INDEXING BOTH CONTENT AND CONCEPT FOR HIGH-DIMENSIONAL MULTIMEDIA DATA

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SERDAR ARSLAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF PHILOSOPHY OF DOCTORATE
IN
COMPUTER ENGINEERING

JUNE 2018

Approval of the thesis:

INDEXING BOTH CONTENT AND CONCEPT FOR HIGH-DIMENSIONAL MULTIMEDIA DATA

submitted by **SERDAR ARSLAN** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of Natural and Applied Sciences _____

Prof. Dr. Halit Oğuztüzün
Head of Department, Computer Engineering _____

Prof. Dr. Adnan Yazıcı
Supervisor, Computer Engineering Dept., METU _____

Examining Committee Members:

Prof. Dr. Ali Doğru
Computer Engineering Dept., METU _____

Prof. Dr. Adnan Yazıcı
Computer Engineering Dept., METU _____

Prof. Dr. İ. Hakkı Toroslu
Computer Engineering Dept., METU _____

Prof. Dr. Ahmet Coşar
Computer Eng. Dept., Unv. Of Turkish Aeronautical Association _____

Assoc. Prof. Dr. Murat Koyuncu
Dept. of Information Systems, Atilim University _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Serdar ARSLAN

Signature:

ABSTRACT

INDEXING BOTH CONTENT AND CONCEPT FOR HIGH-DIMENSIONAL MULTIMEDIA DATA

Arslan, Serdar
Ph.D., Department of Computer Engineering

Supervisor : Prof. Dr. Adnan Yazıcı

June 2018, 100 Pages

While understanding the semantic meaning of multimedia content is immediate for humans, it's far from immediate for a computer. This problem is commonly known as the semantic gap which is difference between human perception of multimedia object and extracted low-level features and it is one of the main problems in multimedia retrieval. Thus, in order to achieve better retrieval performance, low-level content features should be combined with semantic features in an efficient way. Another critical task in this domain is efficient similarity search of multimedia object in large collections. According to various studies in the literature, using query by content and concept approaches together may not only enhance performance, but also functionality of the overall system. In this study, we focus on the retrieval process of multimedia data by combining semantic information with the content of the data in order to try to solve the semantic gap problem in an efficient way. The low-level content features are extracted and mapped from high-dimesional space into low-dimensional space by using a fast dimension reduction algorithm. Thus, we have showed that our approach can reduce the retrieval problem to a spatial-indexing task

and accuracy of the retrieval performed in low- dimensional space is shown to be comparable to that of the retrieval performed in the original space. High-level concept descriptors are combined with these low-level content descriptors as a new dimension and indexed together in a single structure. We also propose another index structure which uses spatial indexing method for low-level features in order to show the effectiveness of our novel approach and we proved that our study has performance enhancement in query response time of retrieving big-sized multimedia objects since it indexes content and conceptual data together for fast retrieval.

Keywords: Content-Based Retrieval, Multimedia, Multidimensional Data Access, Multidimensional Scaling.

ÖZ

ÇOK BOYUTLU ÇOKLUORTAM VERİ ERİŞİMİ İÇİN İÇERİK VE ANLAM DİZİNLEME

Arslan, Serdar
Doktora, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi : Prof. Dr. Adnan Yazıcı

Haziran 2018, 100 Sayfa

İnsanlar için çokboyutlu nesnelerinin içeriğinin algılanması cok kolay ve hızlı bir şekilde olabilirken bu süreç bilgisayarlar için daha zordur. Bu problem genellikle çokboyutlu nesnelerinin anlamsal özellikleri ile matematiksel özellikleri arasında oluşan anlamsal boşluk olarak adlandırılır. Dolayısıyla çokboyutlu nesneleri üzerinde etkin bir sorgulama yöntemi geliştirmek için bu iki farklı tipteki özelliklerin uygun bir şekilde birleştirilebilmesi gerekmektedir. Büyük boyutlu verilerde bilgiye erişim ve sorgulama noktasında etkin bir benzerlik araması önemli bir noktadır. Literaturdeki çalışmalarda görülmüştür ki sorguların örnek nesne, yazı ya da içerik verilerinin birlikte kullanılarak yapılması sistem performansını arttıran bir unsurdur. Hesaplamaların karmaşıklığı ve sistemin kullanılabilirliği diğer önemli konulardır. Bu çalışmada çokboyutlu verilerin etkin bir şekilde sorgulanması ve getirilmesi noktasında anlamsal boşluk olarak adlandırılan genel problemin giderilmesine yönelik etkin bir yöntem uygulanmıştır. Ayrıca sorguların hızlı bir şekilde sonuçlanması için iki farklı dizin yapısı kullanılmıştır ve bu yapılar anlamsal ve

matematiksel özellikleri bir arada kullanmaktadır. Bu yaklaşım ile veriye erişme problemi etkin bir boyut azaltma yöntemi kullanılarak az boyutlu evrende dizinleme noktasına evrilmiş ve sorgulamalar daha az boyutlu evrende yapılarak orijinal cok boyutlu evrenlerine gore daha hızlı bir şekilde yapılmıştır.

Anahtar Kelimeler: İçerik-tabanlı Erişim, Çokluortam, Çokboyutlu Veri Erişimi, Çokboyutlu indirgeme.

I dedicate this thesis to

my family,

my wife, Fatma,

and my daughters Deren and Ekin.

ACKNOWLEDGEMENTS

First and foremost I want to thank my supervisor Prof. Dr. Adnan Yazıcı for all the support and encouragement he gave me. His guidance and positive approach throughout this time period made possible to complete this work.

Additionally, I am grateful to Prof. Dr. I. Hakki Toroslu, and Assoc. Prof. Dr. Murat Koyuncu for their valuable guidance, support and advices that steered me in the right the direction whenever I needed.

I thank to the members of the thesis jury, Prof. Dr. Ahmet Coşar and Prof. Dr. Ali Doğru for reviewing and evaluating my thesis.

I am indebted to my family for all their support and self-sacrifice on my behalf.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Efficient retrieval of multimedia data is a popular research topic since the number of applications in the digital technologies is increasing exponentially. Classical retrieval techniques for this kind of data are insufficent in terms of retrieavl time and search accuracy because multimedia data has different features than any other data, such as complexity, high number of dimensions etc.

Multimedia retrieval has been a very active research topic for decades, and has two major research areas: database management and computer vision [1] [2] [3] and it can be defined as the process of searching for complex objects in a multimedia database. Multimedia retrieval techniques can be classified into three categories: text-based retrieval (TBR), content-based retrieval (CBR), and semantic-based retrieval (SBR) as shown Figure 1-1.

TBR is used to define the objects in the database with annotations, keywords, or descriptions manually. In TBR, The multimedia objects are described for both contents and other metadata of the object such as: file name, format, and size. Then, the user searches the system by formulating queries by using same text attributes in order to retrieve all objects that are satisfying these annotations.

CBR is used for retrieving multimedia objects from a database according to content descriptors of the multimedia data and it has widely been used and also active research area. The content desciptors of a multimedia object is commonly the set of low level features (color, shape, texture… etc.).

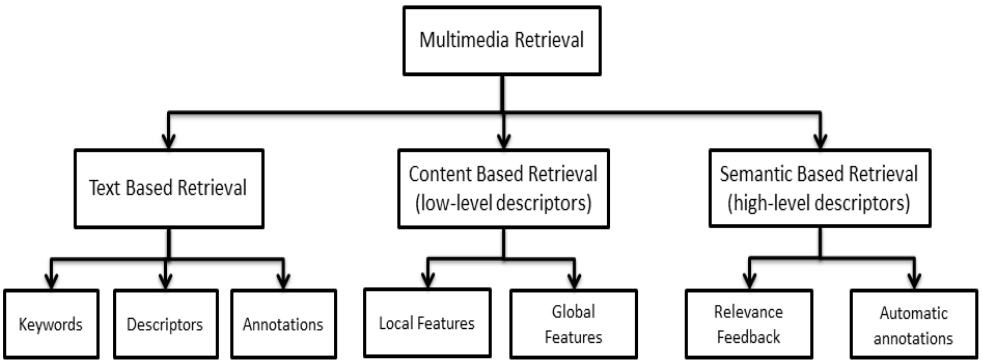

Figure 1-1: Multimedia Retrieval categories.

A typical CBIR has two main parts; feature extraction phase which is mostly done in an offline stage and retrieval phase which can be done offline or online. In off-line feature extraction stage, the system extracts features of each object in the database automatically and stores them in a features database. In retrieval stage, user defines a query by example and the features of the query are extracted similar to the offline stage and then these features are represented for (dis)similarity evaluation. The similarity is measured between the feature vector of the query object and the feature vectors of the objects in the database by using an appropriate distance funciton. After this step, an indexing scheme is used to retrieve desired objects from database in an efficient way. Thus, efficient multidimensional indexing techniques has to be applied to the retrieval process in order to make the multimedia retrieval scalable to large databases. In the literatüre, there are two main categories for multidimensional data indexing:

- Spatial Access Methods (SAM) [4] [5] index the multidimensional space defined by data feature vectors. These index structures clusters data according to their vectors and use a tree data structure with the data nodes in the leaves of the tree [6].

- Point Access Methods (PAM) try to index objects represented as points in a multidimensional space. In PAMs, the points in the database are accessed by either hashing or tree-based structures [7].

One of the major problems in CBR is reducing the *semantic gap* between human understanding and low-level features of multimedia data. While human perception of the multimedia objects' content is described by high-level semantic descriptors, the computer has different and low-level understanding of the same content. This problem is commonly referred to as the semantic gap. This gap between low-level features and high-level semantics has been one of the major problems to better retrieval performance. So SBR tries to solve this problem and tries to identify conceptual informations by using the multimedia objects' low level features. Similar to low-level features in CBR, SBR uses high-level features of objects and extracts these semantics description of objects and stores them in a database. Multimedia retrieval can be queried based on the high-level semantic informations.

Thus, in order to achieve better retrieval performance for a general retrieval system, low-level visual features should be combined with semantic features. In addition to enhancing query result accuracy, using combined approach has search time advantage. Since multimedia objects are big-sized units and retrieving a multimedia object is time consuming process, narrowing the search space and retrieving less objects directly affects overall system performance.

In addition to the above given problems, efficiency considerations like computational complexity and usability are the other important challenge in multimedia data retrieval.

1.1. Contributions of the Dissertation

The thesis study is started with a literature survey, firstly searching and analayzing the high-dimensional indexing literature. After a literature survey, we focus on designing a general information retrieval system.

First of all representation of multimedia data and feature extraction process are analyzed. In order to represent low-level features we have used several MPEG-7 content descriptors. After extraction of these low-level features, in order to represent content of the multimedia data (visual and/or audio descriptors) efficiently, similarity between these data are carried out by using some feature-based and image-based comparison methods. In order to show the effect of the comparison method, we compared the retrieval accuracy and time values of these approaches in the original high-dimensional space and also in the low-dimensional space which is mapped by using a multidimensional scaling technique.

Multidimensional scaling (MDS) is a collection of statistical techniques that attempt to evaluate some set of patterns which uses similarity matrix, into a low-dimensional space and tries to preserve their original pairwise interrelationships as closely as possible. Classical MDS has many appealing fetaures but when the number of points increases the efficiency of this algorithm decreases. Since the number of applications in the multimedia technology is increasing recently, the number of points and dimensions of these points are also increases. Thus, this limitation has become more critical and MDS algorithm should be scalable for these kinds of high-dimensioanl spaces and large databases. In order to solve scaling problem, we have adapted a very fast landmark-based MDS technique [8] to the multimedia data retrieval process.

We have showed that multi-dimensional scaling can reduce the retrieval problem to a spatial-indexing task, where queries can be performed orders of magnitude faster than distance based indexing methods. The accuracy of the retrieval performed in low dimensional space is shown to be comparable to that of the retrieval performed in the original space. Moreover using landmark-based MDS approach with feature-based comparison methods outperforms whole image based comparison methods in terms of query accuracy and retrieval time.

The next contribution of our study is that we focus on similarity measurement and retrieval process of multimedia data by combining semantic information (high-level

descriptors) with the content of the data (low-level descriptors) in order to try to solve the semantic gap problem in an efficient way. We have developed two different access methods in order to index both high-level and low-level descriptors together so that these high and low-level descriptors are accessed by using the same index structure.

For both access methods, a fuzzy object oriented indexing structure called FOOD-Index [9] is adapted in order to retrieve multimedia objects based on their semantic information. In the first access method, we also use a spatial indexing technique, called X-Tree [10], for content descriptors. For the second access method, a clustering based method is integrated with the FOOD-index for content descriptors. The performance comparison of these methods is accomplished by using query times and retrieval accuracy as the main criteria.

1.2. Organization of the Dissertation

The rest of this dissertation is organized as follows: Chapter 1 gives a review of the literature that is related to our work. Chapter 2 presents multidimensional scaling and similarity measurement background and explores some indexing techniques. Chapter 3 and Chapter 4 give details about our proposed methods for accessing content and concept in a single structure. Chapter 5 is about performance experiments and their results. Lastly, Chapter 6 states the conclusions and presents the details about the future works.

CHAPTER 2

BACKGROUND

In this chapter, we present a general backgorund of access structures for multimedia data and also brief description of some multimedia retrieval systems.

2.1. Background

Improvements in the multimedia technologies and the common availability of imaging devices have generated a large amount of multimedia data in scientific, medical, and social applications. This rapid growth of multimedia data has generated the need for new methods that provide efficient content-based retrieval (CBR). In response to this need, a number of content-based retrieval systems have been developed, some of which are listed in Figure 2-1. As demonstrated in the figure, retrieval systems can be divided broadly into two categories: feature-based and annotation-based systems. In the former category, low-level features (and/or objects) extracted from multimedia data are used [11] [12] to compare and retrieve data; whereas in the latter category, text based semantic information is used to characterize the multimedia data and form the basis for the retrieval process. Annotation-based retrieval systems have traditionally relied on manual annotation of images/videos and retrieval by search of keywords within the annotation database. Several image segmentation methods have been proposed to perform automated segmentation and association of these objects to a pre-defined vocabulary [13] [14]. The feature-based retrieval systems allow identification of similar multimedia objects using low-level features [1] [2] [15] [16]. While most of these systems use tree-based or cluster-

based data access structures, other access methods have also been applied, such as hashing and dimension-reduction techniques.

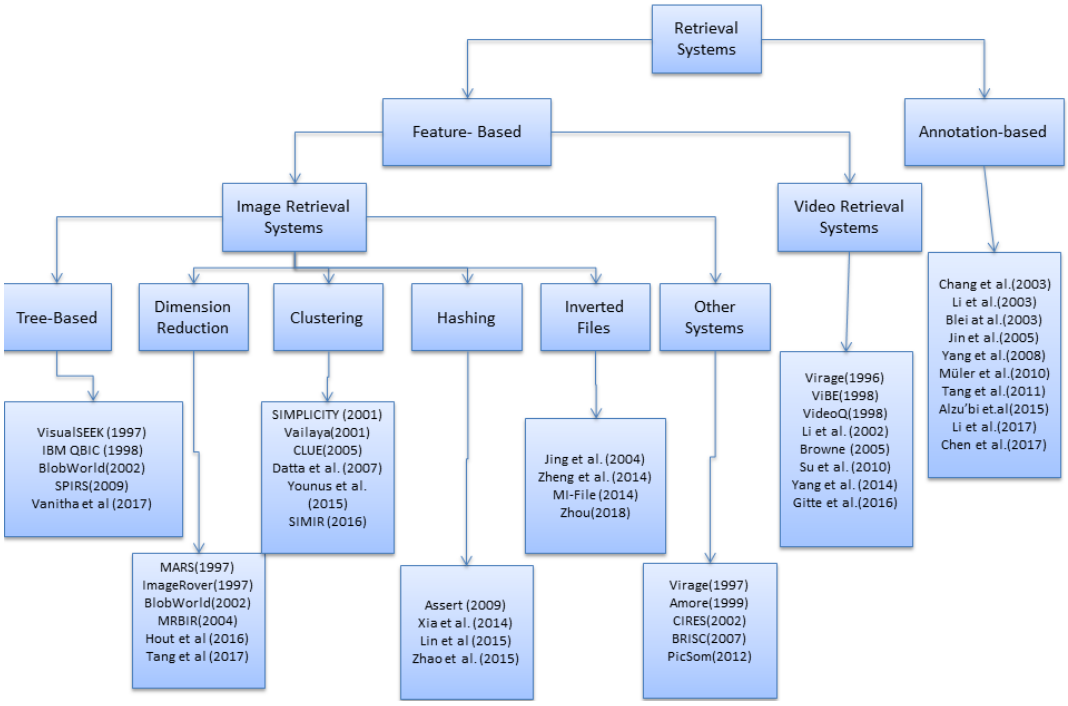

Figure 2-1 Retrieval systems proposed for accessing multimedia data.

In addition to the efforts to provide efficient and accurate multimedia data retrieval, there are also several high-level systems that attempt to address the problem in a different manner, via storage and management of multimedia data as they are perceived by humans [17] [18]. These high-level systems define a data model to represent the multimedia objects and temporal relations among them. They also define appropriate query languages to facilitate access to multidimensional data.

In each of these proposed CBR systems, retrieval performance becomes a critical factor that determines the scaling of their application to real-life databases. Thus, several multidimensional access methods have been proposed to meet performance considerations of information retrieval in high dimensional data spaces [6] [7].

Spatial Access Methods (SAM) [4] [5] index the multidimensional space defined by data feature vectors. These index structures are based on a tree data structure with the data nodes in the leaves of the tree and a cluster hierarchy built on top [6]. SAMs partition either the multidimensional data to be indexed or its underlying data space. The main drawback of spatial-access methods is the drastic decrease in the retrieval performance as the dimensionality of the feature vectors of data objects increases, a phenomenon known as the *curse of dimensionality*. The SAMs are generally outperformed by simple sequential scan when the number of data dimensions exceeds a certain threshold [6].

Point Access Methods (PAM) try to index objects represented as points in a multidimensional space. In PAMs, the points in the database are organized in a number of buckets. These buckets correspond to some subspace of the universe and are accessed by either hashing or tree-based structures [6] [7]. The Grid File [19] is the most commonly known and used index structure for hash-based PAM. On the other hand, there are several tree-based PAMs such as k-d-B Tree [20] [21], LSD-Tree [22], and hB-Tree [23]. Unlike SAM, in tree-based PAMs, the regions are at the same level of the tree and they are mutually disjoint.

In order to alleviate the curse of dimensionality and remove redundant features from the data, Dimension Reduction techniques can be used to pre-process the data and map it to a low-dimensional space. One of the efficient low-dimensional index structures such as B+-Tree can then be used for indexing in this low dimensional space. Such dimension reduction approaches include KPYR [24], Pyramid-Technique [25], iDistance [26], EHD-Tree [27], and Hilbert space-filling curve map [28]. The main drawback of dimension reduction approaches is the decrease in query accuracy since the low dimensional representations do not accurately represent the multimedia data. The specific requirements determine the desirable accuracy vs. speed trade-off, usually parameterized by the number of dimensions in the target space.

Data Approximation Structures such as VA-File [29] , OVA-File [30], VQ-Index [31], BID/BID+ [32], Hierarchical Bitmap Index [33], and BitMatrix [34] [35] construct a vector of approximations that is significantly smaller than the original data. Using these approximations, some of the indexed objects are pruned to reduce the number of distance computations that need to be performed during similarity queries. This type of access structures provide approximate k-nearest neighbor (k-NN) search that again present a tradeoff between the retrieval accuracy and query response time.

Unlike the spatial, point, and approximation access methods that operate in a feature-vector space, the Metric Access Methods (MAMs), such as M-Tree [36], Slim-Tree [37], MRKD-tree [38], and Hierarchical Cellular Tree [39], operate using the relative distances among objects. Whereas other methods partition the *space*, MAMs partition the *data*. As such, MAMs are effective in handling high-dimensional or non-vectoral data, provided that a function satisfying the metric constraints can be defined to compare data objects [40]. Similar to SAMs, MAMs utilize a tree data structure to organize the data into clusters. The critical issue in metric-trees is the degree of overlap between the tree nodes since the overlap between tree nodes directly affects the retrieval performance as in SAMs.

As mentioned above, most work in literature on content-based video retrieval relies on low-level global descriptors such as color, texture, and shape. These descriptors are represented as features and as most of them are extracted directly from digital representations of images/objects of video in the database. However, to a user, the similarity between videos/objects is often high-level or semantic. Thus a concept-based approach is necessary and in concept-based retrieval mechanism, semantic features which mean any mid- or high-level feature about concept of multimedia data are used to satisfy the user needs. The use of higher level semantic descriptors such as cars, objects etc. in order to make multimedia data retrieval efficient, has become popular in TRECVID [41]. However, extracting and evaluating of these high level descriptors are very difficult process.

TREC VID 2004	Access Mechanism	Novice	Expert
Shots Judged Correct	Text query	95%	78%
	Image query	5%	16%
	Concept	0%	6%

Figure 2-2 Percentages of different query types used by each kind of user.

Searching for multimedia data based on only one of low or high-level descriptors cannot be accurate. Since semantic meaning of an object (i.e. car) in a video can be more meaningful for the user, it cannot be used to retrieve a special case of that object (i.e. red car). Moreover, semantic features can be used with low-level features or within object based approaches. There is a study in TRECVID 2004 which examines and analysis user's needs and responses for different type of queries (i.e. text based, low-level feature based, semantic based). And from the results in Figure 2-2, text based querying is used mostly by inexperienced users and retrieves mostly inaccurate results. It has another disadvantage, since text annotation is not automatic process. Another querying method, feature-based one, can be used by experienced users mostly. But concept based querying is the least used one by all types of users.

CHAPTER 3

MULTIDIMENSIONAL ACCESS METHODS

The accuracy of similarity-based retrieval is especially sensitive to the selection of the data comparison function that evaluates the level of similarity between two multimedia objects. In this work, we used both feature based and image-registration based comparison functions. In the former technique, four low-level MPEG-7 feature descriptors [42] are extracted and used: two color descriptors, Dominant Color (DC) and Color Layout (CL); one texture descriptor, Edge Histogram (EH); and one shape descriptor, Region Shape (RS). MPEG-7 [42] was introduced as a standard for representing the audio-visual content.

MPEG-7 focuses on the description of multimedia content and does not standardize the way these descriptions are obtained or how to use them, but only standardizes the descriptions and the way of structuring them. In [43], MPEG-7 visual descriptors are analyzed from the statistical point of view, using mean and variance of the descriptor elements, and cluster and factor analysis. It is shown that the best descriptor combination is Color Layout, Dominant Color, Edge Histogram, and Texture Browsing. The other descriptors are highly dependent on these ``best'' descriptors.

Euclidean Distance is used as the similarity function for these features and the weights of the distance function are determined by either a constant weighting (CW) of each feature or by dynamically adapting the weights using the Ordered Weighting Averaging (OWA) method [44]. Moreover, Earth Mover's Distance [45] is also used as another similarity measure in order to compare with Euclidean Distance. In the latter similarity measurement technique, the whole image is used to compute the degree of similarity of the query image with those from our database. For this

purpose, we have used three different image-registration based similarity measurement methods, Cross Correlation (CC) [46], Mutual Information (MI) [46], Spatial Color Variance (SCV) [47], and Scale Invariant Feature Transform (SIFT). CC and MI are commonly used as the preferred image based matching techniques, especially in image registration applications in biological and Geographic Information System (GIS) domains [46]. CC and MI use a feature matching process without segmenting salient objects in the image [46]. SCV is a histogram based comparison algorithm for colored images, and uses spatial and color information together. SIFT features have successfully been used for the object recognition, robotic mapping, and 3D modeling. The effect of these similarity measurement techniques on the retrieval process is analyzed in terms of response time and accuracy for the access methods studied in this work.

Figure 3-1: Sample images from different categories of the Corel Dataset used in this study.

The performance of the multidimensional access structures is evaluated using three different image data sets. Corel data set [48] which has been used in a number of studies to demonstrate the performance of CBR systems and contains different categories of images with 100 images in each category. This database contains ten different categories; example images from each category are shown in Figure 3-1. ImageNet data set [49] has 4000 images and contains different categories such as car, basketball, football, tennis, bird, furniture, food and random images. The video

image data set [50] has 3142 images and is extracted from news videos and contains miscellaneous images including news, sport, accident etc. The retrieval accuracy is measured in reference to the image classifications defined in the data set. A Sequential Scan method that compares the query with every single database object is also included in the comparison to provide a base line for time and accuracy results.

3.1. Similarity measurement functions

In this work, we use two color descriptors: Dominant Color (DC) and Color Layout (CL); one texture descriptor: Edge Histogram (EH); and one shape descriptor: Region Shape (RS). DC and CL descriptors are used together in the system as the color descriptors since these descriptors complement each other in describing the color features of images. EH is chosen as the texture descriptor, since according to MPEG-7 standard, it is the recommended texture descriptor for non-homogeneous regions and natural images [51]. RS is used as the shape descriptor since it can describe complex objects consisting of multiple disconnected regions as well as the simple objects with or without holes [51]. Note that no single descriptor is able to produce retrieval results comparable to those of the combination of features. This is primarily due to the diversity of the images in the data sets we utilize here, which includes both real images such as the *nature* category, and artificial images such as the *dinosaur* category.

These visual descriptors for the data sets are extracted using the MPEG-7 eXperimentation Model (XM) [52] [53] Software. XM software is the simulation platform for the MPEG-7 Descriptors and comes with two sets of applications: the server (extraction) applications and the client (search, filtering and/or transcoding) applications. We have only used the server applications for low-level feature extraction.

Similarity measurement for the visual descriptors was carried out using the Euclidean Distance measure, which is a metric distance function. The attributes of the MPEG-7 descriptors are briefly explained below. More detailed information about MPEG-7 descriptors can be found in [42] .

The Color Layout distance (d_{CL}) of two images *a* and *b* is defined as:

$$d_{CL}(a,b) = \sqrt{\sum_{i=0}^{5}\left(YC_{a,i} - YC_{b,i}\right)^2 + \sum_{i=0}^{2}\left(CbC_{a,i} - CbC_{b,i}\right)^2 + \sum_{i=0}^{2}\left(CrC_{a,i} - CrC_{b,i}\right)^2} \quad (3.1)$$

where *YC* is the DCT (Discrete Cosine Transform) coefficients for the luminance, *CbC* and *CrC* are DCT coefficients for the chrominance in *YCbCr* color space.

The EH descriptor distance function is defined as:

$$d_{EH}(a,b) = \sqrt{\sum_{i=1}^{80}\left(BinCounts_{a,i} - BinCounts_{b,i}\right)^2} \quad (3.2)$$

where *Bincounts* represents the number of types of edges (i.e. vertical, horizontal, 45^o diagonal, 135^o diagonal, and isotropic) for 16 sub-images of the image.

The RS descriptor distance function is defined as:

$$\boldsymbol{d_{RS}(a,b) = \sqrt{\sum_{i=1}^{35}(MagOfArt_{a,i} - MagOfArt_{b,i})^2}} \quad \textbf{(3.3)}$$

where *MagOfArt* is the array of magnitudes of the shape coefficients.

The distance function for DC is a fuzzy distance function which is introduced in [54] and differs from the distance computation of other features. The fuzzy distance of DC is calculated in two steps. In the first step, if the color value differences are less than a predefined threshold value, the color distance is evaluated in the specified color space, which is RGB for our system, using the following function:

$$d_{DC}(a,b) = \sqrt{\sum_{i=1}^{3}(ColorVal_{\,a,i} - ColorVal_{\,b,i})^2} \qquad (3.4)$$

where *ColorVal* is the dominant color values in the specified color space. If the color value differences are greater than the threshold value, then color similarity is assumed to be one (1). In the second step, the minimum percentage of the related dominant colors of two images is normalized and multiplied by the color similarity to find the final DC similarity of the two images.

Each of the descriptor functions described above is normalized such that any two images from the database gives a descriptor distance value between 0 and 1. The vector of descriptor distance values between two images is denoted as:

$$\nabla = [d_{CL}, d_{EH}, d_{RS}, d_{DC}] \qquad (3.5)$$

3.2. Aggregation of Descriptor Functions

The most common way of combining multiple distance functions is to take their arithmetic average. This restricted, equally weighted merging does not take into account the fact that some of the features may be more relevant for discriminating among the objects than others. In order to recognize the differing utility of the feature

descriptors, we use a *Constant Weighting (CW)* scheme to combine the descriptor distance functions described above. The CW distance between two images is denoted as:

$$d_{CW} = W_{CW}\, x\, \nabla \tag{3.6}$$

where the weight vector is:

$$W_{CW} = [W_{CL}, W_{EH}, W_{RS}, W_{DC}]^T \tag{3.7}$$

such that $W_{CL}, W_{EH}, W_{RS}, W_{DC} \in [0,1]$ and $W_{CL}, + W_{EH} + W_{RS} + W_{DC} = 1$. The weight vector W_{CW} determines the contribution of each of the descriptor distance functions and has been determined by Nelder-Mead simplex optimization method [55].

In contrast to applying the same constant weights for all image pairs, it may be possible to determine the weights dynamically for the image pair being compared. Specifically, it may be desirable for a retrieval task to highlight the feature that two images share the most in their comparison. The Ordered Weighted Average (OWA) method [44] allows dynamic weighting of the features by ordering the descriptor distance values. Let $W_{OWA} = [W_1, W_2, W_3, W_4]$ be a pre-determined weighting vector such that $W_1 \geq W_2 \geq W_3 \geq W_4$ and $W_1, W_2, W_3, W_4 \in [0,1]$ and $W_1, + W_2 + W_3 + W_4 = 1$. And let ∇' be the vector of descriptor distances whose elements are sorted in ascending order, such that the vector I defines the mapping from ∇ to ∇': $\nabla' = [\nabla_{I_1}, \nabla_{I_2}, \nabla_{I_3}, \nabla_{I_4}]$ and $\nabla_{I_1} \leq \nabla_{I_2} \leq \nabla_{I_3} \leq \nabla_{I_4}$. The ordered weighted average distance between two images is then:

$$d_{OWA} = W_{OWA}\, x\, \nabla' \quad (3.8)$$

Notice that the descriptor distance with the smallest value is associated with the largest element in the OWA weight vector. Note also that, since the descriptor distances have been separately normalized to fall in the range from 0 (maximum similarity) to 1.0 (maximum dissimilarity), and that the length of the W_{OWA} vector is 1, the aggregated OWA distance d_{OWA} is also within $[0, 1]$. To illustrate the calculation of d_{CW} and d_{OWA} consider the following descriptor distances for comparing two images:

$$\nabla = [d_{DC}, d_{CL}, d_{EH}, d_{RS}] = [0.325, 0.57, 0.45, 0.25]$$

and let the CW and OWA weights be:

$$d_{CW} = d_{OWA} = [W_1, W_2, W_3, W_4] = [0.35, 0.3, 0.2, 0.15]$$

The aggregated distances are then calculated as:

$d_{CW} = W_{CW}\, x\, \nabla$

$= W_1 * d_{DC} + W_2 * d_{CL} + W_3 * d_{EH} + W_4 * d_{RS}$

= 0,35 * 0,325 + 0,3 * 0,57 + 0,2 * 0,45 + 0,15 + 0,25

= 0,4122

$d_{OWA} = W_{OWA}\, x\, \nabla'$

$= W_1 * d_{RS} + W_2 * d_{DC} + W_3 * d_{EH} + W_4 * d_{CL}$

= 0,35 * 0,25 + 0,3 * 0,325 + 0,2 * 0,45 + 0,15 + 0,57

= 0,3605

Earth Mover's Distance (EMD) function is also used in order to compare with Euclidean Distance measure. The EMD measures minimum cost required to transform one histogram or signature into another. This measure can be applied to

distributions of features as long as space of these features is equipped with some similarity measure (e.g., ground distance). We have used L1 distance as ground distance for this work (*EMD-L_1*), since it makes the EMD be metric. The formal definition of the EMD can be found in [45]. We have used low level MPEG-7 features as the weights of the EMD signatures and their index is used as feature representatives.

3.3. Image-based Similarity Measurement Techniques

In feature-based image comparison, each image is pre-processed and represented as a feature vector; the distance functions are then applied to these feature vectors and the original image data is no longer used in comparison. On the other hand, the whole image based comparison, also known as *image registration*, uses the image data directly to calculate the similarity of two images. In this study, we investigate two popular area-based image registration methods: Cross Correlation and Mutual Information. These methods, sometimes called correlation-like methods or template matching methods [56], merge the feature detection step with the image matching step without detecting salient objects.

We have also included Spatial Color Variance method, a simple color histogram based image comparison method, to form a basis for these image registration methods. Spatial Color Variance is a simple and relatively fast matching method for colored images. Moreover, we have used SIFT features for image registration, since this method is widely used in matching and recognition of objects in the images. A brief overview of each of the image registration methods is given in the following sub-sections.

3.3.1. Cross Correlation

Cross Correlation (CC) is a standard statistical technique and is mostly used in the image registration domain in order to estimate the degree to which two images are correlated [46]. Given two images represented as matrices *A* and *B*, the normalized CC can be computed as:

$$CC = \frac{\sum_i \sum_j (A_{ij} - A')(B_{ij} - B)}{\sqrt{\sum_i \sum_j (A_{ij} - A')^2 \sum_i \sum_j (B_{ij} - B)^2}} \quad (3.9)$$

where *A'* and *B'* are the mean values of *A* and *B* respectively.

3.3.2. Mutual Information

Mutual Information (MI) [46] is a measure of how well one image explains the other in a statistical sense. Mutual Information is commonly used in multimodal image registration. However, since we do not need to register the images, we use this method only for evaluating the degree of similarity between two images. Mutual Information of two images *A* and *B* is calculated as:

$$MI(A, B) = H(A) + H(B) - H(A, B) \quad (3.10)$$

where *H(A)* and *H(B)* are the individual entropy values of the images and *H(A,B)* is their joint entropy. Joint entropy measures the amount of information contained in the two images combined and can be calculated using the joint histogram of the two images.

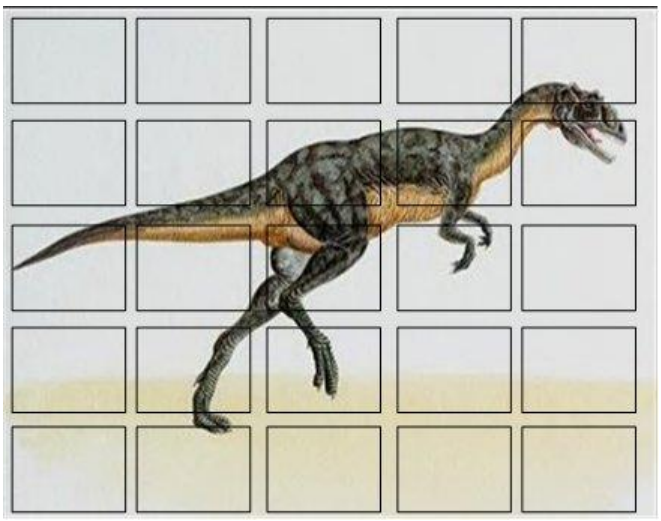

Figure 3-2: Spatial Color Variance - Dividing image into 25 regions.

3.3.3. Spatial Color Variance

Another image-to-image comparison method applied in this work is a color signature based similarity measurement, described in [47]. First, the image is normalized and only the color averages of the image sub-regions are stored. For this purpose, the image is divided into 5x5 regions and is represented as a 25x3 feature vector (Figure 3-2). Spatial Color Variance (SCV) of two images is then calculated as the Euclidean distance between their color average vectors. SCV essentially combines color information with spatial information.

3.3.4. Scale Invariant Feature Transform

Scale Invariant Feature Transform (SIFT) have successfully been used for object recognition, owing to their ability to capture distinctive keypoints that are invariant to location, scale and rotation, and that are robust to affine transformations [57] [58]. However, applications of SIFT have generally been in recognition of transformations of the same object, which is different than the image retrieval task surveyed in this

method. Thus, we have adapted a matching process of SIFT keypoints in order to compare images by using SIFT descriptors in a simple way by using L_2 distance function.

3.4. Multidimensional Scaling

Multidimensional Scaling (MDS) [59] is an important method for mapping pairwise relationships to coordinates. MDS requires a distance matrix between points as input. Its output is a set of vector which consists of low-dimensional coordinates for each point. MDS is used for dimension reduction in general. Firstly, a set of distances between points is measured and MDS algorithm applied to this set in order to find a sparse distance matrix.

The main problem of the classical MDS algorithm that it is not appropriate for large number of points because it requires an entire $N \times N$ distance matrix. This distance matrix is usually stored in memory. The classical MDS algorithm has $O(N^3)$ complexity. Thus, several scalable MDS algorithms have been proposed.

Landmark Multidimensional Scaling (LMDS) [60] requires a small subset of data to increase scalability of the classical MDS. The algorithm uses (dis)similarity matrix D of l landmark data points and embed these points in m-dimensional Euclidean space. While LMDS solves efficiency problem of classical MDS with the help of fast computation beacuse of using small number of landmark points, it also preserves all properties of classical MDS.

First of all, l landmark points are chosen by using some appropriate algorithm (random selection, farthest points etc…). Then the classical MDS algorith is applied to these landmarks in order to find a mapping of the l landmark points into m-dimensional space. The remaining set of points are embedded to m-dimensional space by using distance funciton and a normalization algorithm is applied as final step.

LMDS is much more efficient than classical MDS since it has $O(nl+ l^3)$ computational complexity.

FastMap [61] is an MDS method that construct a low-dimensional embedding by determining one dimension at a time. It uses two farthest pivot points and creates a transformation to n-1 dimensions using distances to compute embedding coordinates. FastMap is an iterated form of LMDS with two landmarks.

3.5. Experiments

The performance of the aforementioned access methods and similarity measurement techniques is evaluated on the images from three different data sets; Corel Database [48], News Video image set originally taken from [50] and ImageNet data set [49]. Corel data set consists of ten categories: mountains, people, buses, flowers, elephants, horses, architectures, dinosaurs, beach, and food (See Figure 3-1: Sample images from different categories of the Corel Dataset used in this study.

). Each category contains 100 images. ImageNet has 4000 images forming different categories images such as basketball images, cars, birds, furniture, food and miscellaneous images each having various number of images. Images from news video data are extracted and used as third data set which has 3142 images and contains categories including news, sport etc. Each image in the categories is used as a query object and the overall retrieval accuracy is evaluated using the precision and recall (PR) curves, averaged over all query objects. Moreover, F-Measure is computed and used as another metric for retrieval performance. The retrieval efficiency is evaluated using the query times and also page access count for k-Nearest Neighbor (k-NN) queries, with varying k values.

BitMatrix and SlimTree index structure implementations in [62] and the $LMDS_{FastMap}$ implementation in [8] were adapted to allow comparison of the images via feature-based or whole image-based distance functions. BitMatrix was built by clustering

CL, DC, EH, and RS feature dimensions into a pre-defined number of ranges. For EH and DC, this number of ranges is eight and for CL and RS, this number is set to four. These are manually optimized numbers and are chosen by performing accuracy tests.

SlimTree access structure is implemented using the XXL API [63]. BitMatrix structure is implemented using Weka [64] and Colt [65] libraries, where Weka API is used for clustering the low-level features and Colt API is used to store the constructed BitMatrix access structure and perform queries on the BitMatrix.

The $LMDS_{FastMap}$ is adapted from [8] to use feature-based and image-based similarity measures for the embedding. The embedded vectors are indexed using the X-Tree spatial access method [10].

For completeness, we perform a Sequential Scan of the entire database to provide a baseline for accuracy and time performance. The performance of the Euclidean Distance measures is compared with that of the Earth Mover's Distance ($EMD\text{-}L_1$), whose implementation is adapted from [66] to deal with MPEG-7 descriptors. In Sequential Scan, a query is compared with every single image in the database and the database objects are sorted by the similarity measure under consideration.

3.5.1. Comparison of Feature-based Similarity Methods

The weights of CW and OWA feature-based similarity measures have been optimized using the Nelder-Mead simplex method [55], with the objective of maximizing the area under the precision-recall (P-R) curve. The optimized W_{CW} weights are:

- for Corel data set: $w_{CL} = 0.455$, $w_{EH} = 0.511$, $w_{RS} = 0.0215$, $w_{DC} = 0.0123$.
- for ImageNet data set: $w_{CL} = 0.448$, $w_{EH} = 0.513$, $w_{RS} = 0.022$, $w_{DC} = 0.015$.

- for news video data set: w_{CL} = 0.433, w_{EH} = 0.528, $ w_{RS} = 0.018, w_{DC} = 0.012.

These values highlight the contributions of the CL and EH feature and provide only marginal contributions to the RS and DC features. We have done some experiments regarding this point. For example, if we ignore RS and DC features, then the area under PR curve values decreases slightly. For the Corel data set, if we apply Sequential Scan and use four low level features, area under PR curve is 0,426. But if we ignore RS and DC features, this value becomes 0,423. On the other hand, for LMDS where space dimension is ten (10), the difference is somewhat larger, 0,383 when four features are used and 0,369 when only two features are used.

For OWA, the following optimized weights were obtained for all data sets:

w_1 = 0.527, w_2 = 0.368, w_3 = 0.054, w_4 = 0.050

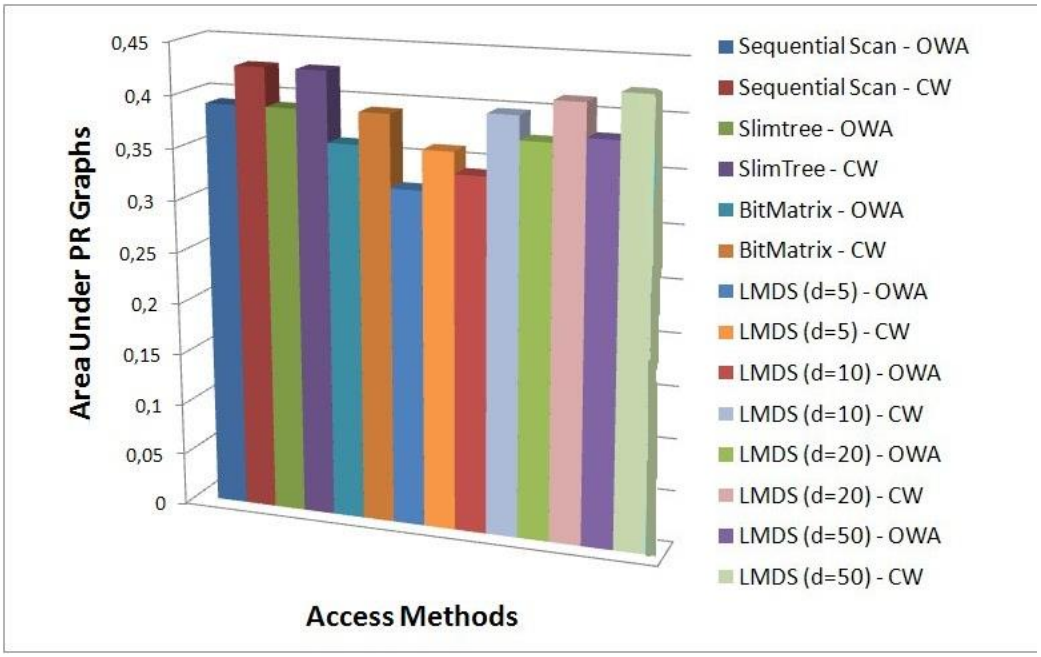

Figure 3-3: Area under PR graphs for k=10 (Corel Dataset)

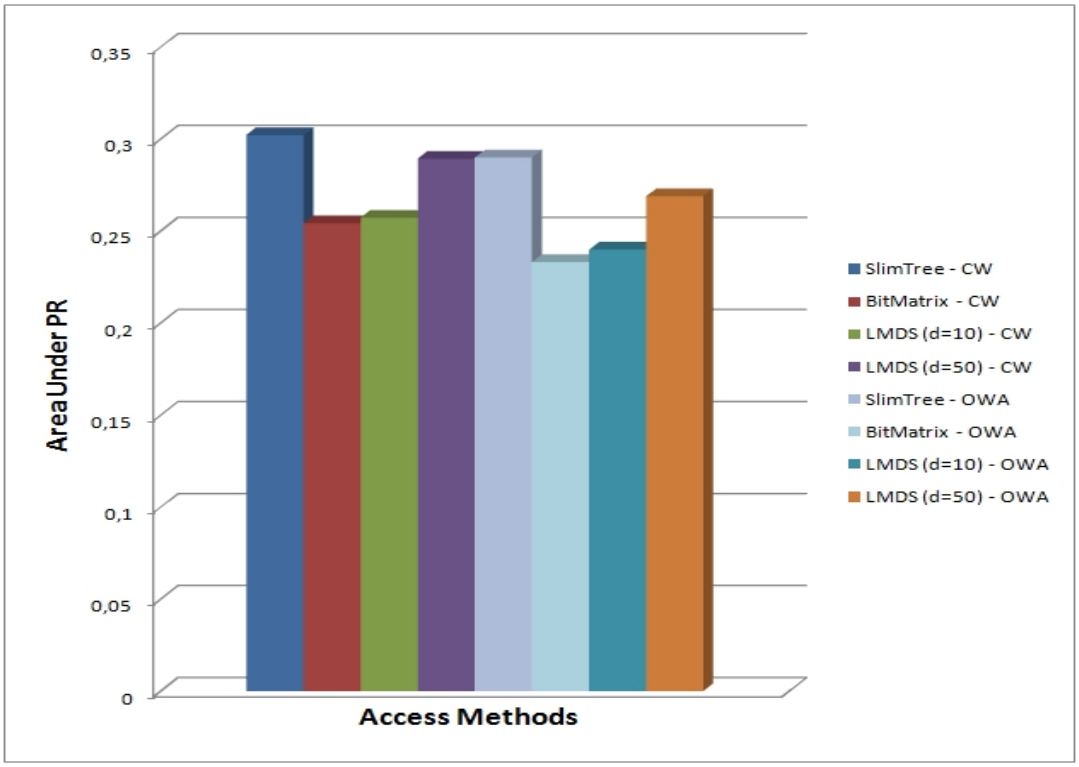

Figure 3-4: Area under PR graphs for k=10 (News Video Dataset)

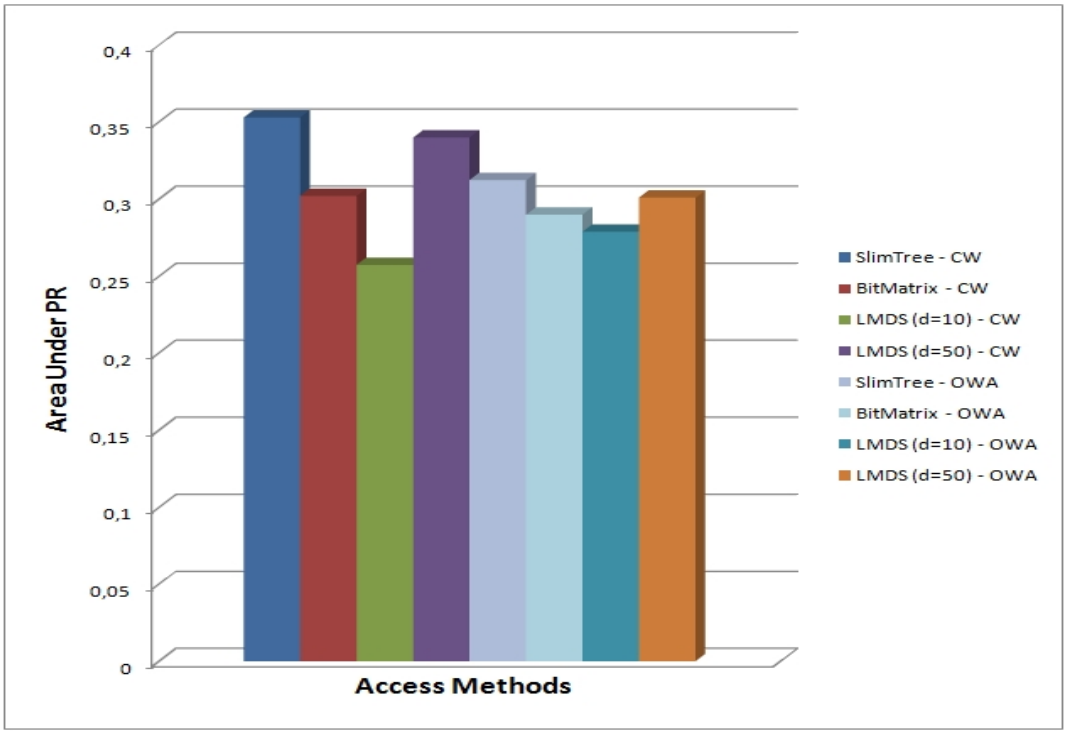

Figure 3-5: Area under PR graphs for k=10 (ImageNet Dataset)

The Precision-Recall results for all multidimensional access methods and for all data sets for 10-NN query are shown in Figure 3-3, Figure 3-4, and Figure 3-5 and a selection of these results for Corel data set are depicted in Figure 3-6. The optimized constant weight measure is found to outperform the OWA measure. Among the multidimensional access methods, SlimTree with constant weights performs the best in terms of retrieval accuracy.

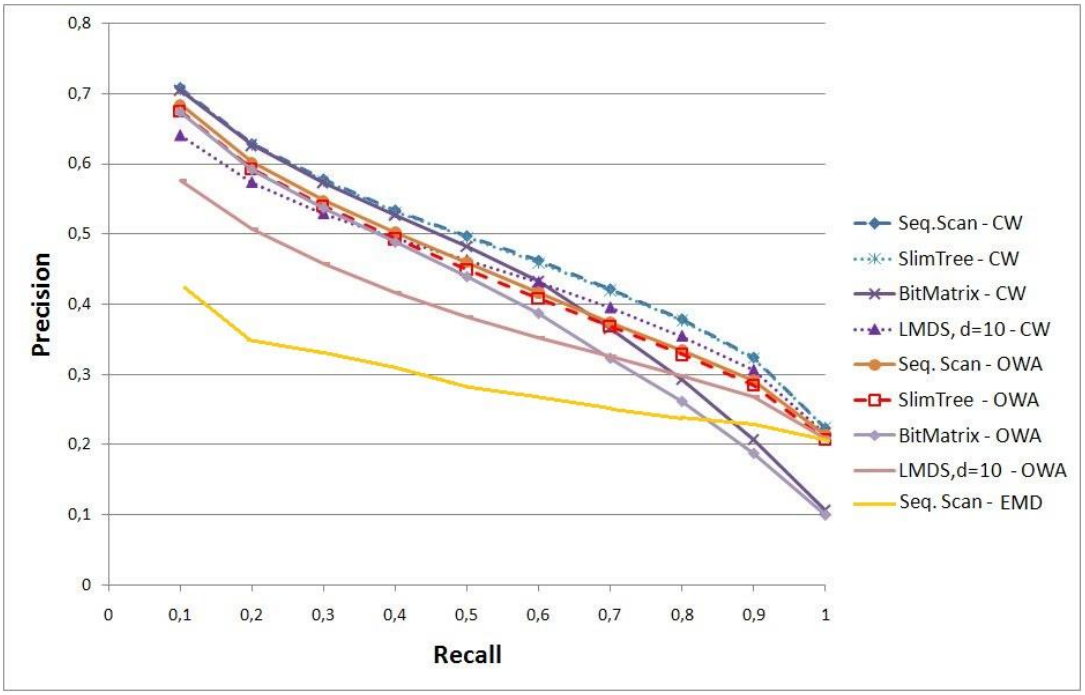

Figure 3-6: PR graphs for feature-based approach for Corel data set.

Note that, since the distance functions used to compare images are metric, the accuracy of SlimTree is guaranteed to be the same as that of the Sequential Scan. On the other hand, BitMatrix and LMDS are approximating the distance function and may give better or worse accuracy than Sequential Scan. As embedded space dimensionality is increased, the accuracy of $LMDS_{FastMap}$ approaches that of the SlimTree. On the other hand, BitMatrix has worse accuracy than Sequential Scan, because of the approximate representation of the objects resulting from the initial

clustering step. Figure 3-6 also shows the performance result for (*EMD-L_1*). Euclidean Distance measure has better PR values than EMD's since whole low-level feature values are used as single cluster in the signatures in a basic way.

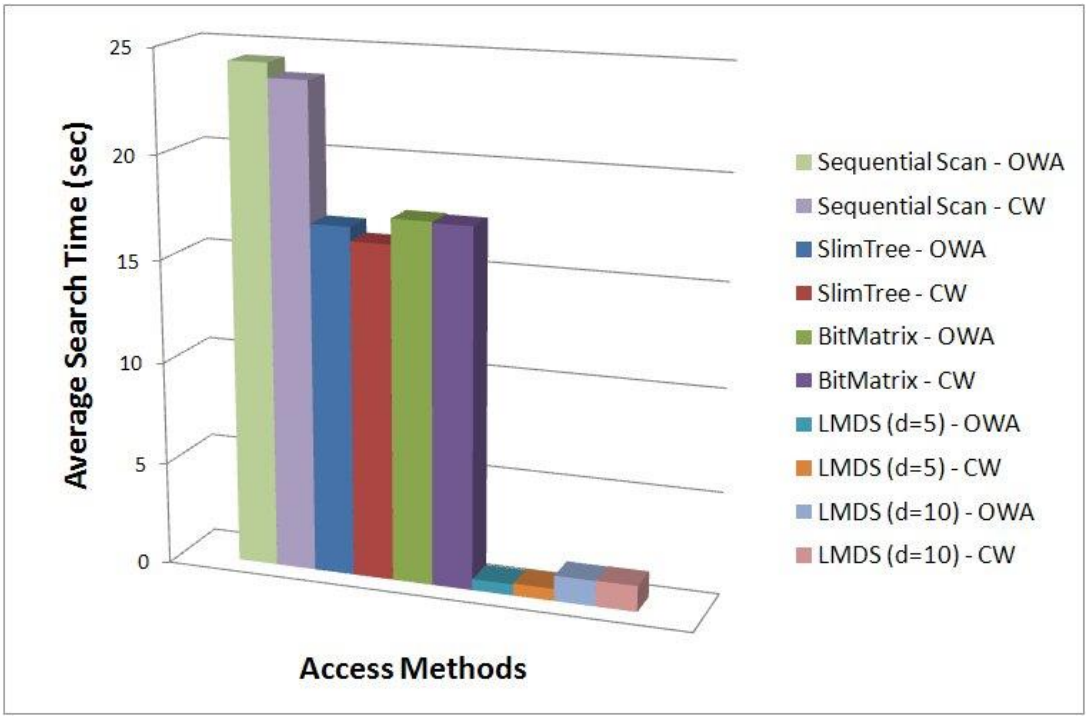

Figure 3-7: Search time graphs for k-NN queries of feature-based approach for Corel data set.

Figure 3-7 shows the query times for varying values of k in nearest neighbor queries for each of the methods for Corel data set. For other data sets, the results are similar to this figure but due to space limitations, they are not included in the paper. $LMDS_{FastMap}$ retrieves results significantly faster than both BitMatrix and SlimTree, owing to the efficiency of X-tree Spatial Access Method. While BitMatrix gives similar precision-recall results to $LMDS_{FastMap}$, it requires significantly more time to find the result objects, due mostly to the time spent during the clustering process in each query evaluation phase. The search time values for SlimTree are slightly better than BitMatrix results, but there is still a significant difference between SlimTree and $LMDS_{FastMap}$, due to SlimTree requiring rather frequent disk page accesses.

Table 3.1 gives a closer look into the LMDSFastMap search time test results for Corel data set. Within the range of dimensions sampled, the search time for k-NN queries increases linearly as the dimensionality of the embedded space increases. Even when the embedded space dimensionality is relatively high (d=100), X-tree indexing over LMDSFastMap embedding is still faster than any other access structure. There is no significant difference in retrieval time for LMDSFastMap using constant weights or OWA.

Table 3.1: Search time values for k-NN queries using LMDS (k=10) for Corel data set.

Space Dimension	Weighting Approach	
	CW	OWA
d = 5	0.594	0.547
d = 6	0.657	0.703
d = 7	0.813	0.828
d = 8	0.938	0.953
d = 9	1.125	1.172
d = 10	1.265	1.248
d = 20	2.647	2.532
d = 30	3.797	3.812
d = 40	5.141	4.969
d = 50	6.250	6.063
d = 100	11.359	11.500

3.5.2. Image Based Similarity

The aim of image-based similarity methods is to perform comparisons of images themselves, directly. CC and MI techniques are mainly used in the image registration domain. SCV is a simpler, less commonly used method and works with pre-compiled color averages of the images. SIFT is mainly used in object recognition. Similar to

the feature-based approaches, for each indexing method, we have evaluated these four methods in terms of their precision and recall performance in similarity queries over the databases.

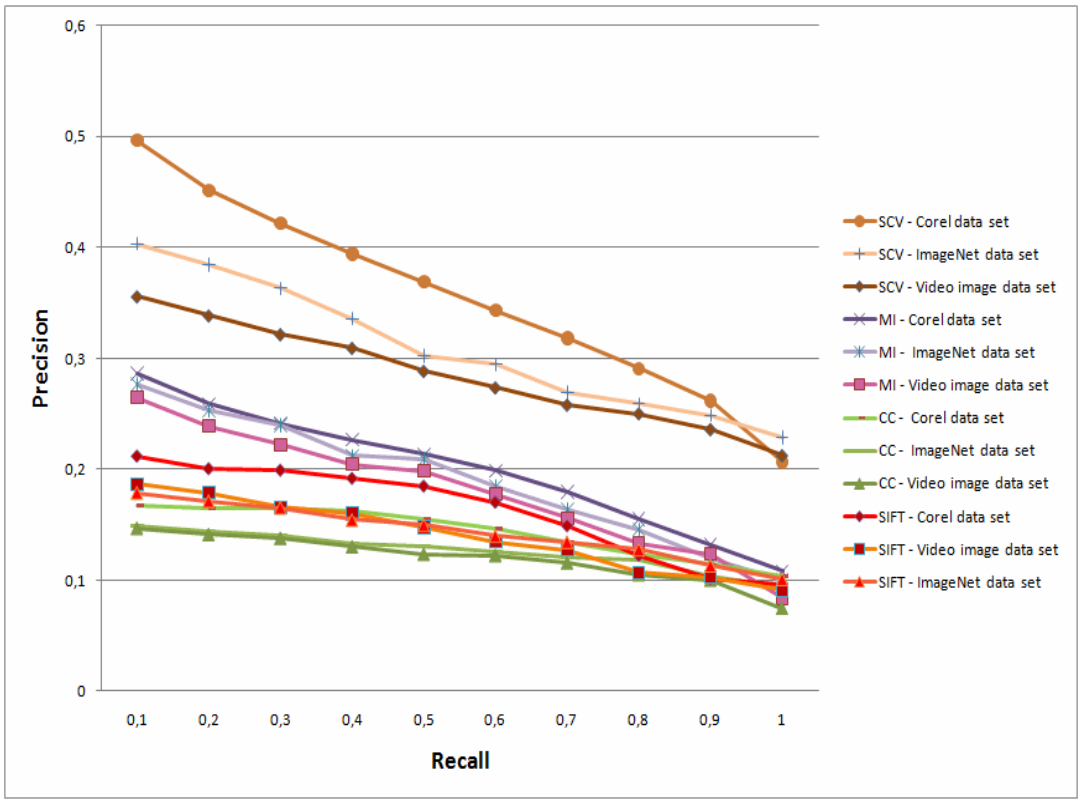

Figure 3-8: Comparison of image-based similarity techniques (Sequential Scan).

The precision-recall curves for Sequential Scan are provided in Figure 3-8. Surprisingly, SCV gives significantly better precision-recall results compared to the other two methods for all data sets. We attribute this to the fact that the images within the same category in the databases share similar spatial color distributions. This was also evident in the constant weighted feature-based retrieval, where the CL descriptor made an important contribution to the similarity measure. Furthermore, both the MI and the CC methods were applied on gray-scale images, making them unable to utilize the discriminative power of the color information. MI method

generally performs better than CC according to [67]. Moreover, using SIFT descriptors has better results than CC method.

The area under the precision recall curves for the indexing methods for Corel data set is given in Table 3.2. The results for other data sets are shown in Figure 3-9, Figure 3-10, Figure 3-11.

Table 3.2 - Area Values Under PR Curves for Corel data set.

Access Method	Similarity Approach			
	CC	MI	SCV	SIFT
Sequential Scan	0.1855	0.2201	0.3202	0.1994
SlimTree	0.1855	0.2201	0.3202	0.1997
BitMatrix	0.1691	0.2082	0.3104	0.1723
LMDS (d = 5)	0.1361	0.1639	0.2878	0.1400
LMDS (d = 10)	0.1456	0.1750	0.2961	0.1579
LMDS (d = 20)	0.1496	0.1898	0.3024	0.1610
LMDS (d = 30)	0.1637	0.2065	0.3000	0.1875
LMDS (d = 40)	0.1710	0.2175	0.3046	0.1922
LMDS (d = 50)	0.1804	0.2150	0.3027	0.1986
LMDS (d = 100)	0.1755	0.2124	0.3014	0.2012

Application of the indexing methods follows the same trend as the feature-based image comparison case. SlimTree has the same accuracy as Sequential Scan, as expected. BitMatrix has worse accuracy than SlimTree but better accuracy than $LMDS_{FastMap}$ for *LMDS* target dimensionality of less than 40. As the target embedding space dimensionality is increased, the accuracy of $LMDS_{FastMap}$ approaches that of the Sequential Scan.

The time requirements of the image-registration and indexing methods are depicted in Figure 3-12. Among three indexing methods, $LMDS_{FastMap}$ again has the best

response time for whole image based comparison. BitMatrix is closer in speed to $LMDS_{FastMap}$ than it was in the feature-based comparison. Since the image registration methods are computationally expensive operations, the SlimTree is the slowest, because of the distance computations it has to perform with the routing nodes in its index tree.

$LMDS_{FastMap}$ performs distance computations only against the small set of landmark images it has selected during the index building phase. Increasing the number of objects to be retrieved does not therefore affect the number of image registration operations to be performed in $LMDS_{FastMap}$.

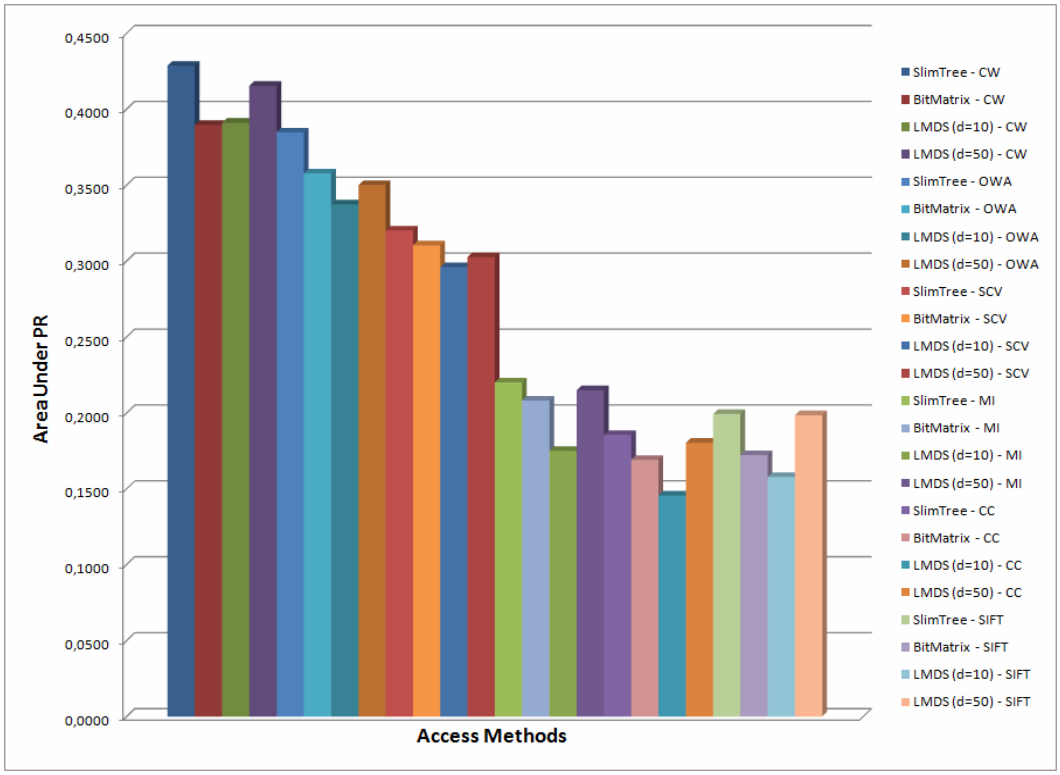

Figure 3-9: Area under PR graphs of k-NN queries (k=10) (Corel Dataset).

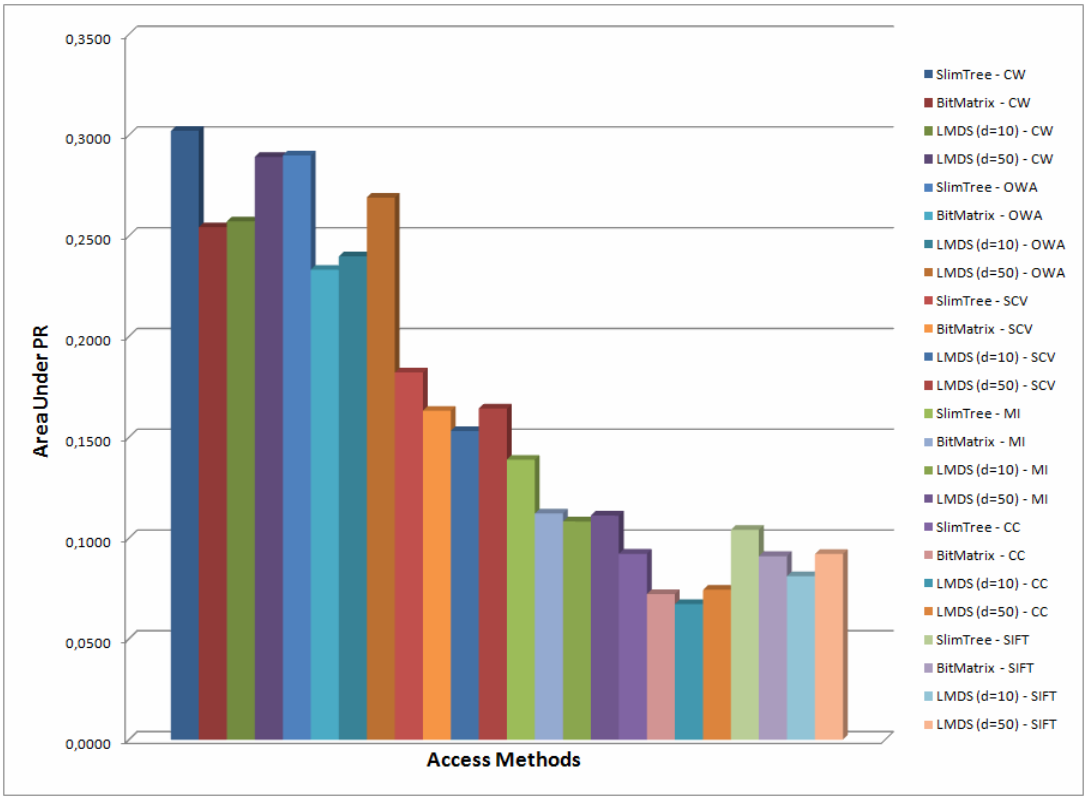

Figure 3-10: Area under PR graphs of k-NN queries (k=10) (News Video Dataset).

Since BitMatrix uses clustering process in both construction and query phases, this structure requires more time than $LMDS_{FastMap}$. For every query phase, the cluster to which query belongs has to be found. After this step, the objects in the same cluster are accessed and similarity between these objects and the query object is calculated.

For the three image-based similarity measurement techniques, using MI in BitMatrix and SlimTree outperforms other techniques for these structures. On the other hand, for $LMDS_{FastMap}$, SCV gives the best results among these techniques for both space dimension 5 and 10. Moreover, using CC causes an increase in the response time in comparison with MI and SIFT, even though no translation or rotation search is performed in the CC implementation.

Sequentially scanning the entire database gives the best accuracy, but is impractical for large databases. Among the data access structures that are proposed to speed-up

the image retrieval, SlimTree metric indexing method provides the best accuracy. Since the distance functions we have utilized form metric spaces, SlimTree is guaranteed to give the same results as the Sequential Scan. On the other hand, SlimTree required significantly more time than other methods to retrieve the images that are similar to the query image, as measured by the given distance function. Due to the intrinsic complexity of the data sets, SlimTree search procedure showed a lack of sufficient pruning and thus a large number of distance calculations were incurred.

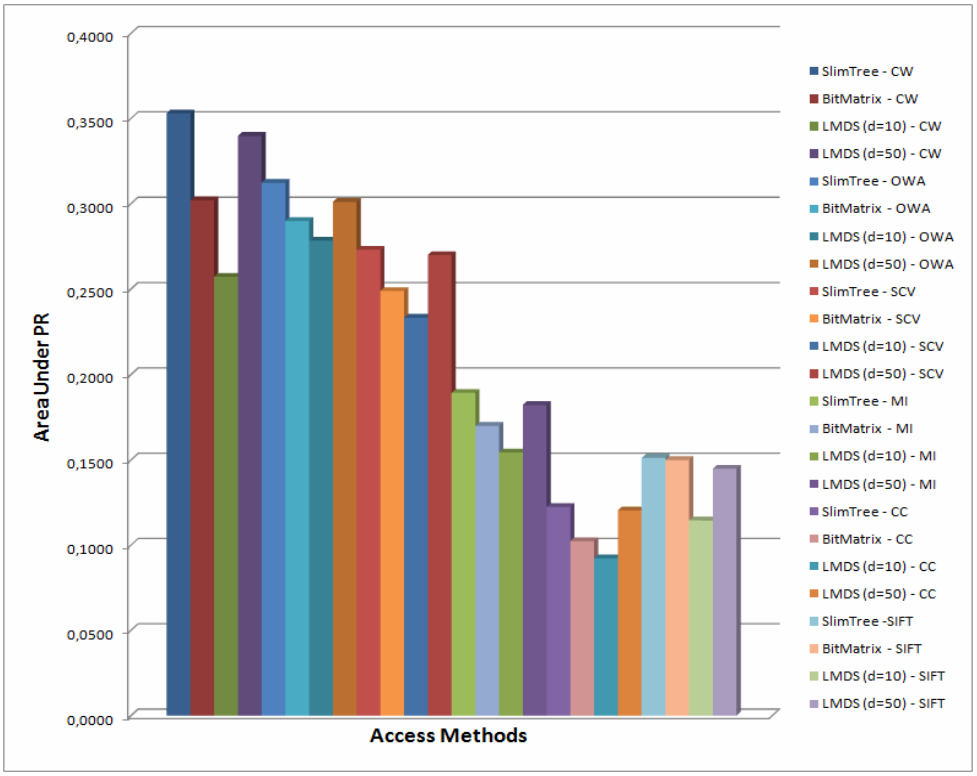

Figure 3-11: Area under PR graphs of k-NN queries (k=10) (ImageNet Dataset).

The clustering-based BitMatrix method and the dimensionality reduction based $LMDS_{FastMap}$ method are both approximate indexing methods that attempt to trade off accuracy for speed. Both of these methods were slightly worse in accuracy than the Sequential Scan. For the $LMDS_{FastMap}$ method, it is possible to increase the dimensionality of the embedded space such that the accuracy approaches that of the Sequential Scan. The running times of BitMatrix were similar to those of SlimTree.

On the other hand, the $LMDS_{FastMap}$ method was an order of magnitude faster than both BitMatrix and SlimTree.

Using feature-based image comparison was both faster and more accurate in retrieving similar images than image-registration based comparison, except for $LMDS_{FastMap}$ method applied to SCV measure which was faster but less accurate than feature-based comparison.

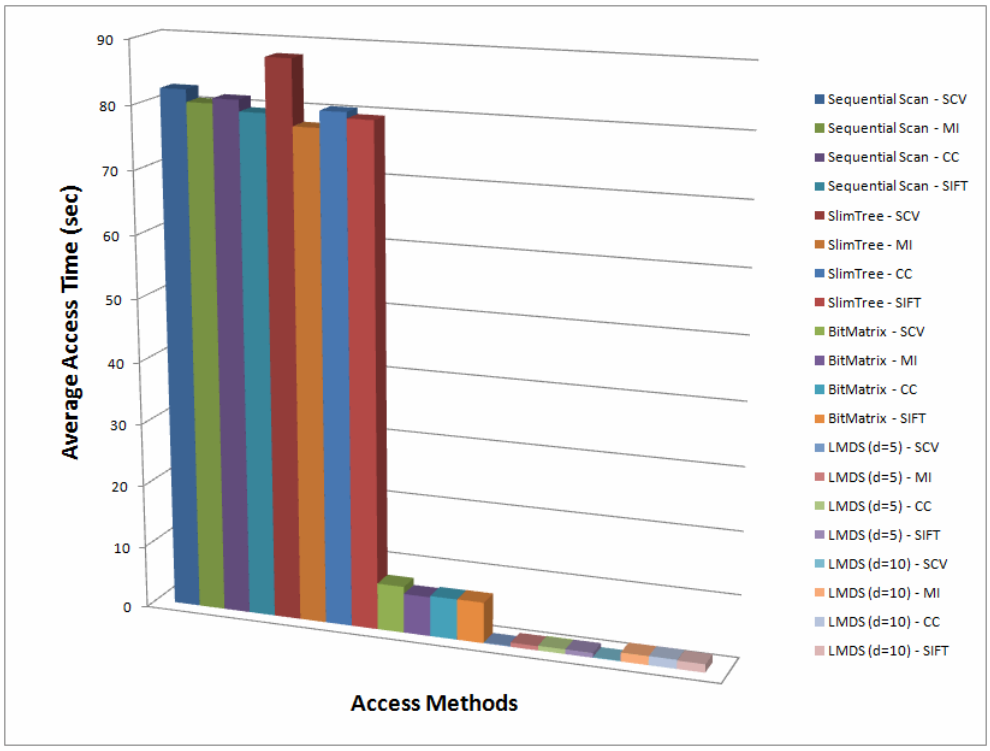

Figure 3-12: Search time graphs for k-NN queries of image-based approach for Corel data set.

To conclude, we have shown that multi-dimensional scaling (*LMDS*) can reduce the retrieval problem to a spatial-indexing task, where queries can be performed orders of magnitude faster than distance or cluster based indexing methods. The accuracy of the embedded space is shown to be comparable to that of the retrieval performed in the original space. We also show that constant weighting scheme can perform better than OWA for all access methods using features for comparing images. Furthermore,

the Spatial Color Variance was found to perform better than other image-based comparison methods in both accuracy and query processing times. Except for Spatial Color Variance's time performance, feature-based approach provides an effective retrieval of images and outperforms the image-registration based similarity measurement methods in both accuracy and running time.

CHAPTER 4

ACCESSING CONCEPT AND CONTENT DATA WITH FOOD INDEX AND SPATIAL INDEXING METHOD

4.1. Overview

The main motivation of this thesis is combining conceptual attributes of the multimedia data with its audio and visual descriptors. For this purpose we have developed two different access mechanism. Both mechanism contains FOOD-Index in order to index conceptual descriptors.

The main difference between these two structures comes from the way which content descriptors are indexed. In the first structure, a spatial indexing structure called X-Tree is adapted in order to represent and use low level features of multimedia data in spatial domain with the help of LMDS. In the second structure, clustering based methods are used and embedded into FOOD-Index.

In this chapter, we give some brief information about the structures we have used in this study, then we explain our first proposed access structure.

4.2. Using FOOD-Index with X-Tree

In this approach, the FOOD-Index structure is integrated with X-Tree in order to satisfy content and also concept based search and retrieval over multimedia data. The overall structure is shown in Figure 4-1.

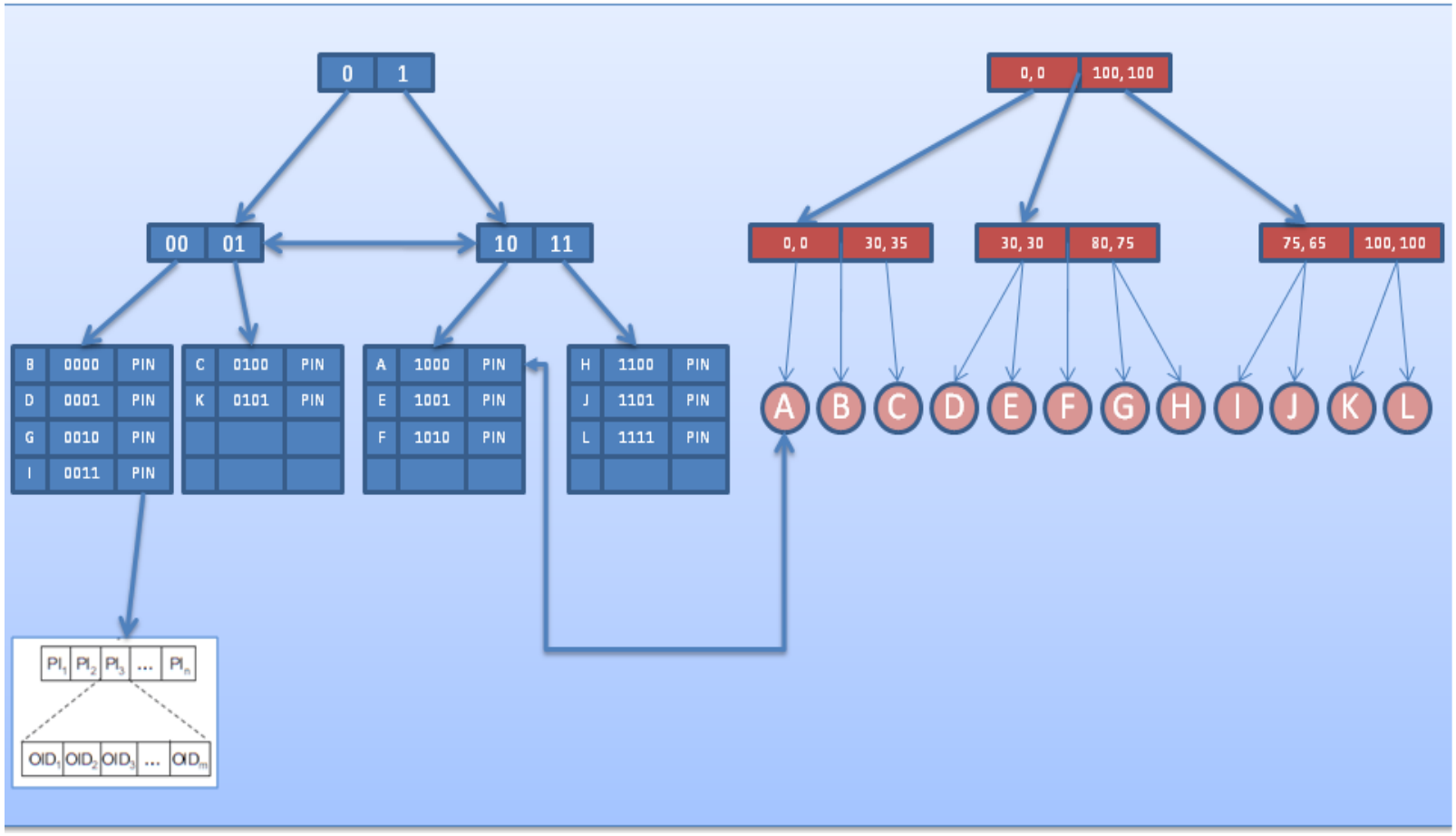

Figure 4-1: FOOD Index with X-Tree

As shown in the figure, the FOOD-Index and X-Tree is integrated by a single directional pointer between leaf nodes of each structure. The details of these structures are given in the next sections.

4.2.1 FOOD-Index

The FOOD-Index is a multidimensional indexing structure which indexes both fuzzy and crisp values in a multidimensional approach. The whole original FOOD-Index structure is shown in Figure 4-2.

The tree structure consists of two main parts; Routing nodes and data buckets. Data buckets are accessed from the root of the tree via routing nodes (Non-leaf and leaf nodes) by using some key value and includes data bucket records for each different key value indexed. Each data bucket record has a pointer to the nodes (Path Instantiation Node – PIN Figure 4-3) that stores all the paths (Path Instantiations - PIs) related to the key value of that data bucket record.

In order to index the path, an index structure called path index structure is adapted. The path index structure is a data structure for indexing object-oriented databases along both aggregation and inheritance hierarchies. Thus, the aggreagation and inheritance relations of the fuzy object oriented database model are handled by FOOD Index.

In the Path Index, G-tree is used for indexing attributes of the objects and also this structure has the ability of search over fuzzy or/and crisp attribute values. In order to handle search mechanism for both fuzzy and/or crisp attributes, these attributes must be converted to bit strings formed by 0's and 1's to provide a common base for the two types of values. In other words, those bit strings are the search keys in the tree.

With the help of Path Index, the FOOD-Index stores the PIs related with the same key value in a single PIN if they fit into a page as shown in Figure 4-3. PIs are implemented as arrays of id values of the objects on the path. While searching, when

a PIN is accessed, the OIDs in the position of the target class within its PIs are determined. The object referred to by these OIDs form the result set of the query.

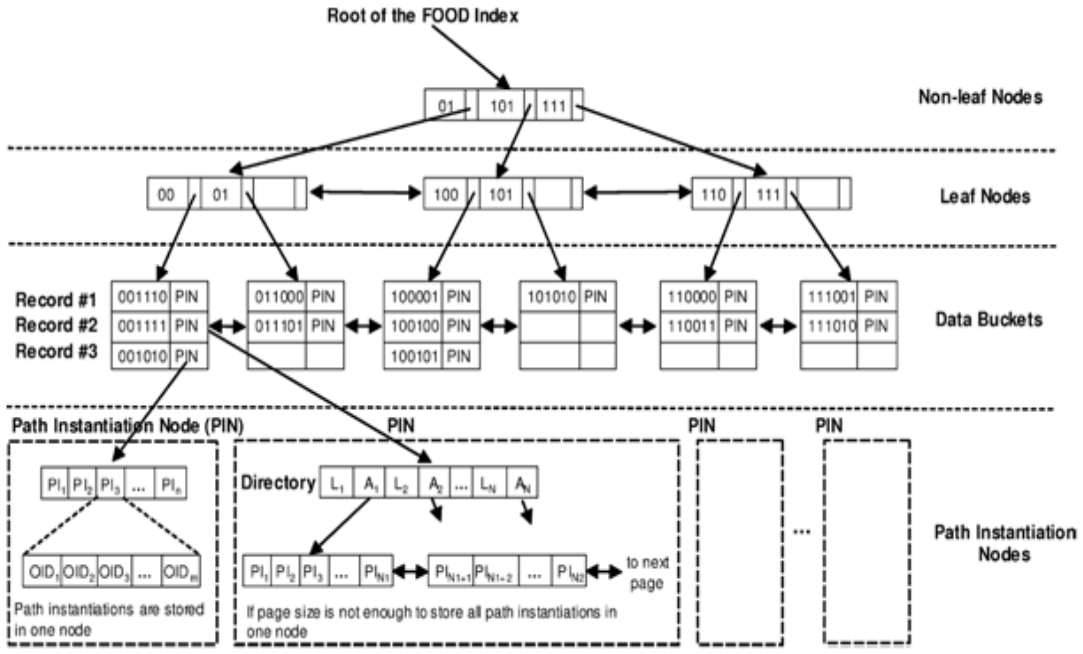

Figure 4-2: FOOD Index Structure.

If PIs do not fit into a page, the FOOD-Index organizes them using a directory structure and PIs are stored in these directory structures according to their lengths as shown in Figure 4-4.

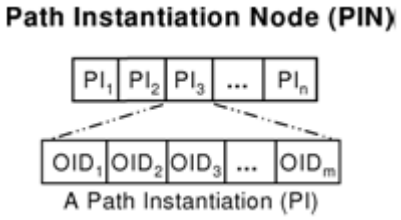

Figure 4-3: Path Instantination Structure.

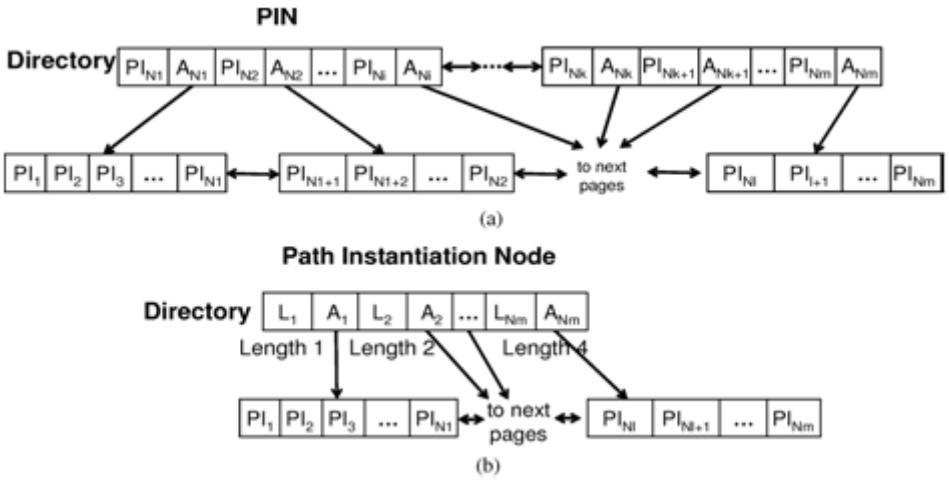

Figure 4-4: Path Instantination Structure.

In order to find related paths to the given query conditions, G-tree structure is used in the FOOD-Index. Search key of G-tree is constructed by using all attributes of the object. For example, if an object has two attributes, attr1 and attr2, the search key should be formed by these two attributes. For this purpose, bit string mechanism is applied in the FOOD-Index in order to evaluate key values. The details of constructing bit strings and algorithms for insertion, deletion and querying over the FOOD-Index can be found at [9].

4.2.2 X-Tree

X-tree (eXtended node tree) [10] is the extension of the R-tree. X-Tree has a term of supernodes which is defined as one or multiple minimum bounding rectangle in order to avoid overlap bounding boxes and, thus, has a larger capacity than a normal R-Tree node. While R-Tree divides a node into smaller nodes when a split necessary, X-Tree increases the capacity of a node on the contrary. This yields a performance gain when there is a large amount of overlap between two nodes after a split since

the probability that both nodes would be accessed by a search operation is very high. Thus, X-tree achieves a performance gain by accessing the supernodes sequentially. On the other hand, supernodes require much more complex disk management procedures and X-Tree also has larger index creation times. It is simply based on R-tree for retrieval process and may be seen as a combination of a linear linked list of supernodes and also R-tree based nodes.

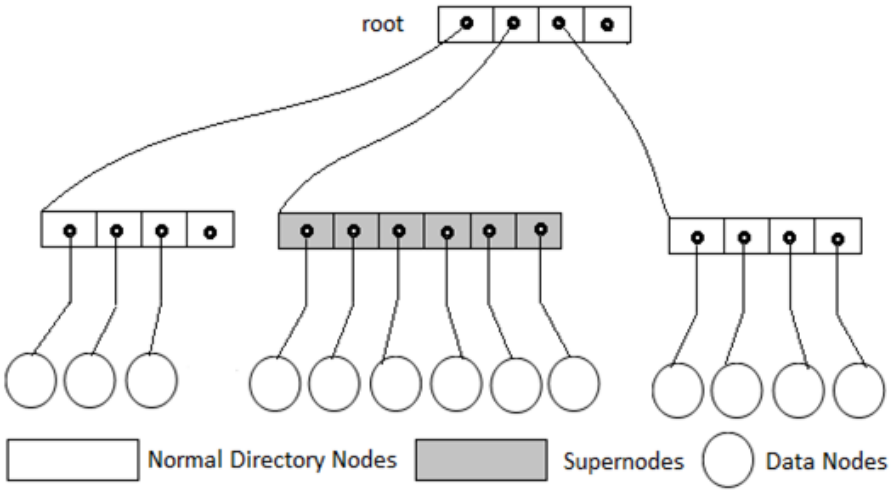

Figure 4-5: X-Tree Structure.

The heterogeneous structure of X-tree is illustrated in Figure 4-5. There are three different types of nodes in X-tree; data nodes, normal directory nodes and supernodes. Data nodes contains pointers to actual data objects represented as Minimum Bounding Rectangles (MBRs). Normal directory nodes is the nodes that holds high-level MBRs together with pointers to sub-MBRs. Finally supernodes are the combined directory nodes and may habe variable sizes in order to avoid splits in the directory.

In order to use X-Tree in our domain, $LMDS_{FastMap}$ algorithm is applied to the objects' low-level feature vectors and space dimension is reduced to 10 (ten), which has the reasonable time and efficiency performance according to Chapter 0. On the other hand, FOOD-Index structure is constructed by using object attribute values as

usual. To integrate both structures, Path Instantiation structure is changed and a new pointer to the X-Tree leaf node is added. Meanwhile, X-Tree leaf node is also adapted in order to access from X-Tree part of the structure to the FOOD-Index part by adding a pointer to the Path Instantiation Node.

Algorithm 1: Insertion

Input: list of coceptual features $F_{concept} = \{f_i\}_{i=1..n}$, list of content features $F_{content} = \{f_i\}_{i=1..n}$, object to be inserted *o*, database objects *O*,
Output: roots for the tree R_F and R_X

1: $d \leftarrow calculateDistances(o, O, F_{content});$
2: $D \leftarrow applyLMDS(o, d);$
3: $B_f \leftarrow \emptyset;$
4: **for all** $f \in F_{concept}$ **do**
  5: $B_f \leftarrow constructBitString();$
6: **end for**
7: $B \leftarrow combineBitStrings(B_f);$
8: $R_F \leftarrow insertFOODIndex(o, B);$
9: $R_X \leftarrow insertXTree(o, D);$
10: $createLinkBetweenTrees(R_F, R_X);$

In order to construct the proposed structure, the insertion mechanism shown in Algorithm 1 is used. Firstly, low level features of objects are extracted and similarities between new object and other objects calculated.This similarity values are used in order to find new object`s coordinates by applying LMDS algorithm.After that process, we first insert object into X-Tree by using its coordinates.Object is then inserted into the FOOD-Index. If the PIN is full then directory node is created and PINs are organized.

Table 4.1 Sample data for proposed structure.

ID	*Name*	*First Conceptual Attribute*	*Second Conceptual Attribute*	*Content Data*
1	A	8	12	XML Data
2	B	7	4	XML Data
3	C	11	3	XML Data
4	D	8	15	XML Data
5	E	6	5	XML Data
6	F	7	12	XML Data
7	G	11	9	XML Data
8	H	5	3	XML Data
9	I	3	5	XML Data
10	J	2	13	XML Data
11	K	4	8	XML Data
12	L	7	3	XML Data

Bulk loading of objects into this structure has similar algortihm to Algorithm 1 but now low level features of all objects in the database are extracted and similarities between each pair of object calculated as shown in Algorithm 2. This similarity matrix is used as input of our LMDS algorithm and the coordinates of objects in the reduced space dimension are obtained. After that process, we first insert object into X-Tree by using its coordinates. Object is then inserted into the FOOD-Index. If the PIN is full then directory node is created and PINs are organized. Finally link between the FOOD-Index` PIN and X-Tree node are constructed.

Algorithm 2: Bulk Load

Input: list of coceptual features $F_{concept} = \{f_i\}_{i=1..n}$, list of content features $F_{content} = \{f_i\}_{i=1..n}$, database objects O,

Output: roots for the tree R_F and R_X

1: $d[] \leftarrow calculateDistances(O, F_{content});$

2: : **for all** $o \in O$ **do**

3: $D \leftarrow applyLMDS(o, d);$

4: $B_f \leftarrow \emptyset;$

5: **for all** $f \in F_{concept}$ **do**

6: $B_f \leftarrow constructBitString();$

7: **end for**

8: $B \leftarrow combineBitStrings(B_f);$

9: $R_F \leftarrow insertFOODIndex(o, B);$

10: $R_X \leftarrow insertXTree(o, D);$

11: $createLinkBetweenTrees(R_F, R_X);$

12: **end for**

Let's assume that we have 12 objects in our database as shown in Table 4.1. Each object has two conceptual attributes. Moreover, each object has low level MPEG-7 descriptors (content descriptors) for visual and audio features in XML format. These descriptors are extracted using MPEG-7 XM software [52]. First of all, distance matrix of whole database is evaluated by using Euclidian Distance Function. After evaluating all distances of each object, $LMDS_{FastMap}$ procedure is applied. The output of this procedure is used as input for X-Tree. $LMDS_{FastMap}$ is used to map the database objects from high-dimensional space to two-dimensional space. Thus, each object is represented in two dimensional space (i.e. coordinates in X-Y space) as shown in *Content* column of Table 4.2.

Table 4.2 Sample data for proposed structure with content dimensions.

ID	*Name*	*First Conceptual Attribute*	*Second Conceptual Attribute*	*Content*
1	A	8	12	0, 0, 5, 12
2	B	7	4	75,80,100,90
3	C	11	3	77,65,88,100
4	D	8	15	30,30,42,35
5	E	6	5	90,90,100,100
6	F	7	12	38,62,45,75
7	G	11	9	5,8,28,30
8	H	5	3	50,51,68,72
9	I	3	5	50,45,70,55
10	J	2	13	60,65,80,75
11	K	4	8	95,85,100,95
12	L	7	3	11,17,20,35

The next step is construction of X-Tree. The content data is used as coordinates of each object in the construction phase since X-Tree indexes the data in spatial domain (Figure 4-6). There are two different X-Trees constructed; one for visual features and the other one for audio features.

While indexing content data, conceptual attributes are also indexed using FOOD-Index. In order to construct FOOD-Index, bit string representations of conceptual attributes *attr1* and *attr2*) is created. For instance, for the object *A*;

attr1 = 8 and bit string value = 1000

attr2 = 12 and bit string value = 1100

combined bit string value for *attr1* and *attr2* = 11010000.

This combined bit string value is indexed in the FOOD-Index. Finally directional link between X-Trees and the FOOD-Index is created. The final structure is shown in Figure 4-7.

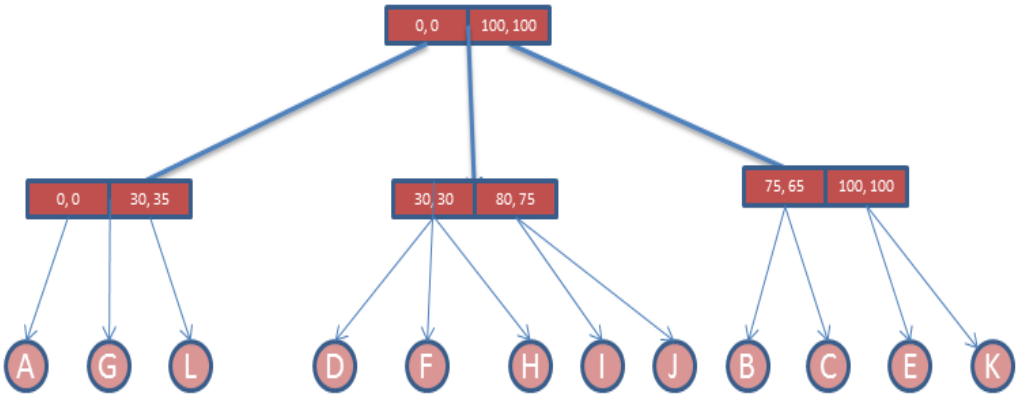

Figure 4-6: X-Tree Structure for sample data.

In order to delete an object from the structure, the object id must be known. First object is deleted from the FOOD-Index since it may result a reorganization process of the tree. If the FOOD-Index structure is reorganized, then links to the X-Tree nodes should also be organized. Otherwise, the node containing the deleted object is deleted easily from X-Tree.

4.2.3 Retrieval of objects

The proposed structure can be used in order to retrieve objects that satisfy user query conditions in terms of content (visually) and concept (semantically). Thus, there are three different methods for querying the structure.

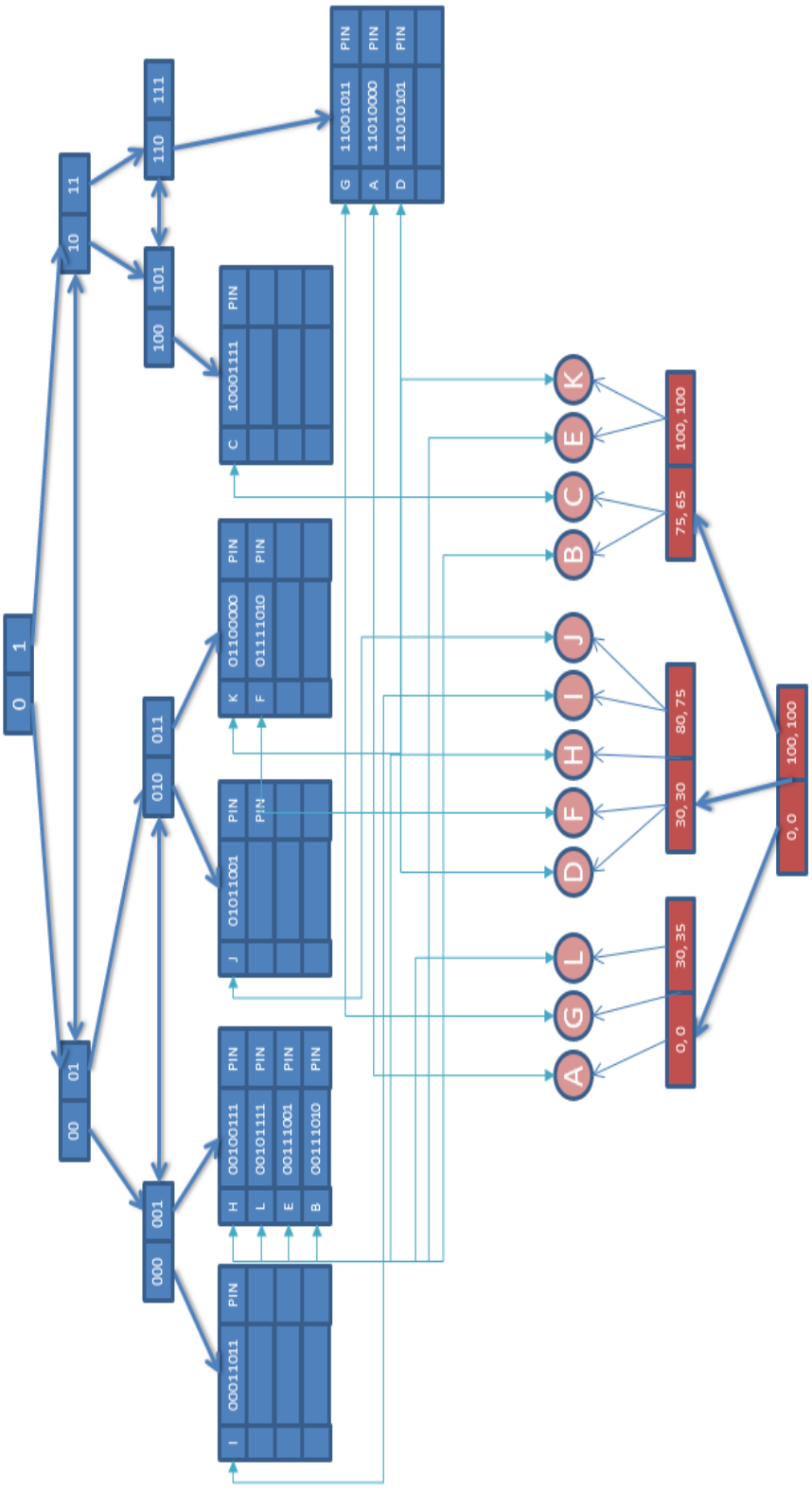

Figure 4-7: FOOD-Index with X-Tree Structure for sample data.

4.2.3.1 Query by content

For the CBR systems, query by example is the most common query strategy since the user can specify the query by giving an example object, such that "retrieve all objects similar to the given query object". Thus, a good CBR system should meet the query by example requirement. The proposed access structure is capable of satisfying the user's visual query with the help of X-Tree. The main steps of the algorithm are shown in Algorithm 3. Firstly, low level features of query object are extracted and query object`s coordinates are found by applying LMDS algorithm. After that process, query object is searched over X-Tree by using its coordinates. Objects which satisfy the query conditions are added to result list. If all the paths to the result objects are required, then for all objects in the result list FOOD-Index is directly accessed and PIs are returned.

Since, the user may want to search just the objects themselves or whole paths to the objects, our structure can easily return relevant query results to the user.

Suppose that the user wants to find *k* objects similar to *object A* in our database (k-NN query). First of all, LMDS is applied to the query object (*q*)'s content descriptors in order to find its space dimension (or mapped dimension) values and assume that these dimension values are {8,15,25,30}. Then, only X-Tree part of our structure is accessed and first *k* objects are returned to the user. For example if *k* value is three(3) then result set contains *A,G* and *L* objects as shown in Figure 4-8.

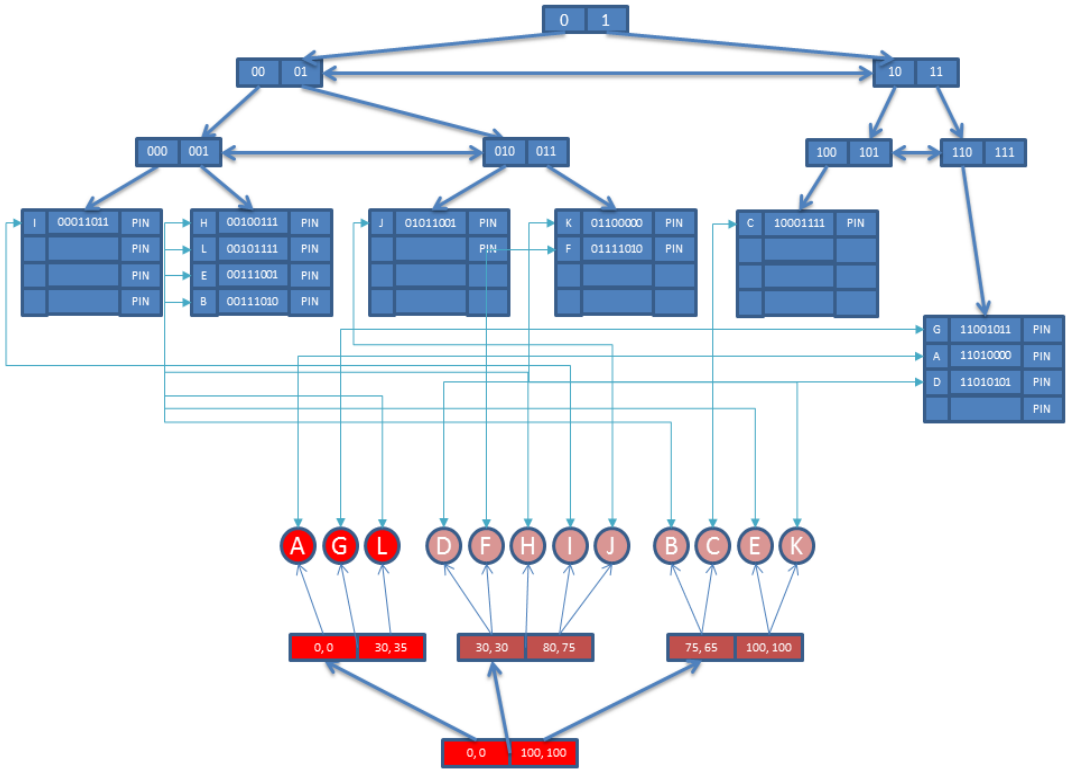

Figure 4-8: Sample content-based query.

4.2.3.2 Query by concept

Another query type our structure can answer is searching the objects by their attributes. In order to achieve this requirement, the FOOD-Index is used. First we construct the bit strings for query object attributes and then search over the FOOD-Index and retrieve all the PIs satisfying query conditions.

The X-Tree is never accessed in this type of querying because the user wants to retrieve objects by only their semantic attributes not by content descriptors.

As an example exact query, suppose that our query is as follows:

Retrieve all objects having attr1 = 7 and attr2 = 3

First of all, bitstrings for both conceptual attributes are constructed and these bitstrings are combined and final bitstring is used to locate relevant objects from FOOD-Index.

attr1 = 7 → Bitstring = 0111

attr2 = 3 → Bitstring = **0011**

Query bitstring = 0**0**1**0**11**1**

In this type of query, only FOOD-Index part of our structure is accessed as shown in Figure 4-9.

The proposed structure also supports range query. For an instance suppose that query is as follows;

Retrieve all objects having attr1 < 8 and attr2 = 3.

In range query, there will be two bitstrings for *attr1* and *attr2, start bitstring* and *stop bitstring*. In order to form final query bitstrings, start bitstrings and stop bitstrirngs are combined separately.

attr1 < 8 → *start bitstring* = 0000 , *stop bitstring* = **0111**

attr2 = 3 → *start bitstring* = 0011 , *stop bitstring* = **0011**

Query start bitstring = 0**0**0**0**1**0**1

Query stop bitstring = 0**0**1**0**1**1**1**1**

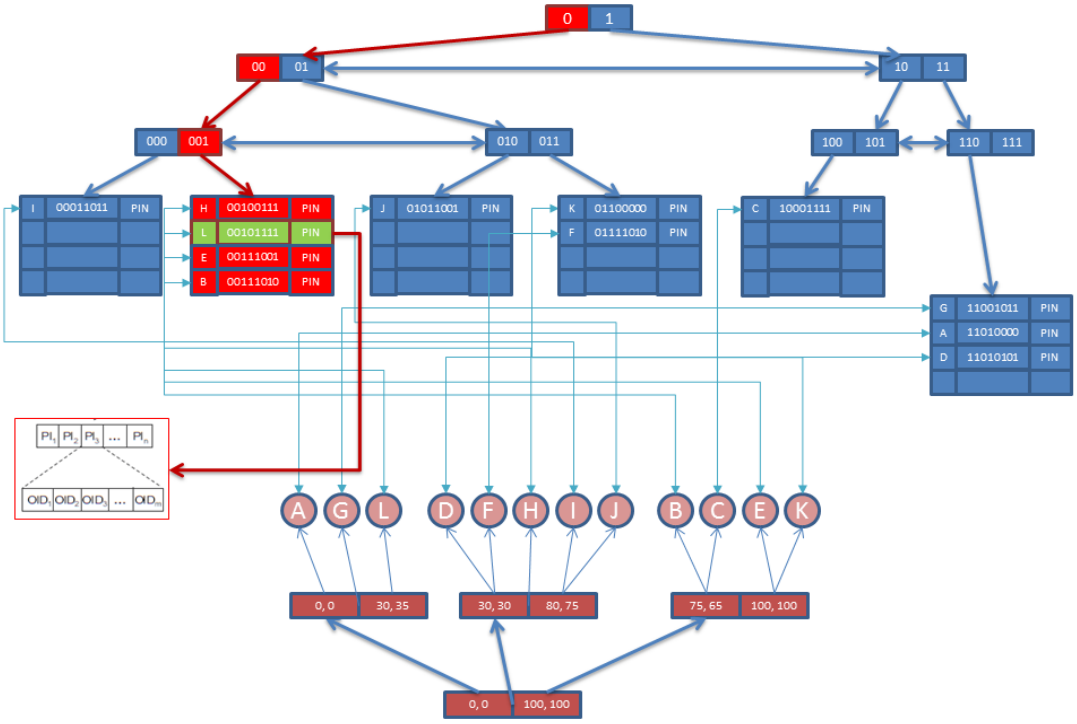

Figure 4-9: Sample concept-based query.

Then, the FOOD-Index is accessed and queried by using query's start and stop bitstrings. All objects having bitstring between *Query start* and *stop bitstrings* are returned as search result. In this example , objects *B,E* and *I* are retruned as illustrated in Figure 4-10.

4.2.3.3 .Query by concept and content

The main advantage of this structure is that searching the objects by their semantic attributes and also their content descriptors is effectively handled. Thus, this kind of search mechanism handled by using both structures together. In this kind of query, firstly the proposed structure is accessed by conceptually (i.e. using FOOD-Index

part). Moreover X-Tree part of the structure is accessed for content-based query and similarity degrees evaluated between X-Tree objects and the query object. If the similarity degree satisfies the query condition, the candidate object is added to the result list. Finally, this result list is joined with FOOD-Index results and returned to the user.

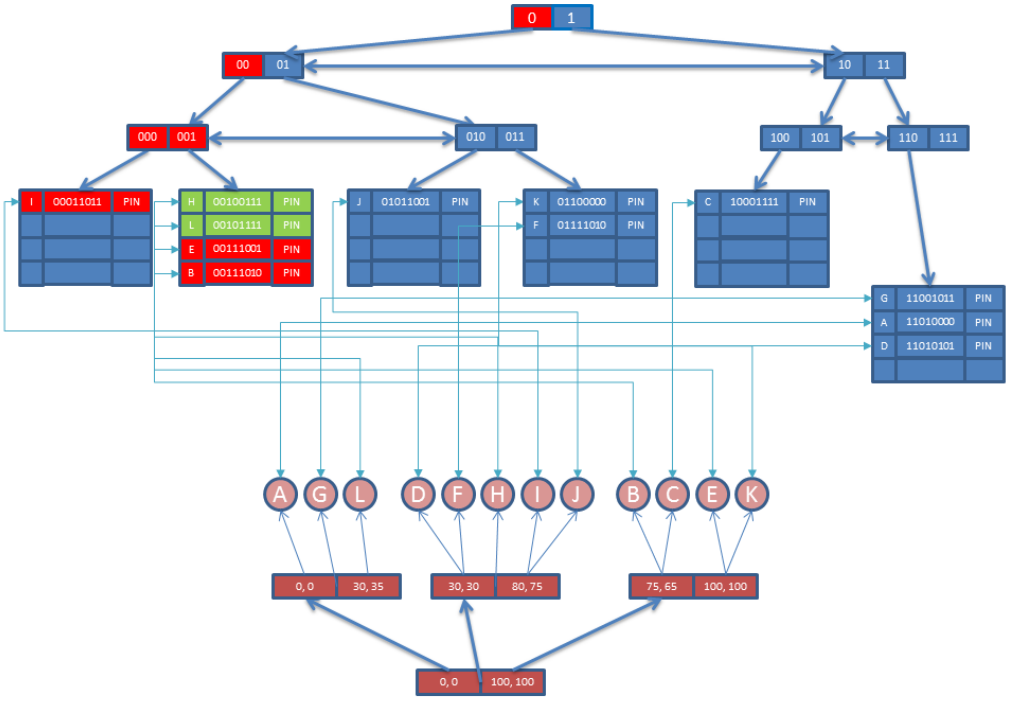

Figure 4-10: Sample concept-based range query.

For example, suppose that the query is;

Retrieve objects having attr1 = 7 and attr2 = 3 and similar to the query content

First of all, combined bit strings for conceptual part are formed as explained in Section 4.2.2.2. Then, $LMDS_{FastMap}$ is applied to query content data in order to evaluate space dimension values and these values are;

Q = {8,15,25,30}

Using both conceptual data (bitstrings) and content data (space dimension values), the query is performed on our structure accessing both FOOD-Index and X-Tree parts respectively. The result objects are joined and returned to the user as final step. In this example FOOD-Index returns *B,E,L* and *H* objects while X-Tree returns *A,G* and *L* objects as shown in Figure 4-11. Only object *L* is returned as query result to the user.

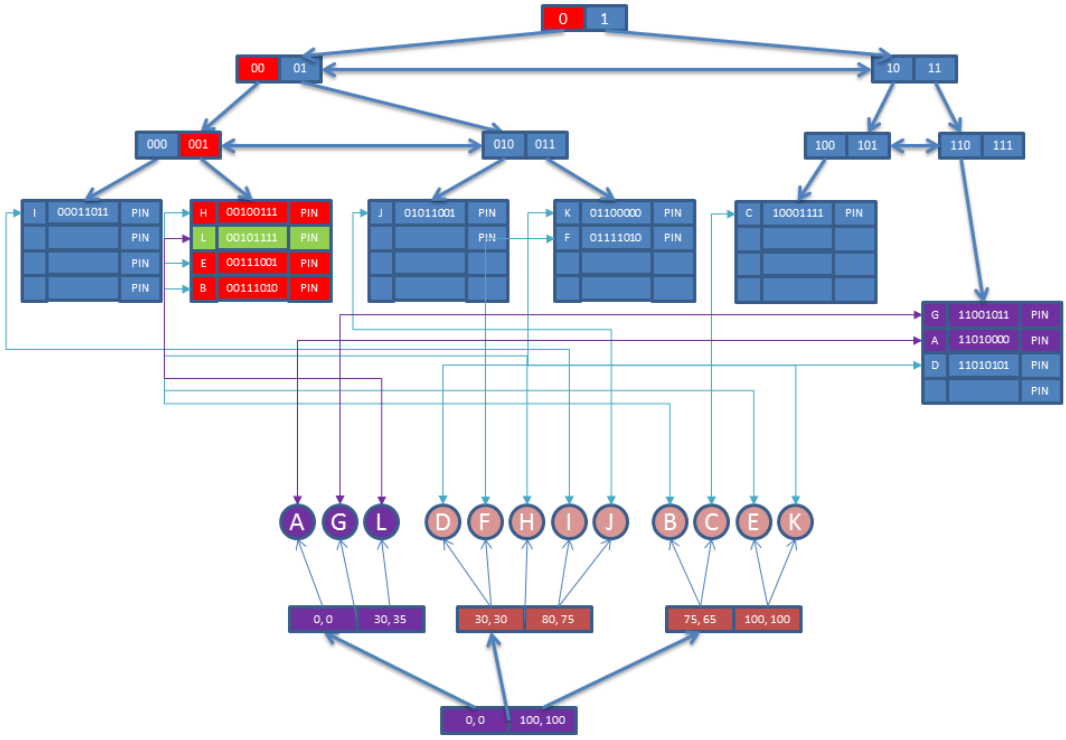

Figure 4-11: Sample concept and content-based range query.

4.2.3.4 .Fuzzy querying

Fuzzy querying method consists of fuzzy conditions in conceptual descriptors and similarity over content descriptors. Fuzzy conditions are defined as equality of indexed conceptual descriptor followed by a fuzzy term and its threshold value. The

details of fuzzy query algorithm and construction process of bit strings can be found [9].

First of all, fuzzy value of the query condition and its membership degree to each fuzzy term is evaluated by using fuzzy membership functions. Then, for each fuzzy term, the start and stop bit strings are constructed using those degree values.

For instance, we have a query such as;

Retreive all objects which has similar content to given image and also has fuzzy condition where conceptual attribute(noise) for audio is medium and threshold value is equal to 0.6 (conceptual_attr1 = medium 0.6).

Table 4.3 Similarity matrix for sample fuzzy-valued attribute (noise).

Similarity Matrix	*Low*	*Medium*	*High*
Low	1.0	0.6	0.3
Medium	0.6	1.0	0.5
High	0.3	0.5	1.0

First of all, we calculate the following values using similarity matrix in showed in

Table 4.3. These values are used to find the previous and next fuzzy terms for the part *(conceptual_attr1 = medium 0.6)* of the query.

$$\mu_{medium}(x) = 0.6 \;\rightarrow\; \mu_{low}(x) = 0.6 * 0.6 = 0.36$$

$$\mu_{medium}(x) = 0.6 \;\rightarrow\; \mu_{old}(x) = 0.6 * 0.5 = 0.3$$

Then we calculate the following threshold values to obtain the fuzzy values of the fuzzy term by using the similarity function showed in Table 4.3..

$$\mu_{medium}(x) = 0.6 \rightarrow \mu_{low}(x) = 0.6/0.6 = 1.0$$

$$\mu_{medium}(x) = 0.6 \rightarrow \mu_{medium}(x) = 0.6/1.0 = 0.6$$

$$\mu_{medium}(x) = 0.6 \rightarrow \mu_{high}(x) = 0.6/0.5 = 1.2$$

$$\mu_{medium}(x) = 0.6 \rightarrow \mu_{\{low,medium\}}(x) = 0.6/0.6 = 1.0$$

$$\mu_{medium}(x) = 0.6 \rightarrow \mu_{\{low,high\}}(x) = 0.6/0.5 = 1.2$$

$$\mu_{medium}(x) = 0.6 \rightarrow \mu_{\{\,medium,high\}}(x) = 0.6/0.5 = 1.2$$

$$\mu_{medium}(x) = 0.6 \rightarrow \mu_{\{low,medium,high\}}(x) = 0.6/0.5 = 1.2$$

Table 4.4 The Search space intervals for the condition "Noise = Medium 0.6"

No	*Fuzzy Term*	*Membership Degree*
1	Low	0.0 – 0.36
2	Low	1.0
3	Medium	0.6 – 1.0
4	High	0.0 – 0.3
5	{Low, Medium}	1.0

Finally, we are interested in the search space of fuzzy terms which are shown in Table 4.4. Then, we construct the bit strings for these space values and corresponding bit sitrings are shown in Table 4.5. Finally the index structure accessed by using these bit strings and B,C,E,F and J objects are returned as shown Figure 4-12.

Table 4.5 Constructed bit strings for the fuzzy condition

No	*Start Bit String*	*Stop Bit String*
1	00110000000000000	00110001011111111
2	00111100000000000	00111100000111111
3	01000010011000000	01010100000111111
4	10000000000000000	10000001001111111
5	01111100000000000	01111100000111111

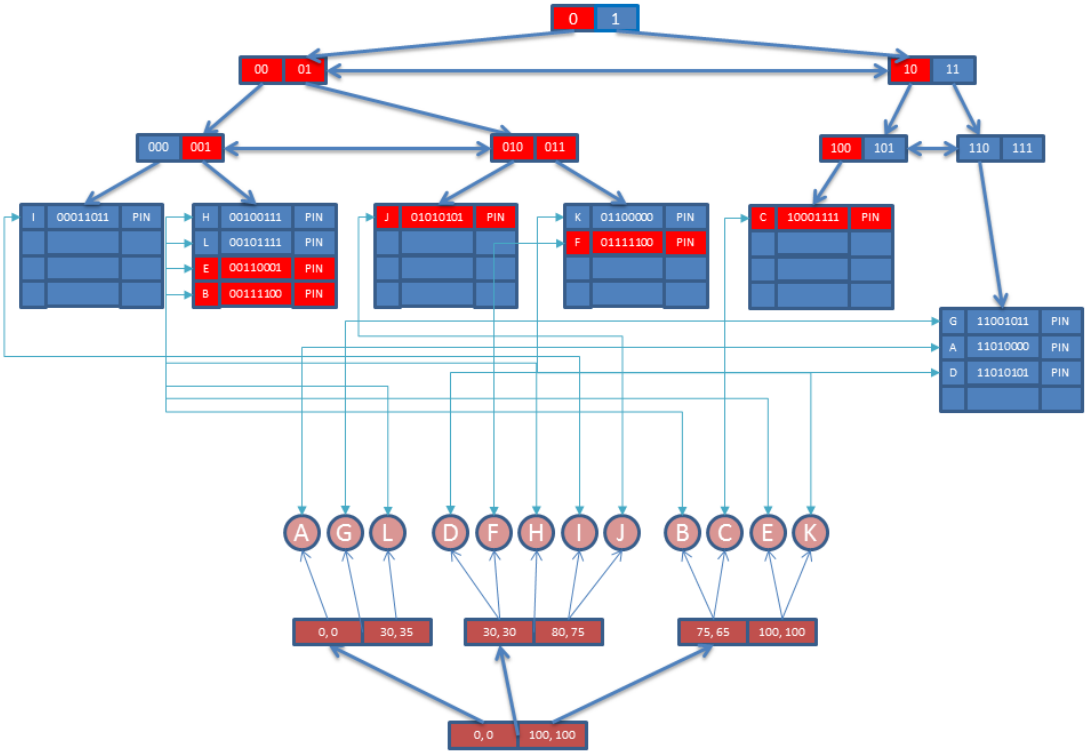

Figure 4-12: Sample fuzzy query.

CHAPTER 5

EMBEDDING CLUSTER INFORMATION INTO FOOD INDEX

5.1. Overview

In the previous chapter, we explained that X-Tree can be used for indexing content descriptors of multimedia data. The structure is formed as the integration of that X-Tree with the FOOD-Index. In this chapter, we will explain the other approach for combining content and concept descriptors in a single indexing structure. In this work, the FOOD-Index is adapted again but in order to represent low level feature descriptors, we have used a cluster based approach. After defining clusters, this cluster information is embedded to the FOOD-Index by using an approach originally used in Array-Index structure [68].

5.2. Array Index

The Array Index is a data partitioning method which uses high-dimensional data partitioning algorithm. It reads partitions and represents these partitions as vector. Each data partition is converted to vector representation m_i of relevant partition. Then a data point R is computed as reference and a distance matrix of all m_i and R pairs evaluated. This distance matrix is used for mapping high dimensional data partitions into one-dimensional distance space. This one-dimensional space is called array-index and mapped data partitions are ordered by their simialrity to R.

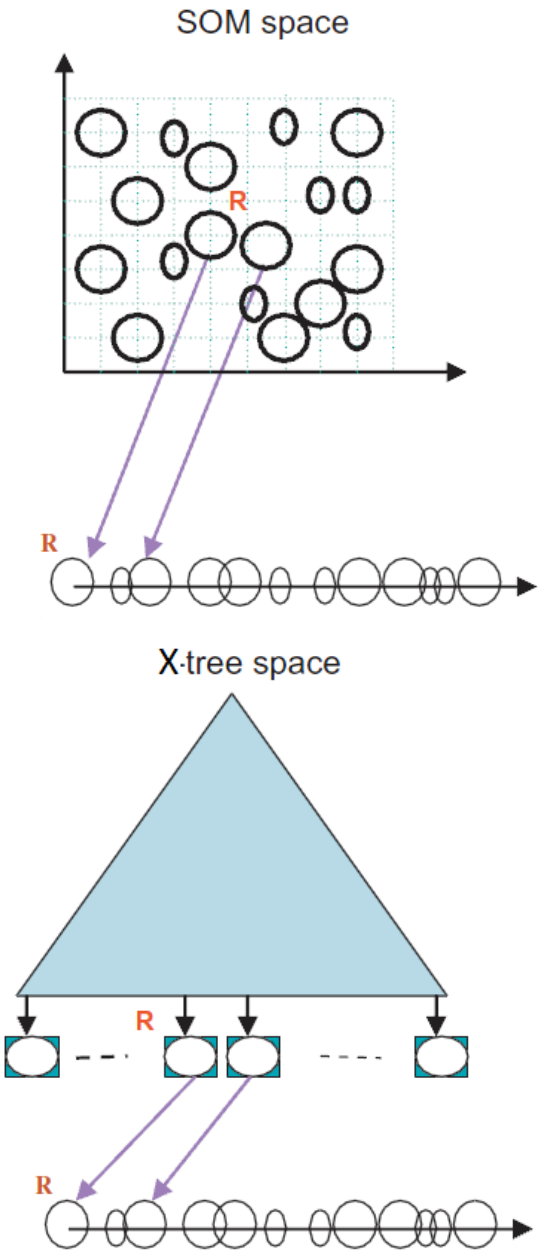

Figure 5-1: Array Index with SOM and X-Tree

The Array Index is a simple sorted array structure since number of data partitions N_c is very small than the number of points in the original space N_p. Winner Partition (WP) is the partition which is the most similar partition to the query. The query is

evaluated as by searching points in the WP first for possible correct results. Then searching continues by using array structure and checks the partitions right and left to WP. The pruning algorithm is used for efficient searching by eliminating irrelevant objects to be visited. Thus, the search algorithm has similar to the fast binary search algoirthm and has complexity $O(logN_c)$.

The example representation of Array Index for SOM and X-Tree is shown in Figure 5-1

5.3. Using Array Index with FOOD-Index

In this work, Array Index is adapted with two different DPMs; K-means clustering and X-Tree. K-means clustering is one of the simplest clustering techniques which is commonly used in biomedical and statistics. It is an unsupervised data mining algorithm used to cluster points into clusters and don't use any relationships between points except distances. The main idea of this algorithm is to define centers for each cluster which has predefined number *k*.

The k-means algorithm uses iterative process to evaluate a final *k* center of clusters. The algorithm takes data set and number of clusters (*k*) as input and produces outputs as centers of *k* clusters. First of all, it selects k cluster centers from data points at random. Then, it assigns objects to the closest cluster center by using some distance function. After this step, the centroid or mean of all objects in a cluster is calculated and these steps are repeated until the same points are assigned to same clusters.

In this structure, k-means clustering is applied to data in the low dimensional space which is represented by using LMDS. One of the main disadvantages to k-means is the number of clusters must be present as an input. Thus, the user has to be similar to the data set and be able to specify an appropriate cluster number. Moreover, different distance functions produce different cluster centers. Thus, K-means is strictly dependent to number of clusters and distance funcitons.

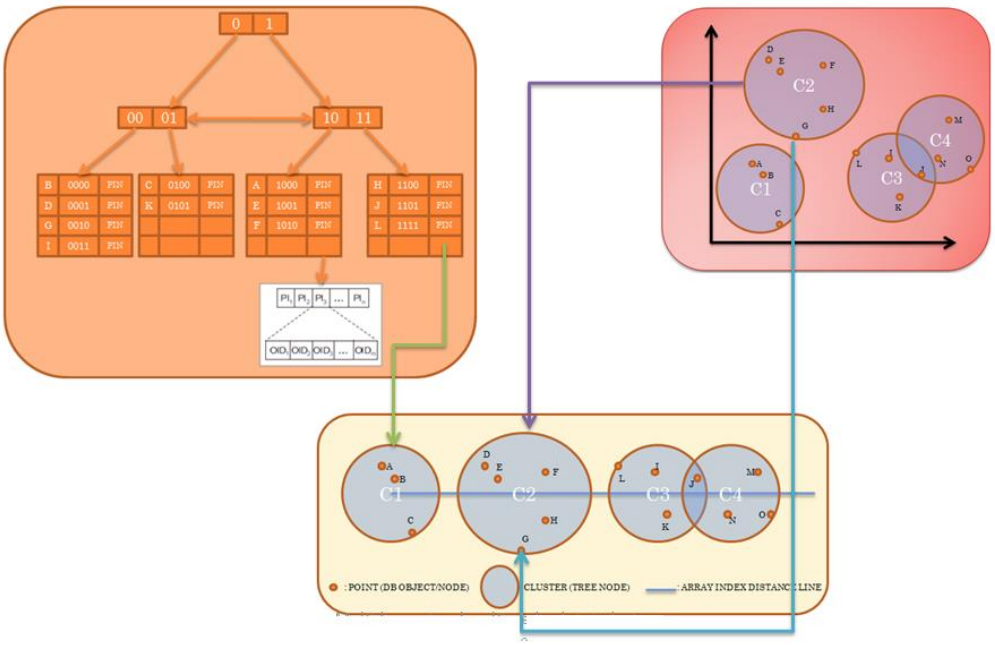

Figure 5-2: Integration of FOOD-Index with Array Index and K-Means Clustering

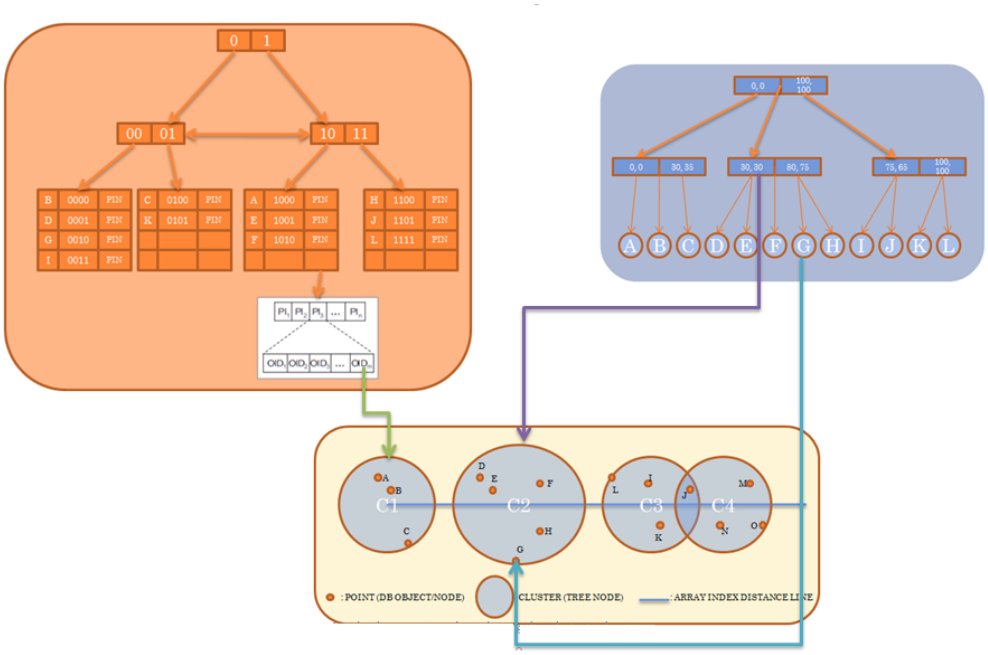

Figure 5-3: Integration of FOOD-Index with Array Index and and X-Tree.

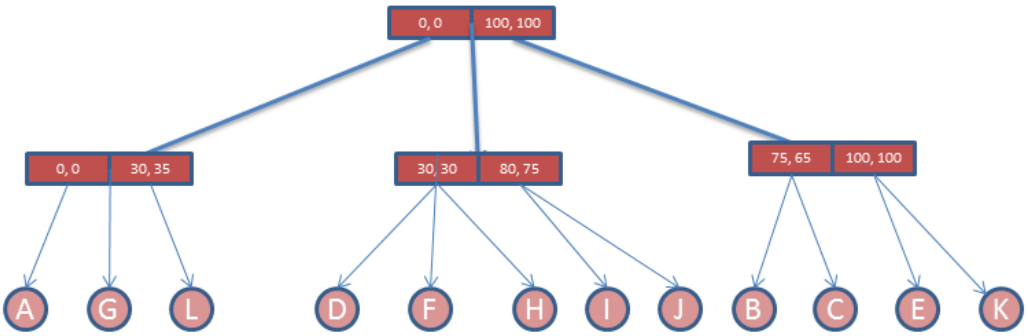

Figure 5-4: X-Tree of sample data.

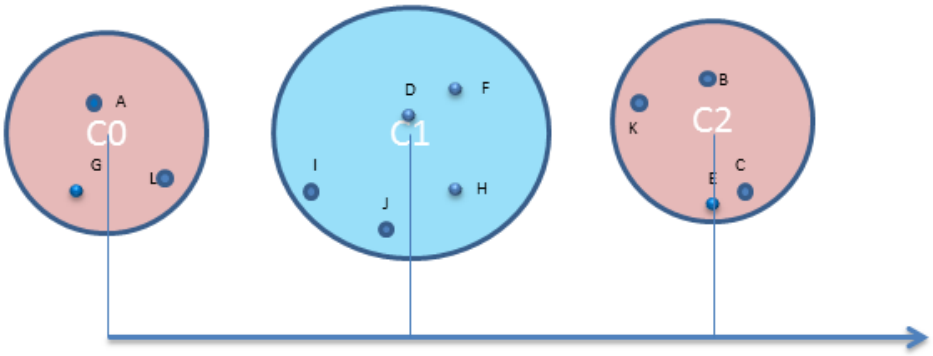

Figure 5-5: Array-Index representation of X-Tree for sample data.

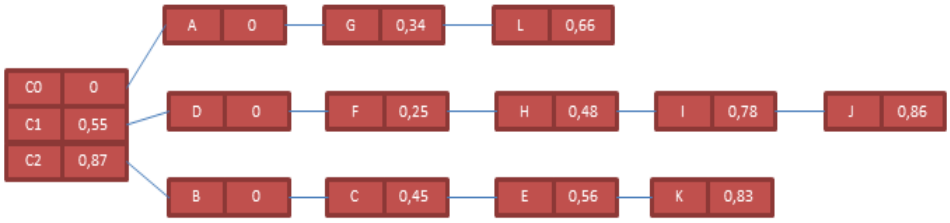

Figure 5-6: Linked list form of Array Index for sample data.

Because of this disadvantage, we have used X-Tree with LMDS as another DPM. Both approaches are shown in Figure 5-2 and Figure 5-3. After finding clusters and represent them in a single dimension as arrays of clusters, the distances of each clusters to first (reference) cluster are calculated and stored in our new structure. Moreover, the objects in each clusters are listed by their distances to center of cluster they belong to. Finally, Array Index is represented as linked list in our new structure Figure 5-6 shows an example representation of Array Index as linked list for the objects shown in Figure 5-4 and Figure 5-5.

5.3.1 Bulk loading algorithm

Bulk loading into the proposed access structure consists of the following steps;

- Firstly, low level features of objects are extracted and similarities between each pair of object calculated.
- This similarity matrix is used as input of our LMDS algorithm and the coordinates of objects in the reduced space dimension are obtained.
- After that process, we first insert object into X-Tree by using its coordinates.
- Array Index is constructed by using X-Tree.
- Objects are then inserted into the FOOD-Index by using their cluster values as another attribute. If the PIN is full then directory node is created and PINs are organized.

Using k-means clustering instead of X-Tree in the creation of Array Index, only third and fourth steps change;

- K-Means clustering is applied to the LMDS output.
- Array Index is constructed by using these cluster definitions.

Table 5.1 Sample data with cluster information.

ID	*Name*	*First Conceptual Attribute*	*Second Conceptual Attribute*	*Content Data*	*Cluster No*
1	A	8	12	0, 0, 5, 12	0
2	B	7	4	75,80,100,90	2
3	C	11	3	77,65,88,100	2
4	D	8	15	30,30,42,35	1
5	E	6	5	90,90,100,100	2
6	F	7	12	38,62,45,75	1
7	G	11	9	5,8,28,30	0
8	H	5	3	50,51,68,72	1
9	I	3	5	50,45,70,55	1
10	J	2	13	60,65,80,75	1
11	K	4	8	95,85,100,95	2
12	L	7	3	11,17,20,35	0

In the previous example in Chapter 4, X-Tree which is constructed for sample data illustrated in Figure 5-4. Array Index is constructed by using this X-Tree and shown in Figure 5-5.

After evaluating each object's cluster, our database contains new attribue called *cluster no* as shown in Table 5.1 . *Cluster no* is used as another dimension in FOOD-Index construction phase. Bit strings for first and second conceptual attributes (namely *attr1* and *attr2*) is also combined with bit stirng of this cluster information. Thus, our new structure stores content information of an object as another conceptual dimension in FOOD-Index. Example index structure for these data is shown in Figure 5-6.

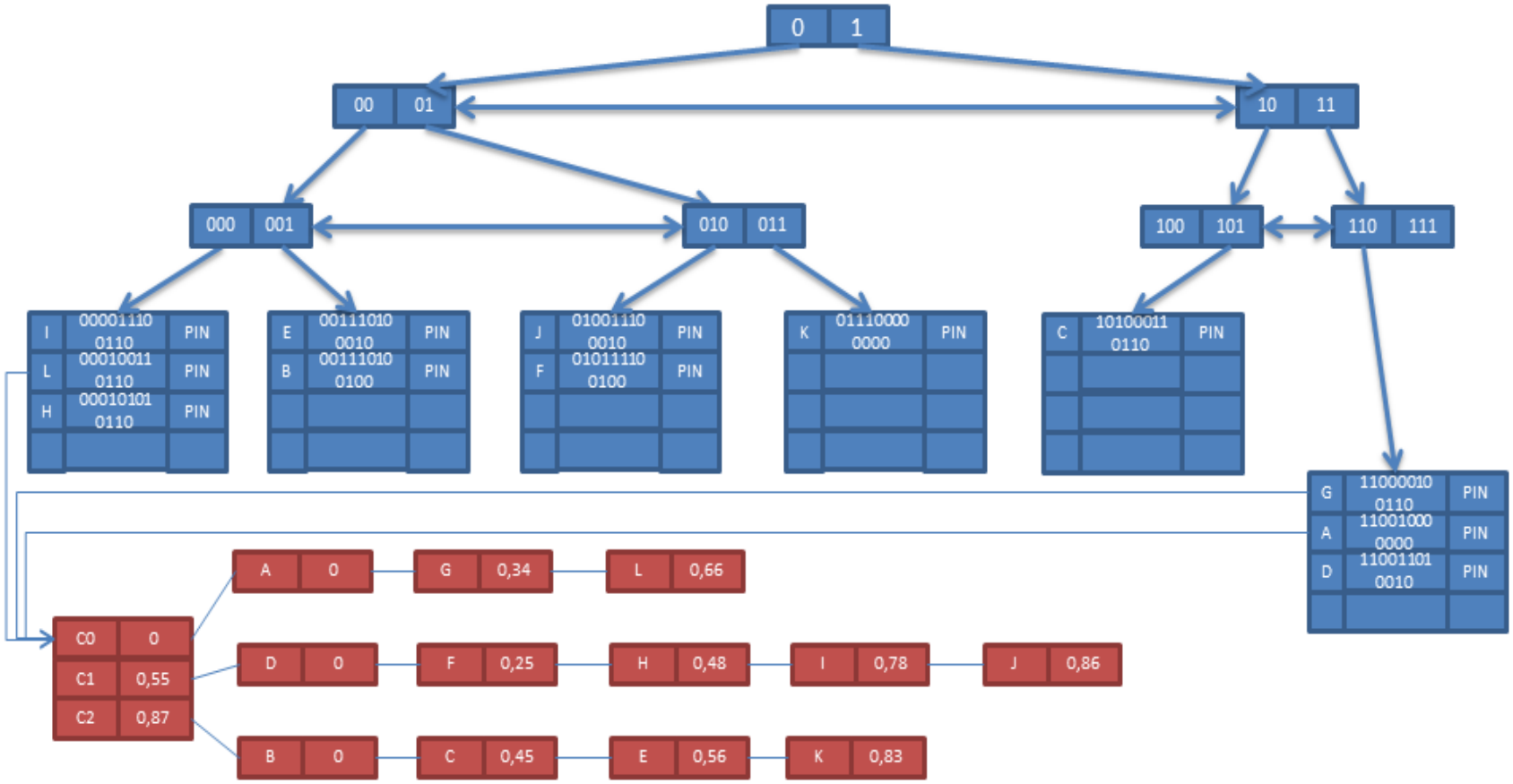

Figure 5-7: FOOD-Index with Array-Index for sample data

Similar to previous structure, bit strings for conceptual attributes are constructed but this time, another bit string is also constructed for *Cluster no*. For instance, for the *Object A*;
attr1 = 8 → bit string value = 1000

attr2 = 12 → bit string value = 1100

Cluster no = 0 → bit string value = 0000

Combined bit string for *Object A* = 110010000000

This combined bit string value is indexed in the FOOD-Index. Finally directional link between linked list for Array Index and the FOOD-Index is created. The final structure is shown in Figure 5-7.

5.3.2 Retrieval algorithm

The proposed structure can be used in order to retrieve objects that satisfy user query conditions in terms of content (visually) and concept (semantically). Thus, there are three different methods for querying the structure;

5.3.2.1 Query by content

The proposed access structure is capable of satisfying the user's visual query with the help of cluster definitions. The main steps of the algorithm are as follows;

- Firstly, low level features of query object are extracted and query object`s coordinates are found by applying LMDS algorithm.
- After that process, query object's cluster is found by calculating the distance to center cluster (R) or using X-Tree by using its coordinates.

- After candidate cluster is located, all objects are evaluated in similarity measurement with the query object.
- Objects which satisfy the query conditions are added to result list.
- If all the paths to the result objects are required, then for all objects in the result list FOOD-Index is directly accessed and PIs are returned.

Since, the user wants to retrieve just the objects themselves or whole paths to the objects, our structure can easily return relevant query results to the user.

Suppose that the user wants to find *k* objects similar to *object A* in our database (k-NN query). First of all, LMDS is applied to the query object (*q*)'s content descriptors in order to find its space dimension (or mapped dimension) values and assume that these dimension values are {8,15,25,30}. Then, the algorithm calculates the distance between the query object and the reference cluster in order to find *Candidate Cluster* (C_c). In this example, C_c is C_0. Finally, the bitstring only for cluster number is constructed since conceptual attributes are irrelevant in this type of query;

Query bitstring for C_0 = XX0XX0XX0XX0

By using this bitstring, all objects in C_c is located and retrieved according to their distances to *object A.* If *k* objects are not retrieved, then algorithm seeks and retrieves other objects from the neighbor clusters by using Array Index pruning algorithms (REF Array Index) .

In this example, objects A, G and L are returned o the user when *k* is equal to three (3) as shown in Figure 5-8.

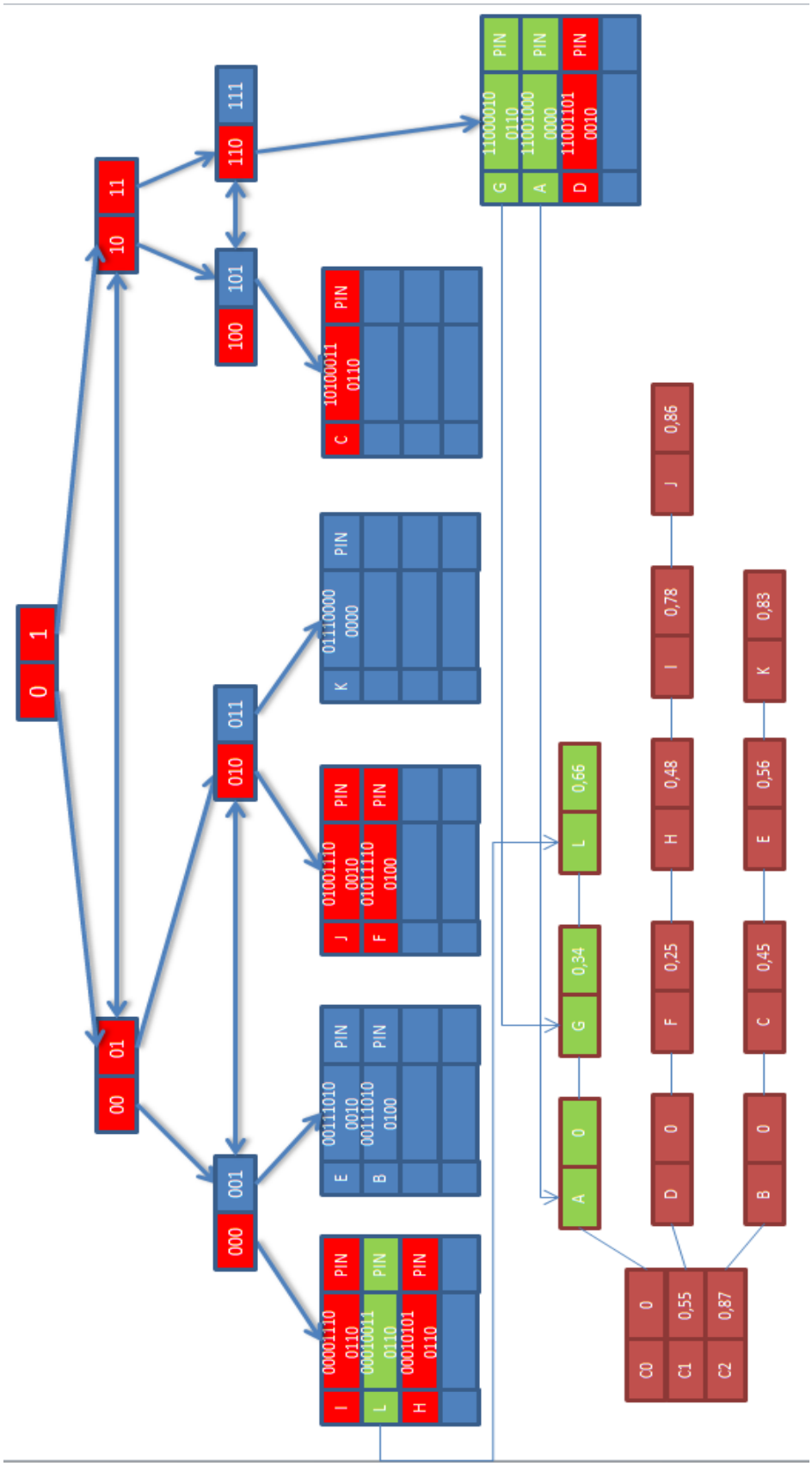

Figure 5-8: Sample content-based query.

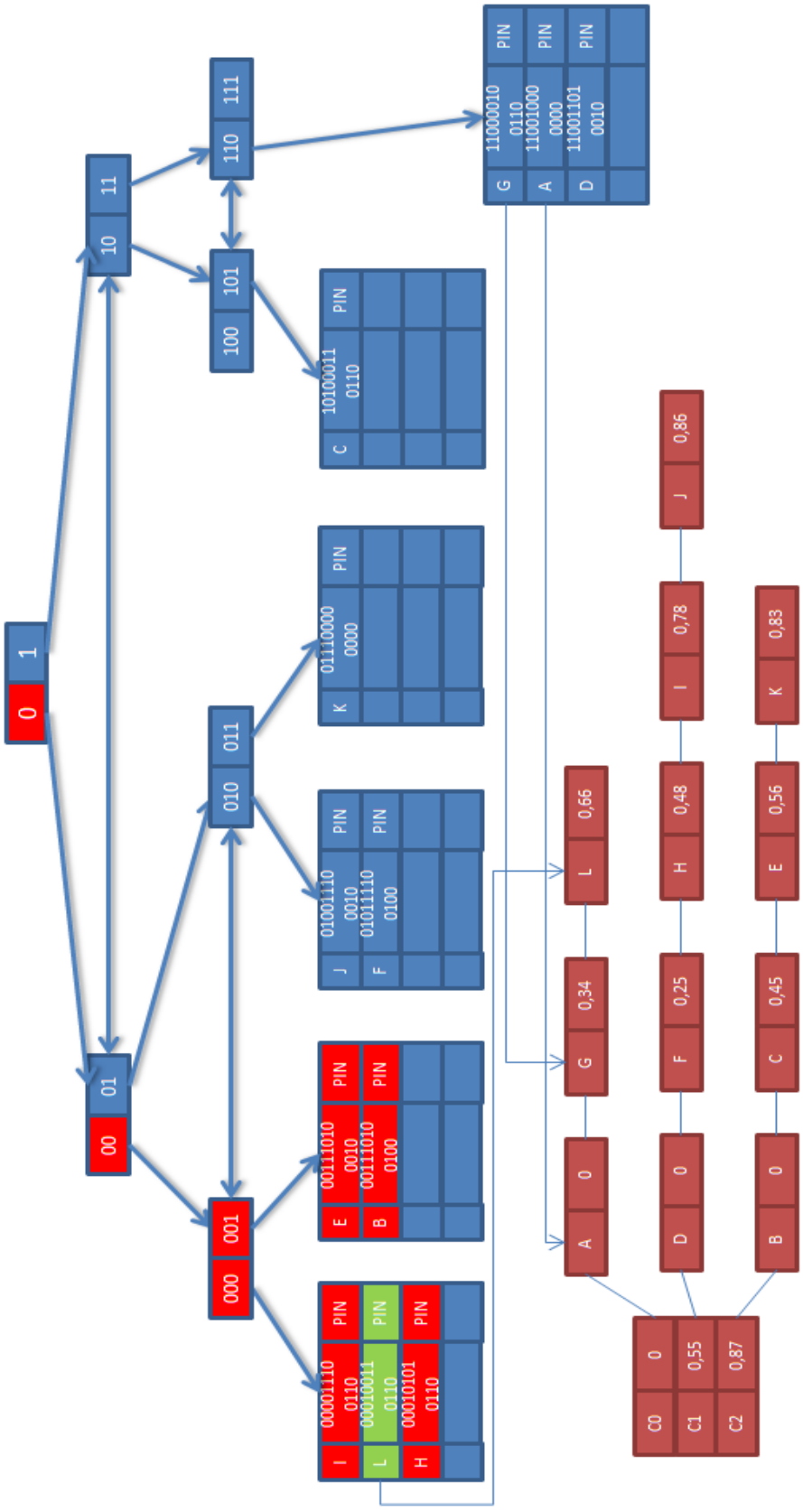

Figure 5-9: Sample concept-based query.

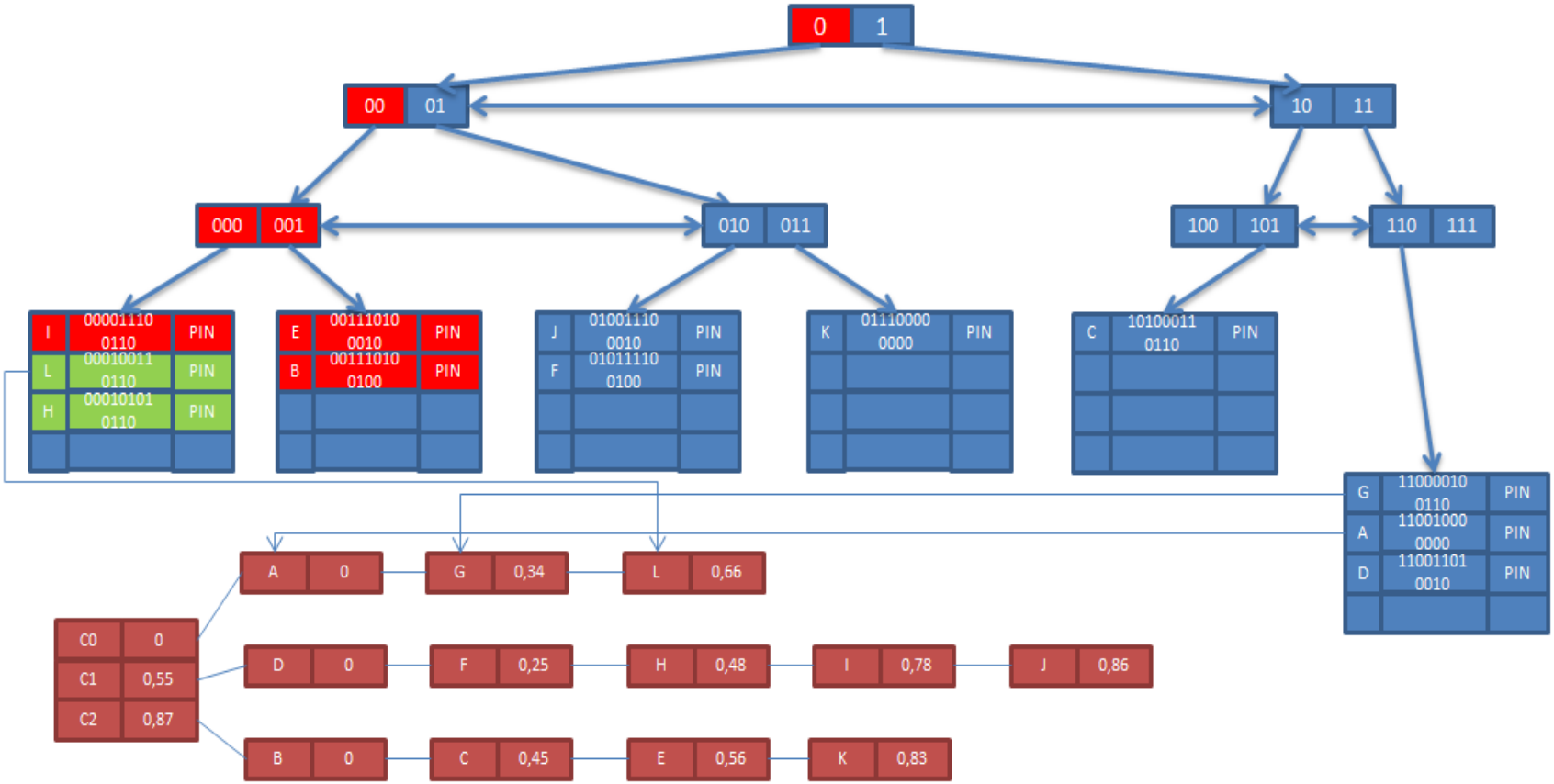

Figure 5-10: Sample concept-based range query.

5.3.2.2 Query by concept

Another query type our structure can answer is searching the objects by their attributes. In order to achieve this requirement, the FOOD-Index is used. The main steps of the algorithm are as follows;

- Construct the bit strings for query object attributes.
- Search over the FOOD-Index and retrieve all the PIs satisfying query conditions.

The cluster is never accessed in this type of querying because the user wants to retrieve objects by only their semantic attributes not by content descriptors.

As an example exact query, suppose that our query is as follows:

Retrieve all objects having attr1 = 7 and attr2 = 3

First of all, bitstrings for both conceptual attributes are constructed and these bitstrings are combined and final bitstring is used to locate relevant objects from FOOD-Index.

attr1 = 7 → Bitstring = 0111

attr2 = 3 → Bitstring = **0011**

Query bitstring = 0**0X**1**0X**1**1X**1**1X**

In this type of query, only FOOD-Index part of our structure is accessed as shown in Figure 5-9.

The proposed structure also supports range query. For an instance suppose that query is as follows;

Retrieve all objects having attr1 < 8 and attr2 = 3.

In range query, there will be two bitstrings for *attr1* and *attr2*, *start bitstring* and *stop bitstring*. In order to form final query bitstrings, start bitstrings and stop bitstrirngs are combined separately.

attr1 < 8 → *start bitstring* = 0000 , *stop bitstring* = **0111**

attr2 = 3 → *start bitstring* = 0011 , *stop bitstring* = **0011**

Query start bitstring = 0**0X**0**0X**0**1X**0**1X**

Query stop bitstring = 0**0X**1**0X**1**1X**1**1X**

Then, the FOOD-Index is accessed and queried by using query's start and stop bitstrings. All objects having bitstring between *Query start* and *stop bitstrings* are returned as search result. In this example , objects *B,E* and *I* are retruned as illustrated in Figure 5-10.

5.3.2.3 Query by concept and content

The main advantage of this structure is that searching the objects by their semantic attributes and also their visual descriptors is effectively handled. Firstly, the FOOD-Index part of the proposed structure is accessed by using combined bitstring corresponding to content and concept descriptors. After finding releavnt objects from FOOD-Index, Array Index part of the structure is accessed if necessary.

For example, suppose that the query is;

Retrieve objects having attr1 = 7 and attr2 = 3 and similar to the query content

First of all, LMDS is applied to the query object (*q*)'s content descriptors in order to find its space dimension (or mapped dimension) values and assume that these dimension values are;

Q = {8,15,25,30}. Then, the algorithm calculates the distance between the query object and the reference cluster in order to find *Candidate Cluster* (C_c). In this example, C_c is C_0. Finally, the combined bitstring for cluster number and conceptual attributes *attr1* and *attr2* is constructed;

Bitstring for *attr1* = 0111

Bitstring for *attr2* = **0011**

Bitstring for C_0 = 0000

Combined bitstring for *Query*= 0**0**01**0**011**1**01**1**0

The FOOD-Index part of our structure is accessed as shown in Figure 5-11. Only the *object L* is returned to the user in this example.

Construction and evaluation of a fuzzy query for this index structure has the same algorithm of previous one since we only use FOOD-Index part of the structures for the conceptual attirbutes and the algorithm is explained in Section 4.2.3.

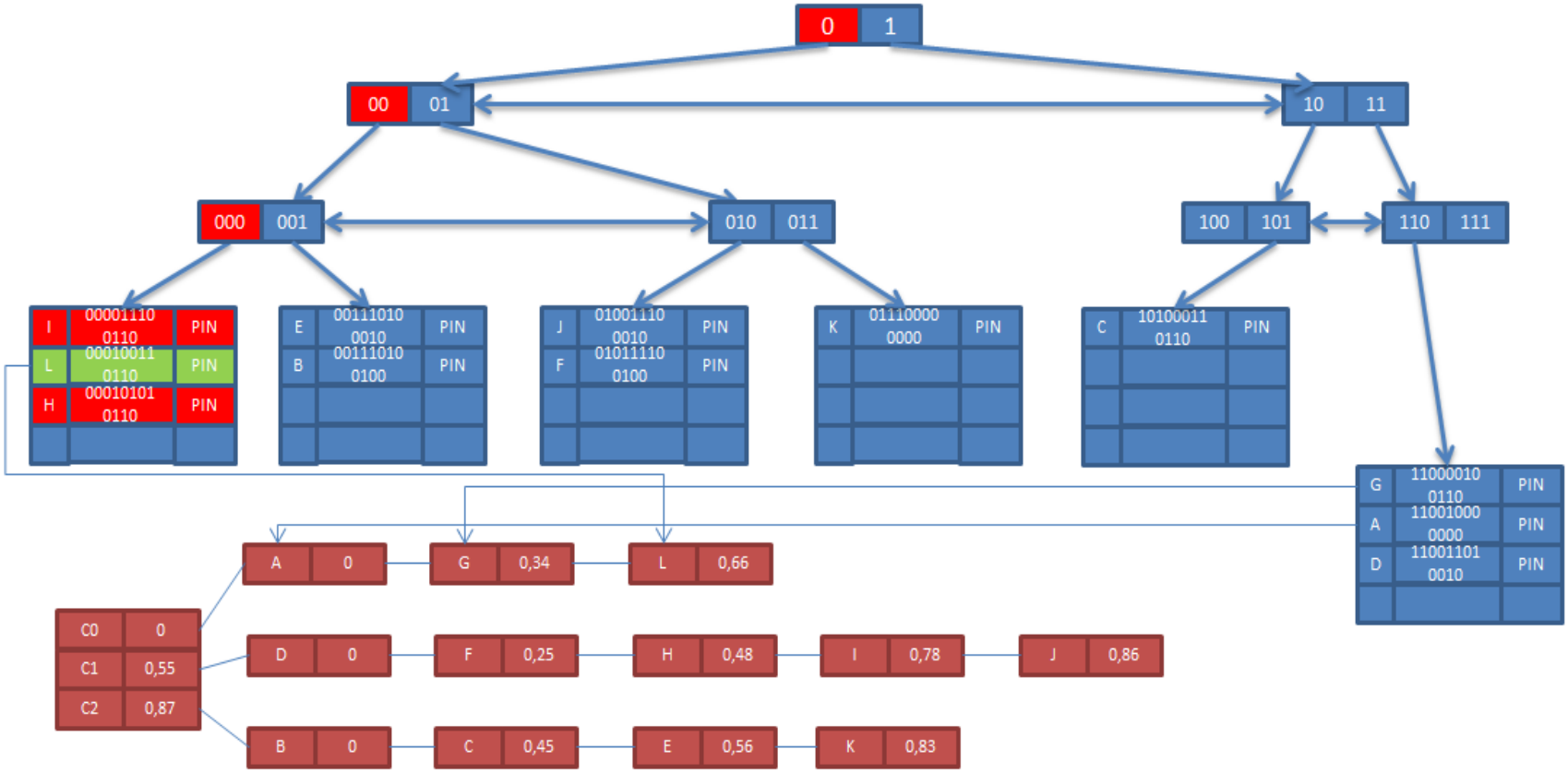

Figure 5-11: Sample concept and content

CHAPTER 6

EXPERIMENTS

In this chapter, we report the results of experiments conducted on different datasets to demonstrate the effectiveness of index structures introduced throughout this dissertation.

The performance of the index structures and similarity measurement techniques is evaluated on two different data sets; the image data set of the Corel Database [48], which consists of ten categories: mountains, people, buses, flowers, elephants, horses, architectures, dinosaurs, beach, and foods (See Figure 3-1) and a video data set.

The image data set contains ten categories each having 100 images. Each image in the entire database is used as the query object. The retrieval accuracy is evaluated using the precision and recall (PR) curves. The retrieval efficiency is evaluated using the query times for k-Nearest Neighbor (k-NN) queries, with varying k values.

The $LMDS_{FastMap}$ is adapted from [8] to use feature-based and image-based similarity measures for the embedding. The embedded vectors are indexed using the X-Tree spatial access method [10]. For completeness, we perform a Sequential Scan of the entire database to provide a baseline accuracy and time performance. In Sequential Scan, a query is compared with every image in the database and the database objects are ordered using the given similarity measure.

The Precision-Recall results for all multidimensional access methods are shown in Figure 6-1. As we see from the figure, using X-Tree with FOOD-Index has the best performance, and Array Index approach comes next since it uses X-Tree as

clustering. If we use K-means clustering, the accuracy of the system decreases since k-means clustering is dependent to k value and requires training process.

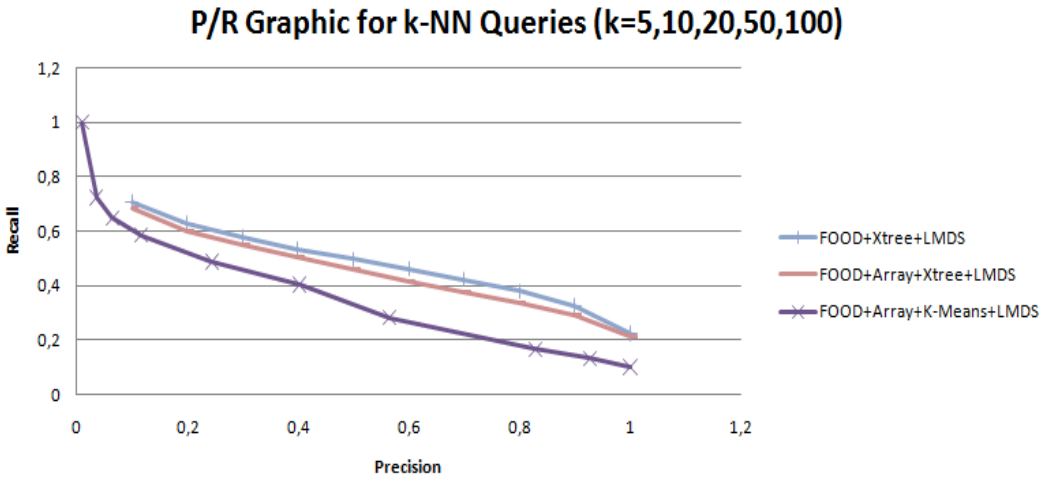

Figure 6-1 Precision/Recall graphs for k-NN queries.

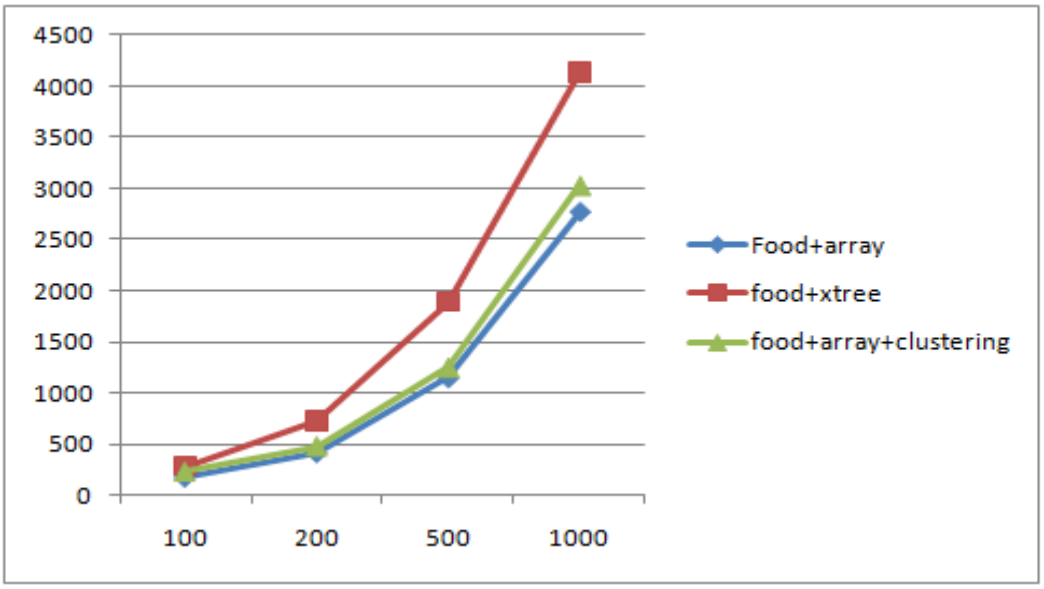

Figure 6-2 Creation times.

Another performance parameter is the time values for creation and retrieval. For comparing creation process, Figure 6-2 gives us some details. First structure requires much more time than second structure even though clustering preprocess is included since X-Tree creation phase increases the creation time.

In order to show the effect of large data, we have used a video data set which contains news videos. There are 76 *videos* in the database and each video has different numbers of *shots*. Total number of *shots* is 1571 for the data set. A *shot* carries various 'indexable' information such as *text concept, audio concept, semantic video/object concept* etc. Moreover, various numbers of *objects* can be found in a single shot. Thus, our structure should index the objects' visual features and concept attributes of each shot. Besides, *audio* is another important factor for video retrieval, so we have to take the audio information into account. Since a shot may have various audio, all the *audio* information of that single *shot* should be also indexed. The overall domain for news video is depicted in Figure 6-3. The concept (high-level) information are stored in *Audio, Text, Semantic* classes and the content (low-level) information are stored in both *Object* and *Audio* classes. There are 2796 *Text* objects, 7719 *Semantic* objects, 4428 *Visual Object* objects and 2428 *Audio* objects.

We created a new class for storing all *concept* (high-level) and *content* (low-level) information of a shot called *IndexedObject*. Our proposed index structures are built on this class. This class contains objects for all high-level and low-level pairs. Thus, the number of *IndexedObject* objects is much more than other objects in the database and our database has 57273 *IndexedObject* objects.

We indexed all these *IndexedObject* objects and evaluated some performance tests. The similarity measurement of visual objects is carried out by the distance functions proposed in Chapter 3 and for audio similarity; we have used L_1 distance function. Examples of queries and results for this system are shown in Figure 6-4, Figure 6-5, and Figure 6-6 .

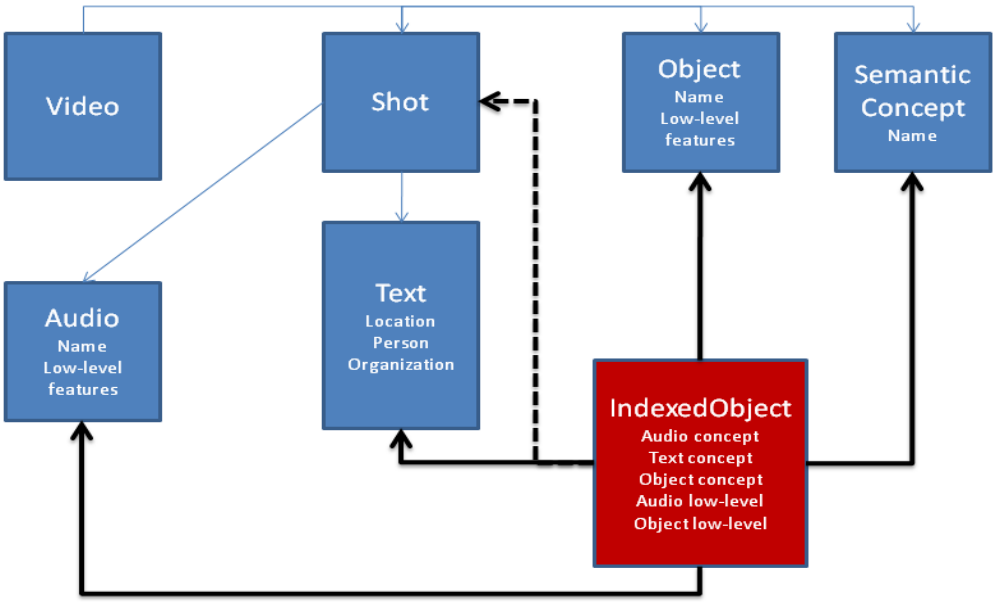

Figure 6-3 Overall domain for video data set.

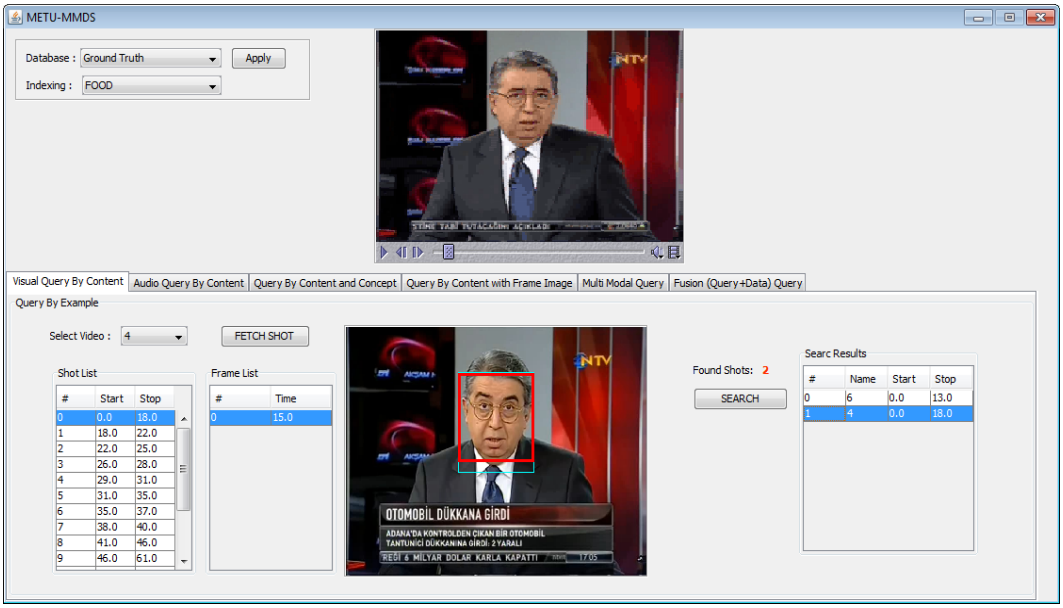

Figure 6-4 Query Example for Video dataset.

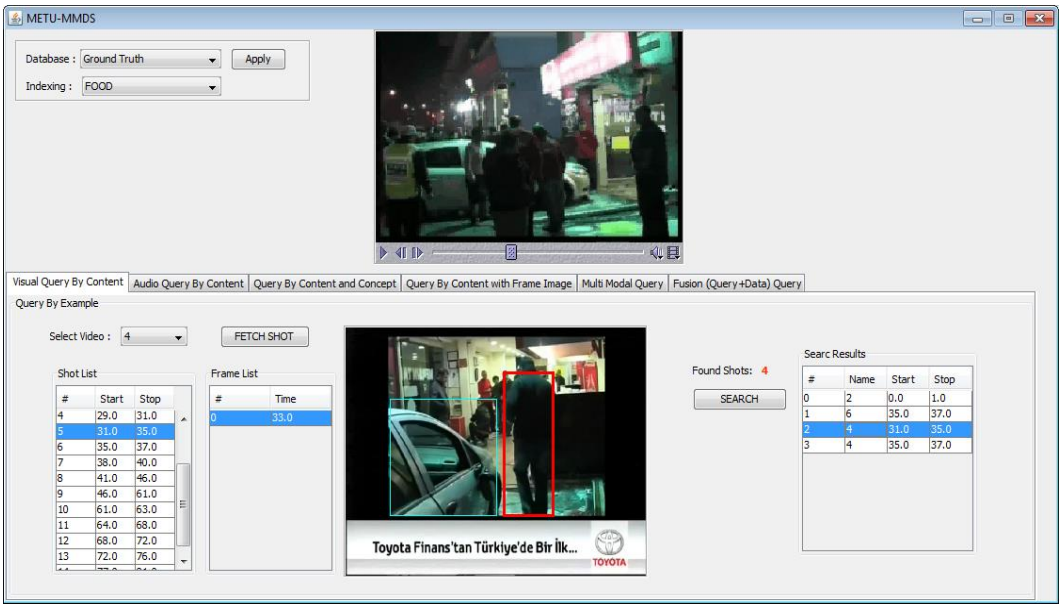

Figure 6-5 Query Example for Video dataset.

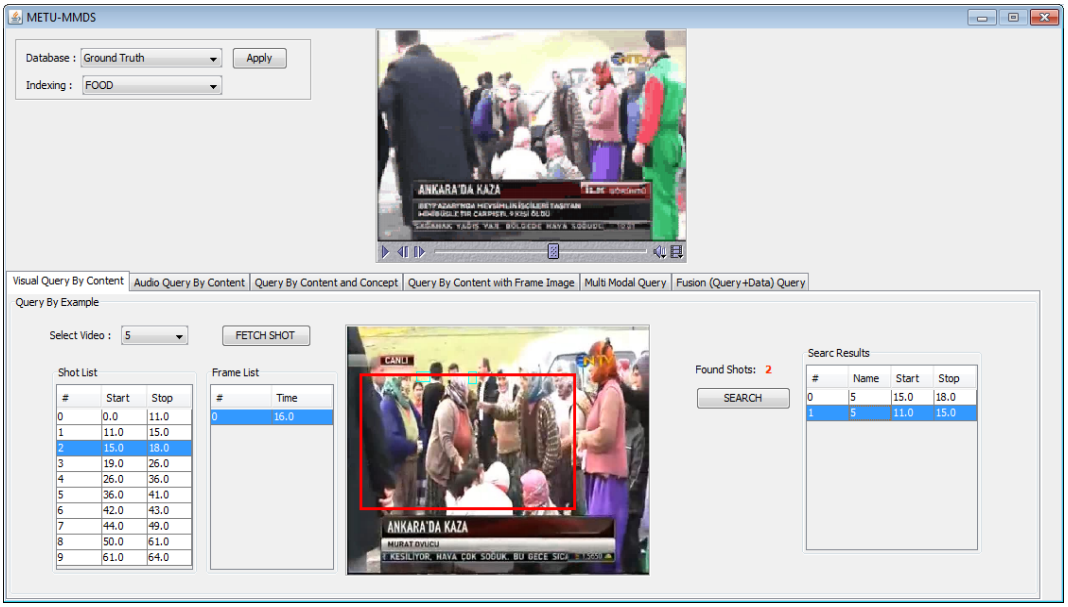

Figure 6-6 Query Example for Video dataset.

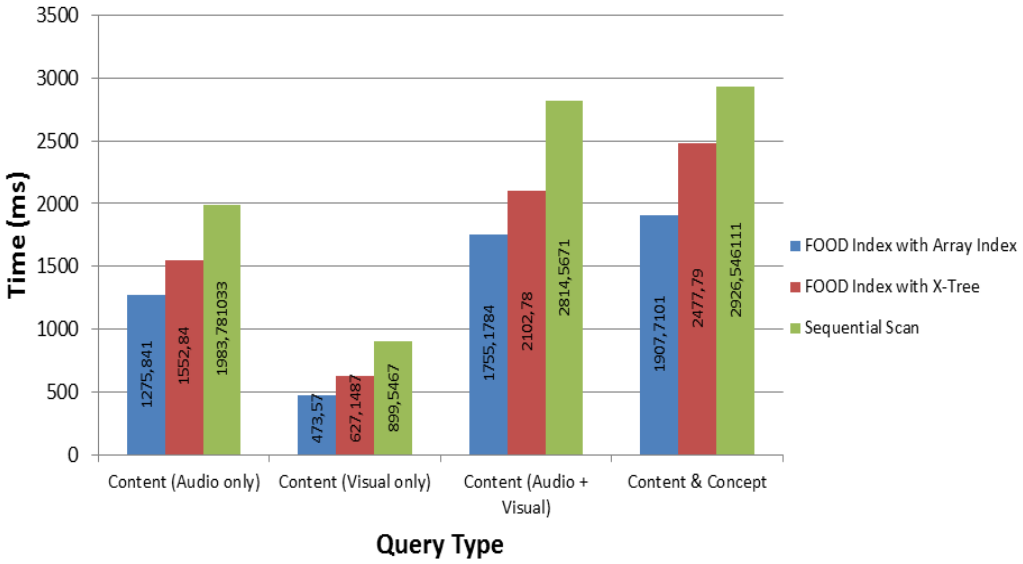

Figure 6-7 Query retrieval times.

Retrieval time is an important performance metric for the index structures. In order to evaluate the speed of the proposed structures, various numbers of queries has been done on both structures and the results are compared with sequential scan querying. The test database contains 57273 *conceptual* objects, 4428 *visual* objects and 2428 *audio* objects. The results are depicted in Figure 6-7. In this experiment, Fuzzy C-Means Clustering [69] method which is similar to K-Means Clustering method is applied in the second structure and cluster size (C) is equal to 64.

Figure 6-7 shows us that using Array Index with FOOD-Index has better search times than both FOOD Index with X-Tree and sequential scan for every query type. Audio-based content querying is conducted on only audio objects and content querying using visual descriptors are conducted on only visual objects. The figure shows us that visual based querying has better results compared to audio based querying since audio descriptors and distance calculation for these descriptors much more complex than visual descriptors and their distance functions.

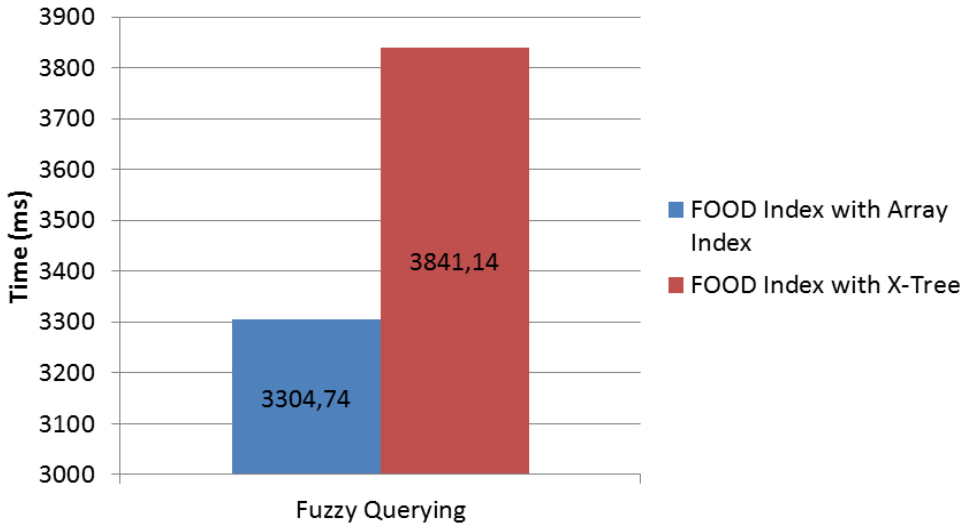

Figure 6-8: Query retrieval times for fuzzy querying.

Another query method tested thorough this study is fuzzy querying. We have evaluated query retrieval time tests for the fuzzy query;

Retreive all objects which has similar content to given image and also has fuzzy condition where conceptual attribute for audio is medium and threshold value is equal to 0.6 (conceptual_attr1 = medium 0.6).

First of all, we find the search space of fuzzy terms by using similarity matrix showed in Table 4.3 and these search space intervals are shown in Table 4.4. Then, we construct the bit strings for these space values and corresponding bit sitrings are shown in Table 4.5.

The retrieval times of example fuzzy query are shown in Figure 6-8 and results are similar to Figure 6-7 since fuzzy querying is a kind of range query and performed over conceptual attributes of the video data set.

The number of access to the indexed objects is another performance metric. Figure 6-9 shows average number of object access for k-NN content based querying. In this experiment, both audio and visual objects are searched and retrieved thus total

number of objects in our database is 6856. The results confirm retrieval time results (Figure 6-7) since queries FOOD-Index with X-tree access much more objects than FOOD-Index with Array Index structure. Thus, retrieval of objects by using first structure requires much more time than the second structure.

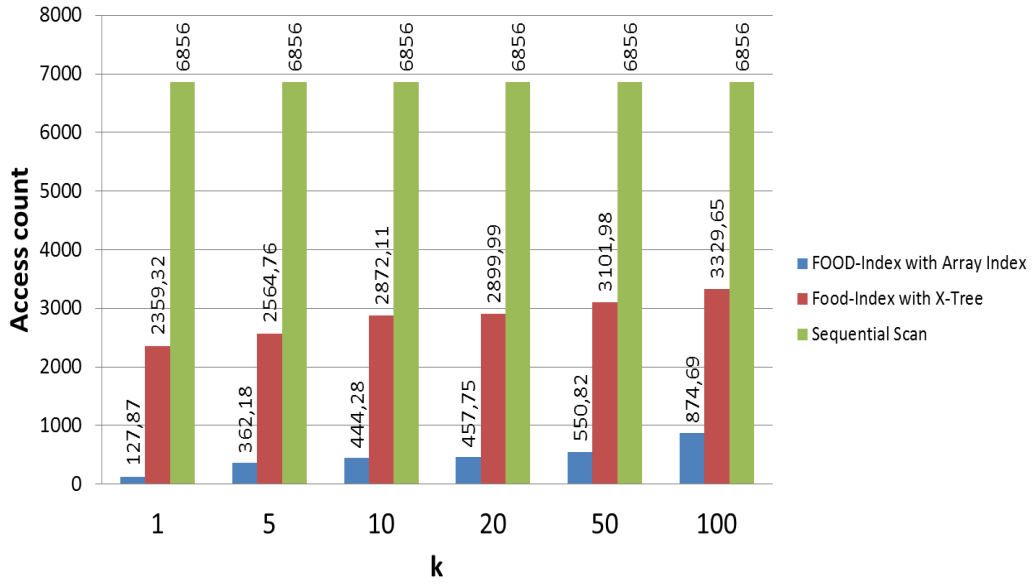

Figure 6-9 Number of object access results.

We also performed object access count tests for varying database sizes. The results depicted in Figure 6-10 show the effect of data set size over the access count for k-NN querying when k is equal to ten (10). FOOD-Index with Array Index structure accesses much less objects than other structure since it searches and locates possible result objects by selecting appropriate cluster(s). After accessing relevant cluster(s), this structure only accesses objects in that cluster(s). This efficient pruning mechanism of Array Index structure yields performance gain. On the other hand, beside object access, clusters are also visited. This requirement increases the retrieval time.

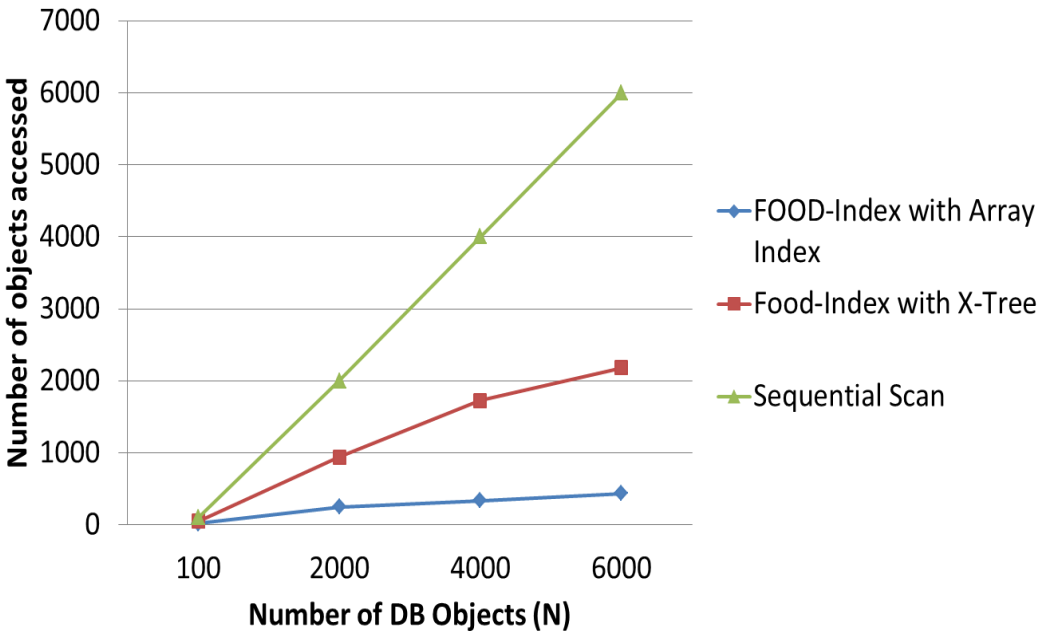

Figure 6-10 Number of object access results for varying data set sizes.

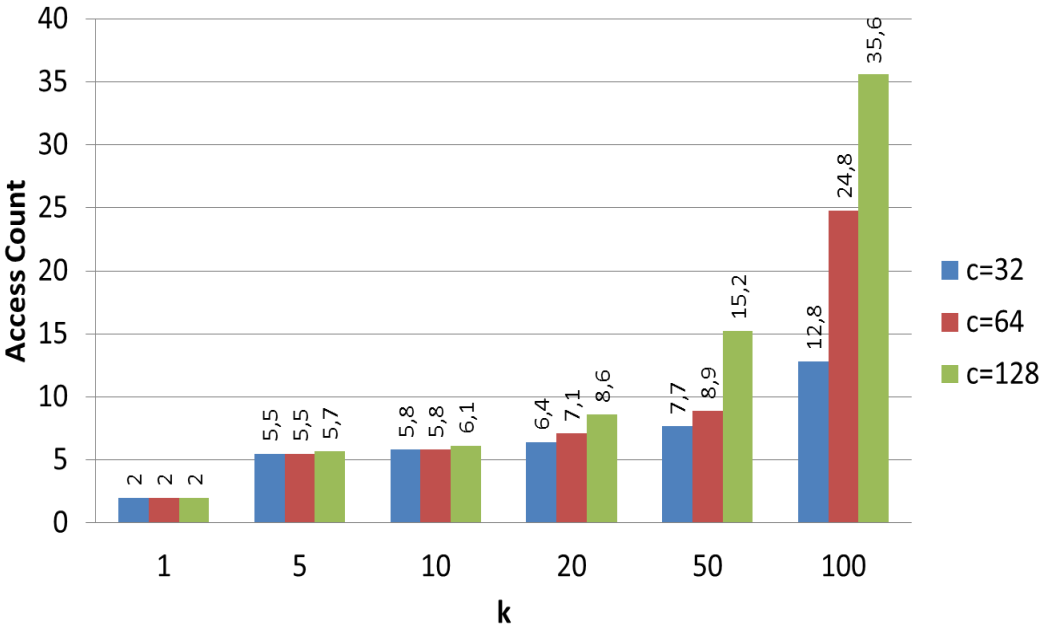

Figure 6-11 Number of cluster access results.

In order to evaluate cost of visiting clusters, we also performed average cluster access number tests on FOOD-Index with Array Index structure and the results are

shown in Figure 6-11. The results show that choosing an efficient cluster number directly affects to the query performance especially when *k* value for k-NN querying increases.

CHAPTER 7

CONCLUSION

In this thesis, we have developed a novel indexing mechanism which uses an underlying clustering algorithm. Our index scheme owns the characteristics that it not only considers content of the multimedia data, but also take conceptual descriptors of these data into account. Thus our index mechanism gets rid of semantic gap problem by combining both content and concept descriptors in a single structure.

In order to construct the proposed structure, we firstly have extracted low-level content features of multimedia data by using MPEG-7 descriptors and compared feature-based comparison with image-based comparison methods in order to find which method has better search performance for content-based retrieval on image data. The experiments on this comparison shows us that feature-based approach has better performance than whole image-based approach in terms of both query accuracy and retrieval time. However, the descriptors which are used in feature-based approach are high dimensional data, thus we have performed dimension reduction technique to be able to index these descriptors in low-dimensional space. The experiments show us that using multidimensional scaling in our structures for content descriptors has accuracy comparable to that of the retrieval performed in the original space. Moreover, multi-dimensional scaling can reduce the retrieval problem to a spatial-indexing task, where queries can be performed orders of magnitude faster than distance based indexing methods.

After representing high-dimensional feature descriptors in low-dimensional space, we have applied clustering algorithm in order to find each multimedia object's cluster information as another conceptual attribute.

For the semantic information, we have used some annotation based descriptors. These descriptors are converted into bit strings and stored in FOOD-Index, which is multidimensional index structure for class attributes and also supports fuzzy and crisp queries for object oriented databases. Cluster information of multimedia object is also converted into bit string, combined with bit string representations of concept descriptors and embedded into the proposed index structure.

In addition to the proposed index structure, another index structure which combines FOOD-Index for concept descriptors and spatial indexing mechanism called X-Tree for content descriptors in a loosely coupled way is implemented to demonstrate the comparison of retrieval performance of our proposed index structure.

The studies conducted throughout this thesis have shown that the proposed index structure is a beneficial approach for improving the retrieval performance. It has been experienced that representing content descriptors in low-dimensional space outperforms the model which represents the low-level features in their original space. Moreover, defining clusters for these low-dimensional content descriptors and use of this cluster information in our study as another index attribute similar to concept descriptors has better query accuracy values than second structure.

The speed performance is another crucial factor for such a retrieval system and the test results show that the both structure performs better than sequential scan in terms of access count since these structures reduce the search space from number of the objects to the cluster size. For queries which include concept and content descriptors together, our study has better performance than the other structure.

REFERENCES

[1] Y. Rui, T. S. Hang and S. Chang, Image retrieval: Current technique, promising directions, and open issues., *Journal of Visual Comm. and Image Representation*, pp. 39-62, 1999.

[2] M. Koskela, J. Laaksonen and E. Oja, Comparison of Techniques for CBIR., *Proc. of the 12th Scandinavian Conf. on Image Analysis*, 2001.

[3] M. Madugunki, D. Bormane, S. Bhadoria and C. Dethe, Comparison of different CBIR techniques, *ICECTECH*, 2011.

[4] M. F. Mokbel, T. M. Ghanem and W. G. Aref, Spatio-temporal Access Methods, *IEEE Data Engineering Bulletin*, cilt 26, pp. 40-49, 2003.

[5] L. V. Nguyen-Dinh, W. G. Aref and M. F. Mokbel, Spatio-Temporal Access Methods: Part 2 (2003 - 2010), *IEEE Data Eng. Bull.*, cilt 33, no. 2, pp. 46-55, 2010.

[6] V. Gaede and O. Gunther, Multidimensional Access Methods, *ACM Computing Surveys (CSUR)*, cilt 30, no. 2, pp. 170 - 231, 1998.

[7] C. Bohm, S. Berchtold and D. A. Keim, Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases, *ACM Computing Survey*, cilt 33, no. 3, pp. 322-373, 2001.

[8] A. Saçan and İ. Toroslu, Approximate Similarity Search in Genomic Sequence Databases Using, *SISAP '08: Proceedings of the First International Workshop*

on Similarity Search and Applications (sisap 2008), 2008.

[9] A. Yazıcı, Ç. İnce and M. Koyuncu, An Indexing Technique for Similarity-Based Fuzzy Object-Oriented Data Model, *Flexible Query Answering Systems*, 2004, pp. 334-347.

[10] S. Berchtold, D. A. Keim and H.-P. Kriegel, The X-tree: An Index Structure for High-Dimensional Data, *VLDB '96 Proceedings of the 22th International Conference on Very Large Data Bases*, 1996.

[11] J. H. Lee, H. Kim and W. Kim, Video Image Retrieval System based on MPEG-7 (VIRS), *Proceedings of the International Conference on Information Technology:*, 2003.

[12] E. E. Tekinalp, O. Onur, M. Soysal and Y. Yasaroglu, A MPEG-7 compliant Video Management System:BilVMS, *Proc. of 4th European Workshop on Image Analysis for Multimedia Interactive*, 77-80, 2003.

[13] Y. M. Hironobu, H. Takahashi and R. Oka, Image-to-word transformation based on dividing and vector quantizing, *MISRM'99 First International Workshop on Multimedia Intelligent Storage and Retrieval Management*, pp. 405-409, 1999.

[14] P. Duygulu, K. Barnard, N. Freitas and D. Forsyth, Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary, *Seventh European Conference on Computer Vision*, pp. 97-112, 2002.

[15] Y. Li, X. Wan and C. Jay, Image DBs, *Introduction to Content-Based Image Retrieval- Overview of Key Techniques*, 2002, pp. 261-284.

[16] T. Sikora, The MPEG-7 Visual Standard for Content Description-An Overview, *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 696-702, 2001.

[17] M. Hacid, C. Decleir and J.Kouloumdjian, A Database Approach for Modeling and Querying Video Data, *IEEE Transactions on Knowledge and Data Engineering*, pp. 729-750, 2000.

[18] A. Ekin, A. M. Tekalp and R. Mehrotra, Integrated Semantic-Syntactic Video Modeling for Search and Browsing, *IEEE Transactions on Multimedia*, cilt 6, no. 6, pp. 839-851, 2004.

[19] J. Nievergelt, H. Hinterberger and K. C. Sevcik, The grid file: An adaptable, symmetric multikey file structure, *ACM Transactions on Database Systems*, cilt 9, no. 1, pp. 38-71, 1984.

[20] H. Lin, P. Huang and K. Hsu, A new indexing method with high storage utilization and retrieval efficiency for large spatial databases, *Inf. Softw. Technol.*, cilt 49, no. 8, pp. 817-826, 2007.

[21] J. T. Robinson, The K-D-B-tree: a search structure for large multidimensional dynamic indexes, *SIGMOD '81: Proceedings of the 1981 ACM SIGMOD international conference on Management of data*, pp. 10-18, 1981.

[22] H. -W. Six, P. Widmayer and A. Henrich, The LSD tree: spatial access to multidimensional and non-point objects, *VLDB '89: Proceedings of the 15th international conference on Very large databases*, pp. 45-53, 1989.

[23] D. Lomet and B. Salzberg, The hB-tree: A multiattribute indexing method with good guaranteed performance, *ACM Transactions on Database Systems*, cilt 15, no. 1, pp. 625-658, 1990.

[24] T. Urruty, F. Belkouch and C. Djeraba, KPYR: An Efficient Indexing Method, *IEEE International Conference on Multimedia and Expo*, pp. 1448-1451, 2005.

[25] S. Berchtold, C. Bohm and H. Kriegel, The Pyramid-Technique: Towards Breaking the Curse of Dimensionality, *Proc. ACM SIGMOD International*

Conference on Management of Data, pp. 142-143, 1998.

[26] S. H. V. Jagadish, B. C. Ooi, K. Tan, C. Yu and R. Zhang, iDistance: An Adaptive B+-Tree Based Indexing Method for Nearest Neighbor Search, *ACM Transactions on Database Systems*, cilt 30, no. 2, pp. 364 - 397, 2005.

[27] Y. Zhuang, Y. Zhuang, Q. Li, L. Chen and Y. Yu, Indexing High-Dimensional Data in Dual Distance Spaces: A Symmetrical Encoding Approach, *Proc. EDBT 2008, 11th International Conference on Extending Database Technology*, 2008.

[28] C. H. Go, A. Lim, B. C. Ooi and K. Tan, Efficient Indexing of High-Dimensional Data Through Dimensionality Reduction, *Data Knowl. Eng.,* cilt 32, no. 2, pp. 115-130, 200.

[29] R. Weber, H. Schek and S. Blott, A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces, 1998.

[30] H. Lu, B. Chin, C. Ooi, H. T. Shen and X. Xue, Hierarchical Indexing Structure for Efficient Similarity Search in Video Retrieval, *IEEE Transactions on Knowledge and Data Engineering,* cilt 18, no. 11, pp. 1544 - 1559, 2006.

[31] E. Tuncel, H. Ferhatosmanoglu and K. Rose, VQ-Index: An Index Structure for Similarity Searching in Multimedia Databases, *Proc. of 10th ACM International Conference on Multimedia,* pp. 543-552, 2002.

[32] B. Cui, H. Shen, J. Shen and K. Tan, Exploring Bit-Difference for Approximate KNN Search in High-dimensional Databases, *Proc. of the 16th Australasian database conference,* pp. 165-174, 2005.

[33] J. Nang and J. Park, An Efficient Indexing Structure for Content Based Multimedia Retrieval with Relevance Feedback, *Proc. of the 2007 ACM symposium on Applied computing*, pp. 517 - 524, 2007.

[34] C. Calistru, C. Riberio and G. David, Multidimensional Descriptor Indexing: Exploring the BitMatrix, *CIVR06*, 2006.

[35] B. Gonalves, C. Calistru, C. Riberio and G. David, An Evaluation Framework for Multidimensional Multimedia Descriptor Indexing, *Proc. of the 23rd International Conference on Data Engineering Workshops,ICDE 2007*, 2007.

[36] P. Ciaccia, M. Patella and P. Zezula, M-tree: An efficient access method for similarity search in metric spaces, *Proc. of the 23rd VLDB International Conference*, 1997.

[37] C. Jr., A. Traina, B.Seeger and C. Faloutsos, Slimtrees: High Performance Metric Trees Minimizing Overlap Between Nodes, *Proc. of the 6th International Conference on Extending Database Technology (EDBT 2000)*, 2000.

[38] J. Yuan, L. Duan, Q. Tian and C. Xu, Fast and Robust Short Video Clip Search Using an Index Structure, *Proc. of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, 2004.

[39] S. Kiranyaz and M. Gabbouj, Hierarchical Cellular Tree: An Efficient Indexing Scheme for Content-Based Retrieval on Multimedia Databases, *IEEE Transactions on Multimedia*, cilt 9, no. 1, pp. 102 - 119, 2007.

[40] G. R. Hjaltason and H. Samet, Index-driven similarity search in metric spaces (Survey Article), *ACM Trans. Database Systems*, cilt 28, no. 4, pp. 517 - 580, 2003.

[41] TREC Video Retrieval Evaluation: TRECVID, https://trecvid.nist.gov/. last visited on 23 12 2017].

[42] International Organization of Standardisation, MPEG-7 Overview (ver. 9), 2003.

[43] H. Eidenberger, How good are the visual MPEG-7 features?, *Visual Communications and Image Processing*, SPIE, 2003.

[44] R. R. Yager, On ordered weighted averaging aggregation operators in multi-criteria decision making, *IEEE Transactions on on Systems, Man and Cybernetics*, cilt 18, no. 1, pp. 183 - 190, 1988.

[45] Y. Rubner, C. Tomasi and L. J. Guibas, The Earth Mover's Distance as a Metric for Image Retrieval, *International Journal of Computer Vision*, cilt 40, no. 9, pp. 99-121, 2000.

[46] B. Zitova and J. Flusser, Image registration methods: a survey, *Image and Vision Computing*, cilt 21, no. 11, pp. 977-1000, 2003.

[47] Java Image Processing Cookbook homepage, http://www.lac.inpe.br/JIPCookbook/index.jsp. last visited on 18 11 2012].

[48] Corel database homepage, http://www.corel.com. last visited on 21 04 2008].

[49] ImageNet Home web site, http://image-net.org/. last visited on 25 11 2009] .

[50] Multimedia Database Research Group Homepage, http://multimedia.ceng.metu.edu.tr/index.php/en/home/. last visited on 01 02 2011].

[51] B. Manjunath, P. Salembier and T. Sikora, Introduction to MPEG-7: Multimedia Content Description Interface, 2002.

[52] Mpeg7XM Software, http://www.lis.ei.tum.de/research/bv/topics/mmdb.htm. last visited on 12 04 2003].

[53] T. Ojala, M. Aittola and E. Matinmikko, Empirical Evaluation of MPEG-7 XM Color Descriptors in Content-Based Retrieval of Semantic Image Categories,

Proc. of 16th Int, Con, on Pattern Recognition, 2002.

[54] K. K. Guner, MPEG-7 Compliant ORDBMS Based Image Storage and Retrieval System, METU, 2004.

[55] J. C. Lagarias, J. A. Reeds, M. H. Wright and P. E. Wright, Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions, *SIAM Journal of Optimization*, cilt 9, no. 1, pp. 112-147, 1998.

[56] L. Fonseca and B. Manjunath, Registration techniques for multisensor remotely sensed imagery, *Photogrammetric Engineering and Remote Sensing*, 1996.

[57] Z. Zhou, Y. Yuan and C. Shi, Object tracking using SIFT features and mean shift, *Computer Vision and Image Understanding*, cilt 113, no. 3, pp. 345-352, 2009.

[58] D. Lowe, Object Recognition from Local Scale-Invariant Features, *Proceedings of the Seventh IEEE International Conference on Computer Vision*, cilt 2, pp. 1150-1157, 1999.

[59] M. A. A. Cox and T. F. Cox, Multidimensional Scaling, *Handbook of Data Visualization*, Springer Berlin Heidelberg, 2008, pp. 315-347.

[60] V. de Silva and J. B. Tenenbaum, Sparse multidimensional scaling using landmark points, Stanford University, 2004.

[61] C. Faloutsos and K.-I. Lin, FastMap: A Fast Algorithm for Indexing, Data-mining and Visualization of Traditional and Multimedia Datasets, *SIGMOD Rec.*, cilt 24, no. 2, pp. 163-174, 1995.

[62] E. Acar, S. Arslan, A. Yazici and M. Koyuncu, Slim-tree and BitMatrix index structures in image retrieval system using MPEG-7 Descriptors, *Content-Based Multimedia Indexing, 2008. CBMI 2008. International Workshop on*, 2008.

[63] XXL Database Research Group, http://www.xxl-library.de/. last visited on 01 01 2008].

[64] Weka Api homepage, http://www.cs.waikato.ac.nz/ml/weka/. last visited on 01 01 2008].

[65] Colt Project homepage, http://dsd.lbl.gov/~hoschek/colt/. last visited on 01 01 2008].

[66] EMD implementaion homepage, 01 06 2010. http://www.cv.tu-berlin.de/~ulas/RaRF.

[67] V. S. Roshni and K. Revathy, Using mutual information and cross correlation as metrics for registration of images, *Journal of Theoretical & Applied Information*, cilt 4, 2008.

[68] Z. Al Aghbari and A. Al-Hamadi, Efficient KNN search by linear projection of image clusters, *International Journal of Intelligent Systems*, cilt 26, no. 9, pp. 844-865, 2011.

[69] J. C. Dunn, A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters, *Journal of Cybernetics*, cilt 3, no. 3, pp. 32-57, 1973.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Arslan, Serdar

Nationality: Turkish (TC)

Date and Place of Birth: 10-July-1980, Kayseri

E-Mail: serdarslan@gmail.com

EDUCATION

Degre	Institution	Year of Graduation
M.S.	METU, Computer Engineering	2005
B.S.	Hacettepe University, Computer Engineering	2001

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2011-….	Turkish Prime Ministry	Soft. Eng.
2007-2011	TUBITAK	Senior Researcher
2001-2007	METU Comp. Center	IT Specialist
2000-2001	ASELSAN	Soft. Eng.

PUBLICATIONS

International Journal Publications

Serdar Arslan, Adnan Yazici, Ahmet Sacan, Hakkı Toroslu, Esra Acar, "Comparison of Feature-based and Image Registration-based Retrieval of Image Data Using Multidimensional Data Access Methods," *Data & Knowledge Engineering,* Vol. 86, July 2013, Pages 124–145.

International Conference Publications

Serdar Arslan, Ahmet Saçan, Esra Açar, I. Hakki Toroslu, Adnan Yazıcı, "Comparison of Multidimensional Data Access Methods for Feature-Based Image Retrieval," 20th International Conference on Pattern Recognition Vol :1 No :1, pp:3260-3263, 2010.

Esra Açar, Serdar Arslan, Adnan Yazıcı and Murat Koyuncu, "Slim-Tree and Bıtmatrıx Index Structures in Image Retrieval System Using MPEG-7 Descriptors," CBMI-2008 Sixth International Workshop on Content-Based Multimedia Indexing, pg:402-409, June 2008, London UK.