HIERARCHICAL CELLULAR TREE: AN EFFICIENT INDEXING METHOD FOR BROWSING AND NAVIGATION IN MULTIMEDIA DATABASES

Serkan Kiranyaz and Moncef Gabbouj

Institute of Signal Processing, Tampere University of Technology, Tampere, Finland
serkan@cs.tut.fi, moncef.gabbouj@tut.fi

ABSTRACT

One of the challenges in the development of content-based multimedia retrieval application is to achieve an efficient browsing and navigation scheme. Since browsing requires the capability of handling the entire database, a particular visualization system and tool(s) for navigation should be provided. Otherwise, browsing may turn out to be a disorienting process. Database items should be organized and especially for large databases the underlying organization scheme such as the indexing structure should provide a hierarchical representation of the database. This paper presents a novel browsing technique based on a new indexing scheme, the *Hierarchical Cellular Tree*, which is designed to bring an effective solution especially for indexing large-scale multimedia databases and furthermore to provide an enhanced browsing capability, which enables user to make a guided tour within the database. A pre-emptive cell search mechanism is introduced in order to prevent the corruption of large multimedia item collections due to the limited discrimination obtained from visual and aural descriptors. In addition, similar items are focused within appropriate cellular structures, which will be subject to mitosis operations when the dissimilarity emerges as a result of irrelevant item insertions. Mitosis operations ensure to keep the cells in a focused and compact form and yet the cells can grow into any dimension as long as compactness prevails. Experimental results show that the *HCT* indexing body can conveniently be used for efficient browsing and navigation operations among the multimedia database items.

1. INTRODUCTION

There are generally two ways to retrieve items from a (multimedia) database: through a query process such as query by example (QBE) and browsing. Query is nothing but a search and retrieve type process and is bound to some strictly defined rules and algorithmic steps. It is a retrieval race against time, so to provide the "most relevant" results in the "earliest" possible time given an "example" query item. However, such a scheme, by its nature, might have limitations and drawbacks. The user may not have a definitive idea what he/she is looking for and even though he/she has a clear idea about the query item, finding a relevant example is not an easy task to accomplish. Therefore, the problem of locating one or more initial query examples can be addressed by some browsing scheme. Database browsing, on the other hand, is a loose process, which usually requires a continuous interaction and feedback from the user and therefore, it is a kind of free-navigation and exploration among the items of a database. Yet, browsing does not lack a purpose: it is to access a set of items in an efficient way even though the definition of the set may not be clear, or rather vague. So it is the browsing algorithm's responsibility to organize the database in such a way that the "unknown" parameters of any browsing action can be resolved as efficiently as possible. Since browsing requires the capability of handling the entire database, a particular visualization system and tool(s) for navigation should be provided. Otherwise, browsing can turn out to be a disorienting process, especially when the user is not guided within the database. For this reason it is essential to provide an organized (perhaps in a hierarchical way) map of the entire database along with the current status of the user (e.g. such as a "You are here!" sign) should be provided during the browsing process.

In order to assist browsing, items should be organized, and especially for large databases, the underlying organization scheme such as the indexing structure should provide a hierarchical representation of the database and a natural support for free-navigation between the "levels" of the hierarchy such as traversing in and out among the levels. In order to provide a more general approach to similarity indexing for multimedia databases, several static and dynamic Metric Access Methods (MAMs) are proposed such as mvp-tree [4], Geometric Near-Neighbor Access Tree (GNAT) [5]; whereas, the dynamic ones include M-tree [3] and its variants, M+-tree [1] and Slim –tree[2]. However, all of them have significant drawbacks and shortcomings for indexing large-scale multimedia databases and hence cannot provide an efficient browsing basis from the user's point of view. In order to provide an efficient indexing scheme for browsing and navigation, we develop a MAM-based, dynamic and self-organized indexing scheme, the *Hierarchical Cellular Tree* (*HCT*). As its name implies, *HCT* has a hierarchic structure, which is formed by one or more levels. Each level has one or more cells. The cell structure is nothing but an acronym of the node structure in e.g. an M-tree. The reason we call it differently is because each cell contains a tree structure, the Minimum Spanning Tree (MST), which refers to the database objects as its MST nodes. Among all indexing structures available, the M-tree shows the highest structural similarity to *HCT*, however there are several major differences in their design philosophies and objectives:

- M-tree is a generic MAM, designed to achieve a balanced tree with a low I/O cost in a large data set. HCT is on the other hand designed for indexing multimedia databases where the content variation is seldom balanced and it is especially optimised for compactness (focused cells).
- M-tree works over the nodes with a maximum (fixed size) capacity M. *HCT* on the other hand has no limit for the cell size as long as the cell keeps a definite "compactness" measure.
- The split (mitosis) policies and objectives are completely different in M-tree and *HCT*.
- M-tree insertion operation is based on sub-optimum cell search Most-Similar Nucleus (*MS-Nucleus*); whereas, *HCT* is designed to perform an optimum search, so called *Pre-emptive* cell search.

The rest of this paper is organized as follows: in Section 2 we introduce the generic *HCT* design. Section 3 is devoted for *HCT Browsing* implemented over MUVIS framework [6]. Section 4 reports some experimental results. Finally Section 5 concludes the paper.

2. *HCT* OVERVIEW

HCT is a dynamic, cell–based and hierarchically structured indexing method, which is purposefully designed for advanced browsing capabilities within large-scale multimedia databases. It is mainly a hierarchical clustering method where the items are partitioned depending on their relative distances and stored within cells on the basis of their similarity proximity. The similarity distance function implementation is a *black box* for the *HCT* and it is a self-organized tree, which is implemented by genetic programming principles. This basically means that the operations are not externally controlled, instead each operation such as item insertion, removal, mitosis, etc. are carried out according to some internal rules within a certain level and their outcomes may uncontrollably initiate some other operations on other levels.

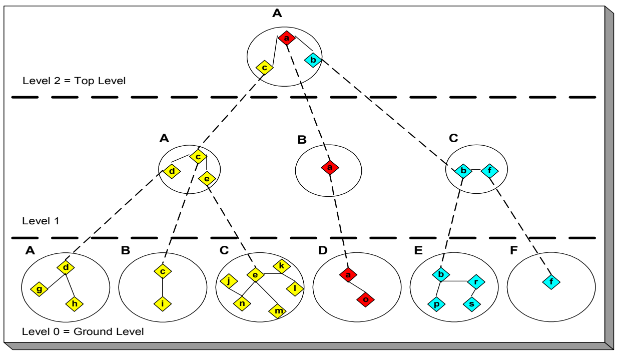


Figure 1: A Sample 3-level HCT body.

2.1 *Cell* Structure

A cell is the basic container structure, in which similar database items are stored. The ground level cells contain the entire database items. Each cell further carries a MST where the items are spanned via its nodes. This internal MST is used to keep the minimum (dis-) similarity distance of each individual item to the rest of the items in the cell. So this scheme resembles MVP-tree structure; instead of using some (pre-fixed) number of items, all of the cell items are now used as vantage points for any (other) cell item. In *HCT*, the cell size is kept flexible, which means there is no fixed cell size that cannot be exceeded. However there is a maturity constraint for the cells in order to prevent a mitosis operation before the cell reaches a certain level of maturity. Otherwise, we cannot obtain healthy information whether or not the cell is ready for mitosis since there is simply not enough statistical data that are gathered from the cell items and its MST. Therefore, using a similar argument for the organic cells, a maturity cell size (i.e. $N_M \geq 5$) is set for all the cells in *HCT* body (level independent).

Cell nucleus is the item, which represents the owner cell on the higher level(s). Since during the top-down cell search for an item insertion, these nucleus items are used to decide the cell into which the item should be inserted, it is essential to promote the best item for this representation at any instant. When there is only one item in the cell, it is obviously the nucleus item of that cell. Otherwise, the nucleus item is assigned by using the cell MST as the item having the maximum number of branches (connections to other items). This heuristics makes sense since it is the unique item to which majority of the items has the closest proximity to it (according to the MST optimality on the minimal branch weights). Contrary to static nucleus assignment of other MAM-based indexing schemes such as M-tree, the cell nucleus is dynamically verified and if necessary updated for *HCT* whenever an operation is performed over the cell in order to maintain the best representation of the (dynamically changing) cell and there is no computational cost for this so far since it can be extracted directly from the MST (branch) data.

Once a cell reaches maturity (a pre-requisite for the compactness feature calculation) then a regularization function, *f*, can be expressed using the following statistical cell parameters:

$$CF_C = f(\mu_C, \sigma_C, r_C, \max(w_C), N_C) \geq 0 \qquad (1)$$

where μ_C and σ_C are the mean and standard deviation of the MST branch weights, w_C, of the cell *C*. r_C is the covering radius, that is the distance from the nucleus to the farthest item in the cell and $N_C > N_M$ is the number of items in the cell *C*. A compact cell can be obtained if all these parameters can be minimized. Accordingly, a regularization function should then be implemented to minimize the compactness feature, CF_C. In the limit, the highest compactness can be achieved when $CF_C = 0$ which means that all the cell items are identical.

2.2 *Level* Structure

HCT body is hierarchically partitioned into one or more levels, a sample is shown in Figure 1. In this example there are three levels that are used to index 18 items. Apart from the top level, each level contains various numbers of cells that are created by mitosis operations which occurred on that level. The top level contains single cell and when this cell splits, a new top level is created above this level. As mentioned earlier, the nucleus item of each cell on a particular level is represented on the higher level. Each level is responsible for taking logs about the operations performed in it, such as number of mitosis operations, the statistics about the compactness feature of the cells, etc. Note that each level dynamically tries to maximize the compactness of their cells although this is not a straightforward process to do since incoming items may not show a similarity to the items present in the cells and therefore, such dissimilar item insertions may cause a temporary degradation in the overall (average) compactness of the level. So each level, while analysing the effects of the (recent) incoming items on the overall level compactness, should employ necessary management steps to provide a trend of improving compactness in due time (i.e. with the future insertions). Within a period of time (i.e. during a number of insertions or after some number of mitosis occurs), each level updates its compactness threshold according to the compactness feature statistics of the mature cells, into which an item is inserted. Therefore, $CThr_L$ value for a level *L* can be estimated as follows:

$$CThr_L = \frac{k_0}{P} \sum_{C \mid N_C > N_M}^{C \in S_P} CF_C = k_0 \mu_{CF_C} \quad \forall C \in S_P \qquad (2)$$

where S_P is the set of mature cells, upon which *P* insertions are recently performed and $0 < k_0 \leq 1$ is the compactness enhancement rate, which determines how much enhancement will be targeted for the next *P* insertions beginning from the moment of the latest $CThr_L$ setting. If $k_0 = 0$ then the cells will split each time they reach maturity and in this case *HCT* split policy will be identical to M-tree.

2.3 *HCT* Operations

Item insertion is a level-based operation and can be implemented per item basis. Let *nextItem* be the item to be inserted into a target level indicated by a number, *levelNo*. The item insertion algorithm, **Insert** can be expressed as follows:

```
Insert (nextItem, levelNo)
➢ Let top level number: topLevelNo and the single cell in top level: cell-T
➢ If(levelNo > topLevelNo) then do:
    o Create a new top level: level-T with number topLevelNo+1
    o Create a new cell in level-T: cell-T
    o Append nextItem into cell-T.
    o Return.
➢ Let the Owner (target) cell in level levelNo: cell-O
➢ If(levelNo = topLevelNo ) then do:
    o Assign cell-O = cell-T
➢ Else do:
    o Create a cell array for Pre-emptive cell search: ArrayCS[], put
      cell-T into it
    o Assign cell-O = PreEmptiveCellSearch (ArrayCS[], nextItem,
      topLevelNo)
➢ Append nextItem into cell-O.
➢ Check cell-O for Post-Processing:
    o If cell-O is split then do:
        ▪ Let item-O, item-N1 and item N2 be old nucleus item (parent) and new nucleus items (2 child)
        ▪ Remove( item-O, levelNo+1)
        ▪ Insert(item-N1, levelNo+1)
        ▪ Insert(item-N2, levelNo+1)
    o Else if nucleus item is changed within cell-O then do:
        ▪ Let item-O and item-N be old and new nucleus items.
        ▪ Remove( item-O, levelNo+1 )
        ▪ Insert( item-N, levelNo+1 )
➢ Return
```

Let *d*( ) be the similarity distance function, ***O*** the object to be inserted, O_N^i and $r(O_N^i)$ the nucleus object and its covering radius for the i^{th} cell, C_i, respectively. Let $d_{\min}$ be the distance to the closest nucleus item (in the upper level). The *pre-emptive* cell search rationale can be expressed as follows: fetch all nucleus items whose cells in the lower level may provide the closest object, $\Delta_i = d(O, O_N^i) - r(O_N^i) \le d_{\min} \quad \forall C_i$, among all nucleus objects that are in the owner cell *C*. Accordingly, the *Pre-emptive* cell search algorithm, **PreemptiveCellSearch,** can be expressed as follows:

```
PreemptiveCellSearch (ArrayCS[], nextItem, LevelNo)
➢ By searching $\forall O_N^i \mid O_N^i \in C_i \wedge \forall C_i \in ArrayCS$ find the most
  similar item, item-MS and $d_{\min}$.
➢ If (curLevelNo = levelNo + 1) then do:
    o Let the owner cell of item-MS: cell-MS in the (target) level
      (with level number: levelNo)
    o Return cell-MS
➢ Create an array for cell search: NewArrayCS[] = $\varnothing$
➢ For $\forall O_N^i \mid O_N^i \in C_i \wedge \forall C_i \in ArrayCS$, do:
    o If ( $\Delta_i = d(O, O_N^i) - r(O_N^i) \le d_{\min}$ ) then do:
        ▪ Find the owner cell of (nucleus) item $O_N^i$ in the lower
          level: $cell - C_N^i$
        ▪ Append $cell - C_N^i$ into NewArrayCS[]
➢ End loop.
➢ PreemptiveCellSearch(NewArrayCS[],nextItem,curLevelNo-1)
```

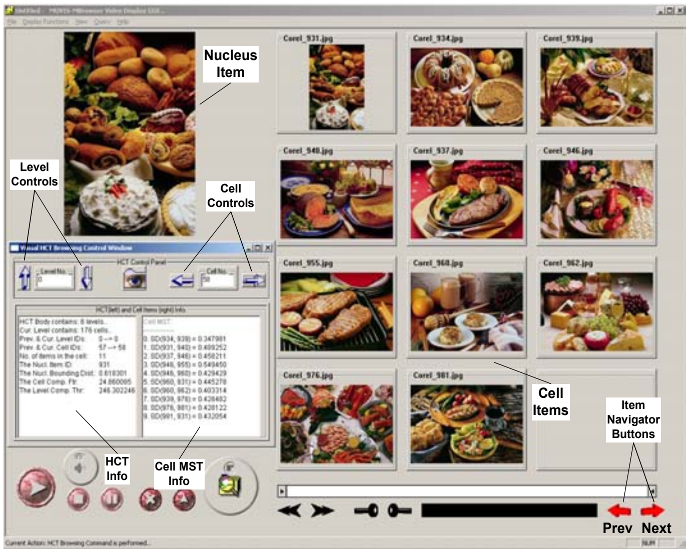
Due to space limitations, item(s) removal is not described in this article.

3. *HCT* BROWSING

The hierarchical structure of *HCT* can be used to give an overview to the user about what lies under the current level so that if well supported via a user friendly GUI, *HCT Browsing* can turn out to be a guided tour of the database items. When the user initiates it, it is designed to show the items in the cell at the top level, so it is the first clue about what the database content or more specifically a brief summary of it. The next logical step for the user is to choose an interesting item in this cell and starts the tour downwards. So this is the first functionality that *HCT Browsing* provides: to choose an item in the upper level and trace down to see the cell it represents (as a nucleus item of that cell). As long as the item chosen belongs to the current cell, the "level down" option will lead to the lower-level cell that is represented by that item. Otherwise, the first cell in the lower level will be shown by default. In the ground level the "level down" option is naturally disabled. The opposite functionality, the "level up" will yield the upper level cell, which is the owner (cell) of the nucleus item of the host cell at the current level and it works at all levels except the top level. This is also a useful functionality to browse similar cells of a particular cell. Therefore, the (slight) variations of a particular content can be visited using both of the level functionalities.

In addition to such inter-level navigation options, *HCT Browsing* provides inter-cellular trips within a certain level. The user can visit the cells sequentially, in a forward or backward direction and one cell at a time. This is especially useful when the user does not have any particular target content in mind and he/she may be just "looking" for interesting items. So in such an indefinite or "open-ended" exploration task, navigating through consecutive cells at a certain level will summarize the overall database content and the amount of summarization obviously depends on the "height" of the navigation performed, that is, simply the current level number.

Figure 2 shows a snapshot of the MUVIS application, *MBrowser*, where the *HCT Browsing* is implemented. Depending on the *HCT* indexing genre (i.e. visual or aural), the aforementioned functionalities of *HCT Browsing* are supported by means of a Control Window along with GUI of *MBrowser*. There are 1000 images (obtained from Corel image database) in this MUVIS database. Some color and texture features are extracted and *HCT* indexing is then performed. As shown in the bottom-left part of the figure, the Control Window allows user to perform inter-level and inter-cellular navigations and furthermore it gives some logs and crucial information to the expert users about the *HCT* body in general and particularly about the current cell and its MST. So any (expert) user can examine the cell compactness, which items are connected to each other within MST, the nucleus item and the most important of all whether or not the current cell is compact and mature. In this example by comparing the cell compactness feature with the current level compactness threshold value, ($CThr_L = 246.3 \, and \, CF_C = 24.86$ for this cell) it can be easily deducted that this is a highly focused cell. As a compact cell, its MST branch weights are quite low and within a close neighbourhood as expected. Another important information that can be gathered from the Control Window is the reliability and discrimination factor of the visual (or aural) features by inspecting the cell items' relevancy along with their (minimal) connections to the cell.



```
HCT Body contains: 8 levels..
Cur. Level contains: 176 cells..
Prev. & Cur. Level IDs:      0 --> 0
Prev. & Cur. Cell IDs:       57 --> 58
No. of items in the cell:    11
The Nucl. Item ID:           931
The Nucl. Bounding Dist.:    0.618301
The Cell Comp. Ftr:          24.860095
The Level Comp. Thr:         246.302246
```

```
Cell MST:
0. SD(934, 939) = 0.347981
1. SD(931, 940) = 0.409252
2. SD(937, 946) = 0.456211
3. SD(946, 955) = 0.549450
4. SD(946, 960) = 0.429429
5. SD(960, 931) = 0.445278
6. SD(960, 962) = 0.403314
7. SD(939, 976) = 0.426482
8. SD(976, 981) = 0.428122
9. SD(981, 931) = 0.432054
```

Current Action: HCT Browsing Command is performed...

Figure 2: HCT Browsing GUI on MUVIS-MBrowser Application.

4. EXPERIMENTAL RESULTS

In the experiments performed in this section, we used 4 different MUVIS image databases:

I. ***Corel_1K*** Image Database: There are 1000 medium resolution (384x256 pix) images from diverse contents such as wild life, city, buses, horses, mountains, beach, food, African natives, etc.
II. ***Corel_10K*** Image Database: There are 10000 low resolution images (in thumbnail size) from diverse contents such as wild life, city, buses, horses, mountains, beach, food, African natives, etc.
III. ***Shape*** Image Database: There are 1500 black and white (binary) images that mainly represent the shapes of different objects such as animals, cars, accessories, geometric objects, etc.
IV. ***Texture*** Image Database: There are 1760 texture images representing the pure textures from several materials and products.

Table I presents several statistics per fitness check status (before and after) and per cell search algorithm: the proposed *Pre-emptive* vs. traditional *MS-Nucleus*. The first two statistics, the percentage of mature cells and their overall item coverage are mainly chosen to show the effect of both algorithms on the maturity of the cells, especially the fitness check operation, which is basically meant for improving in both cases. The three other averaging statistics, compactness, nucleus distance and broken branch weight are about the "quality" of the indexing scheme, that is, how focused (compact) the cells are. For (*HCT*) indexing of these databases, the following regularization function is used:

$$CF_C = f(\mu_C, \sigma_C, r_C, \max(w_C), N_C) = K\mu_C\sigma_C r_C \max(w_C)\sqrt{N_C} \qquad (3)$$

where *K*, is a scaling coefficient. Once the indexing operation is completed, the average compactness is then calculated using CF_C value for all mature cells in the ground level.

We used $N_M = 6$ for maturity and *K*=1000 in the experiments

Table I: Statistics obtained from the ground level of HCT bodies extracted from the sample MUVIS databases.

Statistics (Level 0)	Cell Search Algorithm	Fitness Check	*Corel_1K*	*Corel_10K*	*Shape*	*Texture*
Mature Cell %	*MS-Nucleus*	Before FC	15.962	18.228	13.694	31.783
		After FC	19.324	34.654	50.877	44.444
	Pre-emptive	Before FC	16.393	13.937	12.760	28.114
		After FC	24.631	25.618	19.048	41.048
Item % in Mature Cells	*MS-Nucleus*	Before FC	39.700	47.865	41.143	64.091
		After FC	44.600	65.116	75.214	72.898
	Pre-emptive	Before FC	44.200	40.729	42.357	61.705
		After FC	54.400	56.465	52.571	70.000
Average Compact.	*MS-Nucleus*	Before FC	152.672	289.694	23.636	0.038
		After FC	145.581	384.193	157.738	0.048
	Pre-emptive	Before FC	82.440	65.020	12.097	0.011
		After FC	99.905	77.587	14.104	0.015
Average Covering Radius	*MS-Nucleus*	Before FC	1.049	1.206	0.580	0.127
		After FC	1.042	1.293	0.817	0.133
	Pre-emptive	Before FC	0.935	0.863	0.504	0.098
		After FC	0.986	0.925	0.539	0.107
Average Cell Size	*MS-Nucleus*	Before FC	4.695	4.619	4.459	6.822
		After FC	4.831	6.231	8.187	8.148
	Pre-emptive	Before FC	4.098	4.097	4.154	6.263
		After FC	4.926	5.321	5.128	7.686

An *HCT Browsing* example with inter-level navigations is shown in Figure 3. In this example the user starts the browsing from the 3rd level within 5 levels high *HCT* bodies and due to the space limitations, only a portion of *HCT* body (where the browsing operation is performed) is shown. Note that *HCT* indexing scheme provides more and more "narrowed" content in the descending order of the levels. For example, the user chooses an "outdoor, city, architecture" content in the third level where it yields "outdoor, city, beach and buses" content carrying cell in the second level. He/she then chooses a multi-colour "bus" and then navigating down into the first level, it yields a cell, which owns mostly "buses" with different colors as the content and finally choosing a "red bus" image (nucleus item) yields the cell of "red buses" in the ground level. Another example can be seen through: "outdoor, city, architecture" → "outdoor, city, beach and buses" → "beach and mountains" → "beaches. The cells are getting more and more compact (focused) in the descending order of level and the ground level cells achieves a "clean" clustering of the "interested" items as intended.

5. CONCLUSIONS

In this paper, we proposed *HCT* indexing structure designed for multimedia databases to achieve a dynamic, parameter independent and flexible cell (node) sized indexing structure, which is optimised to achieve as focused cells as possible using the visual and aural descriptors. Thus *HCT* indexing body can then be used as an efficient browsing and navigation infrastructure for multimedia databases.

The numerical results given in Table I approves that regardless of the cell search algorithm performed, the fitness check operation usually improves the amount (percentage) of mature cells and also the percentage of items stored in mature cells significantly without drastically degrading the overall compactness. Furthermore, *Pre-emptive* cell search achieves a major compactness improvement with respect to *MS-Nucleus* cell search algorithm, which induces corruption proportional with the database size. On the other hand, *Pre-emptive* cell search algorithm is not degraded from the increasing database size and therefore achieves a significant *scalability* performance in this aspect. Apart from the database size, the reliability (discriminating factor) of the feature(s) is also an important factor. With improved discrimination factors of the features, more robust similarity distance measures can be done and hence even more focused cells can be formed using *Pre-emptive* cell search algorithm.

Experimental results show that *HCT* indexing body can be used for efficient browsing and navigation among database items. The user is guided at each level by the nucleus items and several hierarchic levels of summarization help the user to have a "mental picture" about the entire database. It further achieves significant improvements on the cell compactness and shows no sign of corruption when the database size is getting larger. Hence the cells keep approximately the same or better level of compactness when the database size is increased significantly (i.e. 10 times). The analysis obtained from different databases suggests that *HCT* usually yields better clustering performance when the discrimination factor of the features is sharper and they provide better relevance for the user point of view.

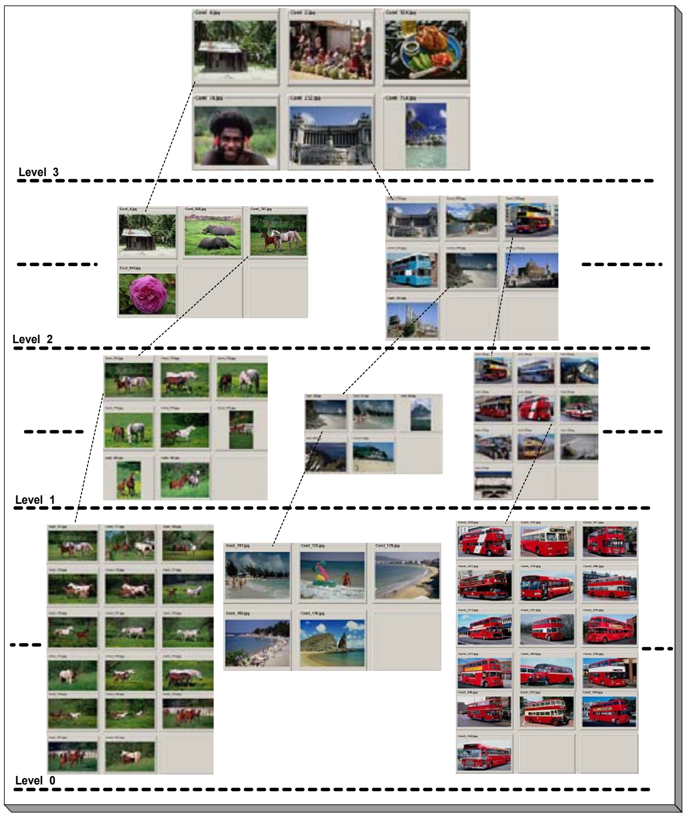


Figure 3: A typical *HCT* Browsing example, starting from the third level within Corel database. The user navigates among the levels shown with the lines through ground level.

REFERENCES

[1] X. Zhou , G. Wang , J. X. Yu , G. Yu, "M+-tree: a new dynamical multidimensional index for metric spaces", Proc. of the Fourteenth Australasian database conference on Database technologies 2003, pp.161-168, Adelaide, Australia, February 01, 2003.

[2] C. Traina Jr., A. J. M. Traina, B. Seeger, and C. Faloutsos, "Slim-trees: High performance metric trees minimizing overlap between nodes", In EDBT 2000, pp. 51-65, Konstanz, Germany, Mar. 2000.

[3] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: an efficient access method for similarity search in metric spaces", In International Conference on Very Large Databases (VLDB), pages 426-435, Athens, Greece, August 1997.

[4] T. Bozkaya, Z. M. Ozsoyoglu, "Distance-Based Indexing for High-Dimensional Metric Spaces", Proc. ACM-SIGMOD: pp.357-368, 1997.

[5] S. Brin, "Near Neighbor Search in Metric Spaces", VLDB, pp. 574-584, 1995.

[6] MUVIS. http://muvis.cs.tut.fi