

CONTENT-BASED AUDIO RETRIEVAL USING A GENERALIZED ALGORITHM

PUNPITI PIAMSA-NGA
S. R. SUBRAMANYA
NIKITAS A. ALEXANDRIDIS
SANAN SRAKAEW
GEORGE BLANKENSHIP

*Department of Electrical Engineering and Computer Science
George Washington University
Washington D.C. 20052 USA
e-mail: {punpiti,subra,alexan,srakaew,blankeng} @seas.gwu.edu*

G. PAPAKONSTANTINOU
P. TSANAKAS
S. TZAFESTAS

*Department of Electrical and Computer Engineering
National Technology University of Athens
Athens Greece*

1. Introduction

Content-based indexing of audio (and multimedia) data has become more important since conventional databases cannot provide the necessary efficiency and performance [1,2]. However, there are three main difficult problems. First, the content of audio data is subjective information; it is hard to give the descriptions in words. The recognition of data content requires prior knowledge and special techniques in Signal Processing and Pattern Recognition, which usually require long computing time. Second, since several audio features can be used as indices [3] (such as pitch, amplitude, and frequency), a method or processing technique designed and developed for one feature may not be appropriate for another. Third, the extremely large data size and the use of a similarity search require extensive computation. Similarity matching is based upon the computation of the distance between a query and each record in the database; the best match is in the data set with the smallest distances. To solve these three problems, we

use a histogram-based feature model to represent subjective features [4], a unified model [5] to represent the data structures of the multimedia data, and a fast, generalized comparison algorithm to reduce the retrieval time.

To reduce the retrieval time, we proposed two techniques in our previous reports, one is a weighted cascade (pipeline) for multiple-feature querying, and the other is the use of a parallel search algorithm [3,4]. The algorithm yielded lower retrieval time and acceptable results. However, if there is only one feature and parallel processing is not available, the retrieval time will be very long. To reduce retrieval time when there is only one feature or parallel processing is not possible, a new fast comparison algorithm is required

In this chapter, we propose a new generalized algorithm for audio (and multimedia) data retrieval using an unrestricted-format query. The algorithm is called the “Virtual-Node (VN)” algorithm. The VN algorithm is adapted from our “Partial-Matching [6]. We have developed and tested the PM algorithm successfully on image retrieval. Here, we extend our previous work by applying the same algorithm on audio databases and then improve the PM algorithm to the new VN algorithm. The results of the VN algorithm are perceptibly more acceptable, and have a smaller retrieval time, than the previous PM algorithm.

The rest of this chapter is organized as follows. Section 2 gives an overview of similarity search types. Section 3 describes the histogram-based binary tree for audio features. Section 4 presents an algorithm for restricted-format queries. Section 5 presents the details of the PM and VN algorithms. Experimental results are given in Section 6, followed by a summary and future directions.

2. Retrieval algorithm

In this Section, the general algorithm for audio (and multimedia) retrieval by content is introduced. Section 2.1 reviews similarity searching. A classification of queries is given in Section 2.2.

2.1 SIMILARITY SEARCH

Exact keyword matching database systems are inefficient and inappropriate for audio (and multimedia) data since the search is based upon the use of a static subjective summary of the data rather than the data itself [1]. Similarity searching is a more appropriate approach. Just as in the exact match approach, the crucial issues are the building of an index table and the retrieval algorithm. The basis of the algorithm is a function that measures the distance between the query and each of the multimedia objects and reports the best matching data, which are those that have the smallest distances to the query.

Two main concerns in the field of similarity searching are the enormous computation requirement and efficiency of the index. Because the similarity search is based on finding the minimal distances between the query and each of the records, the computation time grows with the size of the database. Retrieval from a multimedia

database is based upon similarity searches of the index space. A multimedia index entry should contain the salient features, which have been extracted from the raw data, and encompass the spatio-temporal relationships. The usefulness of an index entry is a function of the computation time used to extract the features that define the entry, the space required to store the entry, and the ease with which the entity identifies the data.

2.2 CLASSIFICATION OF SEARCH ALGORITHMS

There are two types of multimedia data queries: query-by-example (QBE) and query-by-content (QBC). The QBE approach searches for data that are similar to the query. An example of QBE query is “Which records in an audio database are similar to an audio query clip?” The QBC takes a qualitative description of data as query. The example of QBC query is “Which audio clips in an audio database this chapter, all query types are QBE. QBE can be divided into two subtypes by application approaches: search-by-restricted-format (SBRF) and search-by-unrestricted-format (SBUF). A search-by-restricted query looks for data that is similar to the query globally, without regard to the scale of the query. In other words, all records in a database must be resampled into a uniform sampling rate. An example of query using SBRF is “Which are the data items that are similar to the given query, as a whole?” In contrast, a query using SBUF is searches for data with similarities to the query. An example of SBUF query is “List all items that have ‘portions’ that are similar to the

3. An audio feature

In this section, a histogram-based k-tree feature is used as an index for audio data. A summary description of the k-tree model is given in Section 3.1 and the construction of the binary tree of a histogram-based feature is explained in Section 3.2.

3.1 THEK-TREE MODEL

A k-tree model [5] is used to unify the data characteristics of multimedia data. A k-tree is a directed graph; each node has 2^k incoming edges and one outgoing edge with a balanced structure. A *binary* k-tree is used for 1-dimensional data; a *quadtree* for 2-dimension data; and an *octtree* for 3-dimensional data. There are three main benefits for exploiting a k-tree. First, the spatio-temporal information of the data is embedded into the tree structure. It reduces the time to compute distances between two nodes when spatio-temporal relationships are required in a query. Second, a k-tree can exploit multiresolution processing by computing small, global information first and then large, local information when more accurate resolution is required. Third, the complexity of the data structure affects only the degree of the tree. Consequently, an algorithm for a particular type of feature can be reused for a feature of other media types. The comparison of two features is based upon the distance between the histograms that define the features.

Since a histogram-based feature is used as index, we have investigated the effects of using different existing histogram distance functions. Three distance functions were selected to determine a distance between two histograms: Euclidean (d_E), Histogram Intersection (d_I) [6], and Histogram Quadratic (d_Q) [8] distance functions, which are defined below:

$$d_E(h, g) = \sum_{m=0}^{M-1} (h[m] - g[m])^2 \quad (1)$$

$$d_I(h, g) = 1 - \frac{\sum_{m=0}^{M-1} \min(h[m], g[m])}{\min(\sum_{m_1=0}^{M-1} h[m_1], \sum_{m_1=0}^{M-1} g[m_1])} \quad (2)$$

$$d_Q(h, g) = \sum_{m_0=0}^{M-1} \sum_{m_1=0}^{M-1} (h[m_0] - g[m_0]) \cdot a_{m_0, m_1} \cdot (h[m_1] - g[m_1]) \quad (3)$$

The h and g are M -dimensional histograms. The $h[m]$ and $g[m]$ are the frequencies of an element in bin m of histogram h and g , respectively. The *Euclidean distance* function (Eq. 1) only compares the identical bins in respective histograms. All bins contribute equally to the distance and the differences of content between two bins of histogram are not ignored. The *histogram intersection distance* function (Eq. 2) compares only the elements, which exist in the query. Note that the result from the intersection is not symmetric. The *histogram quadratic distance* function (Eq. 3) uses matrix $\{a_{m_0, m_1}\}$, where $\{a_{m_0, m_1}\} = |m_0 - m_1|$. This gives a weight to the histogram distance; bins that are closer are given less weight in the product term than those that are farther apart. In the algorithms, the histogram intersection is used. Intuitively this gives the number of samples common to both histograms.

3.2 A HISTOGRAM-BASED BINARY TREE OF AUDIO FEATURE

In this chapter, we use audio amplitude as the feature of interest and a key index for an audio database. Generally, a normalization technique is used for generating the index. The domain of a feature is reduced to a set of selected values from a universe of potential values of the feature. We assign an identification number of each element in the reduced set. When data is inserted into the system, it is converted or transformed into the selected domain then a histogram is generated. Thus, the feature is represented by a histogram of data in the selected domain. For an audio feature, the amplitude values are picked from the whole infinite universe and then mapped into a set of 256 pulse-code modulation (PCM) values.

The amplitude feature of an audio clip is organized as a binary tree ($k=2$) of histograms. A histogram is constructed from the counts of PCM values that occur in an audio clip. Since there are 256 existing PCM values, a histogram feature of an audio clip is a 256-element vector; each element i contains the number of samples with a PCM value of i . To construct the binary tree, the audio clip is divided into m segments, where $m=2^l$ and l is an integer. The histograms for each segment form the nodes of the tree. Pairs of histograms from consecutive segments are summarized into a new histogram forming a parent node. The process is repeated until a global histogram (at the root of the tree) is generated.

4. Algorithms to search by restricted-format query (SBRF)

In the search-by-restricted-format approach, the size of the data in the multimedia database and the query must be normalized to a uniform format. Normalized audio data has a similar sampling rate and a similar number of samples. Any audio data must be quantized and resampled to the normalized configuration. This is useful for applications that require the search by restricted-format query or the query, which disregards the scale of the data. The pseudocode of SBRF is given below:

Algorithm 1: Search by Restricted-format (SBRF)

```
SearchByRestrictedFormat (In Query, In FeatureOfData [n], In K-TreeLevel,  
                          Out Record (Distance [n], Data [n]))  
1) Begin  
2) NormalizedQuery = NormalizeSize (Query)  
3) FeaturesOfQuery = FeatureExtraction(NormalizedQuery)  
4) For i=0 to n do  
5)     Record.Distance[i] = DistanceSBRF (FeatureOfQuery,  
                                          FeatureOfData[i],  
                                          K-TreeLevel).  
6)     Record.Data[i] = Data[i]  
7) End for  
8) Sort(Record(Distance[n],Data[n]))  
9) End
```

Algorithm 2: DistanceSBRF

```
Out DistanceSBRF (In FeatureA, In FeatureB, In K-TreeLevel)  
1) Begin  
2) Distance=0;  
3) For i=0 to NumberOfNodesAt (K-TreeLevel) do  
4)     X = HistogramDistance(FeatureAtNode ( i, FeatureA),  
                             FeatureAtNode ( i, FeatureB))  
5)     Distance = Distance + (X * X )  
6) End for  
7) Return sqrt(Distance)  
8) End
```

5. Algorithms of search by unrestricted-formatted query (SBUF)

In the search-by-unrestricted-format (SBUF) approach, the queries are assumed to be in a continuous form. The algorithm determines the best match for the query in any portion of each of the audio clips and reports a number of close matches. The simple algorithm for searching by unrestricted-formatted query is shown in Algorithm 3 and Algorithm 4. Note that Algorithm 4 uses brute-force searching method, which has $O(n)$

time complexity. Faster algorithms have been developed to replace the Algorithm 4 and are given in Section 5.1 and 5.2.

Algorithm 3: Search by unrestricted-formatted query (SBUF)

```

SearchByUnrestrictedFormatted(In Query,In K-TreeLevelQuery,
                               In FeatureOfData[n], In K-TreeLevelData[n],
                               Out Record (Distance[n], Data[n]))

1) Begin
2) FeatureOfQuery = FeatureExtraction(Query)
3) For i=0 to n do
4)     Record.Distance[i] = DistanceSBUF(FeatureOfQuery,
                                         K-TreeLevelQuery,
                                         FeatureOfData[i],
                                         K-TreeLevelData[i])
5)     Record.Data[i] = Data[i]
6) End for
7) Sort(Record(Distance[n],Data[n]))
8) End

```

Algorithm 4: The Simple DistanceSBUF

```

Out DistanceSBUF (In FeatureA, In K-TreeLevelA,
                  In FeatureB, In K-TreeLevelB)

1) Begin
2) NumberOfNodesLevelA = NumberOfNodesAtLevel (KTreeLevelA)
3) NumberOfNodesLevelB = NumberOfNodesAtLevel (KTreeLevelB)
4) NumberOfComparisons = NumberOfNodesLevelB-
                          NumberOfNodesLevelA+1
5) Distance=0
6) For i to NumberOfComparisons do
7)     For j to NumberOfNodesLevelA do
8)         Distance = min( Distance , HistogramDistance(
                               FeatureAtNode( j, FeatureA),
                               FeatureAtNode( i+j, FeatureB))
9)     End For
10) End For
11) Return Distance
12) End

```

Two algorithms for searching using an unrestricted-formatted query have been developed, tested, and presented using the k-tree model. The algorithms are called “partial-matching” [6] and “virtual-node.” The mathematical analysis and experimental results on both algorithms show a much faster response time than using the brute-force search technique of Algorithm 4. Both methods have a common prologue derived from Algorithm 3 and Algorithm 4.

5.1 THE PARTIAL-MATCHING (PM) ALGORITHM

The Partial-Matching (PM) algorithm [6] is a generalized algorithm for searching a multimedia database by content using queries with unrestricted formats. This algorithm allows us to search for data in a multimedia database and find those entries that contain the same “object” as the query; since no scaling is performed on the query, it represents the “*unrestricted query format*” approach. This algorithm has been developed and tested for retrieval using image database. The results are very satisfactory, when measured by both retrieval time and perceptual quality. We have adapted this algorithm for an audio database. Algorithm 5 below shows the details of the partial-matching algorithm. An illustration of searching using the partial matching algorithm is depicted in Figure 1.

Algorithm 5: A partial-matching algorithm for DistanceSBUF

PartialMatchingComparison(**In** FeatureOfQuery, **In** FeatureOfData, **Out** Distance)

Step 1: Find candidates

- 1) Calculate the “feature distances” between the root of the query tree and the all nodes in data trees, which have the same height as the query tree in the database, (we assume that data size are larger than the query size.)
- 2) Select those data types for which the distances are below a threshold value. These candidates will be searched at a higher-resolution level in Step 2: below.

Step 2: Determine the position

- 1) On the candidate nodes, generate virtual nodes (nodes that locate across the boundary of real nodes)
- 2) Calculate the distances between root of the query tree and all virtual nodes around the candidate nodes.
- 3) Sort the distances from 2).
- 4) Return the minimum distances with node positions.

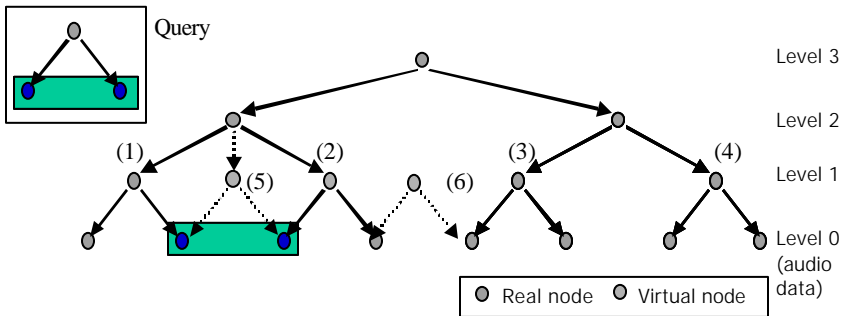


Figure 1: Illustration of searching using the partial-matching algorithm

The process begins at Step 1: of Algorithm 5 which finds the candidate matches – illustrated at node (1), (2), (3), and (4) in Figure 1. Figure 1 demonstrates the comparison between the root of the query and the nodes that have the same height as the query. Suppose we found that node (2) is a candidate. A culling of candidates that do not match (Step 2:) creates virtual nodes at node (5) and node (6). Finally, similar data to the query is found at the position of node (5).

5.2 THE VIRTUAL-NODE (VN) ALGORITHM

The Virtual-node algorithm is a newly proposed generalized algorithm for searching data using an unrestricted-format query. This algorithm is an extension of the partial-matching algorithm. It exploits the multiresolution capabilities of the hierarchical structure to eliminate some unnecessary computations. The detail is shown in the pseudo-code of Algorithm 6.

Algorithm 6: A virtual-node algorithm for DistanceSBUF

VirtualNodeComparison(In FeatureOfQuery, In FeatureOfData, Out Distance)

Step 1: Find candidates

- 1) Calculate the “feature distances” between the root of the query tree and the roots of all data trees in the database, (assume that data tree height is larger than the query height.) In our experiments, the “histogram intersection function” has been used as the “Feature distance” to determine whether the query histogram is contained within the searched multimedia database and the “histogram quadratic function” to determine whether the query histogram is similar to the data.
- 2) Select those data types for which the distances are below a threshold value. These candidates will be searched at a higher-resolution level in Step 2: below.

Step 2: Determine the position

Case a) If query’s tree aligns within the k-tree structure of data:

- 1) Find the feature distances between feature in root of query tree and nodes of data at level L_{i-1} – nodes with solid-line link in Figure 2. If the distance between a child node is equal to the distance between the query and its parent (L_i), the query may be found within that child node.
- 2) Repeat Case a) recursively on the found child node. If there is no distance at level L_{i-1} close to the distance of the parent, the query is “not aligned”. Follow Case b) below.

Case b) If the query data falls in between two or more nodes:

- 1) If no "real node" in tree (nodes connected by solid lines in Figure 2 can be a candidate, "virtual nodes" (node connected by dot lines) between two nodes have to be generated from the parts of their child nodes.

- 2) Repeat the algorithm using a new tree of virtual nodes i.e. use the algorithm within the dashed box in the Figure 2.

Case c) If height of query is equal to a node height:

- 1) Use Algorithm 1: Search by Restricted-format (SBRF) with histogram quadratic distance function to calculate the distance and then:
- 2) Return the distance.

An example of a search using the virtual node algorithm is illustrated in Figure 2. The root node of the query is compared with the root node of the data to determine if the query is part of the data in the tree under node (1) at level 3. If the result shows that the target is in the tree under node (1), the comparison between root of the query and the lower-level nodes continues using the nodes in level 2. If the result at level 2 shows that the query is under node (2) and not under node (3), then, the subtree under node (3) is ignored and the process continues with the node at level 1, node (4) and node (5). If results from both node (4) and (5) do not indicate that a match is below either, node (6) is generated dynamically and then compared with the query.

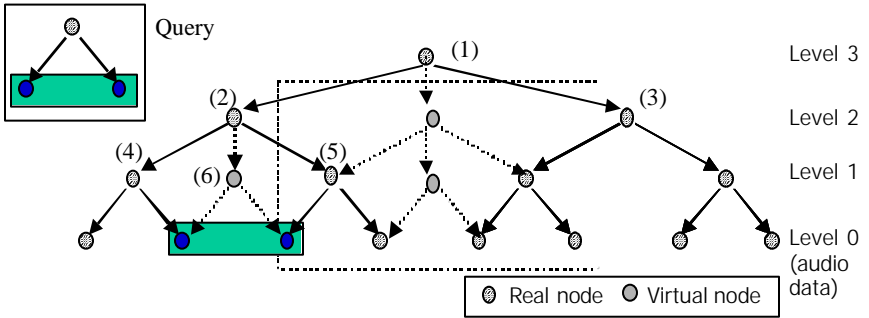


Figure 2 Illustration of example for searching by virtual-node algorithm

5.3 MATHEMATICAL ANALYSIS

Let k be a number of spatio-temporal dimensions of the data; S_d be the average size of all records in the database; and S_q be the size of query. (The size of the data is the number of leaf nodes in a query or data record tree.) The total time complexity to compare two records is given in the following discussion.

For the linear simple DistanceSBUF, the time complexity is $(\sqrt[k]{S_d} - \sqrt[k]{S_q} + 1)^k$. For the partial-matching algorithm, the time complexity is $(S_d/S_q) + 3^k$. For the virtual-node algorithm, the time complexity is $3^k \cdot \log_{2^k}(S_d/S_q)$ plus time of generating the virtual node, which is $(3^k - 2^k) \cdot \log_{2^k}(S_d/S_q)$. The DistanceSBUF algorithm has linear time complexity; it grows with S_d . The new algorithm has time

complexity $O\left(\log\left(\frac{S_d}{S_q}\right)\right)$. However, as k grows, the constant term may have the dominant effect on the processing time.

6. Experiments

6.1 COMPUTATIONAL PLATFORM

The audio database used in this experiment consists of over hundred files downloaded from Sunsite at University of North Carolina [9]. The database represents 3.5 hours of audio clips and requires about 100MB of mass storage. The average length of a clip is 10 seconds. Each clip has the same sampling rate of 8KHz. A Linux/Pentium-II machine was the computation platform. Several audio clips in the database were selected as a set of queries for whole clip searches (the *whole-clip* set). A one-second portion from each clip in the *whole-clip* set was randomly selected to be used for partial clip searches (the *part-of-clip* set). During the search, the query is compared only with the data that is equal to or longer than the query.

In this experiment, the results from the *whole-clip* set was retrieved using a search-by-restricted-format query, while the *part-of-clip set* results was retrieved using an unrestricted-format query.

6.2 EXPERIMENTAL RESULTS

We have performed the experiments for searching audio data using the proposed "Virtual-node" algorithm and compared the results to those using the "Partial-matching" algorithm. The experiments focused on two primary issues: the quality of search results and the retrieval time.

The results from each query are a list of audio clip identified with distances less than a given threshold. The retrieval time, the number of candidate results, and the average position of the expected results in the candidate lists were measured and are shown in Table 1.

Table 1 shows that the retrieval time using "part of clip" is slightly more than the retrieval time using whole clips. We conclude that a search-by-unrestricted-format query (search by "part of clip") does not require a significantly greater processing time than a search-by-restricted-format query (search by "whole clip".)

Table 1 also shows better metrics from the virtual node: the average retrieval time, the number of candidate results, and the position of expected results in the list of candidates.

Table 1 Comparisons between the “partial matching” and the “virtual node” algorithms

Set of queries	PARTIAL MATCHING		VIRTUAL NODE	
	Part of clip	Whole clip	Part of clip	Whole clip
Average query length (sec.)	1	10	1	10
Average retrieval time (sec.)	38	33	35	32
Number of candidates	218	34	22	1
Average position of target clip in candidate list	109	1	8	1

7. Summary and future directions

We have presented a generalized algorithm for content-based audio retrieval using the concept of a "virtual-node." The algorithm is an improvement over our previous "partial-matching" algorithm and also exploits the multiresolution data structure of the unified k-tree model. The experimental results of retrieval from an audio database using the virtual-node algorithm show that the retrieval time is less than, and the perceptive accuracy is better than, the results from the partial-matching algorithm. The results of a restricted-format query using this generalized virtual-node algorithm do not require a significantly longer time than the conventional restricted-format algorithm.

The unified k-tree model can be used for various types of multimedia data; we intend to extend the virtual-node algorithm for other multimedia data types, such as images and video, in the near future. Preliminary experimental results of retrieval from an image database show that the virtual-node approach improves retrieval performance.

Moreover, we are also extending the k-tree model and developing a new search algorithm to allow the searching for data that have a different resolution than that of the query, such as a different sampling rate (audio clip) and different scale (image). Currently, the preliminary results in audio retrieval exhibit a requirement for intensive computation in order to retrieve acceptable results.

8. References

- [1] V. Gudivada and V. Raghavan, “*Special issue on content-based image retrieval*” IEEE Computers, Vol. 28, No. 9, September 1995.
- [2] Z. Kemp, “*Multimedia and spatial information systems,*” IEEE Multimedia, Vol. 2, No. 4, 1995.

- [3] E. Wold et al., "Content-based classification, search and retrieval of audio data," IEEE Multimedia, 1996.
- [4] P. Piamsa-nga, N. A. Alexandridis, S. Srakaew, G. Blankenship, G. Papakonstantinou, P. Tsanakas, and S. Tzafestas, "Multi-feature content based *International Conference on Computer Graphics and imaging*, 1998
- [5] P. Piamsa-nga, N. A. Alexandridis, G. Blankenship, G. Papakonstantinou, P. Tsanakas, and S. Tzafestas, "A unified k-tree model for multimedia retrieval," in *International Conference on Computers and their applications*, Hawaii, March 1998.
- [6] S. R. Subramanya, P. Piamsa-nga, N. A. Alexandridis, and A. Youssef, "A *Scheme for Content-Based Image Retrievals for Unrestricted Query Formats*," International Conference on Imaging Science, Systems and Technology (CISST'98), Las Vegas, July 1998
- [7] M. J. Swain and D. H. Ballard, "Color Indexing" International Journal of Computer Vision, 7:1, 1991.
- [8] J. R. Smith, "Integrated spatial and feature image systems: retrieval, analysis, and Ph.D. Thesis, Columbia University, 1997
- [9] Sunsite at University of North Carolina, "FTP archive," available URL: <http://sunsite.unc.edu/pub/multimedia>

Paper summary

Title:

Content-based audio retrieval using a generalized algorithm

Authors:

Punpiti Piamsa-nga, S. R. Subramanya, Nikitas Alexandridis, Sanan Srakaew, and George Blankenship

Department of Electrical Engineering and Computer Science
George Washington University
Washington D.C. 20052 USA

G. Papakonstantinou, P. Tsanakas, and S. Tzafestas
Department of Electrical and Computer Engineering
National Technology University of Athens
Athens Greece

Corresponding author:

Punpiti Piamsa-nga
Department of Electrical Engineering and Computer Science
George Washington University
Washington D.C. 20052 USA
e-mail: punpiti@seas.gwu.edu

Keywords:

Content-based retrieval,
Audio retrieval,
The k-tree model,
The virtual-node algorithm

Pages:

12 pages