Noname manuscript No.
(will be inserted by the editor)

Improving retrieval accuracy of Hierarchical Cellular Trees for generic metric spaces

Carles Ventura · Verónica Vilaplana · Xavier Giró-i-Nieto · Ferran Marqués

Received: date / Accepted: date

Abstract Metric Access Methods (MAMs) are indexing techniques which allow working in generic metric spaces. Therefore, MAMs are specially useful for Content-Based Image Retrieval systems based on features which use non L_p norms as similarity measures. MAMs naturally allow the design of image browsers due to their inherent hierarchical structure. The Hierarchical Cellular Tree (HCT), a MAM-based indexing technique, provides the starting point of our work. In this paper, we describe some limitations detected in the original formulation of the HCT and propose some modifications to both the index building and the search algorithm. First, the covering radius, which is defined as the distance from the representative to the furthest element in a node, may not cover all the elements belonging to the node's subtree. Therefore, we propose to redefine the covering radius as the distance from the representative to the furthest element in the node's subtree. This new definition is essential to guarantee a correct construction of the HCT. Second, the proposed Progressive Query retrieval scheme can be redesigned to perform the nearest neighbor operation in a more efficient way. We propose a new retrieval scheme which takes advantage of the benefits of the search algorithm used in the index building. Furthermore, while the evaluation of the HCT in the original work was only subjective, we propose an objective evaluation based on two aspects which are crucial in any approximate search algorithm: the retrieval time and the retrieval accuracy. Finally, we illustrate the usefulness of the proposal by presenting some actual applications.

F. Author
first address
Tel.: +123-45-678910
Fax: +123-45-678910
E-mail: fauthor@example.com

S. Author
second address

Keywords Multimedia Retrieval · Content-Based Image Retrieval · Indexing techniques · Metric Access Methods · Hierarchical Cellular Tree

1 Motivation

Content-Based Image Retrieval (CBIR) systems rely on the automatic extraction of visual descriptors based on features such as color, texture and shape. These descriptors are compared using some similarity measures in order to infer how similar two images are. A classic application of CBIR systems is the retrieval of images similar to a provided exemplar. This approach is known as Query by Example (QbE) [NBE$^+$93], and basically generates a ranked list of images in the database according to their similarity to a query image provided by the user. Another way to retrieve elements from an image database is browsing, which is specially useful when the user does not have any example query image or any particular target content in mind.

As a consequence of recent technology developments, large amounts of images are generated and stored. For instance, audiovisual media companies store their broadcasted content in large private systems, which need to be easily accessible. However, an exhaustive search would not be feasible in such large databases due to its linear computation time behavior. In this context, well organized databases and efficient storing and retrieval are absolutely necessary. Therefore, CBIR systems need to incorporate indexing techniques in order to scale up well when working over large databases. Furthermore, these indexing techniques should rely on a dynamic approach which allows insertions and deletions of the indexed elements. Multimedia databases are not static and, therefore, a dynamic approach is essential in order not to reindex the whole database every time an element is to be either inserted or removed.

Many CBIR systems use an L_p norm on an Euclidean space to compare visual features. Such features are usually indexed using some popular indexing techniques such as Locality Sensitive Hashing (LSH) [IM98] and Spatial Access Methods (SAMs) [BM72]. However, there also exist scenarios where the features may require a specific metric. This may happen when the feature contains attributes which should be treated independently or when a non-L_p norm behaves perceptually better than an L_p norm. In such cases, it is preferable to use an indexing technique which allows the features to be indexed in a generic metric space.

One of such cases is the set of visual descriptors defined in the MPEG-7 standard [BSMS02], where suggested similarity measures are often non-L_p norms and some descriptors, such as the Dominant Color Descriptor and the Contour-Shape Descriptor, do not lead to a fixed-size representation. Another case is SIFT [Low99], an algorithm which transforms an image into a large collection of local feature vectors. Although it is a common practice to use the L_2 norm to compare SIFT descriptors, works such as [PW08] and [LO07] identified an alignment problem and proposed the Earth Mover's Distance (EMD) [RTG00], which is a non-L_p norm, to address this problem.

In the context of image retrieval solutions for a national broadcasting company (BuscaMedia project [Bus]), a CBIR system based on the MPEG-7 visual descriptors supporting searches given a query image needs to be built. The goal is to design a retrieval system capable of approximate search techniques [PC09] under the assumption that the user may still prefer to quickly obtain an approximate result rather than to wait longer for the exact answer. Thus, this retrieval system requires an indexing technique which allows working in a generic metric space and speed up the retrieval operation with respect to the time required by an exhaustive search. Furthermore, the retrieval system is also required to provide an image browser for those scenarios where the user does not have any query image available. The Hierarchical Cellular Technique (HCT) [KG07] is an indexing technique which satisfies both requirements and provides the starting point of our work. However, some limitations have been detected in both HCT construction and retrieval scheme. With regard to the HCT construction, the original definition of a parameter named covering radius may lead to a HCT where elements may be wrongly inserted in a cell. Regarding the retrieval scheme, the original approach is based on a search strategy which assumes that the closest nucleus item yields the best subtree during the descend. Besides these two limitations, the evaluation of the original HCT was only subjective. In this paper, we propose some modifications to the original formulation of the HCT and to the retrieval system. In order to objectively evaluate them, the results obtained by the HCT search algorithms for a set of queries are compared with those obtained by an exhaustive search, using some contrasted measures extracted from the literature.

This paper is structured as follows. First, in Section 2 we analyze different indexing techniques to describe their qualities and limitations and choose the one that best fits the requirements. Next, in Section 3, we give an overview of the Hierarchical Cellular Tree (HCT). Section 4 introduces some limitations detected in the original formulation of the HCT and proposes some modifications to both the HCT building and the retrieval system. In Section 5, a set of experiments are presented to show the improvements achieved thanks to the proposed modifications. The retrieval accuracy of the new retrieval scheme over the HCT is objectively evaluated by using different measures in the experiments. Then, some actual applications which use the HCT are presented in Section 6. Finally, in Section 7 we draw the conclusions.

2 Related work

As previously discussed, indexing techniques are required by CBIR systems to scale up well over large databases. The most popular multimedia indexing techniques can be mainly grouped in two categories: (*i*) Locality Sensitive Hashing (LSH), and (*ii*) hierarchical tree structures.

LSH [IM98] is an efficient indexing method to search on large-scale and high-dimensional databases. The LSH algorithm relies on the existence of locality-sensitive hash functions. The principle of LSH is that nearby data points are mapped into the same bucket with a high probability while points faraway are hashed into the same bucket with a low probability.

Despite the popularity and the usefulness of the LSH algorithms, their applicability is limited by the fact that they have to be designed according to some specific metrics. Originally, LSH was designed assuming that the Euclidean distance was used to measure how similar two items are [IM98]. Although this original method was extended to a variety of similarity measures including L_p for $p \in (0, 2]$, Hammming, Mahalanobis, Jaccard, and Arccos distances [AI08], LSH algorithms cannot be designed in a generic metric space. Some efforts have been made to solve this problem, but the proposed approaches are hybrid strategies which do not use real hash functions as in [NKZ10]. Another limitation on the applicability of the LSH algorithms is that elements have to be represented by feature vectors of the same size. Therefore, LSH algorithms cannot be used over feature vectors of variable length, like, for example, MPEG-7 Dominant Color and Contour Shape descriptors.

The indexing techniques based on trees have a hierarchical structure which is formed by one or more levels, each of them holding one or more nodes. Each element belonging to a node from level l represents the elements hosted by one node from the lower level $l - 1$. Therefore, the tree structure gives the user an overview of what lies under the current level. This kind of indexing techniques provides at the same time an efficient browsing scheme given their inherent hierarchical structure. The indexing techniques based on trees can be mainly grouped in two categories: (*i*) Spatial Access Methods (SAMs) and (*ii*) Metric Access Methods (MAMs).

Search structures for vector spaces are called Spatial Access Methods (SAMs) [CNBYM01]. These techniques make extensive use of coordinate information to group and classify points in the space. They assume that the given data are embedded in an N-dimensional vector space, which is referred to as universe or original space. SAMs are responsible for the partitioning of the space, i.e. the process of dividing the space into non-overlapping regions which contain at least one data item.

The applicability of SAMs is limited by the fact that items have to be represented by the points in an N-dimensional feature space and the similarity measure between two points has to be based on a distance function in an L_p metric, such as the Euclidean distance. Furthermore, SAMs suffer from the "curse of dimensionality" problem and do not scale up well to high dimensional spaces, becoming less efficient than sequential indexing for dimensions higher than 10 [WSB98]. Even though approximate searching algorithms [ML09] have been proposed to overcome this problem, the limitation on the choice of the dissimilarity measure remains. As an example, the *k-d tree* [Ben75], which is one of the most popular SAMs, assumes that the similarity measure between two points must be defined in terms of one-dimensional distance functions along each coordinate. Therefore, the k-d tree cannot be used to index features based on distances which require matching algorithms such as EMD or some MPEG-7 similarity measures.

On the other hand, Metric Access Methods (MAMs) [CNBYM01] carry out the indexing process assuming only the availability of a distance function between elements and do not need the elements to be represented as points in an N-dimensional feature space. Therefore, MAM-based indexing techniques work in a generic metric space, where similarity is modeled with a distance that satisfies the triangle inequality.

M-tree [CPRZ97], one of the most popular MAMs, partitions elements on the basis of their relative distances and stores them into fixed-size nodes. Leaf nodes of any M-tree store all indexed database elements whereas internal nodes store the so-called routing element. M-tree insertion operation is based on Most Similar Nucleus cell search, which assumes that the closest routing element yields the best subtree during the descend. The starting point of our contribution is the Hierarchical Cellular Tree (HCT) [KG07], a MAM-based indexing technique similar to M-tree. One of the main differences is that HCT is a tree designed to achieve highly focused cells [KG07]. The reason is that the content variation of multimedia databases is seldom balanced. Consequently, HCT does not depend on a maximum (fixed size) capacity M as the M-tree does, and cells may exhibit variations on size and density. HCT has no limit for the cell size as long as cells are compact enough. Furthermore, the insertion processes differ significantly in terms of cell-search operations. Instead of the Most Similar Nucleus cell search used by M-tree, HCT performs the Preemptive Cell Search, a search algorithm which guarantees that the target cell to which the incoming item should belong is always found. In addition, HCT has a totally dynamic approach whereas M-tree has a conservative structure which may cause degradations in due time. These differentiating factors captured our attention and motivated this work. The reader is addressed to [KG07] for more differences in their design philosophies and objectives. Some CBIR systems based on HCT for efficient retrieval are presented in [KIP$^+$11], [AG11], [YLhF$^+$11] and [YD13].

3 The original Hierarchical Cellular Tree (HCT)

In this section we briefly describe the main features of the original HCT in order to introduce the improvements that are proposed in Section 4.

Hierarchical Cellular Tree (HCT) is an indexing technique designed by Kiranyaz and Gabbouj in [KG07]. The elements are partitioned depending on their relative distances and stored within *cells* on the basis of their similarity (like in clustering). Each group of elements (cell) has a representative, which is called *nucleus*. The distance from the nucleus to the furthest element in the cell is defined as *covering radius*. HCT has a hierarchical structure where the representatives of the cells at a given level l are in turn clustered to build the cells at level $l+1$. The ground level ($l = 0$) cells contain all items in the database (see Figure 1).

The HCT construction over an entire database is the result of dynamically adding each of its elements, one after the other, to the ground level. The insertion of a new item requires finding the most suitable cell from the ground level using an algorithm called *Preemptive Cell Search*. This insertion cell is defined as the cell represented by the closest nucleus element on the ground level.

In order to find this cell, the algorithm first compares the insertion element O with the elements O_i from the top level cell. The distance values $d(O, O_i)$ are jointly used with their respective covering radius $r(O_i)$ to discard the branches that cannot hold the insertion cell. Preemptive Cell Search algorithm descends through all non-discarded branches and iteratively performs the same operations at each level. The algorithm ends when the level $l = 1$ is reached. The closest element from this level to the insertion element is the nucleus of the insertion cell at the ground level. Formally, at a given level l, if d_{min} is the distance to the closest element on that level, then all elements O_i from level l that satisfy

$$d(O, O_i) - r(O_i) < d_{min} \tag{1}$$

are fetched for tracking, where O is the element to be inserted and $r(O_i)$ is the covering radius of the cell at level $l-1$ represented by O_i.

Figure 1 shows a possible hierarchical organization of a set of elements into a three-level HCT. These elements are spatially distributed as shown in Figure 2. In the process of finding the target cell from the ground level where the new element O should be inserted, the Preemptive Cell Search algorithm would discard C_3 since this subtree cannot yield to the target cell ($d(O, O_N^3) - r(O_N^3) > d_{min}$, where $d_{min} = d(O, O_N^1)$). Therefore, the algorithm only considers cells C_1 and C_2 when descends to the next level $l = 1$. Since this is the level above the ground level, the iterative process of descending through all non-discarded branches ends. All the elements belonging to cells C_1 and C_2 are compared with the insertion element O and the cell from the ground level represented by the closest element, i.e. element C, is found. This is the insertion cell where element O is appended to. In contrast with HCT, M-tree uses the Most Similar Nucleus (MS-Nucleus) search algorithm, which consists on descending only through the HCT branch which gives the most similar element at each level. In this example, using the MS-Nucleus instead of the Preemptive Cell Search algorithm would have yield the ground level cell

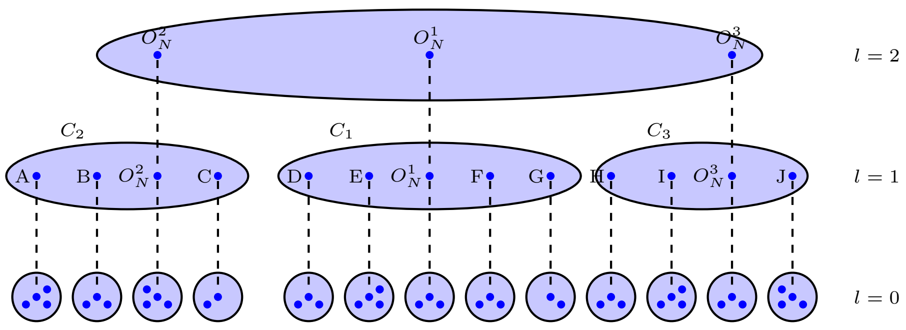

Fig. 1: HCT representation

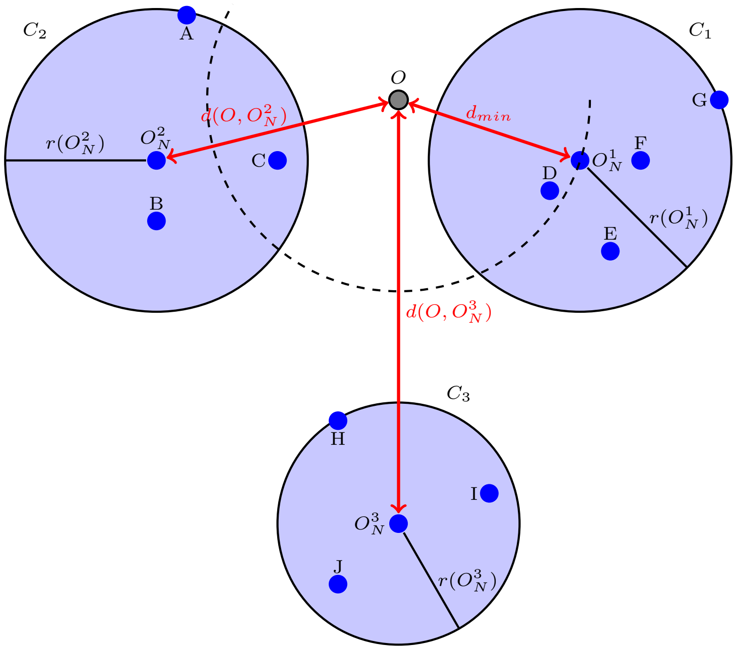

Fig. 2: Insertion stage of element O in a level with cells C_1, C_2 and C_3

represented by nucleus D since this algorithm would have descended through C_1, discarding C_2 and C_3, at first step. Therefore, in Figure 2, MS-Nucleus fails in finding the most suitable cell.

Once the element is added to the insertion cell, this cell passes a generic post-processing check to decide whether the cell must be split (the cell may not be compact enough) or the nucleus has changed due to the insertion. The reader is addressed to [KG07] for more details.

The authors of the original paper also proposed a retrieval scheme, called Progressive Query (PQ) [KG05], to be used over the HCT. PQ performs periodical sub-queries over subsets of database items and allows the user to interact with the ongoing query process. The order in which database items are processed is called Query Path (QP). In [KG07] the QP is created using the Most Similar Nucleus algorithm over the HCT. First of all, the algorithm descends through the sub-tree represented by the closest element at each level until reaches a ground level cell. This cell is the first cell in the QP. Then, the algorithm applies a backtracking strategy by going back to its parent cell at level $l = 1$ and finding the second closest element of that cell. The ground level cell represented by this element is appended to the QP, i.e. it is the second cell of the QP. The algorithm goes back again to its parent cell and looks for the following closest element. Once all the elements from this cell have been considered (and their child cells have been appended to the QP), the algorithm goes back to its parent cell at level $l = 2$ and descends through the branch represented by the second closest element (the closest element from that cell was already tracked through the first descent). The algorithm is iteratively applied until it reaches the top level cell and finds that all its elements have been considered, i.e. there are no new branches to descend through. In other words, all ground level cells have been already appended to the QP.

Following the same example of Figure 2, if O is now a query element, the algorithm first descends through O_N^1 since it is the closest element of the top level cell ($l = 2$ in this example). Then, the elements belonging to cell C_1 at level $l = 1$ are sorted according to their similarity to the query element. Among these elements, D is the closest one. Therefore, the ground level cell represented by D is the first QP cell. Then, the algorithm visits the second closest element, i.e. O_N^1, and appends the ground level cell represented by this element. The following cells being appended are the ground level cells represented by F, E, and G respectively. Once all elements from C_1 have been considered, the algorithm goes back to its parent cell, i.e. the top level cell for the example being considered, and finds the second closest element, i.e. O_N^2. Therefore, the algorithm reaches cell C_2 by descending though this routing element. As before, its elements are also sorted and the ground level cells are appended to the QP according to the similarity order of their representative elements. Thus, ground level cells represented by elements C, A, O_N^2, and B respectively are pushed back. Finally, the algorithm goes back to the top level cell (all elements of C_2 at level $l = 1$ have been already considered) and visits the last (and furthest) element, i.e. O_N^3. Therefore, cell C_3 is visited and its elements are also sorted according to their similarity to the query element. The ground level cells represented by H, I, O_N^3, and J respectively form the tail of the QP. This means that their elements are compared with the query element in the last sub-queries performed by the Progressive Query. The reader is addressed to [KG07] for more details.

4 Contributions to HCT

In this section, based on the previous description, we present some weaknesses that we have detected in the original formulation of the HCT and propose strategies to overcome these limitations. First, we show that the use of the original definition of the covering radius may deteriorate the tree structure. Therefore, we propose a new definition which guarantees the right construction of the HCT. Due to the high computational cost of the new covering radius, we use a recursive algorithm which gives an overestimated value and a method for updating it to its actual value. Then, we propose a retrieval scheme based on the Preemptive Cell Search algorithm in order to improve the original retrieval system based on the MS-Nucleus technique.

4.1 A broader covering radius

The covering radius r of a cell C was defined in [KG07] as the distance from the nucleus to the furthest element in the cell:

$$r = \max_i (d(O_i, O_N)) \tag{2}$$

where O_N refers to the nucleus, O_i to the other cell elements, and $d(x, y)$ to the dissimilarity measure between two elements x and y. However, we consider that the covering radius should include all the elements of the database belonging to the subtree which has cell C as root, and not only the elements of that cell. According to this interpretation, the use of the covering radius defined in [KG07] in the Preemptive Cell Search algorithm for the HCT building can lead to wrongly discarded HCT branches. Consequently, the most suitable cell may not be found when a new element is inserted, resulting in a possible corruption of the HCT.

Figure 3 illustrates how the Preemptive Cell Search algorithm can fail in the process of finding the most suitable cell when the original definition of the covering radius is used. It represents a four-level tree in which the size of the points representing the elements is directly proportional to the level which belongs to and the non-nucleus ground level elements are not represented. The big dots labeled as O_N^1, O_N^2, and O_N^3 represent the elements belonging to the top level cell ($l = 3$), the medium dots (A, B, C, etc.) represents the elements from level $l = 2$ and the small dots the ones from level $l = 1$. Note that any element belonging to a level l also belongs to the lower levels ($l - 1$, $l - 2$,..., 1, 0). For the new element O being inserted, the ground level cell represented at level $l = 1$ by the element labeled as t is the most suitable cell, since t is the closest element among all elements belonging to that level ($l = 1$). However, cell C_3 is discarded because $d(O, O_N^3) - r(O_N^3) > d_{min}$,

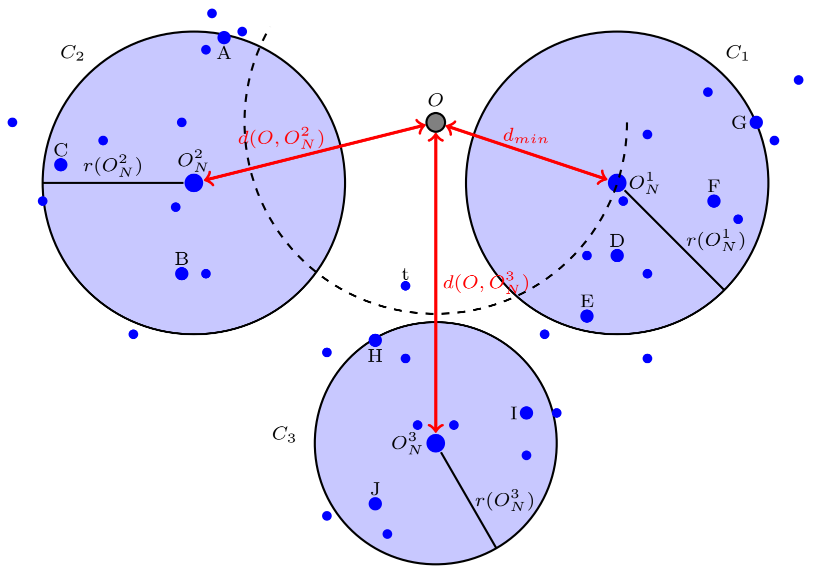

Fig. 3: Preemptive Cell Search fails with the original covering radius

where $d_{min} = d(O, O_N^1)$, and the algorithm does not descend for that HCT branch. As a result, this solution does not find the most suitable cell, i.e. the ground level cell which has t, H and O_N^3 as routing elements at the different levels. The decision of discarding C_3 is wrongly made as its covering radius $r(O_N^3)$ is computed only taking into account the distance from O_N^3 to H, I, and J, i.e. the elements belonging to cell C_3 at level $l = 2$.

As it will be shown in Section 5, HCT construction is the keystone to guarantee the retrieval accuracy of future query requests, so we propose the definition of a *broader* covering radius, which considers all the elements belonging to the subtree, not only the ones belonging to the cell:

$$r = \max_i (d(\hat{O}_i, O_N)) \tag{3}$$

where O_N refers to the nucleus, and $\hat{O}_i$ refers now to all the elements from the ground level which have O_N as a routing element at the level which the cell belongs to. The same example used before in Figure 3 is now illustrated with the broader covering radius in Figure 4. Now, the routing element O_N^3 satisfies $d(O, O_N^3) - r(O_N^3) < d_{min}$ and, therefore, cell C_3 is not discarded. Thus, the algorithm descends through all the branches to the lower level $l = 2$. Now, O is compared with all elements from that level where O_N^1 is still the closest element. The routing element H is not discarded either since it also satisfies $d(O, H) - r(H) < d(O, O_N^1)$. Notice that Figure 4 has been simplified by hiding the ground level elements that are not nucleus. As a consequence, the branches

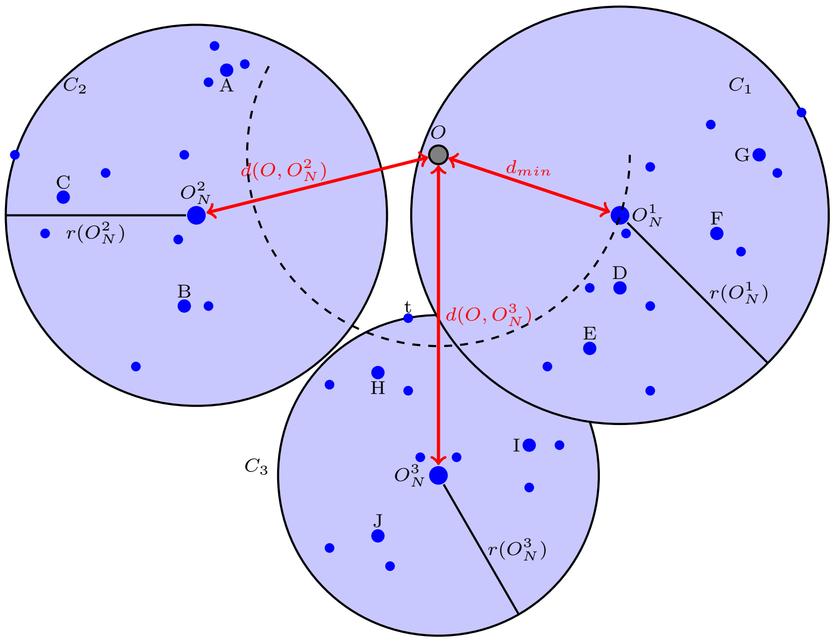

Fig. 4: Preemptive Cell Search with the new covering radius

which would be discarded at level $l = 2$ cannot be specified since the broader covering radius depends on the ground level elements. For instance, the routing element C has a high probability of being discarded since it is highly likely that element C does not satisfy $d(O, C) - r(C) < d(O, O_N^1)$. The Preemptive Cell Search algorithm descends through all non-discarded branches to the level $l = 1$. All the elements from this level belonging to the non-discarded branches are compared with the insertion element O again. Now, the ground level cell represented by the closest element, i.e. the element t, is chosen as the insertion cell. Thus, Preemptive Cell Search algorithm reaches the most suitable cell, i.e. the cell which has the closest nucleus to the insertion element. According to this new definition (see Equation 3), the covering radius values obtained by Equation 2 may be underestimated as O_i in this case refers only to the elements within the cell at the same level.

However, the computation of the exact broader covering radius (Equation 3) has, in general, a high computational cost because whenever a new element is inserted, the similarity measure may have to be computed many times. The worst-case scenario happens when the insertion produces such alterations in the tree structure that it is necessary to recompute all the distances from each cell nucleus to every database element belonging to its subtree for each cell and for each level. Therefore, we use an approximation of the broader covering radius, given by the following expression:

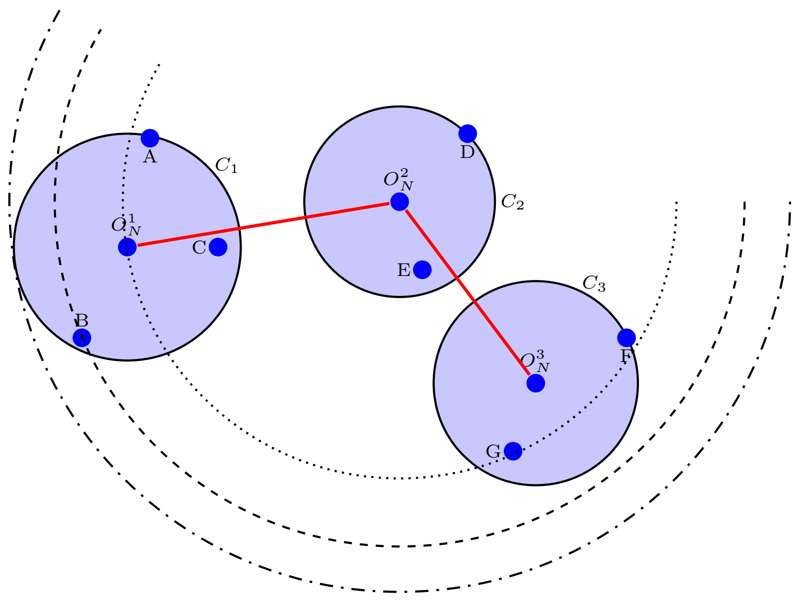

Fig. 5: Original covering radius (inner dotted line), exact broader covering radius (middle dashed line) and overestimated broader covering radius (outer dash dotted line) of the cell of the upper level which hosts the cells C_1, C_2 and C_3 from the lower level.

$$r = \max(r(S_N), \max_i(d(O_i, O_N) + r(S_i))) \tag{4}$$

where $r(S_N)$ refers to the covering radius of the child cell represented by the nucleus element O_N of cell C, O_i refers to the other elements of cell C, $d(O_i, O_N)$ is the distance between each cell element O_i and the cell nucleus O_N, and $r(S_i)$ refers to the covering radius of the child cell represented by O_i. Equation 4 gives an overestimation of the broader covering radius value and was already used in [CPRZ97] for M-trees. It allows keeping the covering radius updated after new insertions with a low computational cost thanks to its recursivity. Each covering radius only depends on the cell elements and on the covering radius of the cells associated with each element. For the cells belonging to the ground level, the covering radius is exactly computed, instead of using the approximation given by Equation 4. Figure 5 illustrates these three posible values (the original, the exact broader and the overestimated broader) of the covering radius in a simple case. It is a two-level HCT in which elements O_N^1, O_N^2 and O_N^3 belong to level $l = 1$ whereas the other ones are the non-nucleus ground level elements.

With the proposed approximation of the covering radius we can guarantee that no HCT branch will be wrongly discarded since we are using an overestimation of the broader covering radius. Therefore, the most suitable cell will always be found whenever a new element is inserted. As we will see in Section 5, a more precise construction of the HCT results in a better performance of the image retrieval system over the HCT. On the other hand, the cost of using this overestimation is that, in general, the number of cells analyzed is larger than the number of cells analyzed when the exact covering radius is used. This implies an increase in the time required by the cell search algorithm used in the insertion operations. This trade-off will be analyzed in Section 5.1.

4.2 Update of the covering radius

As pointed before, as a result of using the proposed overestimation of the covering radius (Equation 4), the number of cells visited through the HCT is larger than the number of cells visited when the exact covering radius is used. In order to reduce the time required by the Preemptive Cell Search algorithm, we propose an updating step that computes the exact value of the covering radius for all the cells of the HCT according to Equation 3. Therefore, the covering radius of each cell C_i is updated by computing the distance between its nucleus O_N to all the elements from the ground level belonging to the sub-tree having C_i as root, and then taking the maximum value. This updating step can be performed either once the HCT construction is completed or periodically during the construction. However, note that too many update operations may imply an increase in the HCT building time. There exists a trade-off between the time required by the updating step and the gain in time for the search operations in the following insertions. This trade-off will be analyzed in Section 5.2.

The objective of the proposed updating step is not only to reduce the insertion time of the elements to be indexed in the future, but also the retrieval times required by the new retrieval scheme proposed in Section 4.3. After the update, the number of visited cells for insertion and retrieval operations is granted to be less or equal to the number of visited cells when the overestimation is used. As a consequence, the number of comparisons between the inserting element and an element from the database is reduced, and the retrieval time is also reduced. Once all covering radius values have been updated, the computation of their values for each new insertion is performed again according to the over-estimation (see Equation 4). The impact of the updating step on the retrieval accuracy and on the retrieval time will be also analyzed in Section 5.1.

4.3 Adaptation of Preemptive Cell Search algorithm for Retrieval

One of the objectives of the HCT is to perform the K Nearest Neighbor (KNN) operation efficiently. KNN consists in retrieving the K most similar elements from the database to the given image query. This section focuses on approximate nearest neighbors algorithms, which aim at retrieving the best candidates in the minimum possible time. There exists a trade-off between the retrieval accuracy and the retrieval time in any approximate searching technique. Kiranyaz and Gabbouj [KG07] proposed to build the Query Path (QP), which is used by the Progressive Query scheme, based on the MS-Nucleus technique as explained in Section 3. However, as it will be shown in Section 5.1, the trade-off between the retrieval accuracy and the retrieval time of this scheme can be improved using a new approach. The limitation of the MS-Nucleus-based Progressive Query is that most of the K most similar elements do not belong to the initial part of the QP. As a consequence, achieving a high retrieval accuracy requires many sub-query operations which result in a high searching time. On the other hand, if considering a low retrieval time, the sub-query operations only consider the initial part of the QP, which result in a low retrieval accuracy.

We propose to perform the retrieval operation with the Preemptive Cell Search, i.e. the algorithm used for the insertion of new elements in the HCT. The reason is that this algorithm guarantees that the most suitable cell is always found. However, this is satisfied only if the covering radius (Equation 3) is not underestimated. Therefore, the proposed overestimation of the covering radius (Equation 4) guarantees the correct construction of the HCT, which becomes essential for improving the retrieval accuracy. Furthermore, the proposed update method of the covering radius is also useful to reduce the retrieval time with respect to the overestimated values.

In fact, we consider two kinds of retrieval operation: cell-based and non-cell-based. The former consists in returning only one cell of the tree structure. Thus, the proposed searching technique based on the Preemptive Cell solves this problem since the ground cell whose nucleus is the nearest one to the query image is always found. Futhermore, the cell-based retrieval operation allows the user to navigate through the tree structure from the returned cell to examine the neighbour cells (see Section 6.2).

The second type of retrieval operation, the non-cell based, is more focused on the KNN operation and does not keep the cellular structure. The most basic option for the KNN scenario would be returning the elements which belong to the retrieved cell sorted in ascending order according to the similarity measure. However, this option has two basic drawbacks: (*i*) the retrieved cell may not provide the user with K elements, and (*ii*) the retrieval of the cell with the nearest nucleus does not ensure that this cell will contain the nearest elements in the database. The former is solved by modifying the searching technique to retrieve a set of cells with enough elements instead of only one cell. Moreover, the probability of retrieving the most similar elements can also be increased by considering multiple cells. We propose that the number of cells N_C to be considered should depend on (*i*) the number of results (K) expected by the user, and (*ii*) a minimum threshold min_C:

$$N_C = \max(min_C, C_{2K}) \tag{5}$$

where C_{2K} is the number of cells that host at least twice the number of expected elements (K).

The adaptation of the Preemptive Cell Search algorithm to retrieve a set of cells arises naturally. The original algorithm descends through the non-discarded branches until level $l = 1$ is reached. Then, the remaining (non-discarded) elements from that level are sorted according to their similarity to the query element and the ground level cell represented by the closest element is selected. Therefore, we select the ground level cells represented by the N_C most similar elements among the non-discarded ones from level $l = 1$. Note that the Preemptive Cell Search algorithm only guarantees that the closest element from level $l = 1$ is found, but the other $N_C - 1$ elements considered may not be the following closest ones. In order to guarantee that the N_C closest elements from level $l = 1$ are found, Equation 1 should be replaced by:

$$d(O, O_i) - r(O_i) < d_{N_C} \tag{6}$$

where d_{N_C} is the distance to the N_C-th closest element instead of the closest one. However, Equation 6 has not been used in the experiments since much more branches should be analyzed, which would result in a high computational cost for retrieval.

The elements hosted by the retrieved N_C cells are sorted according to the distance from each element to the query element. The results given to the user, therefore, no longer keep the cellular structure. The motivation is that we consider the clustering as a method for speeding up the searching process. The most important thing from the user point of view in the KNN scenario is how good the retrieval system is, and not how well the elements have been clustered. The experimental results obtained by this proposed Preemptive Cell as a searching technique for the KNN operation will be presented in Section 5.1, where it will be also compared with the MS-Nucleus-based Progressive Query proposed in the original paper [KG07].

5 Experimental results

In this section, we evaluate the proposed contributions in order to show their usefulness. Results have been obtained using our own implementation of [KG07].

The use of an image database with an available ground truth (where elements to be retrieved are specified for a query set) would evaluate not only the proposed search algorithm used by the retrieval scheme, but also the performance of the visual descriptors and their visual similarity measures. The reason is that the ground truth may have been built from a semantic point of view or using visual criteria which may not be captured by the visual descriptors. Since the scope of this paper is not to evaluate the performance of the visual descriptors nor their similarity measures, we have decided to compare the performance of the searching techniques on the HCT index with respect to an exhaustive search. The basic exhaustive search consists in computing the dissimilarity measure between the element query and all the database items and sorting them in ascending order. This approach gives the optimal retrieval accuracy but the worst retrieval time. While in [KG07] the evaluation is only subjective (a group of people evaluates subjectively the query results), we propose an objective evaluation based on two aspects: (*i*) the retrieval time, which is simply the time required by the query retrieval, and (*ii*) the retrieval accuracy, which results from the comparison between the rankings obtained by each one of the searching techniques.

In order to compare the rankings obtained with the different HCT configurations (henceforth to be referred to as *approximate rankings*) with the ranking provided by the exhaustive search (henceforth to be referred to as *exact ranking*), we use three differents measures that compare two top k lists:

– *Mean Competitive Recall* ($\overline{CR}$) [CPR+07][SMR04][Ger07]. Let k be the number of images we want to retrieve and $A(k,q)$ the set of k images retrieved by the searching technique A for an image query q on our indexed image database. Let $GT(k,q)$ be the set of the k nearest neighbours to the image query q which has been obtained through an exhaustive search. Then, the competitive recall $CR(A,q,k)$ is defined as the number of elements belonging to the instersection of both sets, i.e. $CR(A,q,k) = |A(k,q) \cap GT(k,q)|$. Note that competitive recall is an integer number in the range $[0,...,k]$ and a higher value indicates higher quality. Then, $\overline{CR}$ is the average of the competitive recall over a set of queries (Q):

$$\overline{CR}(A,Q) = \frac{1}{|Q|}\sum_{q\in Q} CR(A,q,k) = \frac{1}{|Q|}\sum_{q\in Q} |A(k,q) \cap GT(k,q)| \qquad (7)$$

  This measure is the most intuitive one but it only analyzes the number of elements retrieved belonging to the exact top k list. Therefore, it does not take into account either the position in the exact ranking or the score (distance) of the retrieved elements.

– *Mean Normalized Aggregate Goodness* ($\overline{NAG}$) [SMR04][Ger07]. Let $W(k,q)$ be the sum of distances of the k farthest elements in the image database from the image query q. Then, NAG is defined as:

$$NAG(k,q,A) = \frac{W(k,q) - \sum_{p\in A(k,q)} d(p,q)}{W(k,q) - \sum_{p\in GT(k,q)} d(p,q)} \qquad (8)$$

where, as in the Mean Competitive Recall, $A(k,q)$ is the set of k images retrieved by the searching technique A for an image query q and $GT(k,q)$ is the set of the k nearest neighbours to the image query q obtained after an exhaustive search. The term $d(p,q)$ refers to the distance between the image query q and each element p from either the set $A(k,q)$ or the set $GT(k,q)$. Note that the NAG is a real number in the range [0,1] and a higher value indicates higher quality. The NAG is only 1 when the k nearest elements are retrieved and is only 0 when the k farthest elements are retrieved. Thus, the Aggregate Goodness is normalized with respect to the worst possible result. Then, the Mean Normalized Aggregate Goodness results from the average of the NAG over a set of queries.

Intuitively, NAG is a measure that evaluates how good the retrieved elements are with respect to the k most similar ones. Thus, this measure models a user who does not mind if the results of the retrieval system are exactly the best ones as long as they are almost as good as the best ones in terms of similarity values.

– *Kendall distance* [FKS03]. This distance is a variation of the standard Kendall's tau metric between permutations [Ken70]. Kendall's tau turns out to be equal to the number of exchanges needed in a bubble sort to convert one permutation to another given one. This distance is modified in [FKS03] to compare top k lists instead of permutations. The main difference to be considered is that the top k lists of two different procedures over the same database can have different elements, i.e. one element may appear in only one of the two lists.

  In constrast with the two previous measures, Kendall distance takes into account the position of the retrieved elements in the exact top k list. However, it does not consider the score obtained by the approximate ranking. Therefore, in a scenario where the retrieved elements rank from $(k+1)$th to $2k$th position in the exact ranking, the penalization will be the same as when the farthest elements are retrieved, i.e. the penalization is maximum. The optimal value for kendall distance is 0, which is the result of comparing two identical top k lists.

In addition to the retrieval time and the measures used for the retrieval accuracy, we have also computed the percentage of success in retrieving the query image, i.e. how often the query image itself can be found among the results of the search.

The HCT has been built over a dataset of 216,317 images. These images are keyframes extracted from professional broadcasted video provided by the *Corporació Catalana de Mitjans Audiovisuals* (CCMA) and *Televisió de Catalunya* (TVC). The content of these videos is generic since we can find news programs, sport events such as soccer matches or Formula 1 races, cultural programs, political debates, etc. Figure 6 shows some images from the database.

In order to evaluate the retrieval system, we use a query set of 1082 images. They have been chosen by selecting an image out of every 200 images of the dataset. Then, for each query image, an exhaustive search was carried out to generate the ground truth. Next, we built two HCTs: (i) one according to the original definition of the covering radius (see Equation 2), henceforth to be referred to as *OriginalHCT*, and (ii) one using the approximation given by Equation 4 and applying the updating method once the HCT has been built (henceforth to be referred to as *NewHCT*). The approximate rankings were obtained applying the MS-Nucleus-based Progressive Query over the OriginalHCT and the proposed Preemptive Cell Search over the NewHCT.

5.1 Impact of the covering radius and the retrieval scheme on the search performance

In this section we present the results obtained by the proposed retrieval scheme, i.e. the Preemptive Cell Search, over the HCT built with the new definition of the covering radius and the updating method. This NewHCT is compared with the Progressive Query over the OriginalHCT, i.e. the original approach [KG07]. Table 1 shows the results in a 40-NN (Nearest Neighbor) scenario. The top part of the table evaluates the retrieval accuracy of the HCT using 4 differents measures. The first measure, named *Retrieved queries*, shows the percentage of success in retrieving the element query in the approximate ranking. The other 3 measures have been previously introduced in this section. At the bottom of the table, the mean and the variance of the retrieval time are given.

In order to compare the new retrieval scheme over the NewHCT with the Progressive Query over the OriginalHCT, we have performed many progressive sub-queries until either the retrieval time or the retrieval accuracy obtained is comparable to the respective value achieved by the NewHCT. Thus, the retrieval accuracy is compared when both approaches give the same retrieval time and viceversa, i.e. the retrieval time is compared when both approaches

Fig. 6: Sample images from CCMA database

give the same retrieval accuracy. The first column in Table 1 shows the results when the Preemptive Cell Search is used over the NewHCT. The search algorithm is applied as explained in Section 4.3, i.e. the number of ground level cells includes at least 80 elements ($2k$, where $k = 40$). These elements are sorted and the top 40 form the approximate ranking. The second column is the result of applying the Progressive Query over a Query Path which has been build using the MS-Nucleus technique until 15000 elements have been appended to it. For this Query Path length, the retrieval time is the same for both approaches. The third, fourth and fifth columns show the results obtained when each of the three different measures of the retrieval accuracy is the same for both approaches. As expected, the retrieval accuracy improves when the number of subqueries performed by the Progressive Query is increased. On the other hand, the greater the number of elements considered, the higher the retrieval time. This is because the number of comparisons is directly proportional to the number of elements considered.

	NewHCT	OriginalHCT			
Num. elements	80	15000	109000	127500	140000
Retrieved queries(%)	99.26	22.64	72.92	78.14	82.62
$\overline{CR}$	27.51	8.04	**27.51**	29.84	32.22
$\overline{NAG}$	0.997	0.973	0.994	0.995	**0.997**
Kendall	313.87	1198.55	389.84	**313.56**	228.41
Mean retrieval time(s)	0.83	**0.83**	5.91	7.30	7.87
Variance retrieval time(s)	0.1466	0.0043	0.0868	0.2965	0.1629

Table 1: Comparison between NewHCT and OriginalHCT for kNN operation with k = 40 results

From Table 1, when the retrieval time is identical for both approaches (1st and 2nd columns), the results given by the retrieval accuracy measures are significantly better for the NewHCT (Preemptive Cell Search over the HCT with the new definition of the covering radius and the updating method) than for the OriginalHCT (Progressive Query over the HCT with the original definition of the covering radius). Whereas the NewHCT approach retrieves 27.51 out of the 40 closest elements on average ($\overline{CR}$), the OriginalHCT only achieves to retrieve 8.04 out of them. We come to the same conclusion for Kendall distance (313.87 for NewHCT and 1198.55 for OriginalHCT) and for $\overline{NAG}$ (0.997 for NewHCT and 0.973 for OriginalHCT). Regarding $\overline{NAG}$, note that small variations in their values may mean great differences between the ranks due to the normalization with respect to the worst possible results. In addition, 99.26% of the query elements are retrieved by the NewHCT in contrast with the 22.64% achieved by the OriginalHCT. This means that the query element is among the first 15000 elements of the Query Path only in 22.64% of the 1082 query retrievals.

When the $\overline{CR}$ is the same for both approaches, i.e. 27.51 out of 40 closest elements are retrieved on average (see 1st and 3rd columns from Table 1), the retrieval time required by the NewHCT is significantly better (0.83 seconds) than the OriginalHCT retrieval time (5.91 seconds). Therefore, the Progressive Query based on the Most Similar Nucleus technique over the OriginalHCT requires more than 7 times the time required by the Preemptive Cell Search over the NewHCT to achieve the same retrieval accuracy (according to $\overline{CR}$). This $\overline{CR}$ is not achieved until 109000 elements are appended to the Query Path, which is approximately half the size of the database. The experiments have been carried out with single threading on a Intel Xeon X5450 @3GHz and 2GB RAM.

5.2 Periodical update of the covering radius during HCT construction

As presented in Section 4.2, the updating step computes the exact value of the covering radius for all the cells of the HCT according to Equation 3. In the previous section, the experiments have been carried out applying the updating method only when the HCT is built for the first time. This section analyzes the impact of using the updating method periodically during the HCT construction.

The periodical update of the covering radius does not have any influence in the retrieval time nor accuracy. This is because the elements are inserted in the same cell whether the covering radius presents an overestimated value or an exact one. Thus, the elements are hierarchically clustered in the same way. However, the periodical use of the updating method has an impact on the time required for the HCT construction. As pointed out in Section 4.2, using the exact value of the covering radius from Equation 3 can reduce the time required by the search algorithm during the insertion process. On the other hand, the computation of the exact covering radius results in an overcharge in comparison with the approximation proposed in Equation 4.

Figure 7 shows the relative gain for the HCT construction time for different updating periods in comparison to the non-update case. The improvement in the HCT construction time is also represented for different sizes of the database. The positive value of the gain indicates that the periodical update does indeed reduce the construction time. This means that the gain in time for the search operations of the following insertions compensates for the extra-time required by the updating method. In particular, there is a range for the period (approximately from 500 to 2500 elements) in which the gain is maximum. Using a value beyond the upper bound also improves the construction time, but the benefits on the searching time decreases as more elements are inserted.

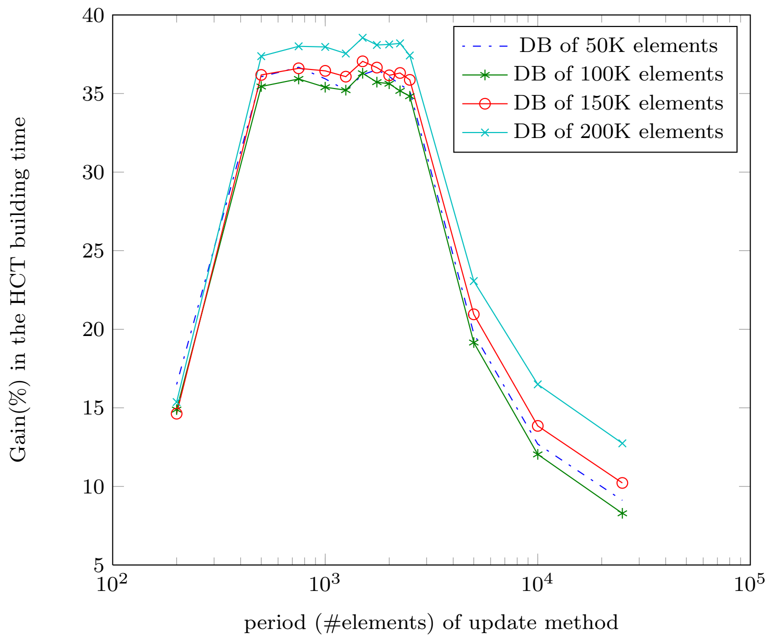

Fig. 7: HCT construction time comparison for update method of the covering radius

The decrease is due to the inefficient insertion of elements caused by the non-optimized covering radius. Analagously, using a period smaller than the lower bound also results in a decrease of the gain. The reason is that the overcost of updating the covering radius is not exploited enough by the search operations of the following insertions.

5.3 Analysis of the pre-fixed parameters

Two of the parameters used in the construction of the HCT are the *Maturity Size* and *Top Maturity Size* [KG07]. The Maturity Size M is the minimum number of elements that a non-top level cell must hold to be split. In the top level cell, the maturity size is referred to as Top Maturity Size. In [KG07], the Top Maturity Size value (24) is greater than the Maturity Size value (6) since the top level cell is understood as a “Table of Contents” of the database whilst giving a summary of the overall HCT body. These values have been adopted in the previous experiments, but they may not be the most appropiate ones if we use the HCT as a browsing tool. The reason is that cells split very frequently and the HCT results in a complex hierarchical structure. The next experiment evaluates the retrieval accuracy and the retrieval times when the Maturity Size is increased. Increasing the value of the Maturity Size, we expect to have a shallower hierarchical structure, i.e. a HCT with fewer levels, and to exploit better the whole available area of the GUI (see the browser application in Section 6.2). We set the same value for the Top Maturity Size since the Maturity Size value is not so small as before and, therefore, can also be considered appropriate for the size of a table of contents. Results are shown in Table 2.

Maturity Size	6	20	30	50
Top Maturity Size	24	20	30	50
$\overline{CR}$	27.51	23.24	21.18	19.32
$\overline{NAG}$	0.9967	0.9943	0.9924	0.9907
Kendall	313.87	472.89	559.34	635.27
Mean retrieval time(s)	0.8319	0.5609	0.5304	0.4967

Table 2: HCT pre-fixed parameters evaluation with Preemptive Cell Search, proposed overestimation of covering radius, update method of covering radius and 40 results

The retrieval accuracy worsens when the Maturity Size value increases whereas the retrieval time decreases. Thus, an increase in the Maturity Size value results in a lower complexity of the tree structure and a lower number of cells visited by the searching algorithm. As a result, the retrieval process becomes faster. In addition, as the Maturity Size is increased, the cells become less compact and the covering radius increases. This situation generates larger and more heterogeneous cells, which reduce the accuracy of the N_C retrieved cells. We can conclude that low maturity size values are more suitable for Query by Example operations (see Section 6.1) whereas higher values may be better for browsing applications (see Section 6.2 and Section 6.3).

5.4 Preliminary visual inspection

So far we have performed an objective evaluation of the HCT through a statistical analysis of some retrieval accuracy measures. Next, we illustrate the performance of our retrieval system through one query request example performed using a user interface, the GOS [GVPT$^+$10]. Figure 8 shows a comparison between the results obtained by using the Preemptive Cell Search technique over the HCT with respect to an exhaustive search over the image dataset. The query image is shown on the top-left. The whole set of images forms the exact ranking, i.e. the ranking obtained when an exhaustive search is performed. The images marked with a green rectangle have been retrieved by the Preemptive Cell Search algorithm, whereas the ones marked with a red

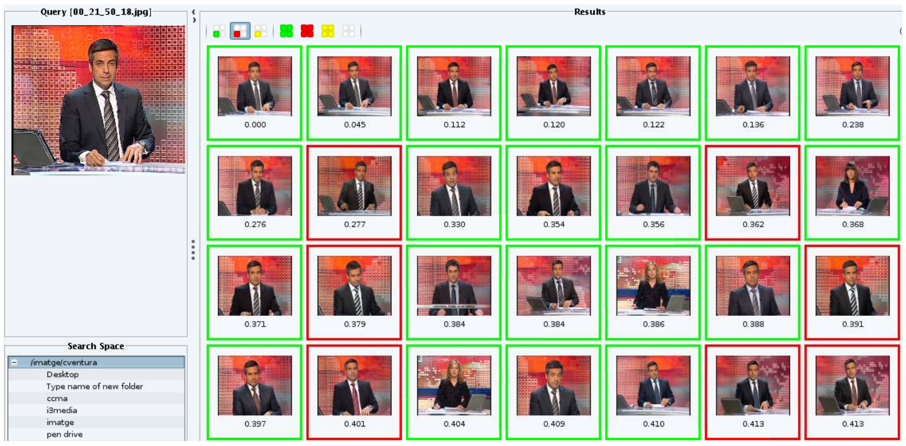

Fig. 8: Comparison between the results obtained by using the Preemptive Cell Search technique over the HCT with respect to an exhaustive search over the image dataset. The images marked with a red rectancle are the missing ones in the approximate ranking

rectangle are the missing ones in the approximate ranking. In particular, these results have been obtained using the HCT with *Maturity Size* = 6 and *Top Maturity Size* = 24, with the updated method and 40 results asked by the user.

6 Applications

As commented in Section 1, there are basically two ways of retrieving images from a dataset: (*i*) searching images similar to a given one (the Query by Example paradigm), and (*ii*) navigating through the dataset by using an image browser. In this section, we present three applications based on the HCT indexing scheme to solve these tasks: Query by Example, image browser and video browser.

6.1 Query by Example

The previous retrieval system, based on the MPEG-7 descriptors and the HCT indexing technique, has been used to develop the Query by Example (QbE) application required by the broadcaster company (Buscamedia project [Bus]). This system was integrated in a GUI (see Figure 8). Moreover, the QbE application has been adapted to a server/client architecture by using a messenger system. The HCT is loaded on a server waiting for the query requests which are launched for the several clients of the retrieval system.

6.2 Navigation through an image database

In addition to the efficient search capabilities, the hierarchical structure of the HCT has also been exploited to design an image browser, which has been integrated in a graphical user interface. An image browser is specially useful when the user does not have any example query image or when the user does not have any particular target content in mind and he/she may be just looking for interesting images. The top level cell can be thought as a “Table of Contents” which summarizes the content of the image database. The size of this summary only depends on the Top Maturity Size parameter, whereas the size of the cells belonging to the other levels depends not only on the Maturity Size parameter but also on the compactness of their elements. As commented in Section 5.3, the larger the Maturity Size parameter, the simpler the hierarchical structure of the HCT. On the other hand, the use of large values for the Maturity Size parameter results in heterogeneous cells, which may be unexpected for the user.

By means of a graphical user interface such as GOS [GVPT$^+$10], browsing through HCT is really user-friendly. First, the user visualizes the images belonging to the top level cell ($l = L$). Then, he/she can descend through any element to a cell of the lower level ($l = L - 1$) represented by the selected image. Note that the representative element for a cell corresponds to the element with the maximum number of connections in the cell’s Minimum Spanning Tree (MST), as in the original formulation of the HCT [KG07]. The user can go on navigating through the tree structure by selecting new elements and visiting lower levels. At any moment, the user can go back towards the top of the tree and descend through a different HCT branch. The hierarchical image organization depends on the configuration used during the HCT creation. Therefore, the elements are hierarchically clustered in function of the visual descriptor used to compare the dataset images. Note that the cells belonging to the ground level ($l = 0$) are more compact than the cells from the upper levels. Thus, the upper cells tend to be less homogeneous than the cells near the ground level. Figure 9 shows an example of navigation through an image database. The images shown at the top form the top level cell. Double clicking on the thumbnails, the system displays its child nodes. When the user reaches the ground level of the HCT, the black frames around the thumbnails indicate to the user that it is a leaf. The arrow icon located in the first position of the thumbnail grid allows the user to go back to the parent cell.

Furthermore, the image browser can also be exploited after a cell-based query operation. As commented in Section 4.3, this retrieval operation allows the user to navigate through the tree structure from the returned cell in order to examine the neighbour cells.

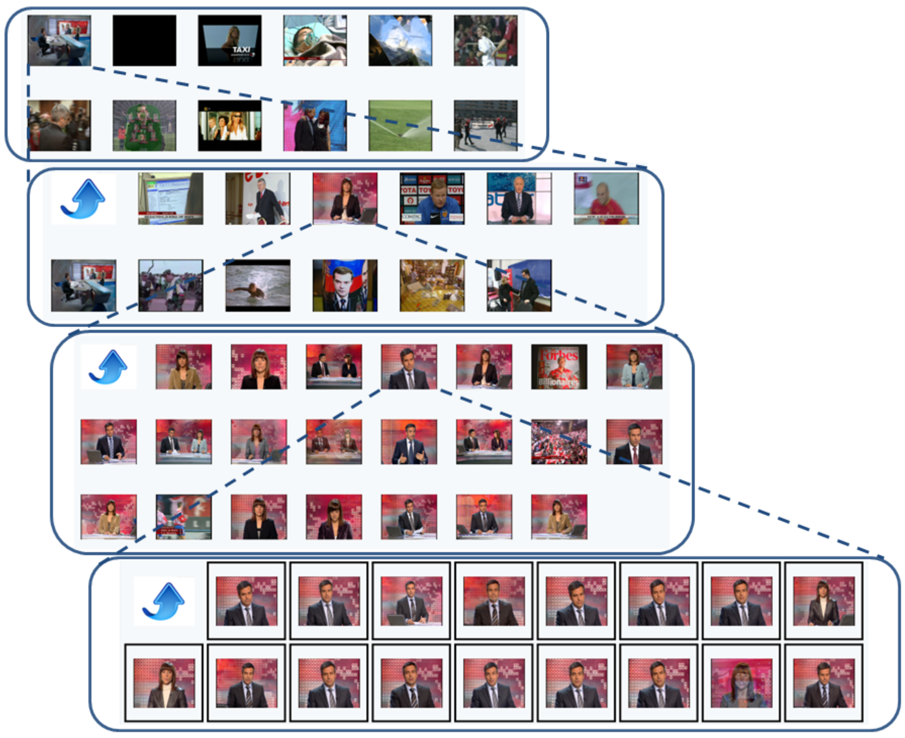

Fig. 9: Example of navigation through an image database

6.3 Video Browsing Tool

The third application based on HCT is a browser [VMG$^+$12] that supports two strategies for video browsing through keyframes: the navigation through visual hierarchies and the search for similar images. In particular, we have developed a video browsing tool which efficiently solves the following "Known Item Search" (KIS) task: finding a preselected segment of interest in a video file by interactive search, i.e. without any text-query. Note that, in this application, the preselected segment of interest, i.e. the query clip, is not available for the retrieval system, so the query clip cannot be processed. Therefore, the user has to find it by navigating through the video file to which the query clip belongs to.

With this purpose, the input video is firstly processed by a keyframe extractor in order to work with a lower number of frames by removing the high temporal redundancy of the video frames. Due to the hierarchical structure of the HCT, this redundancy removal step is not strictly necessary. Highly temporally redundant frames are expected to be clustered in the same cell. Therefore, the HCT computed over the images obtained by the keyframe extractor is expected to have a similar structure to the upper part of the HCT computed over all the video frames. However, the use of the keyframe extractor reduces the complexity of the HCT structure and the time required for its construction.

Once the keyframe extractor has been applied, the following MPEG-7 visual descriptors are extracted for each keyframe: (*i*) Color Structure, (*ii*) Dominant Color, (*iii*) Color Layout, and (*iv*) Texture Edge Histogram. Then, an HCT is built over each of the visual descriptors, considering as a similarity metric the visual distances recommended in MPEG-7.

Depending on the query clip, the user decides on which visual descriptor the navigation will start. The GUI shows to the user the thumbnails belonging to the root node of the HCT for each visual descriptor in a different tab. Since the output of the HCT indexing algorithm is a hierarchical clustering of the keyframes in the video, each tab can be understood as a summary of the video according to each visual descriptor. Furthermore, we can take advantage of the tree structure and go down the tree by double-clicking on an element to visualize which elements it represents in the same way as detailed in Section 6.2. In addition to this, the user can also take advantage of the indexed database to quickly perform a query-by-example operation when a keyframe globally similar to any of the frames in the segment of interest is found during the navigation.

This application was designed in order to participate in the Video Browser Showdown at the 18th International Conference on MultiMedia Modeling (MMM 2012), where we were awarded with the “Best Video Browser” certificate in the novice-run round, in which volunteers from the audience acted as searchers after a short training phase. This allowed us to test the usability of this video browsing tool. More details about this application can be found in [VMG$^+$12].

7 Conclusions

In this article, we have highlighted the flexibility provided by an indexing technique that works in a generic metric space. Some popular indexing solutions, such as the Locality Sensitive Hashing and the k-d tree, impose some restrictions in the use of certain similarity measures. That is the reason why we have chosen the Hierarchical Cellular Tree, a Metric Access Method (MAM) which allows indexing any data according to any similarity measure.

This paper has proposed some modifications to the original HCT [KG07]: (*i*) a new definition of the covering radius and an approximation that overestimates its value, (*ii*) the implementation of a method to update the covering radius to its actual value, and (*iii*) a new retrieval scheme based on the Preemptive Cell Search algorithm. These modifications have resulted in an improvement of the trade-off between the retrieval accuracy and the retrieval time. Some evaluation measures from the literature have been used to assess the retrieval schemes by comparing the rankings obtained by the searching techniques over the HCT (MS-Nucleus-based Progressive Query and Preemptive Cell Search) with respect to the ranking which results from an exhaustive search. On the one hand, for a same retrieval time (0.83 seconds), the retrieval accuracy achieved by the NewHCT is significantly better than for the OriginalHCT. Whereas the NewHCT approach retrieves 27.51 out of the 40 closest elements on average, the OriginalHCT only achieves to retrieve 8.04 out of them. On the other hand, for a same retrieval accuracy ($\overline{CR} = 27.51$) , the retrieval time required by the NewHCT approach is also significantly better (0.83 seconds) than the OriginalHCT retrieval time (5.91 seconds).

Finally, we have presented three applications based on the HCT. The use of the HCT shortens the retrieval times of our image retrieval system based on the MPEG-7 visual descriptors. Moreover, this CBIR system has also been integrated in a GUI, named GOS, over a client/server architecture. The HCT is loaded in memory in a server which runs the query requested by any remote client through the GOS. Furthermore, GOS can also exploit the HCT structure to explore the visual contents in the database. In addition, we have designed a video browsing tool which allows the user to find a preselected segment of interest in a video file by interactive search, i.e. without the need of any text-query.

Acknowledgment

This work was partially founded by the Catalan Broadcasting Corporation through the Spanish project CENIT-2009-1026 BuscaMedia, by TEC2010-18094 MuViPro project of the Spanish Government, and by FPU-2010 Research Fellowship Program of the Spanish Ministry of Education.

References

[AG11] Iftikhar Ahmad and Moncef Gabbouj, *A generic content-based image retrieval framework for mobile devices*, vol. 55, Multimedia Tools and Applications, 2011, pp. 423–442.

[AI08] Alexandr Andoni and Piotr Indyk, *Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions*, vol. 51, Communications of the ACM, 2008, p. 117.

[Ben75] Jon Louis Bentley, *Multidimensional binary search trees used for associative searching*, vol. 18, Communications of the ACM, September 1975, pp. 509–517.

[BM72] R. Bayer and E.M. McCreight, *Organization and maintenance of large ordered indexes*, Acta informatica **1** (1972), no. 3, 173–189.

[BSMS02] P. Salembier B. S. Manjunath and T. Sikora, *Introduction to mpeg-7, multimedia content description interface*, John Wiley and Sons, Ltd., Jun 2002.

[Bus] *CENIT Buscamedia Project*, `www.cenitbuscamedia.es`.

[CNBYM01] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín, *Searching in metric spaces*, vol. 33, ACM Computing Surveys (CSUR), September 2001, pp. 273–321.

[CPR+07] Flavio Chierichetti, Alessandro Panconesi, Prabhakar Raghavan, Mauro Sozio, Alessandro Tiberi, and Eli Upfal, *Finding near neighbors through cluster pruning*, Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (New York, NY, USA), PODS '07, ACM, 2007, pp. 103–112.

[CPRZ97] P. Ciaccia, M. Patella, F. Rabitti, and P. Zezula, *Indexing metric spaces with mtree*, Proc. Quinto convegno Nazionale SEBD, 1997, pp. 67–86.

[FKS03] Ronald Fagin, Ravi Kumar, and D. Sivakumar, *Comparing top k lists*, Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms (Philadelphia, PA, USA), SODA '03, Society for Industrial and Applied Mathematics, 2003, pp. 28–36.

[Ger07] Filippo Geraci, *Fast clustering for web information retrieval*, Ph.D. thesis, Universita degli Studio di Siena, Facoltá di Ingegnieria, Dipartaminto di Ingegnieria dell'Informazione, 2007.

[GVPT+10] X. Giró, C. Ventura, J. Pont-Tuset, S. Cortés, and F. Marqués, *System architecture of a web service for content-based image retrieval*, ACM International Conference On Image And Video Retrieval 2010, 2010, p. 358–365.

[IM98] Piotr Indyk and Rajeev Motwani, *Approximate nearest neighbors: towards removing the curse of dimensionality*, Proceedings of the thirtieth annual ACM symposium on Theory of computing (New York, NY, USA), STOC '98, ACM, 1998, pp. 604–613.

[Ken70] Maurice G. Kendall, *Rank correlation methods [by] maurice g. kendall*, 4th ed. ed., Griffin, London,, 1970 (English).

[KG05] S. Kiranyaz and M. Gabbouj, *Novel multimedia retrieval technique: progressive query (why wait?)*, vol. 152, Vision, Image and Signal Processing, IEEE Proceedings -, June 2005, pp. 356 – 366.

[KG07] ———, *Hierarchical cellular tree: An efficient indexing scheme for content-based retrieval on multimedia databases*, vol. 9, Multimedia, IEEE Transactions on, Jan. 2007, pp. 102 –119.

[KIP+11] Serkan Kiranyaz, Turker Ince, Jenni Pulkkinen, Moncef Gabbouj, Johanna Ärje, Salme Kärkkäinen, Ville Tirronen, Martti Juhola, Tuomas Turpeinen, and Kristian Meissner, *Classification and retrieval on macroinvertebrate image databases*, vol. 41, Computers in biology and medicine, 2011, pp. 463–472.

[LO07] Haibin Ling and Kazunori Okada, *An efficient earth mover's distance algorithm for robust histogram comparison*, vol. 29, Pattern Analysis and Machine Intelligence, IEEE Transactions on, May 2007, pp. 840–853.

[Low99] D.G. Lowe, *Object recognition from local scale-invariant features*, Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on, vol. 2, 1999, pp. 1150 –1157 vol.2.

[ML09] Marius Muja and David G. Lowe, *Fast approximate nearest neighbors with automatic algorithm configuration*, International Conference on Computer Vision Theory and Application VISSAPP'09), INSTICC Press, 2009, pp. 331–340.

[NBE+93] Wayne Niblack, Ron Barber, William Equitz, Myron Flickner, Eduardo H. Glasman, Dragutin Petkovic, Peter Yanker, Christos Faloutsos, and Gabriel Taubin, *The qbic project: Querying images by content, using color, texture, and shape*, Storage and Retrieval for Image and Video Databases, 1993, pp. 173–187.

[NKZ10] David Novak, Martin Kyselak, and Pavel Zezula, *On locality-sensitive indexing in generic metric spaces*, Proceedings of the Third International Conference on SImilarity Search and APplications (New York, NY, USA), SISAP '10, ACM, 2010, pp. 59–66.

[PC09] Marco Patella and Paolo Ciaccia, *Approximate similarity search: A multi-faceted problem*, vol. 7, Journal of Discrete Algorithms, March 2009, pp. 36–48.

[PW08] Ofir Pele and Michael Werman, *A linear time histogram metric for improved sift matching*, Proceedings of the 10th European Conference on Computer Vision: Part III (Berlin, Heidelberg), ECCV '08, Springer-Verlag, 2008, pp. 495–508.

[RTG00] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas, *The earth mover's distance as a metric for image retrieval*, vol. 40, International Journal of Computer Vision, November 2000, pp. 99–121.

[SMR04] Pavan Kumar C. Singitham, Mahathi S. Mahabhashyam, and Prabhakar Raghavan, *Efficiency-quality tradeoffs for vector score aggregation*, Proceedings of the Thirtieth international conference on Very large data bases - Volume 30, VLDB '04, VLDB Endowment, 2004, pp. 624–635.

[VMG$^+$12] Carles Ventura, Manel Martos, Xavier Giró, Verónica Vilaplana, and Ferran Marqués, *Hierarchical navigation and visual search for video keyframe retrieval*, Proceedings of the 18th international conference on Advances in Multimedia Modeling (Berlin, Heidelberg), MMM'12, Springer-Verlag, 2012, pp. 652–654.

[WSB98] Roger Weber, Hans-Jörg Schek, and Stephen Blott, *A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces*, Proceedings of the 24rd International Conference on Very Large Data Bases (San Francisco, CA, USA), VLDB '98, Morgan Kaufmann Publishers Inc., 1998, pp. 194–205.

[YD13] Ji Cheng Yang and Xiang Rong Ding, *Movie Audio Retrieval Based on HCT*, vol. 321, Applied Mechanics and Materials, 2013, pp. 1129–1132.

[YLhF$^+$11] Ji-Chen Yang, Yan-Xiong Li, Xiao hui Feng, Qian hua He, and Jun He, *Speaker retrieval based on minimum distance in HCT*, Wireless Mobile and Computing (CCWMC 2011), IET International Communication Conference on, 2011, pp. 274–277.