Exploring Strategies for Minimizing Overlap Between Nodes in a Multimodal Metric Tree

Ricardo C. Sperandio, Zenilton K.G. Patrocínio Jr., Hugo B. de Paula, Silvio J.F. Guimarães

Pontifícia Universidade Católica de Minas Gerais – PUC Minas, Brazil
rcarlini@gmail.com, {zenilton, hugo, sjamil}@pucminas.br

Abstract. *Slim²-tree* is a multimodal metric tree which enables video indexing and retrieval by using information from multiple modalities. Experimental results have demonstrated its efficiency when compared to other multimodal solutions. This article explores different strategies related to the use of a *post-processing* algorithm for the *Slim²-tree* – named multimodal *Slim-down*, which tries to minimize the overlap between tree nodes. Experiments have also shown the performance improvement obtained by the policy, in which any element that presents the larger distance value to the pivot for any modality is selected as candidate to be moved. Moreover the results are better when that policy is repeatedly used during insertion.

Categories and Subject Descriptors: H.2.4 [**Database Management**]: Systems—*Multimedia databases*; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Indexing methods*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Retrieval models and Search process*

Keywords: content based video retrieval, metric access methods, multimedia database, multimodal video retrieval

1. INTRODUCTION

Video is an effective medium for storing information about events of the real world, and a vast amount of video materials exists, covering a wide range of applications. However, widespread use of video in computer applications is often prohibited by the lack of effective tools to store, index and recover this type of media. To avoid misunderstanding, in this article a digital video encompasses information from different data sources, such as visual and acoustic data. The trivial way to implement video query systems is through the use of previously registered annotations for each video in the database.

This approach may be inadequate for large databases of videos, since it involves great human effort in generating annotations [Shao et al. 2008]. Furthermore, ambiguous or incomplete descriptions can negatively impact search results. Therefore, the use of the video content for indexing and retrieval may provide an alternative approach, which gives rise to content-based video retrieval systems (*Content-Based Video Retrieval* – CBVR). In this scenario, similarity search is the fundamental tool for those systems, and there were several studies in the last two decades about this topic [Almeida et al. 2010]. However, similarity search for digital videos has been generally focused solely on the use of visual data, completely neglecting another important source of video information – the acoustic track.

This article is an enhanced version of a previous work [Sperandio et al. 2013] that proposes a new multimodal metric tree – *Slim²-tree* – which enables video indexing and retrieval by using information from multiple modalities. Experimental results have shown its efficiency when compared to other multimodal solutions. It is worth mentioning that tests results have also shown that *Slim²-tree* supports queries using only one modality with a computational cost similar to a unimodal solution. This is an important property of *Slim²-tree*, since it can be used instead of several others.

The authors are grateful to PUC Minas, CNPq, CAPES and FAPEMIG for the financial support of this work.

This article has the following improvements on top of the previous work: (i) the text have been reviewed and only the major issues related to the creation and use of *Slim²-tree* remain; (ii) the dataset is better described and some frame samples are presented; (iii) a detailed description of a multimodal *post-processing* algorithm – named multimodal *Slim-down* – is given; and (iv) different strategies related to the use of multimodal *Slim-down* (*i.e.*, policies for choosing elements to be moved and the context for its application) are fully described and evaluated. Experiments have also shown that multimodal *Slim-down* can improve the test results, specially when the candidate to be moved defines at least one of the radii of the hyperregion associated with the tree node.

The remainder of this article is organized as follows. Section 2 covers the main concepts and related works. The new multimodal metric index structure – *Slim²-tree* – is briefly reviewed in Section 3. Section 4 gives a detail description of the multimodal *Slim-down* along with different strategies for its use. In Section 5, experimental results are shown and analyzed. Finally, Section 6 draws some conclusions and proposes future directions for research.

2. RELATED WORK

The support to multimodal queries – the ones that involve data from different nature, such as visual and acoustic – is extremely valuable for CBVR. For example, queries such as "*find news videos in which president Dilma Rousseff talks about 2014 FIFA World Cup*" require more than one modality to be answered adequately. According to Atrey et al. [2010] and Yan and Hauptmann [2007], the combination of multiple modalities may consistently improve, in terms of precision, the obtained results when compared to the use of each modality separately. However, the use of multiple modalities involves the adoption of a fusion strategy. In the work of Atrey et al. [2010], different fusion techniques are described and analyzed, specially *early fusion* techniques, in which several features extracted from the data are combined before their effective use; and *late fusion* techniques, in which features are processed separately and their results are combined later.

The need of indexing and retrieval of more complex types of data, such as images, sounds, videos, and others, initiated a series of studies, since these types of data are not totally ordered. Techniques, such as *R-tree* and *KD-tree*, among others, are based on vector space [Gaede and Günther 1998] and represent data as points in the space in order to group and organize the information that they represent. However, for CBVR the huge number of dimensions generates a performance problem known as *"the curse of dimensionality"* [Chávez et al. 2001]. For this scenario, metric access methods were developed to support similarity queries based on a distance function used to measure the (dis)similarity between data. These methods are based on the following approach: partitioning the set of objects using a distance function and choosing one of them as the representative (pivot) of the entire set. During the search, the representatives are used to reduce the search space using the *triangular inequality* – which allows the computation of lower and upper bounds for the distances between the query object and every element of a set based on the distance between the query object and the set pivot.

The first metric access methods proposed in the literature, such as *VP-tree* and *FQ-tree*, were static, and they did not support neither insertions nor removals after their creation. The first dynamic metric access method was *M-tree* [Ciaccia et al. 1997]. It is a balanced tree, in which data are stored in the leaves and internal nodes have pointers to direct the search to the correct leaf. This paradigm has become very popular, and many access methods extending the original *M-tree* have been proposed. In relation to this work, two extensions should be mentioned: *Slim-tree* and *M²-tree*.

The first one – *Slim-tree* – tries to minimize the overlapping between regions [Traina et al. 2000]. Similar to *M-tree*, the space is divided in regions (represented by subtrees) that are not disjoint. Each one of those regions is defined by one representative object (O_r) – used as reference to the others in that subtree, and by a covering radius which should be large enough to cover all elements stored in that subtree. This allows the search algorithms to ignore entire tree branches using the *triangular*

inequality. Beyond that, *Slim-tree* tries to minimize the overlapping between regions by applying a heuristic called *Slim-down* which moves objects between leaf nodes in order to reduce the covering radius of those nodes. During insertion, *Slim-tree* has also proposed a heuristic based on minimal occupancy (`MinOccup`) – which is responsible for balancing the objects distribution among subtrees, and the use of *Minimum Spanning Tree* (MST) for node splitting and pivot selection.

Another important extension of *M-tree* is *M²-tree* which was proposed to support complex queries between objects represented by multiple feature descriptors [Ciaccia and Patella 2000]. This approach combines in a single structure all descriptors related to different metric spaces, allowing the use of distinct distance functions for each feature. Besides, *M²-tree* adopts a monotonic increasing *score* function to combine the distances of each distinct feature in a single value (*early fusion*) which is used during insertion and search. Unfortunately, the authors only provide a very brief description of it, *i.e.*, they claim that *M²-tree* works in a very similar way to *M-tree* – using the same policies and heuristics – except for the adoption of a *score* function to implement the *early fusion*.

Other structures for multiple features have also been proposed in the literature. In the *MOSAIC-tree* [Goh and Tan 2000], each feature is stored in a distinct layer using *R-trees*. The leaf nodes from a superior layer are connected to the root of the associated tree in the layer below. But the use of multidimensional trees only allows its application with low dimensional feature descriptors due to "*the curse of dimensionality*". The *MFI-tree* [He and Yu 2010] adopts a single distance value calculated by a linear combination of the several distances which is used to index the object. The major issue is how to determine the weights for that linear combination in a scalable and data-independent approach, but the authors left that problem for a future work. More recently, the *TEMPOM²-tree* [Döller et al. 2012] proposes the use of two data structures in parallel, one is a *M²-tree* for content-based video indexing and another for representing the video temporal structure. Finally, the *M³-tree* [Bustos et al. 2012] allows searches through a dynamic combination of distinct metrics (defined by the user when a query is executed), instead of using a fixed combination such as *MFI-tree*.

3. SLIM²-TREE

An efficient access method for multimodal and unimodal video retrieval, named *Slim²-tree*, was proposed by Sperandio et al. [2013]. Here we briefly review the major issues related to its creation and use for performing similarity queries.

Slim²-tree supports the use of features from distinct domains (or modalities) in a single index structure and it adopts a monotonic increasing *score* function for dealing with fusion (*early fusion*) – similar to *M²-tree*, but one should notice that *M²-tree* only supports multimodal queries. Since *Slim²-tree* borrows (and adapts to the "*multimodal world*") some of the node management policies used during insertion from the *Slim-tree*, it is named after it (although it is good to remember that a single *Slim-tree* can not easily handle multimodal queries by itself).

Unimodal data structures, such as *M-tree* and *Slim-tree*, are capable of indexing objects from a single metric space $\mathcal{M} = (\mathcal{D}, d)$ – in which $\mathcal{D}$ represents the data domain and d is a (metric) distance function used to assess the (dis)similarity between objects. *Slim²-tree* allows indexing of objects represented by a collection of metric spaces $\mathcal{M}^n = \{(\mathcal{D}_i, d_i), \forall i = 1, \ldots, n\}$, in a way that each pair $(\mathcal{D}_i, d_i)$ is associated with a different modality. Since *Slim²-tree* nodes represent hyperregions of that collection of metric spaces, it is possible to use triangular inequality for search and insertion operations. Thus,

Table I. Structure of an entry belonging to a leaf node.

Symbol	Definition
$O_j.F_i$	Feature value for object O_j in domain (or modality) $\mathcal{D}_i$.
$oid(O_j)$	Object identifier O_j.
$d_i(O_j, P(O_j))$	Distance, in domain $\mathcal{D}_i$, from object O_j to its pivot.

Table II. Structure of an entry belonging to an internal node.

Symbol	Definition
$O_r.F_i$	Feature value for representative O_r in domain (or modality) $\mathcal{D}_i$.
ne	Number of entries stored in subtree $T(O_r)$.
$ptr(T(O_r))$	Pointer to the root of subtree $T(O_r)$.
$r_i(O_r)$	Covering radius associated to object O_r in domain $\mathcal{D}_i$.
$d_i(O_r, P(O_r))$	Distance, in domain $\mathcal{D}_i$, from object O_r to its pivot.

there are n distinct domains $\mathcal{D}_i$, one for each modality, adopting a particular (and, perhaps, distinct) distance function d_i defining a hyperregion that is associated to each *Slim²-tree* node. Therefore, the hyperregion associated with a tree node is defined by all those n distance functions, and it is not just a simple hypersphere (as in *Slim-tree*) but it represents the data space in a more complex way. Thus, when n features $F_1, \ldots, F_n$ are considered along with their distance functions $d_1, \ldots, d_n$ and a representative object $O_r = (O_r.F_1, \ldots, O_r.F_n) \in \mathcal{D}_1 \times \ldots \times \mathcal{D}_n$, the hyperregion associated to O_r in the collection of metric spaces $\mathcal{M}^n$ can be seen as the non negative orthant of the n domain space.

In a leaf node, an entry for each object O_j is used to store the object identifier $oid(O_j)$, the values for its n features F_i $\forall i = 1, \ldots, n$, along with n distance values $d_i(O_j, P(O_j))$ between O_j and its parent (or pivot) $P(O_j)$, *i.e.*, the routing object O_r which is stored in the node N^p (from a level immediately above the leaf node) and which is used to make a reference to node N (where object O_j is stored). Table I presents the structure of an entry belonging to a leaf node. Thus, a leaf node entry should be defined as follows:

$$entry(O_j) = \langle oid(O_j), array\_of\langle O_j.F_i, d_i(O_j, P(O_j)) \ \forall i = 1, \ldots, n\rangle\rangle.$$

On the other hand, in an internal node, each entry for a routing object (or pivot) O_r is used to store: a pointer $ptr(T(O_r))$ to the root of subtree $T(O_r)$ covered by (related to) the routing object O_r; the number of objects stored in the subtree $T(O_r)$; the n values for the features F_i assigned to the routing object O_r during promotion, along with n values for covering radii $r_i(O_r) > 0$ and n distance values $d_i(O_r, P(O_r))$ between O_r and its parent $P(O_r)$ which is stored in a higher level of the tree (except for the tree root). Table II presents the structure of an entry belonging to an internal node. Therefore, an internal node entry should be defined as follows:

$$entry(O_r) = \langle ne, ptr(T(O_r)), array\_of\langle O_r.F_i, d_i(O_r, P(O_r)), r_i(O_r) \ \forall i = 1 \ldots n\rangle\rangle.$$

The creation and use of *Slim²-tree* for performing similarity queries are delineated in the following. In the work of Sperandio et al. [2014], all algorithms for creating and searching are fully described.

3.1 Construction of a *Slim²-tree*

The building algorithms for *Slim²-tree* specify how objects are inserted in the tree structure as well as how node overflows are treated. First, *Slim²-tree* building parameters should be set: the number of modalities – n; the domain of each modality with its distance function, *i.e.*, $\mathcal{M}^i = (\mathcal{D}_i, d_i), \forall i = 1, \ldots, n$; the *score* function S_f to be used in *early fusion* strategy; and the maximum number of objects per node – m. As shown by Ciaccia and Patella [2000], any monotonic increasing function combining the distances of all modalities could be used as *score* function, such as $\max\{d_i, \forall i = 1, \ldots, n\}$, or $\sum_{i=1,\ldots,n} \theta_i d_i$, with $\theta_i \geq 0, \forall i = 1, \ldots, n$ and $\sum_{i=1,\ldots,n} \theta_i = 1$.

During the `insertion` of a new object, the procedure recursively descends the *Slim²-tree* trying to locate the most suitable leaf node for accommodating the new object O_n. This could possibly trigger a node split if the selected leaf is full. The principle used to select the most suitable leaf node to insert the new object is to locate a subtree $T(O_r)$ which does not need an enlargement of its covering radii, *i.e.*, $d_i(O_r, O_n) \leq r_i(O_r), \forall i \in 1, \ldots, n$. If multiple subtrees are qualified, the choice is made

using one of the following heuristics (which are similar to the ones of *Slim-tree*): (i) `Random`: choose a subtree randomly; (ii) `MinDist`: choose the subtree whose pivot is closer to the new object; and (iii) `MinOccup`: choose the subtree which contains the smallest number of objects. In this article, the choice of the subtree where the new object is inserted is made by `MinOccup` since Traina et al. [2000] have demonstrated its efficiency for *Slim-tree*.

The search for routing objects which do not need an enlargement in their covering radii is optimized though the use of *triangular inequality*, reducing the computational cost since many distance calculations are avoided. If there is no routing object satisfying $d_i(O_r, O_n) \leq r_i(O_r), \forall i \in 1, \ldots, n$, the choice will be made heuristically minimizing the increasing of the *score* function S_f, *i.e.*, choosing a routing object $\tilde{O}_r$ such as

$$\tilde{O}_r = \arg\min_{O_r} \{S_f(d_i(O_r, O_n) - r_i(O_r)), \forall i \in 1, \ldots, n\},$$

and this is possible since S_f is monotonic increasing. Here, the main idea is to minimize the average covering volume for each domain associated with the routing object.

Similar to what happens with other dynamic trees, *Slim²-tree* is built in a *bottom-up* approach. When an overflow occurs in a node N, a new sibling node N' is created (in the same level) and all entries are divided between the two nodes. Moreover, the copies of two objects O_{r_1} and O_{r_2} are selected and promoted to the father node and they will be used as pivots (routing objects) for the new regions represented by the nodes N and N'. When the tree root is divided, a new root is created and the tree increases one level. An "ideal" policy for promotion should select two routing objects and divide the other elements into two subsets in a way to reduce the regions volume and, consequently, the overlapping between them. The division policy adopted in *Slim²-tree* follows an approach based on MST – first used by *Slim-tree* – see [Sperandio et al. 2013; 2014] for a detailed description.

3.2 Processing Similarity Queries in *Slim²-tree*

Slim²-tree supports a multimodal range query (with a fixed value for search radius) and a multimodal search for the k-nearest neighbors. More important, it also allows both multimodal and unimodal (in which only one modality is used) search using the same structure for that.

Range queries in *Slim²-tree* could be unimodal – in which only one modality is considered – or multimodal, when the *score* function is used to analyze all modalities simultaneously. Given a collection of indexed objects C, a query object O_q and a query radius $r(O_q)$, a unimodal range query $\texttt{range}_i(O_q, r(O_q), C)$ for a given modality i consists in returning all objects $O_j \in C$ for which $d_i(O_j, O_q) \leq r(O_q)$, for a given modality i. On the other hand, a multimodal range query using a monotonic increasing *score* function S_f – $\texttt{range}_{S_f}(O_q, r(O_q), C)$ – consists in locating all objects $O_j \in C$ for which $S_f(d_i(O_j, O_q), \forall i = 1, \ldots, n) \leq r(O_q)$. It is worth mentioning that the *score* function S_f should be the same used during insertions. Another range query that is possible to be made is a multimodal search which allows distinct values for each modality radius $\texttt{range}_r(O_q, r_1(O_q), \ldots, r_n(O_q), C)$. In this case, the search will return all objects $O_j \in C$ for which $d_i(O_j, O_q) \leq r_i(O_q), \forall i = 1, \ldots, n$.

Search algorithms are omitted due to space limitations – see [Sperandio et al. 2013; 2014] for a detailed description. In all of them, *triangular inequality* is used to optimize the search by reducing the number of both distance calculations and disk accesses.

The `kNN_Search` algorithm searches for the k-nearest neighbors given a query object O_q – supposing that *Slim²-tree* has at least k objects. The steps of the `kNN_Search` are similar to the ones of a range search; however, the criteria used to obtain performance improvements are dynamic, since query radii are defined based on the distances between O_q and O_k – which represents the k-th closest neighbor in any moment during the search. The implementation of `kNN_Search` could also be unimodal or multimodal – using only one feature and distance function for the unimodal case; or adopting the *score* function S_f for the multimodal version – see [Sperandio et al. 2014].

4. MULTIMODAL *SLIM-DOWN* ALGORITHM

Similarly to the work of Traina et al. [2000], a multimodal *post-processing* algorithm is proposed for *Slim²-tree* – multimodal *Slim-down*. The main idea is to move an element from a leaf node to a sibling node without increasing the radius (or radii) of the hyperregion associated with the receiving node. If the selected object is the farthest from the pivot of the supplying node, this may allow a reduction of the radius (radii) of the hyperregion associated with it. Therefore, the use of multimodal *Slim-down* may help improving search performance of *Slim²-tree*, since it could minimize the overlap between nodes.

Multimodal *Slim-down* algorithm (see Algorithm 1 and Algorithm 2) is similar to the one adopted by *Slim-tree*, *i.e.*, the farthest object O_f belonging to a node N (which is responsible for defining the radii of the associated hyperregion) is selected and it is moved to a sibling node N' which already covers the object, so there is no need of modifying the radii of N'. This heuristic procedure is applied until no more exchanges between nodes are possible or until the maximum number of iterations is reached (to prevent an infinite loop) – see *lines 24–27* in Algorithm 2. However, since a *Slim²-tree* node is associated with a hyperregion defined by multiple domains, different policies could be used to select the farthest object O_f within a tree node N. Two of those policies are considered in this article (see *lines 15–20* in Algorithm 2): (i) `POL1`: choose the node element that presents all distance values to the pivot (for each modality) larger than the others; and (ii) `POL2`: choose any node element that presents the larger distance value to the pivot (for any modality) when compared to the others.

The first policy `POL1` selects an element that is responsible for defining all the radii of the associated hyperregion (this policy is the one implemented in *line 16* of Algorithm 2), while the second policy `POL2` chooses any element associated with at least one of the radii of the hyperregion associated with the node. `POL2` is obtained by changing *line 16* to one that looks for just one larger distance value, *i.e.*, $\exists i \in 1, \ldots, n \mid d_i(O_j, O_p)) > D_f[i]$ (*line 18* should also be adjusted accordingly). Although policy `POL1` is more similar to the original unimodal version of *Slim-down*, in which generally only one element is responsible for defining the radius of the hypersphere in *Slim-tree*, it may not provide the best results since it might be harder to find an element that is solely responsible for defining all the radii of the associated hyperregion, and, therefore, no object would be moved between nodes. On the other hand, policy `POL2` always finds at least one candidate object which is related to the maximum distance from the pivot for each modality and, consequently, it may produce better results.

Traina et al. [2000] only present results for one scenario of *Slim-down* application (*i.e.*, when it is executed after the tree creation), since they did not obtain different results for distinct scenarios. However, here two different contexts are considered for the use of multimodal *Slim-down*: (i) `SDW1`: when it is executed only once after the *Slim²-tree* is fully constructed; and (ii) `SDW2`: when it is executed repeatedly after num_sdw insertions. There is a possible interference between the context of application (`SDW1` or `SDW2`) and the selected policy (`POL1` or `POL2`). Actually, `POL1` should only be affected by `SDW2` for very small values of num_sdw, since it would be easier to find an element

Algorithm 1 Multimodal *Slim-down* algorithm applied to *Slim²-tree*.

```
 1: procedure SlimDown(RootNode)
 2:     num_repeat ← 0
 3:     abort ← false
 4:     repeat
 5:         moved ← false
 6:         abort ← SlimDownalyze(RootNode)          ▷ Starts recursive procedure.
 7:         if abort = true then
 8:             break
 9:         end if
10:     until moved = true                           ▷ Repeat again every time an object is moved.
11: end procedure
```

Algorithm 2 Recursive procedure responsible for analyzing tree nodes and executing *Slim-down*.

```
1: procedure SlimDownalyze(N)
2:     Input: N – Subtree root node (initially Slim²-tree root node).
3:     if N ¬isLEAF then
4:         for all O_r ∈ N do                          ▷ Recursively call to SlimDownalyze for each node entry.
5:             abort ← SlimDowalyze(ptr(O_r))
6:             if abort = true then
7:                 return abort;
8:             end if
9:         end for
10:    else if N isLEAF and N ¬isROOT then             ▷ Try to reduce the overlap involving a leaf node.
11:        O_p ← Pivot(N)                              ▷ Representative of N.
12:        O_f ← O_p                                   ▷ Initialize the farthest element of N.
13:        D_f[i] ← d_i(O_f, O_p)∀i ∈ 1, ..., n        ▷ Initialize distances between O_f and O_p.
14:        moved ← false
15:        for all O_j ∈ N do                          ▷ Look for the farthest element of N.
16:            if d_i(O_j, O_p) > D_f[i]∀i ∈ 1, ..., n then
17:                O_f ← O_j                           ▷ A new farthest element is found.
18:                D_f[i] ← d_i(O_j, O_p)∀i ∈ 1, ..., n
19:            end if
20:        end for
21:        N^p ← Parent(N)                             ▷ Parent node of N.
22:        N^s ← N ∈ {N^p \ {N}}                       ▷ Set of siblings of N.
23:        num_repeat ← num_repeat + 1
24:        if num_repeat > 3 × NEntries(N^p) then
25:            abort ← true                            ▷ Maximum number of iteration is reached.
26:            return abort
27:        end if
28:        for all N' ∈ N^s do                         ▷ Search for suitable node among the siblings of N.
29:            O_p' ← Pivot(N')
30:            if (d_i(O_f, O_p') ≤ r_i(O_p')∀i ∈ 1, ..., n) and N' ¬isFULL then
31:                moved ← true                        ▷ A suitable node is found.
32:                break
33:            end if
34:        end for
35:        if moved = true then
36:            Move(O_f, N, N')                        ▷ Move O_f from N to N'.
37:        end if
38:    end if
39: end procedure
```

that were solely responsible for all radii of a tree node. For larger values of *num_sdw*, `POL1` may not promote any performance improvement, since it might be harder to find any suitable candidates. This is similar to what it is expected when `POL1` is applied at end of Slim-tree construction (`SDW1`). However, `POL2` may promote a greater improvement of *Slim²-tree* performance, since it might find many candidates, especially when `SDW2` is adopted (as shown in the analysis of test results).

5. EXPERIMENTAL RESULTS

In this section, we present some experiments in order to assess the performance of *Slim²-tree*. Our video corpora consists of TV broadcast, recorded directly and continuously from cable TV channels, and Internet retrieved videos. The dataset is composed by 48 videos of different categories/genres, like documentary, movie, news, TV series, and others (almost 21 hours). Each video was divided in fixed length videoclips of 01 second and, then, each one of these clips has its visual and acoustic features extracted and stored in the *Slim²-tree* – a total of 74,744 videoclips. Those short-length videoclips were used to avoid degradation with time, especially in relation to the acoustic data which may change a lot in a short period of time. Table III shows some information about our video dataset and Fig. 1 presents some samples of frames belonging to videos from each distinct class of the dataset. As one can see in Fig. 1, since videos were recorded continuously, the dataset presents a great variety of content even inside of the same class. And this poses a great challenge during the search of some videoclips among the others of the same class (as it will be shown forward), since for some classes there is not a strong correlation (neither visual nor acoustic) among the contents of their videoclips. The experiments were performed on a Xeon 2.40GHz with 48 GB of main memory. All structures

Table III. Information about video dataset and queries.

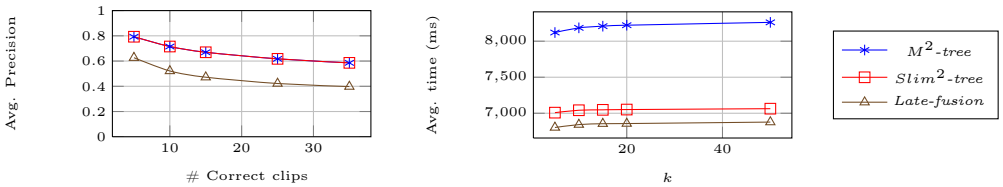
Category	Class	# Videos	# Clips	Length	# Queries	Percentage
Musical	Adele	04	5,878	01:37:59	284	4.83%
	RogerWaters	04	18,102	05:01:43	876	4.84%
Documentary	Africa	03	2,701	00:45:01	89	3.30%
	Ocean	01	2,699	00:44:59	129	4.78%
Sports	F1	05	4,844	01:20:47	234	4.83%
	Soccer	16	9,954	02:46:01	481	4.83%
News	TV news	03	6,671	01:51:12	322	4.83%
Cartoon	Simpsons	03	3,865	01:04:27	129	3.33%
Variety	Nigella	03	5,169	01:26:10	235	4.55%
	TopGear	03	11,027	03:03:48	538	4.88%
TV series	BigBang	03	3,834	01:03:55	183	4.77%
Total		48	74,744	20:45:44	3,500	—

were implemented from scratch in Java under Debian Linux 3.9.8-1/ *OpenJDK RT (IcedTea 2.3.9)*.

For each videoclip, only one visual keyframe is extracted (in the middle of the clip), while acoustic frames of 01 second are sampled at 22.05 Hz (16 bits) monaural. In our experiments, visual data are described using GIST [Oliva and Torralba 2001] and acoustic information is represented by *Mel-Frequency Cepstral Coefficient* – MFCC [Ganchev et al. 2005]. Most of the works using the GIST descriptor resize the image in a preliminary stage, producing a small square image. Following the work of Douze et al. [2009], we adopt a size of 32×32 pixels – and visual frames are rescaled to that size irrespective of their aspect ratio. After resizing, color GIST descriptor is calculated for each visual frame producing a 960-dimensional feature vector. In the acoustic domain, MFCC is used to describe the one-second audio frames adopting a 40 ms window with 25% overlap which generates a 13-dimensional feature vector for each window – those vectors are concatenated in a single 377-dimensional feature vector for each acoustic frame. A simple *min-max* normalization is applied for both domains. Finally, a total number of 149,488 (visual and acoustic) descriptors are stored in a *Slim²-tree*.

Since we are not able to find any data structure developed specifically for multimodal content-based video retrieval, we compare the results of *Slim²-tree* with those obtained by an implementation of *M²-tree* and by a *late fusion* strategy based on the combination of the results generated by two distinct *Slim-trees* – one for each modality. We have also developed an implementation of *M²-tree* based on the standard policies of *M-tree*, as described by Ciaccia and Patella [2000]. The *score* function $S_f = \max\{d_i, \forall i = 1, \ldots, n\}$ was used with *Slim²-tree* and *M²-tree* – the same function adopted

Fig. 1. Some samples of frames belonging to videos from each distinct class of the dataset.



(a) Precision (b) Time

Fig. 2. Average precision and time for multimodal k-NN search (Global).

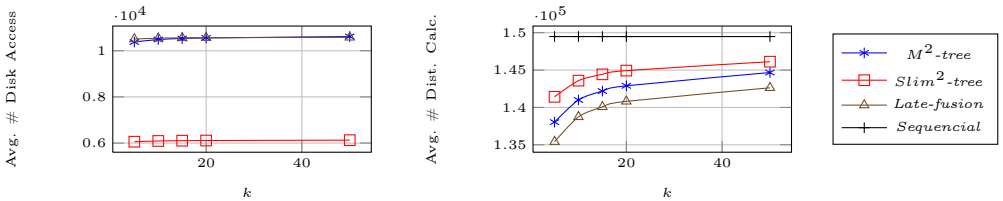
by Ciaccia and Patella [2000]; and the maximum number of objects per node m is set to 30.

Tests were made using a set of 3,500 queries selected randomly from the dataset ($\approx 5\%$ of the video dataset). Table III also presents information about the query clips. In order to evaluate query results, we consider as correct (relevant) answers any clip that belongs to the same class of the query. This was done due to the lack of a standard benchmark for evaluating multimodal CBVR using visual and acoustic data. Moreover, the use of this criteria to establish a ground-truth emulates better the subjective evaluation found in a human assessment.

5.1 Results for Similarity Queries

Fig. 2(a) presents average precision values obtained by *Slim²-tree*, *M²-tree*, and *late fusion* strategy for all 3,500 queries. It can be easily seen that *Slim²-tree* and *M²-tree* present similar results and both are superior to those of *late fusion* strategy. Moreover, the *mean average precision* – MAP values obtained by *Slim²-tree*, *M²-tree* and *late fusion* strategy are 21.62%, 21.62% and 21.45%, respectively (which are statistically equivalent with a confidence level of 95%), but they present distinct values for computational time. Fig. 2(b) shows the average time per query associated with *Slim²-tree*, *M²-tree* and *late fusion* strategy, while Fig. 3 presents the average number of disk accesses and distance calculations for multimodal k-NN search. *Slim²-tree* exhibits an expressive reduction of the disk accesses when compared to the other approaches ($\approx$ 40%). Considering the distance calculations, *Slim²-tree* and *M²-tree* have presented very similar results (less than 1.3% of difference). In Fig. 4, the average precision values for some video classes are shown (one class for each category). It is worth mentioning that results obtained by both *Slim²-tree* and *M²-tree* are superior to the ones achieved by *late fusion* strategy. In particular, the improvement is larger for categories and classes in which there is a greater correlation between visual and acoustic content, as well as in the situations when they exhibit little variability through the entire duration of the video content, because this may help in the recognition of a clip of the same class – see the results obtained for classes *Adele* and *F1*.

Slim²-tree also performs unimodal search (based on only one modality). In order to assess this capacity, the results obtained for unimodal search by *Slim²-tree* are compared to those obtained by a *Slim-tree* using only one modality and by an adaptation of *M²-tree* developed for this purpose (notice that the original version of *M²-tree* does not exhibit that capacity). Since those methods are exact, precision results obtained by these three approaches are identical (which was already expected), but



(a) Disk access (b) Distance calculation

Fig. 3. Avg. number of disk accesses and distance calculations for multimodal k-NN search (Global). The number of disk accesses for sequential access is omitted since it is much larger than the others.

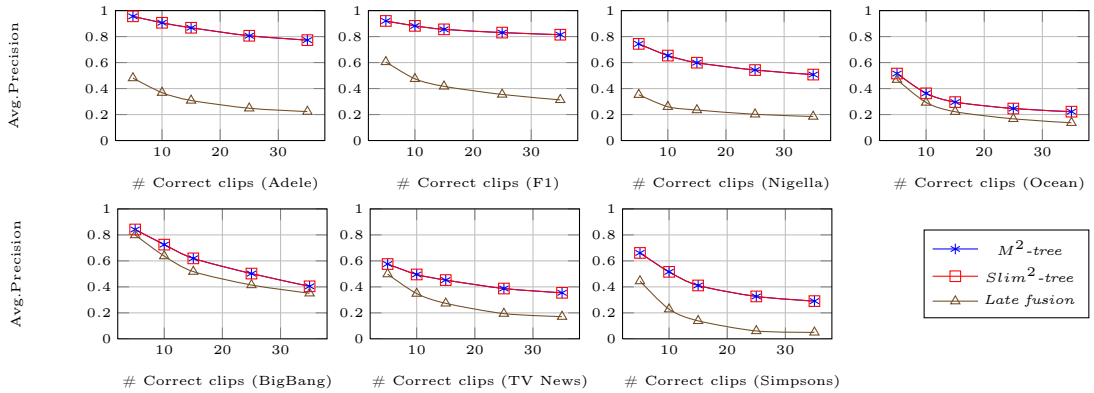
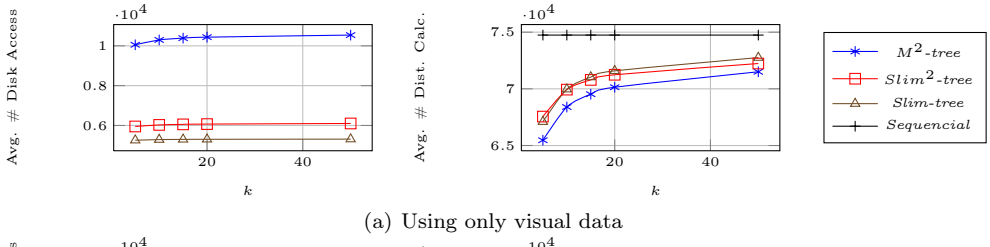


Fig. 4. Average precision per class for multimodal k-NN search.

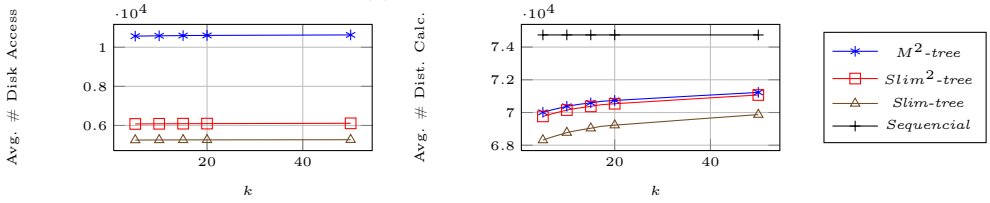
the computational cost associated with the unimodal search made by *Slim²-tree* is very close to the one presented by the unimodal search made by a *Slim-tree* using only one modality. Fig. 5(a) presents the average values of disk accesses and distance calculations for the k-NN search using only visual modality, while Fig. 5(b) shows the same metrics when only acoustic modality is used. Considering the distance calculations, both *Slim²-tree* and *M²-tree* adaptation have achieved results that are close to the ones presented by a *Slim-tree* for only one modality. But, in relation to the disk accesses, *Slim²-tree* exhibits a performance quite superior to *M²-tree* adaptation, and again its results are close to the ones presented by a *Slim-tree* for only one modality. This is a great advantage of *Slim²-tree* when compared to the other analyzed multimodal approaches, because it is possible with a single data structure (the *Slim²-tree*) to perform both multimodal and unimodal searches. Moreover, *Slim²-tree* is capable of answering unimodal queries with a computational cost very similar to the one associated to a "specialized" data structure built using only one modality.

5.2 Results for Multimodal *Slim-down*

In order to assess the multimodal *Slim-down* algorithm, the same set of 3,500 queries was used again. Since *Slim²-tree* represents an exact solution for similarity queries no difference in the quality of the results (values of average precision) should be expected, but performance numbers should improve if



(a) Using only visual data

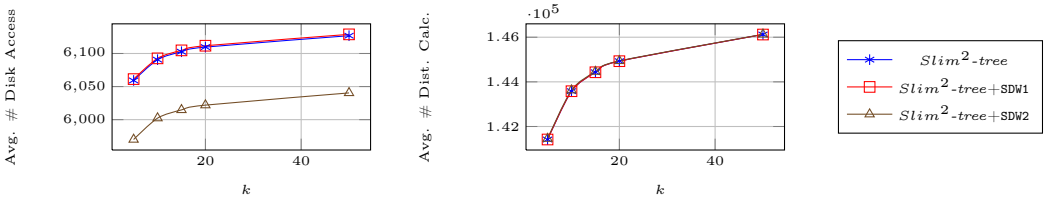


(b) Using only acoustic data

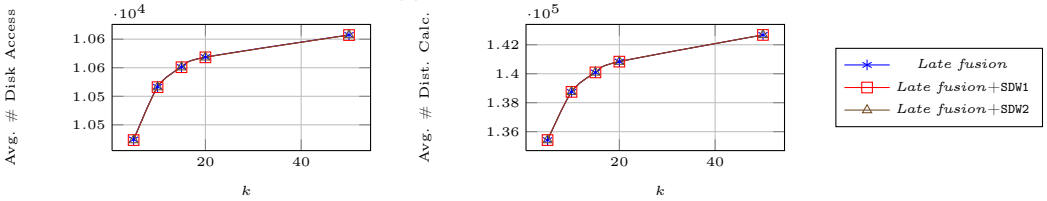
Fig. 5. Average number of disk accesses and distance calculations for unimodal k-NN search (Global). The number of disk accesses for sequential access is omitted since it is much larger than the others.

Table IV. Total number of disk accesses, distance calculations and time spent by *Slim-down* using POL2.

Context	SDW1	SDW2
# Disk accesses	2,190	225,374
# Dist. calculations	1,090	15,599
Time (ms)	1,947	79,568



(a) Results for *Slim²-tree*



(b) Results for *Late fusion*

Fig. 6. Average number of disk accesses and distance calculations for multimodal k-NN search before and after using *Slim-down* (Global).

multimodal *Slim-down* really helps minimizing the overlap between nodes. Tests were made with the two policies – POL1 and POL2, considering both context – SDW1 and SDW2 (with $num\_sdw = 60$ which is the double of the *Slim²-tree* node capacity). Independently of the context (SDW1 or SDW2), tests with policy POL1 did not present any improvement of *Slim²-tree* performance and their results will be omitted. This can be explained by the fact that policy POL1 has only selected a very few candidates to be moved, and, therefore, a great minimization of the overlap between nodes was not possible.

Table IV shows the total number of disk accesses, distance calculations and time spent with multimodal *Slim-down* with policy POL2. It is worth mentioning that context SDW2 took 40 times more time them SDW1. However, tests with POL2 considering context SDW2 have achieved a reduction of 1.5% in the average number of disk accesses without compromising the average number of distance calculations, while context SDW1 did not show any improvement, see Fig. 6(a). *Slim-down* was also applied to the unimodal *Slim-trees* used for the *late fusion* approach. But in this case, no improvement has been observed independently of the context used (SDW1 or SDW2) – see Fig. 6(b). Fig. 7 presents the average time for some video classes. As one can see the impact of multimodal *Slim-down* over the average time of multimodal k-NN search is greater for lower values of k (≤ 20).

6. CONCLUSION

Slim²-tree is a new multimodal metric tree which enables video indexing and retrieval by using information from multiple modalities. Experimental results have demonstrated its efficiency when compared to other multimodal solutions. Moreover, it also supports queries using only one modality with a computational cost similar to an unimodal solution. This article explores different strategies related to the use of a *post-processing* algorithm for the *Slim²-tree* – named multimodal *Slim-down*. Experiments have also shown the performance improvement obtained by the policy, in which any element that presents the larger distance value to the pivot for any modality is selected as candidate to be moved, i.e., any element which defines at least one of the radii of the hyperregion associated with the node. Moreover the results are better when that policy is repeatedly used during insertion. Future works should address the impact of different *score* functions in the *Slim²-tree* results. Moreover, it is

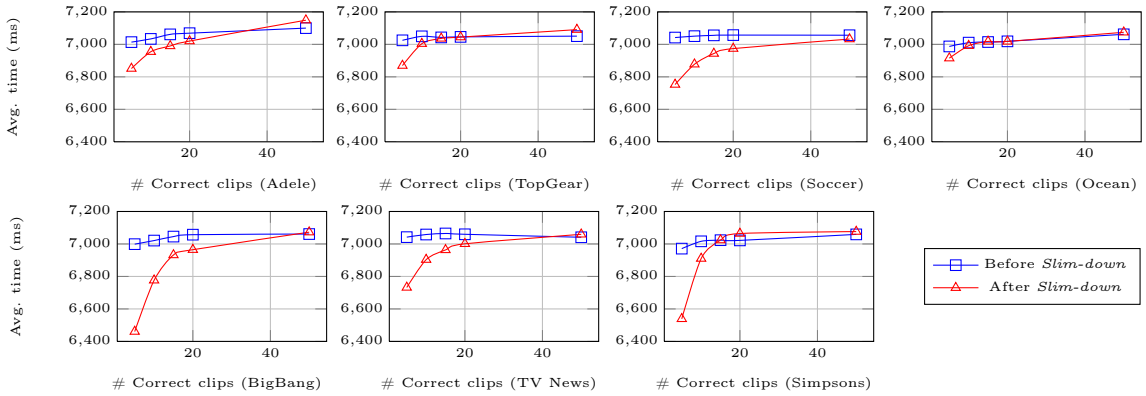


Fig. 7. Average time per class for multimodal k-NN search before and after using *Slim-down*.

also important to assess the *Slim^2-tree* performance when facing an increasing number of modalities and when multiple features of a same modality are used.

REFERENCES

Almeida, J., Valle, E., Torres, R. S., and Leite, N. J. DAHC-Tree: An Effective Index for Approximate Search in High-Dimensional Metric Spaces. *Journal of Information and Data Management* 1 (3): 375–390, 2010.

Atrey, P. K., Hossain, M. A., El Saddik, A., and Kankanhalli, M. S. Multimodal Fusion for Multimedia Analysis: A Survey. *Multimedia Systems* 16 (6): 345–379, 2010.

Bustos, B., Kreft, S., and Skopal, T. Adapting Metric Indexes for Searching in Multi-Metric Spaces. *Multimedia Tools and Applications* 58 (3): 467–496, 2012.

Chávez, E., Navarro, G., Baeza-Yates, R., and Marroquín, J. L. Searching in Metric Spaces. *ACM Computing Surveys* 33 (3): 273–321, 2001.

Ciaccia, P. and Patella, M. The M^2-Tree: Processing Complex Multi-Feature Queries with Just One Index. In *Proceedings of the DELOS Network of Excellence Workshop on Information Seeking, Searching and Querying in Digital Libraries*. Zurich, Switzerland, 2000.

Ciaccia, P., Patella, M., and Zezula, P. M-Tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *Proceedings of the International Conference on Very Large Data Bases*. Athens, Greece, pp. 426–435, 1997.

Döller, M., Stegmaier, F., Jans, S., and Kosch, H. TempoM^2: A Multi Feature Index Structure for Temporal Video Search. In *Proceedings of the International Conference on Multimedia Modeling*, K. Schoeffmann, B. Merialdo, A. Hauptmann, C.-W. Ngo, Y. Andreopoulos, and C. Breiteneder (Eds.). Lecture Notes in Computer Science, vol. 7131. Springer, Klagenfurt, Austria, pp. 323–333, 2012.

Douze, M., Jégou, H., Sandhawalia, H., Amsaleg, L., and Schmid, C. Evaluation of GIST Descriptors for Web-Scale Image Search. In *Proceedings of the ACM International Conference on Image and Video Retrieval*. Santorini, Fira, Greece, pp. 19:1–19:8, 2009.

Gaede, V. and Günther, O. Multidimensional Access Methods. *ACM Computing Surveys* 30 (2): 170–231, 1998.

Ganchev, T., Fakotakis, N., and Kokkinakis, G. Comparative Evaluation of Various MFCC Implementations on the Speaker Verification Task. In *Proceedings of the International Conference on Speech and Computer*. Vol. 1. University of Patras, Patras, Greece, pp. 191–194, 2005.

Goh, S.-T. and Tan, K.-L. MOSAIC: A Fast Multi-Feature Image Retrieval System. *Data & Knowledge Engineeering* 33 (3): 219–239, 2000.

He, Y. and Yu, J. MFI-Tree: An Effective Multi-Feature Index Structure for Weighted Query Application. *Computer Science and Information Systems* 7 (1): 139–152, 2010.

Oliva, A. and Torralba, A. Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope. *International Journal of Computer Vision* 42 (3): 145–175, 2001.

Shao, J., Shen, H. T., and Zhou, X. Challenges and Techniques for Effective and Efficient Similarity Search in Large Video Databases. *Proceedings of the VLDB Endowment* 1 (2): 1598–1603, 2008.

Sperandio, R. C., Patrocínio, Jr., Z. K. G., Paula, H. B., and Guimarães, S. J. F. An Efficient Access Method for Multimodal Video Retrieval (in portuguese). In *Proceedings of the Brazilian Symposium on Multimedia and the Web*. Salvador, Brazil, pp. 31–38, 2013.

Sperandio, R. C., Patrocínio, Jr., Z. K. G., Paula, H. B., and Guimarães, S. J. F. An Efficient Access Method for Multimodal Video Retrieval (accepted). *Multimedia Tools and Applications*, 2014.

Traina, C., Traina, A., Seeger, B., and Faloutsos, C. Slim-Trees: High Performance Metric Trees Minimizing Overlap Between Nodes. In *Proceedings of the International Conference on Extending Database Technology*, C. Zaniolo, P. Lockemann, M. Scholl, and T. Grust (Eds.). Lecture Notes in Computer Science, vol. 1777. Springer-Verlag, Konstanz, Germany, pp. 51–65, 2000.

Yan, R. and Hauptmann, A. G. A Review of Text and Image Retrieval Approaches for Broadcast News Video. *Information Retrieval* 10 (4-5): 445–484, 2007.