FLEXIBLE DESCRIPTOR INDEXING FOR MULTIMEDIA DATABASES

Catalin Calistru

INESC-Porto
FCT PhD grant SFRH/BD/12426/2003
cmc@inescporto.pt

Cristina Ribeiro, Gabriel David

FEUP/INESC-Porto

mcr@fe.up.pt, gtd@fe.up.pt

ABSTRACT

Current multimedia applications require efficient tools for modeling and search. While solid models must support a large variety of concepts like sets, sub-sets, part-of hierarchies and integrate standards like Dublin Core, MPEG-7 (descriptors, descriptor schemes), TV-Anytime, MPEG-21 (digital item, digital item adaptation, digital item identification), the search algorithms must deal with heterogeneous multimedia data (text, image, audio and video). The search in audiovisual data requires the use of heterogeneous metadata representing a wide range of features, from low-level ones (color, motion) to high level ones (subject, mood), or domain-dependent concepts.

The current work is part of the development of a prototype Multimedia Database and addresses the problem of choosing the proper storage and search strategy for the descriptors in the multimedia database. It includes a review of the main applicable data structures and search techniques and a case study from the point of view of our system requirements.

1. INTRODUCTION

Multimedia databases may have big diversity both in the nature of stored objects and in the application domains. The database may be a repository of objects which have been collected and appropriately described (as in a digital library) or may consist of the heterogeneous information assembled and dynamically modified in a Web site, or still result from the production process of a publisher or a broadcaster.

Many standards have now been established for multimedia information, both the objects themselves and for the corresponding metadata, the focus of this paper. Metadata may describe the context in which the MM objects have been created, are stored or can be used, or describe aspects of the object content. Metadata standards can be applied to traditional descriptive metadata (Dublin Core), to content metadata (MPEG-7) and to the various aspects of assembling components, handling digital rights or adapting the content to specific players (MPEG-21). Standards help to clarify concepts and promote interoperability, but they are currently not appropriate as data models in a MM DB [1].

We have argued in favor of a multimedia database model structured around a limited number of central concepts that capture the main aspects of the objects to be stored and can easily import descriptions using the current standards. An appropriate model should easily handle large volumes of information, be extensible and integrate flexible search mechanisms.

Relational DB management systems provide a natural setup for highly structured data, efficient implementation for the most common data operations and a standard interrogation language and respond well to the scalability and search flexibility criteria. However, they lack the extensibility which can be provided by XML, which has become a de facto standard for data representation and interchange.

A multimedia database must account for the storage and retrieval of both the multimedia objects and the associated metadata. An object's metadata includes generic descriptors like titles and dates, applying to any object, and others that are specific to certain types of objects (an audio descriptor has no meaning for a textual object, for instance).

The scope of this work is the representation and indexing of metadata in a multimedia database where items can be assembled from diverse sources. Current developments are being tested in a prototype multimedia database (MetaMedia [2]). Its data model has three main underlying principles. The first one is that multimedia objects are usually represented in a hierarchical manner, allowing sets of items to be treated as objects that can have associated descriptions. The second one is that the same kind of descriptors can be used for an individual object and for a set of related objects, allowing descriptors to be inherited from a collection to a sub-collection down to individual objects. The third one is that the structure of descriptors should be left as open-ended as possible.

The first two principles have been followed in the standards for archival description [3, 4, 5] and prove themselves very useful when it comes to the representation of large collections: metadata is frequently available for sets of items rather than individual ones, and inheritance can make it use-

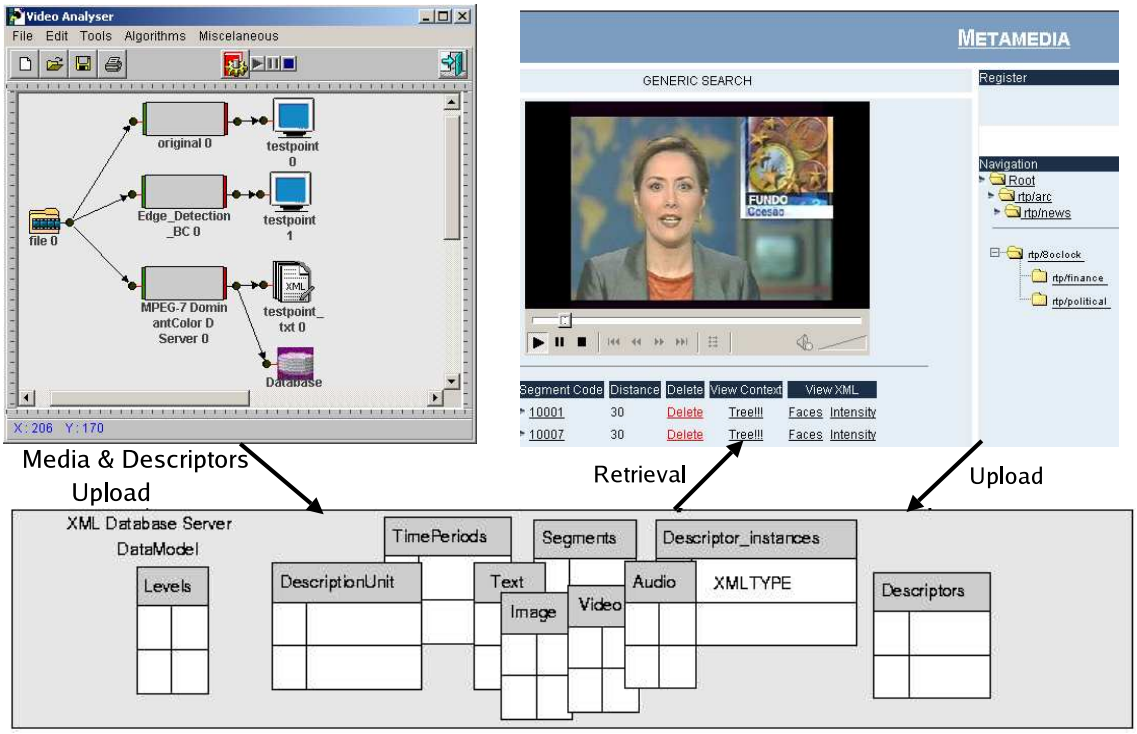


Fig. 1. The Multimedia Database Architecture.

ful further down in the hierarchy.

The third principle intends to account for the ever growing possibilities of analysis and feature extraction from multimedia objects. A large set of descriptors has already been identified in the multimedia description standards [6, 7]. The use of flexible XML models for descriptors allows the incorporation of new ones without any changes in the data model.

The paper is centered on the problem of choosing the proper storage and search strategy for the multimedia descriptors. It includes the identification of the main requirements for the MetaMedia database and a review of the data structures and access methods that have been proposed for similar tasks. In a second step existing approaches are evaluated from the point of view of the MetaMedia requirements.

The paper is organized as follows. Section 2 presents the general architecture of our current workbench, highlighting its main concepts. In Section 3 we focus on the database model characteristics and on the requirements that our workbench would ideally support. Section 4 makes an overview of data structures specialized for indexing. The next section provides an analysis of the main search techniques from the point of view of the MetaMedia requirements. Finally, Section 6 presents conclusions and outlines the work to be done in the sequel.

2. A MULTIMEDIA DATABASE

The ideas proposed in what follows have been developed in the MetaMedia multimedia database prototype. The architecture of the database comprises a data model supporting storage and a search/upload interface, and the connection to a video analysis tool [2]. The proposed data model takes into consideration the relevant aspects of the storage and retrieval processes and has been designed as an assembly of modules dedicated to the representation of separable aspects of an object's description [8].

The MetaMedia database model has not been built for a specific category of multimedia objects. This means that its design has been supported on the identification of abstractions that might be simultaneously useful for different kinds of objects and meaningful for the user interfacing the database. The model has been used on several data sets including documents from a historic archive, a photo collection (both descriptions and digitized versions) and video and audio from the MPEG-7 test sets. It is currently being used with MPEG-21 Digital Items.

The MetaMedia model is also not strictly oriented to a specific data format, nor to a specific standard. There is a discrepancy between the richness of the available description standards and the performances of the search and retrieval tools for multimedia databases. Neither Dublin Core, nor MPEG-7, nor TV-Anytime, nor MPEG-21 are designed

for search intensive operations, but rather for describing objects at different levels (content, context, adaptation, identification, rights management, etc). Given the breadth of the description standards it is not surprising to find that they are not disjoint, the same information being available in different formats. Combination of various formats is also allowed by recent schemes. The Digital Item declaration schema (part of MPEG-21), for example, allows the insertion (at certain levels) of descriptors coming from any other format. Dublin Core, MPEG-7, or TV-Anytime can provide a *Title* descriptor for a Digital Item; the choice of a specific one is left for the content provider or metadata authoring tool.

The MetaMedia model clearly separates metadata concerning context, such as Author or Rights, and content metadata, such as MPEG-7 audiovisual descriptors. Figure 2 shows the structure of the main classes in the data model. Three main concepts are associated with the objects to be described. The first is the Unit of Description (DU), capturing the notion of an object or collection of objects that can be given a context in terms of creation and relationship with other objects. The second is the Segment, capturing the notion of some part of an object that can be independently analyzed in terms of content and be used by itself. A segment has no context of its own, getting it from the Unit of Description of the object it belongs to.

In terms of structure both DU's and segments are organized as part-of hierarchies. A DU is either a root unit or is related to the DU describing the collection of units where it belongs, which has its own DU. The same applies to a segment, which may be a part-of another segment. Any item that can be individually retrieved has an associated context, either the one in its own DU or that of the object including it.

The third basic concept for the description of content is that of a Descriptor. The sense in which Descriptor is used is the one established by the MPEG-7 Standard—a representation of a feature [9]. A Descriptor Value is an instantiation thereof.

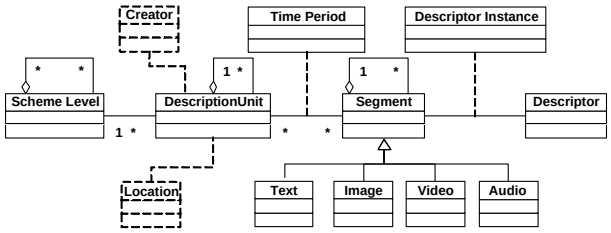


Fig. 2. The Data Model.

A convenient component for a multimedia database environment is an analysis tool for automatically generating content descriptors. The Video Analysis Framework (VAF) [10] is an application that provides an integrated environment for executing video analysis tools. The VAF supports operations such as video parsing and video representation with different levels of abstraction (scene, key-frame and objects), each one with its own video engine. A graphical user interface (GUI) allows users to select tools, assemble their processing graph and determine the flow of information. The VAF has been integrated with MetaMedia, allowing the automatic upload of descriptions (see Figure 1).

2.1. Content and Context Retrieval interface

Retrieval capabilities are central in multimedia databases. MetaMedia currently has a search interface where queries can be formulated based on contextual information, the structure of the object collections and the value of content descriptors. Handling the diversity of descriptors is a major issue in retrieval, requiring specialized structures and effective access methods.

3. A FLEXIBLE MODEL FOR OBJECTS DESCRIPTORS AND METRICS

A common characteristic for the diversity of multimedia objects, is that they have a conceptual structure. For instance MPEG-21 Digital Items come with a defined hierarchical structure. Hence, a top level Item can represent either a collection of musical albums, or an album related to its parent item (the collection) , or a track as a part of the album. At each level an item can be independently consumed having its own content and contextual information or inheriting from its parent.

We will assume that metadata comes in the form of Descriptors which can be elements from any other XML based standard like Dublin Core, MPEG-7, TV-Anytime, MPEG-21 parts (Digital Item Adaptation, Digital Item Identification, Digital Rights Management, etc). The descriptors can vary from simple low-level descriptors, like the ones that describe motion activity or color, to higher level descriptors like a summary one for instance. Typically new descriptors can be added at any semantic level, whether it is a completely new descriptor, or it is a combination of some preexisting lower levels descriptors [11].

Figure 3 illustrates the use of our database conceptual layer (description units and descriptors in this example) for the representation of heterogeneous documents. The left part of the figure illustrates a hierarchy of description units constructed for the RTP archive data set, while the right part is a hierarchy built from an MPEG-21 Digital Item. RTP is the Portuguese National Broadcasts, and a set of news Programs have been used as a data set in the prototype.

Such huge variety of descriptors has a direct impact on the search task and on the model side. Basically each new descriptor implies, a new criteria for the items to be searched on, and possibly a new metric to be integrated in the search

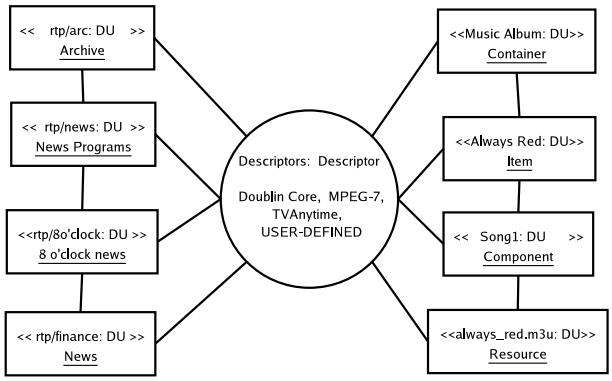


Fig. 3. Uniform Descriptors

engine. Thus, a flexible model for metrics imposes a fine grained representation of the descriptor values and their types.

An MPEG-7 *DominantColor* descriptor for example, describes the color components in a string format (a list of components) but when applying some metric, a numerical format for the components would be definitely preferred.

```
<Descriptor type=''DominantColorType''>
   ..........
    <ColorValueIndex>
       12 3 15
    </ColorValueIndex>
   .......
</Descriptor>
```

MetaMedia belongs to the family of multimedia database systems representing content information as well as contextual information (a picture along with owner and copyright information), for large objects (hundreds of MBytes, perhaps GBytes) whose retrieval is expected to be based on similarity. Common requirements for multimedia databases include an interactive response mode which may raise efficiency issues, and a flexible user interface accomodating both naive and advanced users. Interactive response mode, imposing requirements on efficiency and a flexible user interface which would accommodate naive users and advanced search for specialists are also common requirements for the current multimedia databases.

Beside, we have identified some particular requirements, which have not been widely taken into consideration, but which are important for our model:

1. **Preprocessing**: the amount of preprocessing work like normalizing the feature vectors, dimensional reduction, might be an important factor when dealing with heterogeneous descriptors.

2. **Frequent updates**: even if the majority of the multimedia databases are seen by their nature as statical archives where mostly appends can happen, in our case updates are quite often. For instance, if the quality of the feature extraction tool improves, the results of applying it to an existing object will be incorporated as update.

3. **Varying dimensionality**: the dimensionality of our search space is directly related to the number of descriptors, meaning that each descriptor contributes with a number of dimensions. For example adding a histogram descriptor increases the dimensionality with the number of bins the histogram have been quantized (8, 16, 32, 166).

4. **Any dimensional subspace**: at query time the user must be able to choose any subset of features.

5. **Support for different metrics**: searching by different components of the feature space often implies using different metrics for each component. The aggregation may be done in different ways: arithmetic aggregate functions or fuzzy aggregate functions.

6. **Support for weighted queries**: is used whenever the user want to stress the relevance of some features.

Considering these requirements as "criteria" we have tried to identify the best existing approaches for our setup. In the following section a short review of this work is presented, followed by a MetaMedia simple case study, conclusion and future work.

4. REVIEW

Recent multimedia applications make intensive use of various database capabilities. Among these, similarity search is required as a basic functionality of the database system. In the general case the usage of such an application is that given an multimedia query object O usually defined at query time, we want to find the most similar objects to O.

The similarity search approaches that have been proposed up to now fall in three distinct classes each one having particular solutions.

4.1. Spatial Access Methods (SAM)

The first class, Spatial Access Methods (SAM), also called feature-based methods, partition the space based on the values of the vectors along each independent dimension and is independent of the distance function used to compute the distance among objects in the database or between query objects and database objects. Even so, there are situations when the only information that we have is the distance between the objects, also called distance-only data. The idea behind SAM for this situation is to construct a feature space

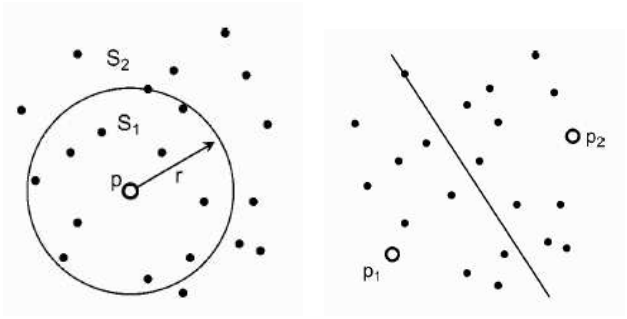
and search the collection of feature vectors for similar ones. One solution for building such a feature space is to derive "features" purely based on the inter-object distances. Such methods are called **embedding methods**. Another solution is to extract important features from the multimedia objects and map the features into high-dimensional feature vectors. Since the database may be very large, consisting of millions of objects with hundreds of features, a dimensionality reduction is frequently required.

According to the method they partition the space these structures can be **data partitioning methods (DP)** which use minimum bounding rectangles (MBR) such as *R-tree* with its variants $R^* - tree$, $R^+ - tree$ and *X-tree*, bounding spheres such as *SS-Tree*, MBR and bounding spheres such as *SR-Tree*, generic minimum bounding regions(hyper rectangle, cube, sphere) such as *TV-tree* and **space partitioning methods (SP)** which partition the whole space such as the *kDB-Tree*, *Hybrid-tree*, *SH-tree*. [12], contains a detailed description of the spatial access methods that we have mentioned here.

4.2. Metric Access Methods (MAM)

The second class, metric access methods (MAM), have gained an important role due to the fact that the datasets of the multimedia databases are often organized in metric spaces. An example of non-spatial data sets that cannot be mapped to points in a multidimensional space is text datasets which use the *edit metric* to measure the distance between two texts. Another reason for resorting to metric access methods is the fact that conventional spatial approaches stop being efficient in high-dimensional data. Even if *dimensionality reduction* techniques exist, they have important drawbacks (do not preserve information from the original high-dimensional space to the mapped low-dimensional one, overhead introduced by the additional set of features, space requirement, update speed). Metric access methods only consider relative distances of objects (rather than their absolute positions in a multi-dimensional space) to organize and partition the search space, and just require that the function used to measure the distance(dissimilarity) is a *metric*. The advantage of distance-based indexing methods is that distance computations are used to build the index, but once the index has been build, similarity queries can be often performed with a significantly lower number of distance computations than a sequential scan of the dataset.

Typical for distance-based indexing structures are the metric-trees, which are binary trees that recursively partition the data set into two subsets at each node level. Two main partitioning schemes have been denoted by Hjaltson and Samet[13]: *ball partitioning* and *generalized hyperplane partitioning*. With the ball partitioning approach, the data set is partitioned based on the distance from one specified object, called *vantage point* or *pivot*, generating two subsets: the first subset inside the ball around the pivot, and the second subset outside the ball.



(a) Ball partitioning (b) Hyperplane partitioning

Among ball partitioning trees the most referenced are *Vantage-Point Tree (VP-tree)*, *Multi-Vantage Point Tree (MVPT)* [14], *Vantage-Point Forest (VPF)*, *Burkhard-Keller Tree (BKT)*, *Fixed Queries Tree (FQT)*.

With the hyperplane partitioning method, at each step two centers c_1 and c_2 are selected. The elements closer to c_1 than to c_2 go into the left subtree and those closer to c_2 go into the right subtree.

Among hyperplane partitioning structures we enumerate *Bisector Tree (BST)*, *Generalize Hyperplane Tree (GHT)*, *Geometric Near-neighbor Access Tree (GNAT)*. and the *M-tree*[15]. In [16], can be found a detailed description of the metric access methods that we have mentioned here.

4.3. Multi-feature queries (MF), aggregation algorithms

The third class of methods for querying high dimensional data assume that the objects in the database have m_i dimensions for each of the n features, with $i = 1..n$. The total dimensionality would be $\sum_{i=0}^{n} m_i$. Supposing the case of images, if a color descriptor introduces three dimensions (red, green and blue components) and the histogram introduces a number of dimensions equal to the number of bins the histogram has been quantized (suppose 8), there will be a total of 11 dimensions. Multi-feature methods treat each dimension as a separate list, and their goal is to obtain the result of the query by accessing a minimum number of lists and as few as possible objects in each of the visited lists. As in the case of SAMs and MAMs these algorithms are very often compared with the sequential scan (the naive algorithm), which is still very useful for some specific distributions of the data [17]. Unlike SAMs and MAMs, multi-feature queries cannot be considered indexing methods, either because they operate in middleware systems (Garlic [18]), like the *Fagin's Algorithm*, *Threshold Algorithm*, *Quick-Combine* [19], or they operate directly with the original data like *BOND* [20].

4.3.1. Fagin's style algorithms

We include in this category Fagin's Algorithm [21], *Threshold Algorithm, medrank* [22] and *Quick-Combine* [23]. They act in two phases:

1. the first phase is common to all of the algorithms in this category, and produces ranked list for each dimension.

2. the second phase combines these ranked lists using different score aggregation. The algorithms differ in the type of aggregation function that is used and the termination condition.

Any combining function can be used, including weighted queries or sub-space oriented queries.

4.3.2. BOND

Branch and Bound on Vertically Decomposed Data (BOND) [20], adopts vertical decomposition as storage organization. It decomposes the data into multiple tables, one for each dimension. Therefore, the information for a specific object is distributed to multiple tables. The algorithm accumulates the distances between the query object and all data vectors, by scanning the dimensional projections one-by-one. After processing a few dimensions, *partial* distances of k-nearest neighbors are exploited to discard safely from further consideration those vectors that cannot possibly participate in the result. Iterative application of this process reduces the candidate set and the last steps are performed on just a small database sample.

5. CASE STUDY

Taking in consideration what it was presented in the previous subsection, a practical example will be examined in order to motivate the choice of the search approach. We will consider for example a set of ten two dimensional segments ($s_1 ... s_{10}$) and a query object q. We will assume that the user asks for the closest s_i to q. The dimensions will be named (d_1 and d_2). Table 1 illustrates the data set on which we will illustrate and analyze some representative indexing methods. No normalization of the dimensions was performed because we deal with a very simple example. Indexing algorithms that will be analyzed lie in the previously presented categories: *spatial access methods*, *metric access methods* and *aggregation algorithms*. We will extend the analysis to the MetaMedia assumptions: a much higher dimensional data set with a *meaningful* distribution [17], frequent updates, different metric support,etc.

From the spatial access methods we have chosen the *R-tree* which is a representative for data partitioning approaches (remember that spatial access methods include data partitioning and space partitioning methods) and have proved to be quite efficient for low-dimensional spaces (2, 3, 4). The left part of the figure 4 was obtained by running a Java implementation of the R-tree found at [25]. It can be seen in Figure 4 that the R-tree builds MBRs (minimum bounding rectangles) around the data. The containment hierarchy obtained(the index) from the data partitioning is presented in the right part of the Figure 4.

dimensions		d_1	d_2
points	s_1	200	70
	s_2	100	150
	s_3	170	240
	s_4	280	160
	s_5	130	180
	s_6	120	120
	s_7	180	130
	s_8	100	300
	s_9	240	280
	s_{10}	250	120

Tab. 1. Data set example

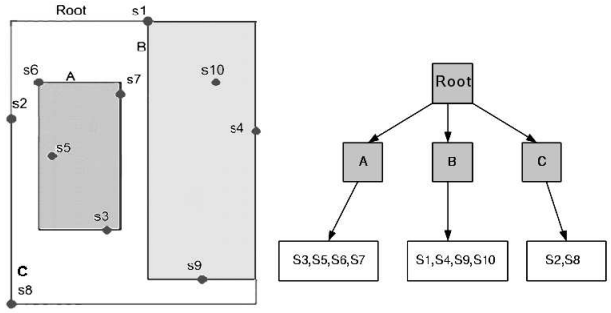


Fig. 4. R-Tree from data set

The nodes of the tree contain (depending of the variants of the R-tree) information about the MBR that they cover(coordinates for instance). This kind of information grows exponentially when increasing the dimensions, leading to the growth of each tree node and of the index itself. The bigger the nodes are, the fewer can be stored in a disk page(which has a fixed amount of space set by the operating system). The number of nodes per disk page is called the *fan-out* of the tree. As the fan-out decreases with the dimensionality, accessing such an index becomes more difficult. Another important issue is the *high-overlapping* between the MBRs stored at the same level in the tree(cannot be observed in our example because we have few data and few dimensions), leading to an increased number of branches to be searched.

From the metric access methods(MAMs) category the

	features	SAM	MAM	MF
1	preprocessing	dim.red, normalization	dist. computations	normalization
2	frequent updates	no(except small dim.)	no	yes
3	varying dimensionality	no	no	yes
4	any dimensional subspace	no	no	yes
5	different metrics	yes	no(except M-tree[24])	yes
6	weighted queries	no	no	yes

Tab. 2. Results

M-tree approach was chosen, because M-tree is one of the best among metric-based indexes. Like the R-tree, the M-tree is a representative for a whole family of trees built on the same principles(M+-tree, M2-tree, Slim-tree). Figure 5 illustrates the construction of the M-tree for our small dataset. The M-tree is a dynamic, balanced tree where all the objects being indexed are stored in the leaf nodes, while the internal nodes, also called *routing objects*, store pointers to child nodes and the covering radius for the children they "enclose". M-tree allows different metrics (that have to belong to the same class as mentioned in [24]) to be used for accessing such an index, but high-overlapping is also a performance issue.

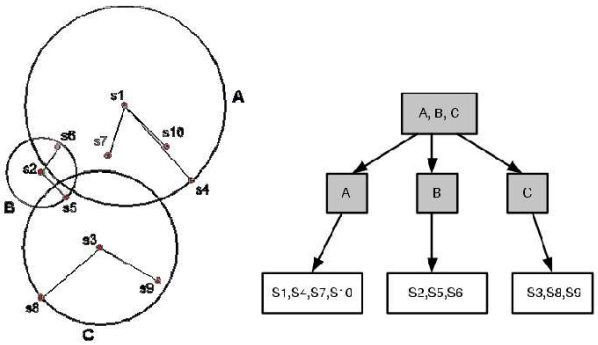


Fig. 5. M-Tree from data set

The case of the aggregation algorithms, from which the BOND was chosen for exemplification is quite straightforward for such a simple setup. Assuming a monotone aggregation function, the objects are processed by BOND in a sequential scan manner, analyzing only the information in the first list ($d1$). Based on this partial scores and on the upper and lower bounds of the aggregate function, objects are discarded from the list of the qualifying objects.

6. CONCLUSIONS

In the previous sections we have focused on a representation of descriptors that accommodates their diversity and on the indexing methods required for similarity-based searching. We have identified the requirements of a multimedia database that influence the choice of descriptor representation and we have presented a short review of the most relevant structures intended for search purposes.

Table 2 centralizes our results. For diverse reasons, most of the presented methods, while fruitful in their specific scenarios, are too restrictive for our database setup. SAMs for instance, although efficient for low dimensionality such as 2-10, are very sensitive to the increase of the dimensionality. MAMs are bound to a specific metric as a global measure, and the same metric must be used to build the index and to search; any modification in terms of dimensionality or metric requires heavy computations and the index reconstruction. With respect to the requirements identified in Section 3, aggregation algorithms (see Section 4.3) seem to fit best. They allow full access on all of the feature components and can cope with domain specific metrics. They also provide easy adaptation to varying dimensionality, being opened to a high number of uploads and insertions.

Future work involves the test of these structures on our search environment and its validation with more significant data sets.

7. REFERENCES

[1] A. Murat Tekalp Ahmet Ekin and Rajiv Mehrotra. Integrated semantic-syntactic video modeling for search and browsing. *IEEE Transactions on MultiMedia*, 6(6):839–851, 2004.

[2] Cristina Ribeiro, Gabriel David, and Catalin Calistru. A multimedia database workbench for content and context retrieval. *MMSP2004 IEEE Workshop*.

[3] ISAD(G). ISAD(G)–General International Standard Archival Description. http://www.ica.org/, consulted on 2004-01-28, 1995.

[4] ISAAR(CPF). ISAAR(CPF): International Standard Archival Authority Record for Corporate Bodies, Persons, and Families. http://www.ica.org/, consulted on 2004-01-28, 1995.

[5] EAD. Encoded Archival Description (EAD), consulted on 2004-01-28. http://www.loc.gov/ead/, 2002.

[6] MPEG-7 Requirements Group. MPEG-7 Overview v.9 consulted on 2004-01-28. http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm, 2003.

[7] MPEG-21 Requirements Group. MPEG-21 Overview v.5, consulted on 2004-01-28. http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm, 2002.

[8] Cristina Ribeiro and Gabriel David. Metadata Model for Multimedia Databases. In *Proceedings of the International Cultural Heritage Informatics Meeting, ICHIM01*, 2001.

[9] Frank Nack and Adam T. Lindsay. Everything you wanted to know about MPEG-7: Part 2. *IEEE MultiMedia*, 6(4):64–73, 1999.

[10] Luís Filipe Tavares, Luís Teixeira, and Luís Corte-Real. Generic Framework for Video Analysis. In *Proceedings of the 12th Portuguese Conference on Pattern Recognition, RecPad 2002, Aveiro, Portugal*, 2002.

[11] Aleksandra Mojsilovic. Semantic metric for image library exploration. *IEEE Transactions on MultiMedia*, 6(6):828–838, 2004.

[12] Christian Böhm, Stefan Berchtold, and Daniel A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Comput. Surv.*, 33(3):322–373, 2001.

[13] Gisli R. Hjaltason and Hanan Samet. Index-driven similarity search in metric spaces. *ACM Trans. Database Syst.*, 28(4):517–580, 2003.

[14] Tolga Bozkaya and Meral Ozsoyoglu. Indexing large metric spaces for similarity search queries. *ACM Trans. Database Syst.*, 24(3):361–404, 1999.

[15] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *VLDB '97: Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 426–435. Morgan Kaufmann Publishers Inc., 1997.

[16] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José; Luis Marroquín. Searching in metric spaces. *ACM Comput. Surv.*, 33(3):273–321, 2001.

[17] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? *Lecture Notes in Computer Science*, 1540:217–235, 1999.

[18] M. J. Carey, L. M. Haas, P. M. Schwarz, M. Arya, W. F. Cody, R. Fagin, M. Flickner, A. W. Luniewski, W. Niblack, D. Petkovic, J. Thomas, J. H. Williams, and E. L. Wimmers. Towards heterogeneous multimedia information systems: the garlic approach. In *RIDE '95: Proceedings of the 5th International Workshop on Research Issues in Data Engineering-Distributed Object Management (RIDE-DOM'95)*, page 124. IEEE Computer Society, 1995.

[19] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In *PODS '01: Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 102–113. ACM Press, 2001.

[20] P. de Vries Arjen, Nikos Mamoulis, Niels Nes, and Martin Kersten. Efficient k-nn search on vertically decomposed data. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 322–333. ACM Press, 2002.

[21] Ronald Fagin. Combining fuzzy information: an overview. *SIGMOD Rec.*, 31(2):109–118, 2002.

[22] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 301–312. ACM Press, 2003.

[23] Ulrich Güntzer, Wolf-Tilo Balke, and Werner Kießling. Optimizing multi-feature queries for image databases. In Amr El Abbadi, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal, Nabil Kamel, Gunter Schlageter, and Kyu-Young Whang, editors, *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 419–428. Morgan Kaufmann, 2000.

[24] Paolo Ciaccia and Marco Patella. Searching in metric spaces with user-defined and approximate distances. *ACM Trans. Database Syst.*, 27(4):398–437, 2002.

[25] Marios Hadjieleftheriou. R-tree visualization. http://www.dbnet.ece.ntua.gr/ mario/rtree/.