

[5].

In this paper, a generalized in-object search algorithm called the "Generalized Virtual-Node (GVN)" algorithm to search on a universal model representing the characteristic features of any multimedia datatypes is proposed. Types of multimedia data is classified and types of querying, and then design unified algorithms for each querying approach on the same the unified k-tree structure. The performance measurements of the algorithms by the use of audio databases are demonstrated. In general, the retrieval time using the proposed search algorithms is quite shorter than using other algorithms, such as "Partial-Matching" algorithm [3].

2 MULTI-CHANNEL MULTIDIMENSIONAL DATATYPES

Multimedia data can be viewed as the raw data or the features that categorize it. Raw multimedia data consists of data structures with diverse characteristics such as images, audio, video, and motion pictures [1]. All types of multimedia data can be recognized as multidimensional, multi-channel signals in spatio-temporal domain. Stereo audio is a one-dimensional, two-channel signal. It usually is a sequence of audio amplitude in the temporal domain. A digital audio waveform signal in a multimedia system is sampled and encoded from an analog signal. Other datatypes are recognized in the similar fashion as the audio. For example, image is a two-dimensional multi-channel signal, where data in both dimensions are spatial data and each channel can be data from different frequencies, such as red, green, blue, gray, infrared, and ultraviolet. Each element of image (pixel) represents color at the corresponding position. The resolution of an image

depends on the sampling rate and the amount of possible details required for each pixel. The processing of multimedia data has a familiar trade-off; one must select an importance ranking of data quality, storage, and computation speed.

More importantly, since any composite data is recognized as multi-channel, multidimensional signals, a query that searches across different types of media can be processed simultaneously. The example of the query across the media is “to find a president giving a speech about preventing crime.” The traditional search mechanism across image and audio data must separate the query into two subqueries, one for image and the other for audio, and then use the database JOIN operation onto the results of those two subqueries with a constraint of the time synchronization to finalize the query results. However, since all channels of data are considered to be a piece of composite data, the JOIN operations can be eliminated from the cross-media querying if the concept of multi-channel, multidimensional signal has been used.

3 AUDIO DATABASE

To create an audio database, characteristic features (contents) of audio are extracted from the data and used them as index key. The k-tree model [2] is used to unify the index structure. A k-tree is a directed graph; each node has k incoming edges and one outgoing edge with a balanced structure. The use of this k-tree provides three main benefits. First, the spatio-temporal information of the data is embedded into the tree structure; it reduces the time when a query comes with a constraint of timing or location. Second, a k-tree can exploit multiresolution processing. Third, the complexity of 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 k-tree structure is used to retain location information and the histogram is used to store the characteristics of each portion the data that corresponding to that part of the tree. This generalized model is depicted in Figure 1. First, the feature of interest is extracted by either general mathematical models or special pattern recognition methods. This process is called *feature extraction*. Second, the universe of datatype is reduced into a smaller finite set and each data in the database must be mapped to fit into this finite set. This process called *feature quantization*. For instance, for the color feature, the infinite number of colors in nature has to be reduced into finite numbers. Third, if size of a data item is not power of two, dummy data will be virtually added into each dimension to make the size of each dimension to be the lowest power-of-two number that is greater than the largest dimension of data. Then, a histogram is constructed and the k-tree of that feature’s histogram is formed.

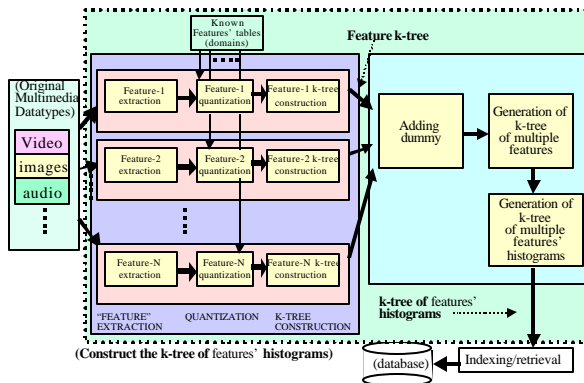


Figure 1: Generalized indexing/retrieving model

4 IN-CLIP SEARCH ALGORITHM

The basic idea of the algorithm is to eliminate unnecessary branches in the k-tree structure, where each branch has information of images in the corresponding area. Color histogram is used as the interest feature. The "histogram intersection" distance function [5] is used to determine what branches of tree the more precisely search should continue. The Algorithm 1 is the details of the limited version of the Virtual-node (VN) algorithm [3], where the each dimension of data or query must have same power-of-two size (Data size and query size is not necessary to be equal.) The Algorithm 2 is the generalized version of Algorithm 1. The Algorithm 2 does not need that the sizes of data and query have to be power of two. Figure 2 shows the examples of virtual nodes for one-dimensional data. In the details of the algorithms, the illustration can be explained easily by using binary tree instead of quad tree; however, the concept of the algorithm is the same if the algorithms are applied into other types of multimedia data.

Algorithm 1: The Virtual-Node (VN) in-object search algorithm

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

- A.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 – of the stored data. If the distance between a child node is equal to that between the query and its parent (L_i), the query may be found within that child node.
- A.2 Repeat Case A) recursively on this child node. If there is no distance at level L_{i-1} close to the distance to the parent, the query is “not aligned”. Follow Case B) below.

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

- B.1 If no node in k-tree (darker nodes in Figure 2) can be a candidate, virtual nodes (dot-line nodes) between two nodes have to be generated from the parts of their child nodes.
- B.2 Repeat the whole algorithm into a new tree i.e. use the whole algorithm within the dashed box in Figure 3.

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

- C.1 Use histogram distance function to calculate the distance then
- C.2 Return the distance and location.

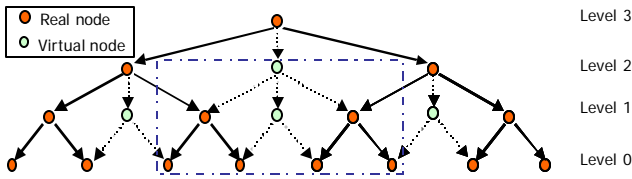


Figure 2: A virtual-node structure for one-dimensional data (k=1)

Algorithm 2: The Generalized Virtual-Node (GVN) in-clip search algorithm

```

ExtendedQuery=AddDummies (Query)
FeatureOfExtendedQuery = FeatureExtraction(ExtendedQuery)
VirtualNodeComparison (FeatureOfExtendedQuery, FeatureOfExtendedData, ROOT, distance1,
    TentativeLocation1)
IF (distance1 < threshold1) THEN BEGIN
    Find “QueryRepresentative,” the largest node in the k-tree of FeatureOfQuery, where no parts of
    dummies are included.
    VirtualNodeComparison (QueryRepresentative, FeatureOfExtendedData, TentativeLocation1,
        distance2, TentativeLocation2)
    IF (distance2 < threshold2) THEN BEGIN
        Find the final distance by calculating the distance between the query and area of data where the
        beginning of the area is at TentativeLocation2.
        Distance = distance2
        Location = TentativeLocation2
        RETURN
    END
END
END

```

An example of a search one-dimensional data using the GVN algorithm is illustrated in Figure 3. Suppose a query is a sequence of three consecutive nodes of feature, which has only two values (A and B) in its universe. A dummy node (**D**) is appended to a query’s feature. The query’s feature with a dummy is used to construct a binary tree (k=1) of histogram. Similarly, a 6-node-long feature of a search’s target is appended by two dummy nodes and then a binary tree of the searched target is constructed and shown in Figure 3 (b). All comparisons use the histogram intersection function [5] as a distance function. The search begins at (1) by comparing the root of the query (**Q1**) and the root of the searched target. Suppose the comparison result at (1) shows that the target is located at nodes under (1) then **Q1** continues compare with nodes (2) and (3). The comparison at (2) shows that the target should be under (2).

To determine more accurate precision, the query’s subtree that does not have a dummy is used as the key of the search. In Figure 3 (a), the search continues by using the query **Q2**, which is the largest subtree of the query that does not have a dummy node. The search continues at (4) and (5). It is noticeable that (5) is possibly a result, then the second largest tree (**Q3**) is used to compare at the consecutive location to (5). The comparison at (6) does not give a good result;

thus a virtual-node in Figure 3 (e) is generated and Q_2 is used as a search key again. Then a possible result is also found at (7) and Q_3 is used to compare again with the node next to results of (7). Finally, the position of the target is found.

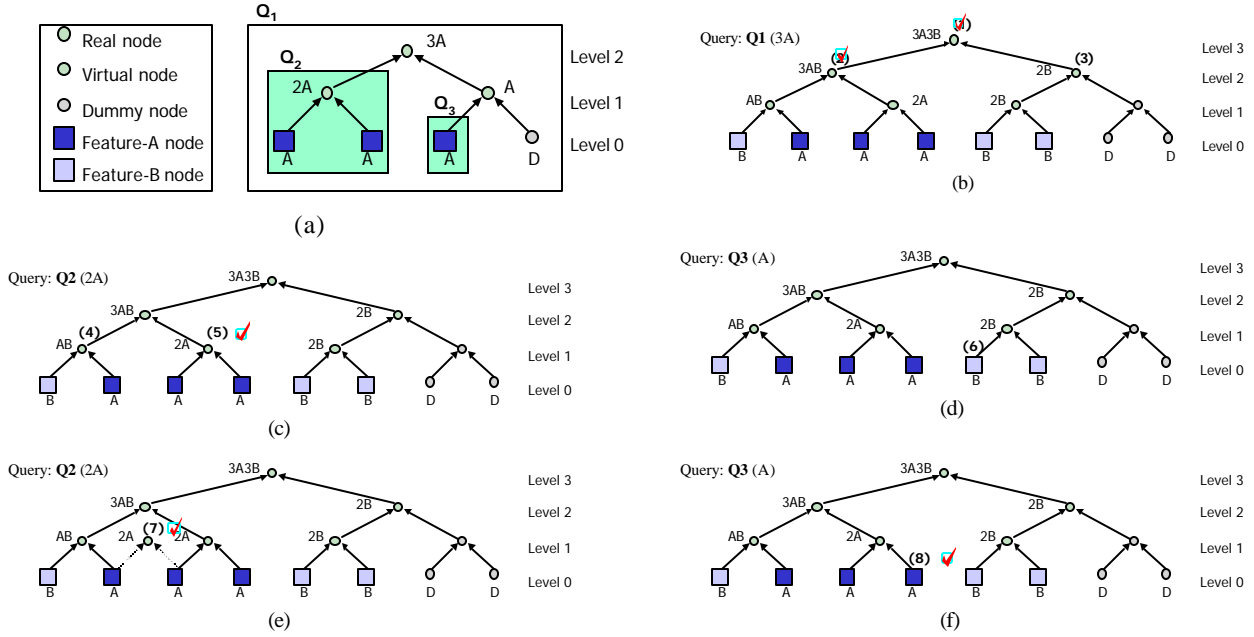


Figure 3: Illustration of searching using the generalized virtual-node algorithm

5 ALGORITHM ANALYSIS

Let k be a number of spatio-temporal dimensions of the data; S_d be the average size of data 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 time complexity to compare two records is given in the follows:

For the brute-force, 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 GVN algorithm, the time complexity is $3^k \cdot \log_2(S_d/S_q)$ plus time of generating the virtual node, which is $(3^k - 2^k) \cdot \log_2(S_d/S_q)$. The brute-force 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, a k grows, the constant term may have

more effect to the processing time.

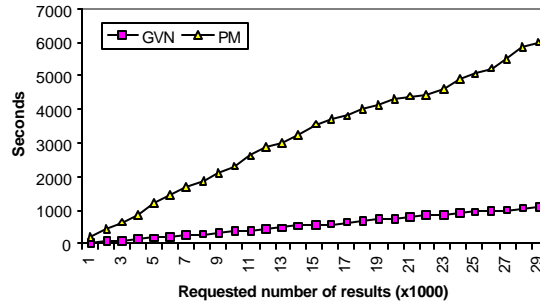
6 EXPERIMENTS

The experiments have been performed for searching audio data using the GVN algorithm. The experiments were concerned primarily with two issues: the search results and retrieval times. The audio database used in this experiment consists of several hundred files, where the total length is approximately 3.5 hours. Each audio clip has the same 8-kHz sampling rate. The results from each query are a list of audio clip identifications with their distances. Two metrics for measuring the effectiveness of the retrieval are *recall* and *precision*.

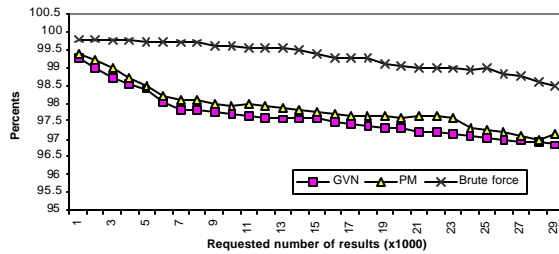
To measure the recall and precision, one-second audio clip in the database was randomly selected as a query. The selected clip is used to impose onto each clip in the audio database, where the imposed position is randomly located and then kept as a reference. All imposed clips are used to generate a tested database. The index of this database is generated by using "audio-amplitude histogram feature." The audio-amplitude histogram is constructed by counting the number of PCM of each amplitude value in a particular part of an audio clip. The k -tree is a binary tree of amplitude histograms. The GVN, brute-force, and the Partial-Matching (PM) [3] algorithms are used as the search algorithms. During the search, if the positions of the retrieved results are with the length of query (one second for this case) as far as the imposed positions, the target is assumed to be *hit*. If they are not within one-second from the imposed positions, target is assumed to be *false alarm*. If a clip is not retrieved, the target is *missed*. Recall and precision, in other words, can be defined by:

$$\text{Recall} = \frac{\text{number of hits}}{\text{number of hits} + \text{number of misses}}, \text{ and Precision} = \frac{\text{number of hits}}{\text{number of hits} + \text{number of false alarms}}$$

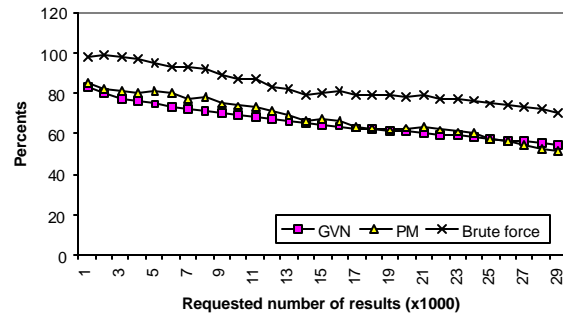
This process above is iterated several times with different target clip. All measured results are used to calculate the averages of recall, precision, and retrieval time for all GVN, brute-force, and PM algorithms. The experiment results are presented in Figure 4. Figure 4 shows that retrieval time using the GVN algorithm is much faster than using the PM algorithm, while there are little differences for recall and precision (the retrieval time of brute-force is much longer than both PM and GVN.)



(a) Retrieval time



(b) Recall



(c) Precision

Figure 4: Experimental results

7 CONCLUSION

A generalized algorithm for content-based retrieval that allows searching inside a clip of audio data (and a piece of multimedia data) is presented. This algorithm is developed onto the unified k-tree multiresolution model of multimedia data; therefore, it can be reused onto other types of data with less effort. The experimental results on audio retrieval system show that the retrieval time is faster than other algorithms, such as brute-force and Partial-matching algorithms, while both quantitative and qualitative accuracy is maintained.

REFERENCES

- [1] V. Gudivada and V. Raghavan, Special issue on content-based image retrieval systems, *IEEE Computers*, Vol. 28, No. 9, September 1995.
- [2] P. Piamsa-nga and N. A. Alexandridis, A universal model for content-based multimedia retrieval, *International Journal of Computer and Their Applications*, March 1999.
- [3] P. Piamsa-nga, S. R. Subramanya, N. A. Alexandridis, S. Srakaew, G. Blankenship, G. Papakonstantinou, P. Tsanakas, and S. Tzafestas, Content-based audio retrieval using a generalized algorithm, *Advances in Intelligent Systems: Concepts, Tools, and Applications*, Kluwer Academic, 1998.
- [4] J. R. Smith, Integrated spatial and feature image systems: retrieval, analysis, and compression, *Ph.D. Thesis*, Columbia University.

- [5] 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, Nevada, July 1998.
- [6] S. R. Subramanya, R. Simha, B. Narahari, and A. Youssef, Transform-based indexing of audio Data for multimedia databases, *International Conference on Multimedia Computing System*, Ottawa, Ontario, Canada, June 3-6, 1997.
- [7] M. J. Swain and D. H. Ballard, Color Indexing, *International Journal of Computer Vision*, 7:1, 1991.