

Efficient Similarity Search by Summarization in Large Video Database

Xiangmin Zhou, Xiaofang Zhou and Heng Tao Shen

School of Information Technology and Electrical Engineering
University of Queensland,
St. Lucia, QLD 4072 Australia,
Email: {emily,zxf,shenht}@itee.uq.edu.au

Abstract

With the explosion of video data, video processing technologies have advanced quickly and been applied into many fields, such as advertisements, medical etc.. To fast search these video data, an important issue is to effectively organize videos by data compacting and indexing. However, practically, many useful distances for video comparison are suitable to human perception, but non-metric. Therefore, traditional high dimensional data structures can not be utilized to index videos directly when non-metric measures are applied. In this paper, we propose a compact video representation based on global summarization, by which each video in database is mapped into a digital string(a series of cluster id). Consequently, the inter-frame similarity measure is transformed into inter-cluster comparison. Then, we propose an efficient index strategy based on sequence decomposition and reconstruction, by which the spatial index methods can be utilized with non-metric measures for video similarity search. We employ an optimal B⁺-tree with an inverted list attached, for quickly identifying similar clusters and locating potentially similar videos respectively. Finally, a clustering based query summarization technique is proposed, which can greatly reduce the IO and CPU cost in the query processing by batch mapping.

Keywords: Video Summarization, Decomposition and Reconstruction, Batch Query Mapping, Multi-Symbol Representation.

1 Introduction

Recently, content-based video search has attracted much attention, due to its wide applications in many areas such as advertising, news video broadcasting and personal video archive. In a video search system, a user typically wants to retrieve videos which are most similar to a video example or the video in his mind.

For *efficient* video similarity search in large video database, there are two typical approaches: compact video representation and effective video index. The first approach is to reduce the computational cost of similarity measure between two videos by representing them in a much simpler way. As a video sequence typically consists of a large number of frames which are high dimensional vectors, and inter-video

similarity is generally measured by identifying the similarity match for every single frame in each sequence, which results in high computational cost for large video databases. The second approach is to reduce the number of sequence comparisons in large video database by indexing videos efficiently. For a video database which contains large numbers of videos, identifying similar videos by exhaustive scanning all videos is apparently undesirable. We need to index videos (or their summaries) in an efficient way, thus limiting the search space and reducing the number of inter-video comparisons. However, the existing similarity search methods have not addressed both of two issues with a non-metric distance function.

In (Shen, Ooi & Zhou 2005), the authors proposed ViTris for video summarization, and optimal B⁺-tree for indexing, however, they neglected the temporal ordering in a video sequence, thus possibly compromising the effectiveness of similarity search. In (Lee, Chun, Kim, Lee & Chung 2000), the authors proposed to summarize each video sequence by several Minimum Bounding Rectangles (MBRs), and to use R-tree (Guttman 1984) for indexing. However, this video representation method was customized for mean Euclidean distance measure, which preserves the temporal order in a video sequence only in a strict manner. Gaps within video sequences cannot be dealt with well.

In this paper, we propose a summarization based video similarity search approach, which has the following main features: Firstly, we symbolize each video sequence as a digital string. This video symbolization makes each video be represented compactly, moreover, it facilitates video sequence matching. Secondly, we propose an index strategy based on sequence decomposition and reconstruction, which can use traditional index structure to manage video summaries for efficient video search with non-metric measures. We perform the video sequence matching by using an edit distance variant, which is a non-metric measure. It takes into consideration not only the temporal ordering inherent in video sequences, but also the inter-frame similarity between two compared sequences. And finally, we employ some optimizations to improve the search performance and effectiveness, which includes novel query summarization method and multi-symbol representation method.

The rest of the paper is organized as follows. The related work to video search is presented in Section 2. Section 3 describes our approach, including video representation, indexing, and similarity search. Section 4 presents our performance study, and Section 5 concludes the whole paper.

2 Related Work

In general, the method used to address the video sequence matching problem depends on the technique used to represent the sequences (Adjeroh, Lee, & King 1999). One popular video representation technique is to represent each video sequence as key frames. This type of representation is mainly used for video browsing (Chang, Sull & Lee 1999, Zhu, Wu, Fan, Elmagarmid & Aref 2004). Several video representations (Lee et al 2000, Cheung & Zakhor 2003, Shen et al 2005) are proposed in recent years for effective video similarity search. We only briefly review a few representatives.

Lee et al. (Lee et al 2000) proposed to partition each video sequence into subsequences, and represent each subsequence by a Minimum Boundary Rectangle (MBR). Accordingly, the query processing is based on these MBRs, instead of scanning data elements of entire sequences. Thus, it is fast and needs small storage overhead compared with a sequential scan. To index these MBRs, a R-tree structure is utilized. The main problem with this representation is that, it can not be used for similarity measure with local time shifting. As such, some similar videos may be considered as dissimilar due to frame insertion or deletion.

In (Cheung et al 2003), authors proposed a video representation named *Video Signature* (or ViSig). Its basic idea is to summarize each video with a small set of its sampled frames, called video signature, and estimate the percentage of similar frames shared by two sequences by computing the percentage of similar ViSig frame pairs. In our previous work (Shen et al 2005), we introduced a video representation model called ViTri. A sequence is first summarized into a small number of clusters which contain similar videos, and each cluster is modelled as a hypersphere in a high-dimensional space, which is represented by a video triplet (or ViTri). As such, the inter-video similarity is measured by the inter-cluster similarity of them, which is in turn estimated by the number of similar frames shared among the clusters (Shen et al 2005). To further facilitate the search process, ViTris are indexed by an optimal B⁺-tree. However, both of these two video representation methods do not consider the temporal ordering in similarity measure.

We improve the previous work for content-based video search by symbolizing video sequences, which reduces the dimensionality of frame and keep the temporal order in video sequence. Many high-dimensional indexing methods have been proposed (Bohm, Berchtold & Keim 2001). In our work, for non-metric measure, we propose a sequence decomposition and reconstruction based indexing strategy to index the clusters and symbol segments (a Triplet representing a series of identical symbols) by employing an index structure which is closely related to iDistance (Jagadish, Ooi, Tan, Yu & Zhang 2005) or similar indexing methods. In these methods, high-dimensional data are first transformed into single dimensional data, and single-dimensional indexing methods such as B⁺-tree are then used. For example, in iDistance, by selecting a reference point and computing the distance of each high-dimensional point to the reference point, a high-dimensional point is transformed into a single-dimensional value.

3 Our Approach

In this section, we describe in detail our similarity search approach, which includes the following aspects: how a video sequence is represented, how sequences are decomposed and reconstructed, how indexes are

constructed, and finally, give a query video, how similarity search is performed and optimized.

3.1 Video Representation

As mentioned earlier, each video sequence consists of a large number of frames, which results in high computational cost in similarity search. Thus, we need to represent video sequences in a compact and effective way.

Based on the observation, that nearby frames in a video are very similar, and also similar videos usually share similar frames, we can summarize a video sequence by the clusters that contain similar frames. Specifically, suppose each video frame in a database is represented as a high-dimensional image feature vector, we employ a hierarchical clustering method (e.g., k-means) and perform clustering over all frames in the database. As such, the whole database is formed into a set of clusters, each containing similar frames with inter-frame distances (measured by Euclidean distance) less than or equal to ϵ (called *clustering threshold*). We represent a cluster C by $\langle id, O, r, N \rangle$, where

1. id is the cluster identifier;
2. O is the cluster centre which indicates the position of the cluster in the original high dimensional space;
3. r is the radius of boundary hypersphere of the cluster.
4. N is the number of frames in the cluster.

Suppose each video frame is represented by its cluster id , we can symbolize each video sequence as a digital string which consists of the cluster ids of its video frames. This video symbolization makes each video be represented compactly, moreover, it facilitates video similarity matching with consideration of temporal ordering.

3.2 Two-layer Video Indexing

To facilitate video search, we need to organize video data by indexing. Since video sequences can not be indexed directly by traditional index structures when a non-metric distance measure is utilized. We adopt video decomposition strategy before video data are indexed. In a video sequence, a series of consecutive frames that are mapped into the same symbol consists of a *video segment*, which can be represented by $\langle vid, pos, len \rangle$, where vid is the identifier of the video where the symbol appears; pos is the first position of the symbol in the video; len is the length of the segment that shows the frequency of the symbol appearance in the video.

We build index on compact video representations. Our index structure includes two parts: (1) An optimal B⁺-tree for indexing clusters which contain similar video frames in the database; (2) An inverted file for indexing the decomposed video segments. As B⁺-tree is used for indexing one-dimensional data, we first map each cluster center, O , a high-dimensional vector, into a one-dimensional value based on O 's distance to a selected reference point, and then use this one-dimensional value as a key in B⁺-tree. For selecting the reference point, we borrowed the method employed in (Shen et al 2005), where an optimal reference point lies on the line identified by the first principle component and out of its variance segment.

Figure 1 shows the index structure. Each leaf node in the structure may have several entries, and each entry contains a key with a cluster and a pointer to

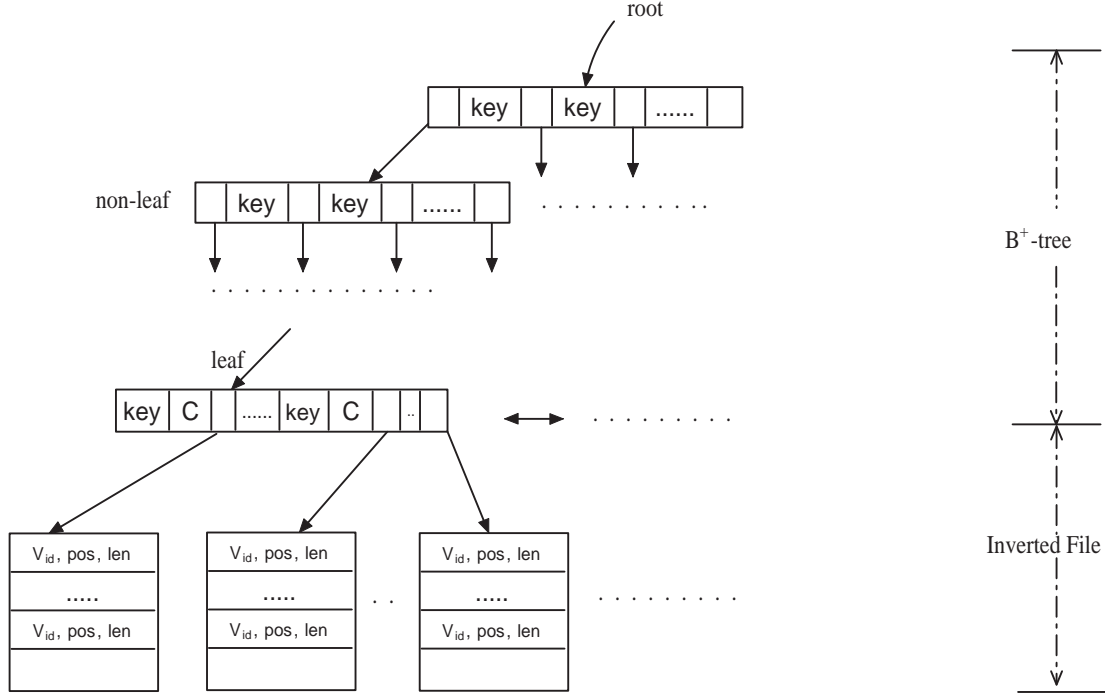


Figure 1: The two-layer index structure.

the corresponding position of the inverted file. The inverted file is used for locating a video segment, i.e., which video sequence a symbol appears in.

3.3 Similarity Search

Given a query video, we need to search the database for most similar videos. During a search, three steps are mainly involved: query mapping, sequence reconstructing, and sequence matching. In the following, we describe each step in detail.

Query mapping For each frame in a query video, we first map it to a set of symbols which are most similar to itself and whose distances from it are within a given similarity threshold ϵ . With our index structure, we can easily identify these symbols. According to the triangular inequality, these symbols' key values should be between $|x - \epsilon|$ and $|x + \epsilon|$, where x is the distance of the query frame to the selected reference point. However, in this process, a frame may not be mapped into any cluster. It shows that this frame is dissimilar with any video frame in the database. In this case, this frame is represented by a special symbol '-', which is dissimilar with any symbol. Further, according to these symbols (except '-'), the potentially similar video segments can be retrieved by the inverted file. By this process, any dissimilar video segment is filtered out. Thus, we can symbolize the query video with the potentially similar videos identified.

To further reduce the number of similar candidates, we adopt the following two filtering strategies: (1) filtering small symbol segments; (2) filtering common symbol segments. The former one is to filter out the candidates that has low similarity with the query. If only few small segments belong to a certain video, these segments will be filtered out from the candidate set. The latter one is to filter out the discriminative segments. If most of the video candidates include the segments consisting of the same symbol, these segments can not contribute to the final query results. These symbol segments will be thrown off.

Sequence reconstructing For video segments obtained by query mapping, we need to recompose symbol sequences according to the location information of each similar segment. To ensure the proper similar results, this process has to conform to the following rule, i.e., the similarity between a reconstructed symbol sequence and the query should be the same as that between its original one and the query. However, there may be cases in which some frames in a similar sequence are not similar to any frame in a query at all, and thus these frames cannot be identified. Therefore, it is impossible to achieve the completely same symbol sequences with the original ones by reconstruction.

Since the identified symbols have the same similarity with query as '-', by sequence reconstruction, we represent these unidentified frames as '-', and facilitate future similarity matching. For example, suppose a query video $Q : < f_1 f_2 f_3 >$ is mapped to $< 223 >$ ('2' and '3' are cluster ids), and also suppose, after query mapping by the index, two video sequences in the database are possibly similar to Q , S_1 represented as $< 11222 >$, and S_2 represented as $< 3344 >$. As no symbol in Q is similar to '1' and '4', S_1 and S_2 will be instead represented as $< - - 222 >$ and $< 33 - - >$ respectively.

Sequence Matching By the above steps, only possibly similar videos are obtained and reconstructed. To refine the results and decide the final query results, we adopt Probability-based Edit Distance, or PED, as a distance measure for sequence matching.

Edit distance is widely used in string matching. Though it is possible for us to use it directly for the similarity measure between video symbol sequences, better performance can be achieved by PED, which takes into consideration not only the temporal ordering inherent in video sequences, but also the inter-symbol similarity of two different symbols in symbol sequences.

For a video symbol sequence $S < s_1, s_2, \dots, s_m >$, its PED to a query symbol video $Q < q_1, q_2, \dots, q_n >$ is computed as follows:

$$PED(S, Q) = \begin{cases} 0 & m = n = 0 \\ m & m > 0 \text{ and } n = 0 \\ \min\{PED(S^{m-1}, Q^{n-1}) + p, \\ \quad PED(S^m, Q^{n-1} + 1), \\ \quad PED(S^{m-1}, Q^n + 1)\} & \text{otherwise} \end{cases} \quad (1)$$

Here, p is decided by the probability distance d between s_i and q_j . We measure d by $\frac{|C_i - C_j|}{|C_i|}$, where C_i is the cluster which contains s_i , C_j is that covers q_j , and $|C_i|$ represents the number of video frames in C_i ; and $|C_i - C_j|$ represents the number of frames which are in C_i but not covered by C_j . Note that this measurement of the similarity degree is probability-based. Given a T (called *probability threshold*), if d is less than T , these two symbols are similar, then $p = 0$; else, $p = 1$.

The whole similarity search process is shown in Figure 2

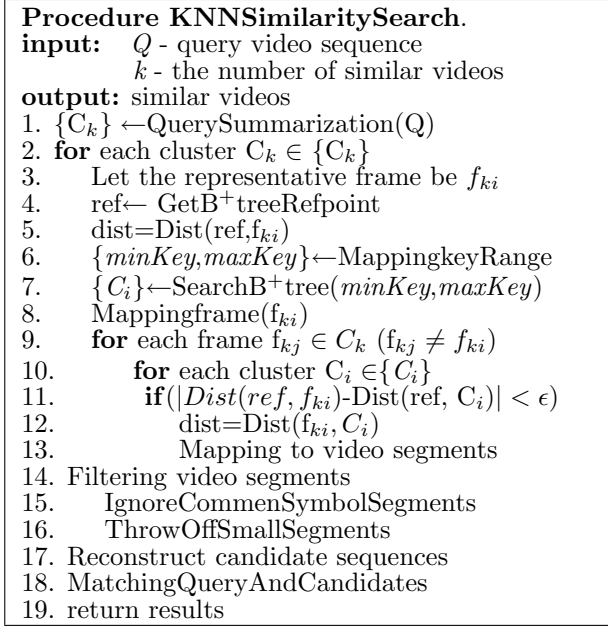


Figure 2: : The similarity search algorithm.

3.4 Optimizations

Some optimizations are employed to reduce the information loss in video representation and improve the performance of similarity search.

3.4.1 Multi-Symbol Representation

To capture more information from original high dimensional space and improve the query result of similarity search, we proposed multi-symbol representation for query video. By this representation, each frame in a video is mapped to a set of clusters that cover it. Thus, each frame corresponds to a set of symbols. And then, the whole sequence is transformed into a symbol set sequence, denoted as $(S_1^Q, S_2^Q, \dots, S_n^Q)$.

For all the clusters containing a certain frame, each of them is similar with the symbolized frame to different extent. To obtain the optimal performance, the symbols in a certain symbol set is accessed orderly. Here, they are sorted by the distances between this frame and its cluster centers in the original space.

Among them, the cluster having the smallest Euclidean distance to the frame is also called *principal cluster*. Accordingly, the id of the principal cluster is the *principal symbol*. The inter-symbol similarity is decided by the the probability distance between the principal symbols.

Given two sequences and the inter-frame similarity threshold, ϵ , in order to judge the inter-frame similarity of two sequences in the original high dimensional space, a multi-symbol match method is employed to compare a specific symbol in the symbol sequence data with a symbol set of the query sequence. This symbol will be compared with each in symbol set orderly. If two symbols are same or completely dissimilar, the comparison process will be stopped. Otherwise, the comparison operation continues over the next one in its symbol set, until the completely similar or dissimilar symbol is obtained or all symbols in the symbol set is visited. If all the symbols in symbol set are overlapped with one in the symbol sequence data, the probability distance is applied.

This multi-symbol match is based on the following theory.

Theorem 1. *Given the symbol set of a frame f , namely $s = \langle c_1, c_2, \dots, c_n \rangle$, and a symbol c' , if c_m is similar with c' , then it is impossible for c' to be dissimilar with another symbol c_n in the symbol set, where c_m and c_n are two different symbols in the symbol set.*

Proof. Since c_m and c_n are different symbols of f , f falls in the intersection part of them.

Suppose c_m be completely similar with c' , all the relations of symbol set are shown in Figure 3, we have f similar with c' . Then

$$d(f, c') \leq \frac{\epsilon}{2}$$

Suppose on the contrary that c_n and c' are dissimilar completely. according to the rule of dissimilarity between symbols, any frame f' in the cluster of c_n , will have a distance more than $\frac{\epsilon}{2} + r'$ from c' . Since frame f is in the cluster of c_n , we have

$$d(f, c') > \frac{\epsilon}{2} + r'$$

These two assumptions contradict each other, thus would not exist in the meanwhile. \square

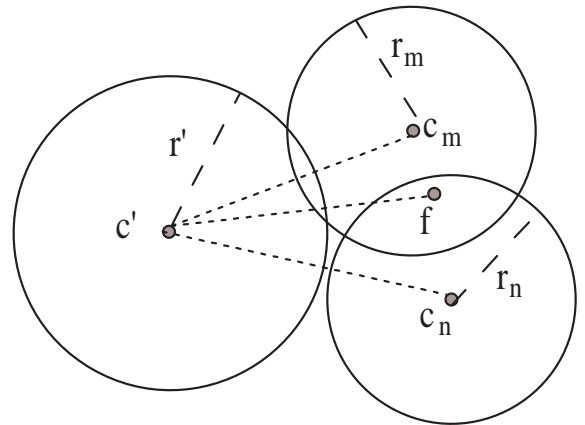


Figure 3: Similarity between symbol and symbol set

Figure 4 summarize the algorithm for the similarity measure between a symbol and a symbol set.

Matching Video Symbol and Symbol Set.

1. Given video symbol c' and symbol set s , here $s = \langle c_1, c_2, \dots, c_n \rangle$
2. **for**(All symbols in the symbol set s , c_i)
3. **If**(c_i and c' are (dis)similar completely)
4. GetInterSymbolSimilarity;
5. StopCheckingSymbolSet;
6. **else**
7. CheckNextSymbolInSymbolSet;
8. **if**(Not found completely (dis)similar pair)
9. MeasurePDistToPrincipleSymbol;

Figure 4: : Similarity Measure between Symbol and Symbol Set.

3.4.2 Batch Query Mapping

For mapping a query video to the corresponding symbols and further retrieving the potential similar segments, the naive method is to map each video frame one by one. As a query sequence may also consist of a large number of frames, naive method is costly. In order to improve the query mapping and the similarity search, based on observation 1, we proposed batch query mapping that performs a batch of individual frame mappings by the index.

Observation 1. *In a video, there exists a lot of very similar frames, which share the same query space in the process of mapping, thus accessing the same disk blocks. Batch processing for them can avoid the unnecessary IO access.*

We propose a query summarization method based on *local clustering*, which perform clustering over query frames by their inter-frame similarity. Since nearby frames in a video are usually quite similar. By doing so, a query sequence is formed into a small number of query clusters which contain similar query frames, where the inter-frame distance in the same cluster is no more than the similarity threshold, ϵ .

Due to the small number of the query clusters, the number of disk re-accesses is reduced greatly. For each frame cluster, with radius r_1 , the cluster center is chosen as a representative. The symbolization for the frames in the whole cluster is completed by finding the first symbol, with radius R_1 , of this representative, and then obtaining all neighboring cluster spaces with distances no more than $R_1 + r_1$ from the first cluster center. Meanwhile, the distance between each cluster in query space and the representative is used to perform the filtering based on triangular inequality, reducing the CPU cost of mapping. Consequently, both IO and CPU cost for query mapping are reduced greatly, thus enhancing the performance of similarity search.

4 Performance Study

In this section, we present an empirical study to evaluate the proposed approach over large real video data sets. The evaluation is based on two aspects: (1) the effectiveness of video symbolization; (2) the efficiency of video indexing.

4.1 Setup

The experiments are conducted over two real datasets from 6000 15s video clips and 896 10s clips, which are recorded by using Virtual Dub at PAL frame rate of 25fps (Shen et al 2005). Each frame is represented as a 64-d vector in the RGB color space. In the same

Para	Description
ϵ	Clustering threshold
T	Probability threshold
K	Number of most similar sequences

Table 1: Parameters used in the experiments.

set of tests, the lengths of video sequences used are similar.

For the effectiveness, we evaluate symbolization representation by comparing with key frames approach, where key frames are obtained by shot boundary detection method (Nagasaka & Tanaka 1992), to test the information loss. As the original data has no information loss, we examine the effectiveness by employing the video symbolization and the original videos and see how close the results generated by the PED over symbol sequences are to those by the edit distance (ED) in the original space. ED in original space is different from PED in how to decide the p . If the Euclidean distance between two frames is more than ϵ , $p=1$; otherwise, $p=0$. The set of query results by ED in the original space is denoted as rel , and that from summarization method is ret , the Accuracy of matching is defined as:

$$Accuracy = \frac{|rel \cap ret|}{|rel|} \quad (2)$$

For the efficiency, we compare our proposed two-layer indexing method with optimal reference based B^+ -tree, the high performance of which has been proved in (Shen et al 2005), and sequential scan as a base. By varying the size of video dataset, the efficiency of two-layer video indexing and ViTris indexing are compared. We measure the search efficiency in terms of IO and CPU cost. IO cost is evaluated by the number of page accesses, and the size of each page is set to be 4k. For CPU, we use the number of distance calculations, since it dominates the time of CPU cost and is also objective. All the experiments were performed on a Sun Enterprise E420(4*450MHz CPU's with 4GB RAM). Table 1 summarizes the parameters used in the experiments.

4.2 Effectiveness of Symbolization

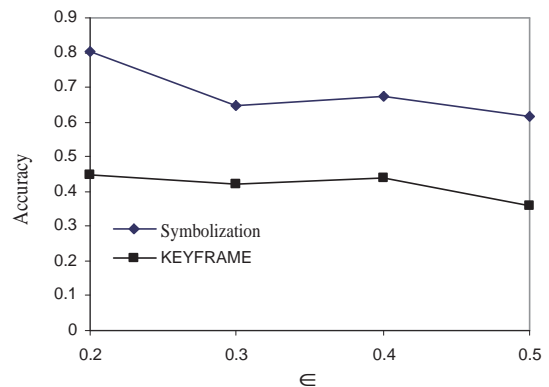


Figure 5: Accuracy By ϵ Varying in KNN Search on 15s Clips

To examine the effectiveness of symbolization, we need to estimate the value of optimal probability

threshold of similar symbols, T , in probability measure. According to our analysis, $T=0.5$ is the optimal symbol similarity threshold.

We evaluate our representation approach by turning the ϵ value, for testing the information loss of it with clustering threshold. Figure 5 shows the test results. We fixed T to 0.5, and changed ϵ from 0.2 to 0.5. Obviously, the accuracy of our method is much better than that of key frames representation. Meanwhile, with the increasing of ϵ , the effectiveness tends to degrade for both methods, because of the more information lose for them. For the symbol sequence similarity search, since the overlap extent is small for small ϵ , inter-symbol similarity is very close to inter-frame similarity, leading to high effectiveness of it. With the ϵ increasing, due to the high overlapping, the stability of inter-symbol comparison degrades accordingly, causing the degradation of effectiveness.

4.3 Efficiency of Batch Mapping

During query mapping, a query video which consists of large numbers of frames may need to be summarized first to reduce the cost. In this set of experiments, we compare the performance of batch mapping with that of the naive mapping approach.

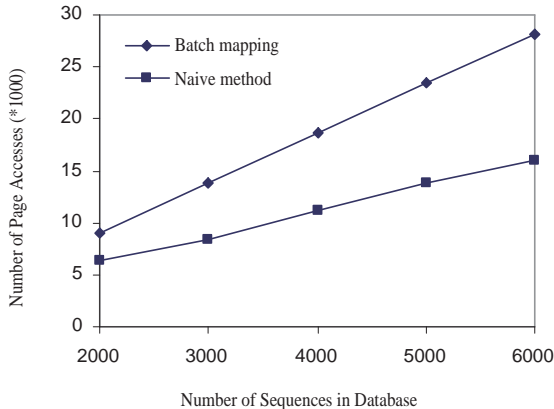


Figure 6: IO in Batch Mapping

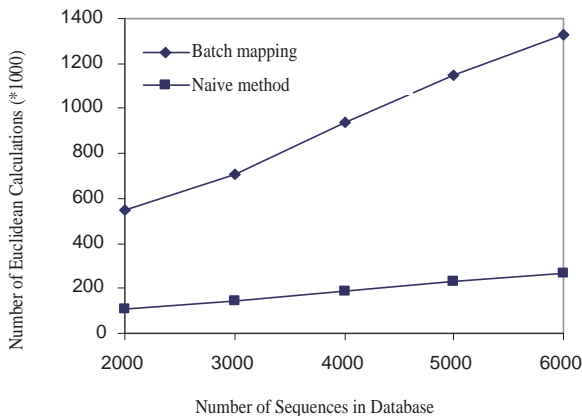


Figure 7: CPU in Batch Mapping

Figure 6 and 7 show the IO cost and CPU cost respectively. As shown in the figures, the batch mapping improves the performance a lot, and both IO cost and CPU cost are reduced greatly. For naive approach, each query frame needs to be mapped to the index space one by one. Also, the same index space may be accessed repeatedly as nearby frames

in a query video are usually similar and share the same index space, thus incurring much more IO and CPU costs. When the batch mapping is used, similar frames are clustered together, thus query mapping has less IO costs. Meanwhile, in the batch mapping, the filtering based on triangular inequality is performed, accordingly, the CPU cost of batch mapping is also reduced greatly.

4.4 Efficiency of Two-layer Video Indexing

In this part, we performed experiments to test the two-layer index method by comparing with ViTris indexing and the sequential scan as a base. The IO cost and CPU cost are shown in the Figures 8 and 9.

Obviously, compared with ViTris indexing and sequential scan, the filtering ability of two-layer index has been improved greatly. Since two-layer index gives up a large number of candidates, in which only very few similar query symbols appear or only common symbols which are not discriminative, the number of IO and that of probability edit distance calculations are reduced noticeably, up to half. At the same time, it is clear that the filtering ability is more efficient for small K , with the increasing of K , becoming a bit weaker, but still keep high efficiency.

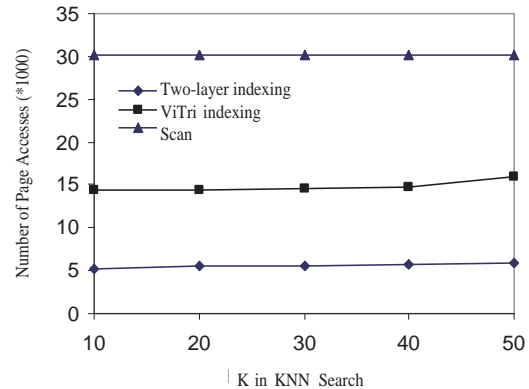


Figure 8: Effect of K in KNN Search

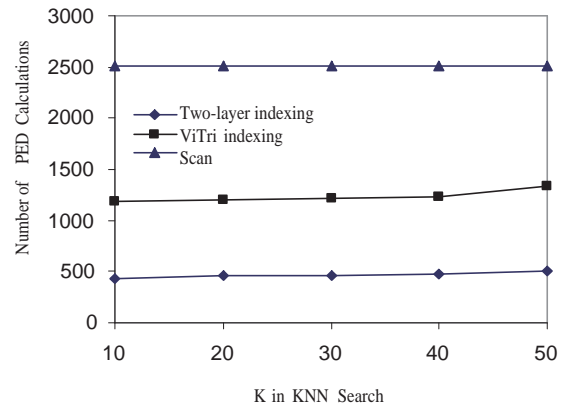


Figure 9: Effect of K in KNN Search

4.5 Effect of Dataset Size

We compare the efficiency of our two-layer video indexing with ViTris indexing in case of varying the number of 15s video clips varying from 2500 to 5000. Figure 10 and 11 show the IO cost and the number

of frame comparisons in two-layer video indexing by comparing with that of B⁺-tree based ViTri indexing.

We can see that, with the increasing of the data size, the IO and CPU costs increase for both of them. For another, obviously, the two-layer video index needs far less IO and frames comparison than ViTri indexing, since two-layer video index performs high dimensional distance calculations only in the process of frame mapping, while not in similarity measure. Moreover, query summarization based batch mapping can filter out most of the unconcerned symbols, thus reducing the cost of two-layer video indexing greatly. Comparing with B⁺-tree based ViTri indexing, for the same video dataset, the cost of two-layer video indexing is improved one order of magnitude.

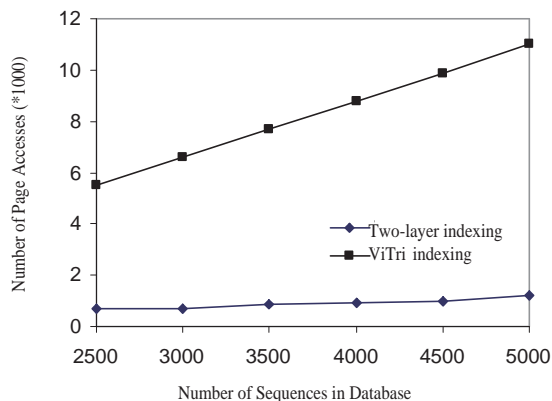


Figure 10: Efficiency by varying dataset size

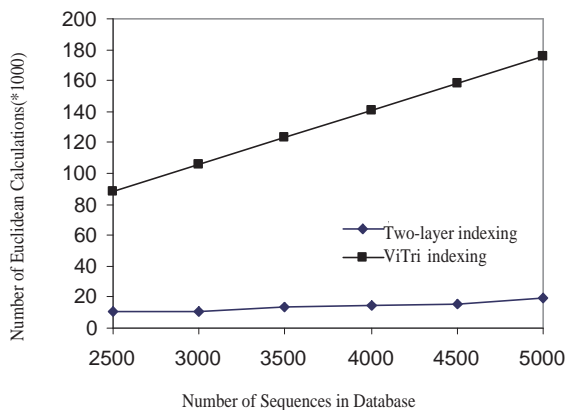


Figure 11: Efficiency by varying dataset size

5 Conclusions

In this paper, we have proposed a summarization based video similarity search approach, which has the following features: by video symbolization, each video sequence is symbolized as a digital string, which facilitates similarity matching with consideration of temporal ordering; by decomposition and reconstruction based video indexing strategy, video data are indexed and similar videos can be efficiently retrieved with non-metric similarity measure. Further, optimizations have been employed to further improve the search performance. We have done extensive performance study and our results have shown that our approach is very efficient, while keeping high search accuracy.

6 Acknowledgments

This research was supported by ARC grant DP0663272 and an Endeavour IPRS.

References

- Adjeroh, D. A., Lee, M. C. & King, I. (1999), A distance measure for video sequences, *in* 'Computer Vision and Image Understanding', Vol. 75, pp. 25–45.
- Bohm, C., Berchtold, S. & Keim, D. (2001), Searching in High-dimensional Spaces: Index Structures for Improving the Performance of Multimedia Databases, *in* 'ACM Computing Surveys', Vol. 33, pp. 322-373.
- Chang, H. S., Sull, S. & Lee, S. U. (1999), Efficient video indexing scheme for content-based retrieval, *in* 'IEEE Transactions on Circuits and Systems for Video Technology', Vol. 9, pp. 1269-1279.
- Cheung, S-C. S. & Zakhor, A. (2003), Efficient video similarity measurement with video signature, *in* 'IEEE Transactions on Circuits and Systems for Video Technology', vol. 13, 2003, pp. 59-74.
- Guttman, A. (1984), R-Trees: A Dynamic Index Structure for Spatial Searching, *in* 'Proceedings of the ACM SIGMOD International Conference on Management of Data', ACM Press, Boston, Massachusetts, USA, pp. 47-57.
- Jagadish, H. V., Ooi, B. C., Tan, K-L., Yu, C. & Zhang, R. (2005), iDistance: An adaptive B+tree based indexing method for nearest neighbor search, *in* 'ACM Transactions on Data Base Systems (TODS)', Vol. 30 ACM Press, New York, NY, USA, pp. 364–397.
- Kim, S. H. & Park, R-H (2002), An efficient algorithm for video sequence matching using the modified Hausdorff distance and the directed divergence, *in* 'IEEE Transactions on Circuits and Systems for Video Technology', Vol. 12, pp. 592-596.
- Lee, S-L., Chun, S-J., Kim, D-H., Lee, J-H. & Chung, C-W. (2000), Similarity Search for Multidimensional Data Sequences, *in* 'Proceedings of the International Conference on Data Engineering', Vol. 12, pp. 599-608.
- Nagasaka, A. & Tanaka, Y. (1992), Automatic Video Indexing and Full-Video Search for Object Appearances, *in* 'Proceedings of the IFIP TC2/WG 2.6 Second Working Conference on Visual Database Systems II', Vol. 12, North-Holland, pp. 113–127.
- Shen, H. T., Ooi, B. C. & Zhou, X. (2005), Towards Effective Indexing for Very Large Video Sequence Database, *in* 'Proceedings of the ACM SIGMOD International Conference on Management of Data', pp. 730-741.
- Zhu, X., Wu, X., Fan, J., Elmagarmid, A. K. & Aref, W. G. (2004), Exploring video content structure for hierarchical summarization, *in* 'Multimedia System', Vol. 10, pp. 98-115.