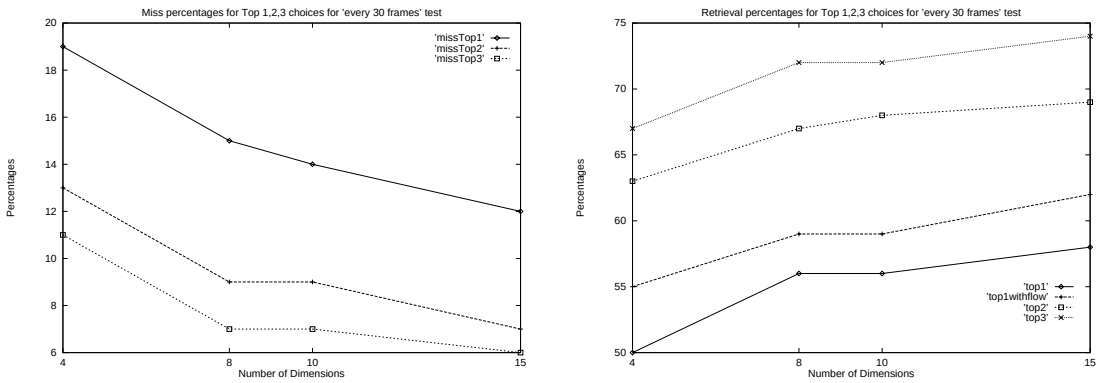
the retrieved key frame was not the most recent in the temporal ordering, but another key frame from the same set of key frames generated from that clip. This happened primarily in two situations. (1) When there were many shots (and thus key frames) that had exactly the same content, and thus the query found a match with one of those alternate shots instead of the shot from which the query was taken. (2) When one continuous shot contained many scenes (and thus key frames) due to changes in content (e.g. due to camera motion) and where the key frame of the next scene was more similar to the query than the key frame of the current scene.

Queries that missed (type C). None of the top choices were from the same clip.

Some type C misses can be justified because we had many clips of the same "program", a football game for example. Figure 11(a) shows a line plot of the percentage of queries that were missed (type C) in the first test as a function of the number of dimensions and the number of top choices retrieved. The graph shows that the miss rate drops to zero for just the top choice at 15 dimensions, while for the top two and three choices it drops to zero at 8 dimensions.

Figure 11(b) shows line plots of the percentages of queries that yielded correct results (type A) as the top choice ('top 1'), as the top choice when using flow ('top1withflow'), and when returning the top two ('top2') and top three ('top 3') frames. The graphs show that better correct retrieval rates and lower miss rates are achieved by increasing *FastMap* dimensions or nearest neighbor choices. We observe a substantial increase in correct retrievals for the 'top choice with flow', as compared with correct retrievals when the flow information is not utilized. The percentages of type B results can be calculated from the type A and type C percentages shown in the two plots. From the graphs, it can be seen that we were able to attain over 95% correct recall with only three frames retrieved.

For the second the set of tests, one frame was selected from every 30 frames in a clip as a single query frame, yielding a total of 473 test frames. This experiment was



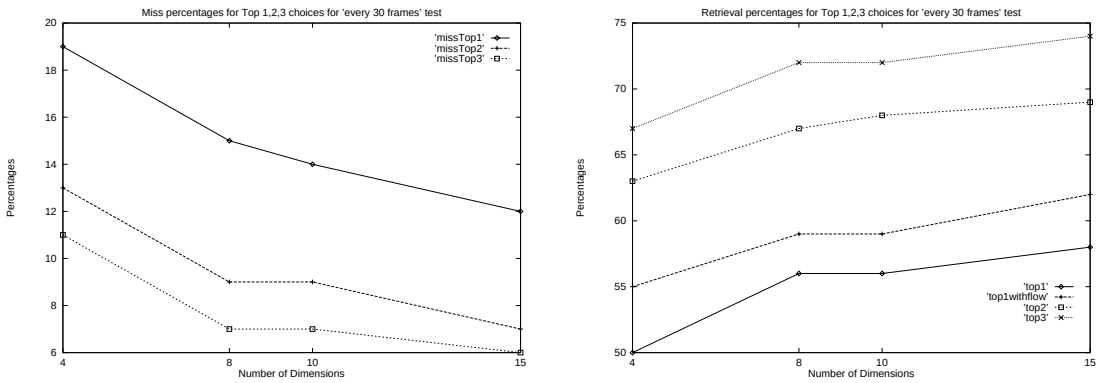
(a) (b)

Figure 11: Retrieval results on video : (a) Recall miss percentages for the 'frame offsets 1 to 6' test. (b) Correct retrieval percentages for the top 1,2,3 choices, and also for the top choice with flow confirmation.

conducted to study how the technique performs on simulated random queries. In this experiment, a query could be quite distant from the the key frame of the scene that it belongs to. Key frame retrieval was performed using each of these query frames and its 'flow'. By using the frame numbers of shot boundaries, correct results were identified. Tests were carried out for 4, 8, 10, and 15 dimensions, while varying the number of nearest neighbors from 1 to 3.

Figures 12(a) and (b) show the graphs for the 'every 30 frames' test. The miss rate for low dimensions is quite high for just the top choice, but drops to an acceptable level for higher dimensions with the top three choices. Due to the randomness of the queries, one would not expect results similar to those of the 'frames 1 to 6' test. As in the previous test, we observe an increase in correct retrieval with rearrangement of the top choices according to flow similarity. The results of such a test depend largely on the set of key frames used. The more the set of key frames is representative of the entire content of the video, the better is the absolute performance. Only a relative comparison with some *ideal* retrieval system can be used to evaluate the performance of our technique.

To evaluate the accuracy of *FastMap* we compared *FastMap* retrieval with an ideal retrieval system employing the Euclidean distance metric with the original features.



(a) (b)

Figure 12: Retrieval results on video : (a) Recall miss percentages for the 'every 30 frames' test. (b) Correct retrieval percentages for the top 1,2,3 choices, and also for the top choice with flow confirmation.

We performed the 'every 30 frames' test with all 1800 DC coefficients and used a pure Euclidean distance metric to find the nearest neighbors of each of the query test frames. The percentage of queries that resulted in misses (result type C) was 13.5%. Figure 13 shows the plot of miss percentages for each of dimensions 4, 8, 10, and 15 and for the top one to top three nearest neighbors. The percentage of misses obtained by using pure Euclidean distance without any dimensionality reduction is shown by the horizontal line. With four dimensions, taking the top two choices performed slightly better than just using Euclidean distance, whereas for dimensions 8 and 10, the top two choices were much better. For 15 dimensions, even the topmost choice was sufficient to yield better performance than the Euclidean distance metric.

Figures 14(a) and (b) illustrate some sample query results. The leftmost frame is the query, followed by the three top matches to its right. With the above-mentioned number of key frames in the database, these experiments show that queries can be processed in a fraction of a second on a SunSPARC 20 workstation.

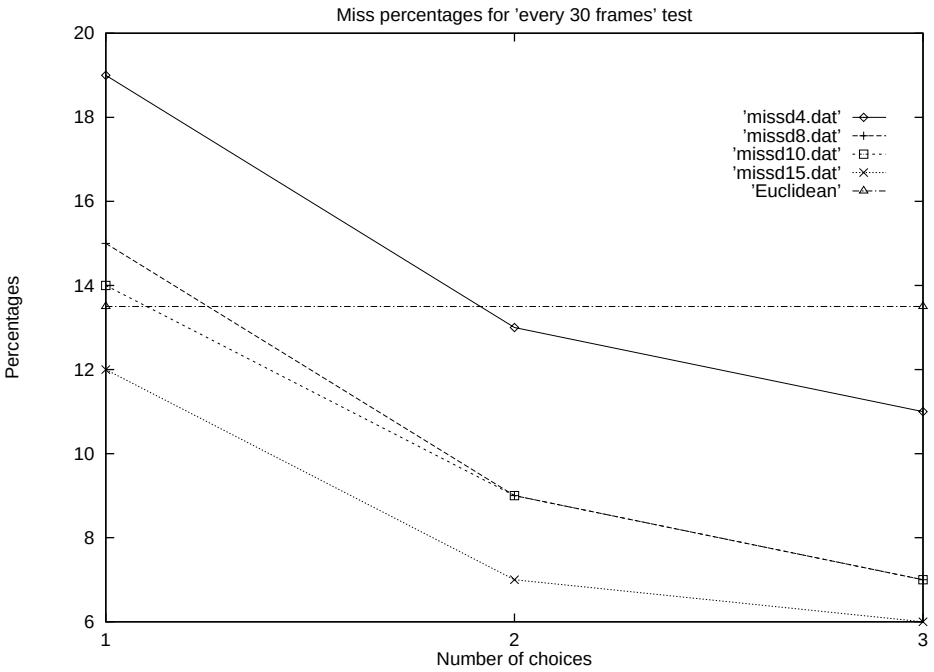
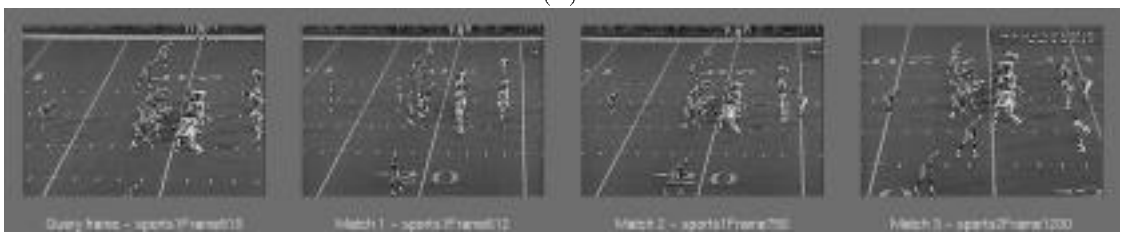


Figure 13: Recall miss percentages for 'every 30 frames test' showing comparison with Euclidean distance results.

7 Conclusion

We have presented techniques for indexing and retrieval of MPEG-compressed video directly from the compressed domain without performing expensive decoding computations. Video is parsed into shots, subshots, and scenes, and key frames are selected. Features are then extracted from these key frames. We have discussed ways of generating a framework in which the I, P, and B frames can be considered equivalent, thereby avoiding any restrictions imposed by the MPEG encoding process. Using the *FastMap* algorithm, the DC coefficients of the key frames are transformed into manageable vectors for archiving, and indexing is performed using these low-dimensional vectors. The motion information is further used to test the potential candidates to yield more robust results.

(a)



(b)

Figure 14: Retrieval results on video : (a) Example 1 from the 'every 30 frames' test. (b) Example 2 from the 'every 30 frames' test.

References

[1] G. Ahanger and T.D.C. Little. A survey of technologies for parsing and indexing digital video. *Journal of Visual Communication and Image Representation*, 7:28–43, 1996.

[2] E. Ardizzone, M. La Casia, V. Di Gesú, and C. Valenti. Content-based indexing of image and video databases by global and shape features. In *Proc. of the International Conference on Pattern Recognition*, pages 140–144, 1996.

[3] E. Ardizzone, M. La Casia, and Davide Molinelli. Motion and color-based video indexing and retrieval. In *Proc. of the International Conference on Pattern Recognition*, pages 135–139, 1996.

[4] Berkeley Plateau Multimedia Research Group. `mpeg_encode` (Berkeley MPEG Tools). Software, August 1995.

[5] S.F. Chang. Compressed-domain techniques for image/video indexing and manipulation. In *Proc. of the IEEE International Conference on Image Processing*,

volume 1, pages 314–317, 1995.

[6] C. Faloutsos and K. Lin. "FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets". In *Proc. of the ACM SIGMOD Conference*, pages 163–174, 1995.

[7] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: the QBIC system. *IEEE Computer*, 28(9):23–32, September 1995.

[8] F. Idris and S. Panchanathan. Indexing of compressed video sequences. In *Proc. of the SPIE Conference on Storage and Retrieval for Still Image and Video Databases IV*, volume 2670, pages 247–253, 1996.

[9] V. Kobla, D.S. Doermann, and K.I. Lin. Archiving, indexing, and retrieval of video in the compressed domain. In *Proc. of the SPIE Conference on Multimedia Storage and Archiving Systems*, volume 2916, 1996.

[10] V. Kobla, D.S. Doermann, and A. Rosenfeld. Compressed domain video segmentation. CfAR Technical Report CAR-TR-839 (CS-TR-3688), 1996.

[11] D. Le Gall. MPEG: A video compression standard for multimedia applications. *Communications of the ACM*, 34:46–58, 1991.

[12] D. Le Gall. The MPEG video compression algorithm: A review. In *Proc. of the SPIE Conference on Image Processing Algorithms and Techniques II*, volume 1452, pages 444–457, 1991.

[13] A. Nagasaka and Y. Tanaka. Automatic video indexing and full-video search for object appearances. In *Proc. of the Working Conference on Visual Database Systems*, pages 119–133, 1991.

[14] K. Shen and J. Delp. A fast algorithm for video parsing using MPEG compressed sequences. In *Proc. of the IEEE International Conference on Image Processing*, volume 2, pages 252–255, 1995.

[15] G.K. Wallace. The JPEG still image compression standard. *Communications of the ACM*, 34:30–44, 1991.

[16] B.L. Yeo and B. Liu. On the extraction of DC sequence from MPEG compressed video. In *Proc. of the IEEE International Conference on Image Processing*, volume 2, pages 260–263, 1995.

[17] H.J. Zhang, C.Y. Low, S.W. Smoliar, and J.H. Wu. Video parsing, retrieval and browsing: An integrated and content-based solution. In *Proc. of the ACM Multimedia Conference*, pages 15–24, 1995.