A QUICK SEARCH METHOD FOR MULTIMEDIA SIGNALS USING FEATURE COMPRESSION BASED ON PIECEWISE LINEAR MAPS

Akisato Kimura, Kunio Kashino, Takayuki Kurozumi and Hiroshi Murase

NTT Communication Science Laboratories, NTT Corporation
3-1, Morinosato-Wakamiya, Atsugi-shi, Kanagawa, 243-0198, Japan
E-Mail : {akisato, kunio, kurozumi, murase}@eye.brl.ntt.co.jp

ABSTRACT

We propose a quick algorithm for multimedia signal search. The algorithm comprises two techniques: feature compression based on piecewise linear maps and distance bounding to efficiently limit the search space. When compared with existing multimedia search techniques, they greatly reduce the computational cost required in searching. Although feature compression is employed in our method, our bounding technique mathematically guarantees the same recall rate as the search based on the original features; no segment to be detected is missed. Experiments indicate that the proposed algorithm is approximately 10 times faster than and as accurate as an existing fast method maitaining the same search accuracy.

1. INTRODUCTION

This paper proposes a method for quick search through a long multimedia signal stream (a *stored signal* or a *database*) to detect and locate a known audio or video signal (a *reference signal* or a *query*) based on signal similarity. We use the term "multimedia signal" to refer to an audio or video signal. The method discussed in this paper is basically applicable to both. For simplicity, however, we will specifically focus on the video retrieval in the following sections.

There are many works in the literature on multimedia signal search or retrieval. One of the major approaches is based on the symbolic indices that correspond to specific objects, events, or meanings [1, 2, 3]. For example, when all the home-run scenes are indexed in a recorded baseball game, one would instantly find those scenes based on the indices. In this approach, the method to generate such indices has been a major research issue.

Another approach to signal search is based on signal similarity. In this approach, it is assumed that a reference signal is given as a query. The task is thus comparison between the reference signal and each section of a signal stored in a database. It is clear that the main research issue in this approach is speed. Specifically, feature vectors for audio or video signals often tend to be high-dimensional, which is not necessarily suitable for the various tree-search algorithms developed in the database field.

In coping with the high-dimensionality problem, it is natural to think of dimension reduction, or more generally, feature compression. Example of feature compression methods are the subspace method [5] based on Karhunen-Loeve (KL) transform, project clustering [6], signal decomposition [7], and singular value decomposition (SVD) [8]. However, these methods usually degrade search accuracy.

Here, we propose a quick and accurate search algorithm for multimedia signals based on histogram matching. The method comprises a feature compression technique and a distance bounding technique. The feature compression is done by piecewise linear maps based on Karhunen-Loeve transform. Distance bounding means limiting the search space using the lower bounds of distances between signals while searching. Our method also uses the distance between the original features and the mapped features, and this greatly reduces the number of necessary feature comparisons while guaranteeing that no segment to be detected is missed.

Keogh *et al.* have recently reported a fast signal searching method based on feature compression [9, 10]. Unlike them, we use a feature histogram as a signal representation. This is because, as we have already reported [4], the histogram itself is a very efficient representation in comparison with the original features, such as waveforms, spectrograms, color distributions, DFT coefficients, and so on.

This paper is organized as follows: Section 2 overviews the algorithm called TAS, which is our previous method based on feature histograms. Section 3 explains the core part of our new algorithm. Section 4 evaluates the speed and the accuracy of the algorithm using a recording of real TV broadcasting. Finally, Section 5 gives conclusions.

2. TIME-SERIES ACTIVE SEARCH (TAS)

Fig. 1 outlines the TAS method. In the preparation stage, the feature vectors are calculated from both the reference signal and the stored signal. The feature vector $\boldsymbol{f}(k)$ is written as

$$\boldsymbol{f}(k) = (f_1(k), f_2(k), f_3(k), \cdots, f_N(k)),$$

0-7803-7402-9/02/$17.00 ©2002 IEEE

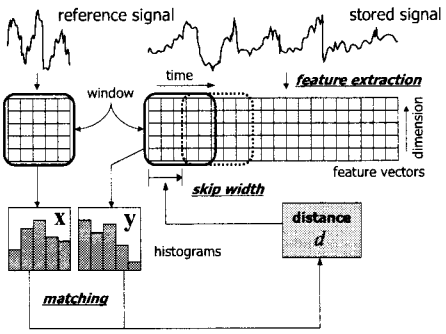

Fig. 1. Overview of TAS method

where N is the number of dimensions and k is the sample time. For example, sets of normalized short-time power spectra are used as a feature for audio signals, and sets of average RGB values in subimages are used as a video feature. The feature vectors are then quantized using a vector quantization (VQ) algorithm. In the search stage, the windows are applied both to the reference feature vectors and to the stored ones. Next, histograms, one for the reference signal and one for the stored signal, are created by counting the number of the feature vectors over the window for each VQ codeword. The distance between these histograms is then calculated. When the distance falls below a given value (a *search threshold*), the reference signal is detected. In the last step, the window on the stored signal is shifted forward in time and the search proceeds.

The main point of TAS is that it models a signal using feature histograms. The distance between the reference and stored histograms over the windows can be determined in several ways. In this paper, we use L_2*-distance* (*Euclid distance*), which is defined as

$$d_2 = d_2(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \sqrt{\sum_{i=1}^{L} (x_i - y_i)^2}, \qquad (1)$$

where $\boldsymbol{x} = (x_1, x_2, \cdots, x_L)^T$ and $\boldsymbol{y} = (y_1, y_2, \cdots, y_L)^T$ are the histograms for the reference and the stored signals, L is the number of dimensions (i.e. number of VQ codeword) and $(\cdot)^T$ means transposition of a vector. When the distance is calculated for one segment, the lower bound of the distance can be determined by

$$\underline{d}_2(n) = d_2(n_1) - \frac{n - n_1}{\sqrt{2}D}, \qquad (2)$$

where $d_2(n_1)$ is the distance when the stored signal window is at the n_1-th frame and D is the window width determined by the number of frames. This means that the histogram matching for the sections that give the lower bound of the distance not smaller than the search threshold can be

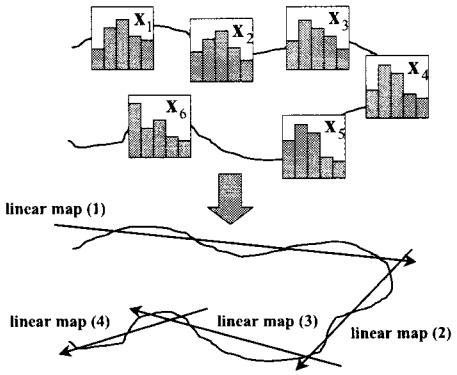

Fig. 2. Outline of map determination

omitted while guaranteeing that no segment to be detected is missed. The skip width w is given by

$$w = \begin{cases} \text{floor}(\sqrt{2}D(d_2 - \theta_1)) + 1 & (\text{if } d_2 > \theta_1) \\ 1. & (\text{otherwise}) \end{cases}$$

where floor(x) means the greatest integer less than x, and θ_1 is the search threshold.

3. METHODS

Although the histogram itself used in TAS can be viewed as a compressed feature derived from the original feature, the high-dimensionality problem can still arise. To further accelerate searching, we here introduce a feature compression techniqu and a distance bounding technique. The first step of the former technique is to determine piecewise linear maps from a stored signal. The second step is feature compression by mapping original features with piecewise linear maps. The latter technique, the distance bounding, also involves a skipping as TAS. The following sections will describe a series of those steps.

3.1. Map Determination

Fig. 2 outlines this step. Suppose that the original features (e.g. color distribution in each frame) were already calculated on the stored signal and the histograms were also created with a predefined-length (e.g. 10 seconds) window corresponding to every feature frame (e.g. 29.97Hz). In the present method, the window length (i.e. the reference signal duration) is fixed, while in TAS the length varies from one search to another. This is because histograms are created during the search in TAS case, while histograms for the stored signal are created prior to the search in our new method.

Here we note that a histogram sequence is "continuous" by nature in the histogram space, and introduce piecewise lower-dimensional representation of the sequential histogram path. That is, the histogram path is then equally divided into a certain number (e.g. 1000) of segments. Then,

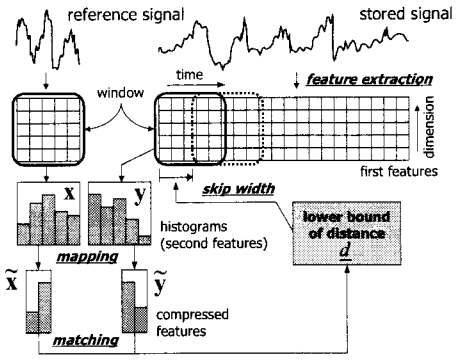

Fig. 3. Overview of the proposed method

KL transform is performed with respect to every segment to create a subspace for each segment. Finally, we determine maps of sequences with a minimum number of components such that the contribution rate exceeds a certain value.

3.2. Feature Compression

Feature compression is done by mapping the histograms to their corresponding subspace. The point here is calculating *mapping distance*. The mapping distance is defined as the distance between the histograms before and after mapping. Hereafter, we refer to this final form of a signal feature as the "compressed feature". Thus, compressed feature vectors finally comprise mapped histograms and mapping distances.

The processes mentioned above are query-independent and is done prior to the search.

3.3. Distance Bounding

The following processes are done after a reference signal is provided.

Fig. 3 overviews the search steps. They are similar to those of TAS. First, we create a histogram for the query signal in the same way as for the stored signal (the *reference feature*). Next, we transform the histogram according to each linear map constructed in the previous step, to create a compressed features. Then, the compressed stored feature and the compressed reference feature corresponding to the matching point are matched. In the matching, we calculate the distance between two compressed features $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{y}}$. This distance gives a lower bound of the distance between two histograms, $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$\begin{aligned} \{d_2(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})\}^2 &= \{d_2(g(\boldsymbol{x}), g(\boldsymbol{y}))\}^2 \\ &\quad + \{d_2(\boldsymbol{x}, g(\boldsymbol{x})) - d_2(\boldsymbol{y}, g(\boldsymbol{y}))\}^2 \quad (3) \\ &= \min\{d_2(\boldsymbol{x}, \boldsymbol{y})\}^2, \quad (4) \end{aligned}$$

where $g(\cdot)$ means a linear map corresponding to the matching point, and the minimum is taken over all $(\boldsymbol{x}, \boldsymbol{y})$ given $g(\boldsymbol{x})$, $g(\boldsymbol{y})$, $d_2(\boldsymbol{x}, g(\boldsymbol{x}))$ and $d_2(\boldsymbol{y}, g(\boldsymbol{y}))$.

Note that we calculate the lower bound using the distance in the original histogram space as well as the compressed space. This generally gives a tighter bound in comparison with the method only using the information contained in the compressed space.

3.4. Skipping

We can calculate a lower bound of the distance at the n-th frame from the distance between the compressed feature at the n_1-th frame using (2). Then, we can also calculate the skip width in the same way as TAS, guaranteeing that no segment to be detected is missed. This skip mechanism is still employed in the present method.

4. EXPERIMENTS

4.1. Conditions

In the experiments, we used a video recording of a 24-hour TV broadcast as a stored signal. The frame rate was 29.97 Hz, and image size was 320 × 240 pixels. The tests were carried out on a PC (Pentium III 966 MHz). In the feature extraction, each frame was first divided into 6 (2 × 3) areas, and then the average of RGB-values was calculatied in each erea. Therefore, the number of dimensions of an original feature vector was 18. Those feature vectors were calculated on every frame, and then quantized. The codebook size for the feature vectors was 256. Then, histograms of the feature vectors were created. Therefore the histogram feature dimension was 256 before compression. Those parameter values were empirically chosen. For simplicity, parts of the stored signal were used as reference signals: ten 15-s segments were randomly chosen.

4.2. Compression Performance

First, we tested the compression performances. The results are shown in Fig. 4, where the horizontal axis is the number of maps and the vertical axis expresses the average number of dimensions of the mapped features. The contribution setting is shown in the inset. The data indicate that the average dimensionality of the features monotonically decreases as the number of maps increases. For example, the average number of dimensions decreases to 9.02 when the contribution rate is 97.5% and the number of maps is 1000.

4.3. Search Performance

We then tested the search speed and the search accuracy. In the search speed test, two measures were used: the number of matches and the CPU time for the search. The CPU time here means the time required for matching. The times for extracting the original features, making histograms, and feature compression were excluded because these processes can be done prior to the search in practical situations.

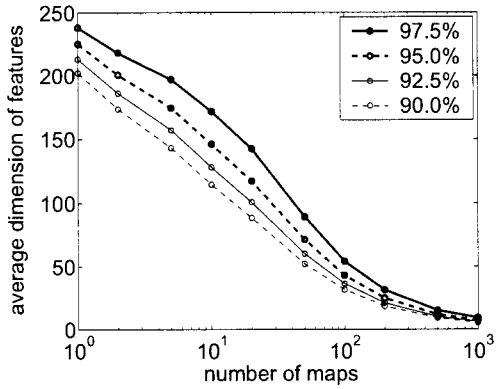

Fig. 4. Compression preformance of the proposed method

	CPU time (msec)	number of matches	precision rate(%)
SSDA	840.00	2592604	100.0
no bounding	32.78	48145	11.4
TAS	158.20	18472	100.0
proposed	**14.62**	21000	100.0

Table 1. Search performance when a 24-h TV broadcasting was scanned (The recall rate is guaranteed to be 100%)

In the search accuracy test, the precision rate was measured. The precision rate is defined as the ratio of "the number of correctly-detected outputs (i.e. local minimum points in the distance)" to "the number of all outputs (i.e. local minimum points)" from the algorithm. Note that the recall rate is guaranteed to be 100% and was not examined here. The search threshold of 45 was chosen so that there are no incorrect detections and no misses in TAS. The number of linear maps was 1000, and the contribution rate was 97.5%.

Table 1 lists the results. The proposed method is over ten times faster than TAS that does not employ feature compression. It is natural that the number of matches is increased in the proposed method in comparison with TAS due to the histogram dimension reduction, but the increase is small. The proposed method also yielded the same accuracy as TAS in this test. There were no incorrect detections or no misses.

In addition, we compared the proposed method with (i) "SSDA", where the Sequential Similarity Detection Algorithm (SSDA) [11] technique was used for feature matching in the compressed feature space instead of performing skipping as the proposed method, and with (ii) "no bounding", which was the same as the proposed method except for the distance bounding was not performed. Both skipping and bounding proved to be effective.

5. CONCLUSIONS

We have proposed a very quick search algorithm for multimedia signals based on feature compression and distance bounding. In our experimentation, the algorithm was over ten times faster than existing fast methods in terms of CPU time. Due to space limitations, we mainly discussed a video example in this paper. The application to the audio signal will be reported in a separate paper. Future work will include further investigation of the optimal decision of the dimensionality of compressed feature space and the number of maps.

Acknowledgment : The authors thank Drs. Ken-ichiro Ishii, Noboru Sugamura and Norihiro Hagita for their help and encouragement.

6. REFERENCES

[1] H. D. Wactler: "Infomedia — Search and Summarization in the Video Medium", *Proc. of Imagina 2000 Conference*, 1983.

[2] Y. Rui et al.: "Content-Based Image Retrievel with Relevance Feedback in MARS", *Proc. of ICIP 1997*, Vol.2, pp. 815-818, 1997.

[3] R. Mohan: "Video Sequence Matching", *Proc. of ICASSP98*, Vol. 6, pp. 3697-3700, 1998.

[4] K. Kashino et al.: "Time-Series Active Search for Quick Retrieval of Audio and Video", *Proc. of ICASSP99*, Vol. 6, pp. 2993-2996, 1999.

[5] E.Oja: *Subspace method of pattern recognition*, Research Studies Press Ltd., 1983.

[6] C. Agarwal et al.: "Fast Algorithm for Project Clustering", *Proc. of ACM SIGMOD 99*, 1999.

[7] K. Wang et al.: "Selective Feature Extraction via Signal Decomposition", *IEEE Signal Processing Letters*, Vol. 4, No. 1, pp. 8-11, 1997.

[8] K. V. R. Kanth et al.: "Dimensionality Reduction for Similarity Searching in Dynamic Databases", *Proc. of ACM SIGMOD 98*, pp. 166-176, 1998.

[9] E. Keogh et al.: "Dimensionarity Reduction for Fast Similarity Search in Large Time Series Databases", *Journal of KAIS*, to appear.

[10] E. Keogh et al.: "Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases", *Proc. of 2001 ACM SIGMOD on Management of Data*, 2001.

[11] D. I. Barnea et al.: "A Class of Algorithms for Fast Digital Image Registration", *IEEE Trans. on Computer*, Vol. C-21, pp. 179-186, 1972.

[12] M. Sugiyama: "Fast Segment Search Algorithms", *Technical Report of IEICE*, SP98-141, pp. 39-45, 1999 (in Japanese).