Efficient and Flexible Cluster-and-Search for CBIR*

Anderson Rocha[1], Jurandy Almeida[1], Mario A. Nascimento[2]
Ricardo Torres[1], and Siome Goldenstein[1]

[1] Instituto de Computação, Universidade Estadual de Campinas – Brasil
{anderson.rocha, jurandy.almeida, rtorres, siome}@ic.unicamp.br

[2] Department of Computing Science, University of Alberta – Canada
mn@cs.ualberta.ca

Abstract. CBIR is a challenging problem both in terms of effectiveness and efficiency. In this paper, we present a flexible cluster-and-search approach that is able to reuse any previously proposed image descriptor as long as a suitable similarity function is provided. In the clustering step, the image data set is clustered using a hybrid divisive-agglomerative hierarchical clustering technique. The obtained clusters are organized in a tree that can be traversed efficiently using the similarity function associated with the chosen image descriptors. Our experiments have shown that we can improve search-time performance by a factor of 10 or more, at the cost of small loss, typically less than 15%, in effectiveness when compared to the state-of-the-art solutions.

1 Introduction

There is a permanent demand for automatic tools to store, organize, browse, and search collections of images, e.g., satellite imagery and commercial collections of stock photography to name but a few. Content-Based Image Retrieval (CBIR) systems aim at addressing these tasks. Arguably, the most challenging task within the domain of CBIR still remains being able to minimize search/retrieval time while keeping the effectiveness as high as possible.

In a typical CBIR system, images are processed and represented as metadata — often an n-dimensional feature vector. In this process, different image descriptors could be used (c.f., Sec. 3), depending on the image domain or the intended use of the system. Oftentimes, the user is interested in searching for images similar to a given query image. Thus, at query time CBIR is used to compare the metadata from the query image to the metadata of all images in the data set. The searching process can be as simplistic as comparing the query image to every other image in the data set, so-called a linear scan, or more sophisticated using clustering or indexing structures.

In this paper, we focus on improving the search time aspect of a CBIR system by employing a flexible cluster-and-search approach (c.f., Sec. 4). Towards this goal, the main contributions of this paper are twofold:

* The authors thank the financial support of Fapesp, CNPq, and Capes. Mario A. Nascimento's work has been partially supported by NSERC Canada.

1. We propose a *hybrid* clustering approach which alternates between agglomerative and divisive clustering paradigms.
2. We allow the use of *any* image descriptor to be used for the image metadata.

For the first point, we present a flexible ***D**ivisive-**A**gglomerative **H**ierarchical **C**lustering* (DAHC) technique. It combines features from both divisive and agglomerative clustering paradigms in order to yield good quality clusters, which are organized in a well-defined tree structure. At search time, only representative elements of clusters need to be compared to the query image, thus efficiently pruning large portions of the tree, diminishing the number of required comparisons, and therefore reducing query time substantially.

The second point makes our proposed framework very flexible. DAHC does not impose any restriction on the descriptor used. It only uses the set of elements to be analyzed, which are not the images themselves but rather their extracted features (metadata), along with a suitable similarity function.

Our experiments, using two real data sets and different image descriptors previously proposed in the literature, reveal that (1) our hybrid clustering technique is more efficient and more effective than using either a divisive-hierarchical-only or partitional-only clustering approach, and (2) it improves search time by up to two orders of magnitude while incurring in a loss of effectiveness typically below 10% when compared to using the same image descriptors without clustering and performing a full linear scan of the data set.

In the following, we present an overview of previous works and present and validate our proposed framework.

2 Related work

Data clustering algorithms can be partitional or hierarchical [6]. In this paper, we rely on the latter. Hierarchical clustering strategies can be divided into two basic paradigms: bottom-up agglomerative or top-down divisive. Agglomerative strategies begin with each element as a separate cluster and merge them into successively larger clusters. Divisive strategies start with one cluster and divide it into new clusters. In both strategies, the process is recursively repeated for each obtained cluster until some convergence criteria are reached.

The clustering paradigm can be used in different contexts such as unsupervised learning [2] and dimensionality reduction in micro-array description [9].

In the past few years, some researchers have presented clustering techniques for CBIR. Shyu et al. [10] have introduced a unified framework to facilitate conceptual database clustering for CBIR using Markov Model Mediators (MMMs). Antani et al. [1] have developed clustering techniques for hybrid text/image query-retrieval for medical images. Malik et al. [14] have proposed a technique to overcome problems of region growing algorithms such as seed point selection and processing order. In their approach, the pixels are consecutively merged in order to create representative clusters. In turn, Stehling et al. [11] proposed an adaptive agglomerative clustering algorithm to segment images into high-similarity regions.

Bhatia [3] has introduced a hierarchical clustering technique for image databases. In such approach, the stored models are hierarchically represented into the database instead of using a flat structure (nonhierarchical). However, this technique presents the undesirable requirement of changing the way the images are physically stored thus breaking up the logical and physical data independence in the database.

Kinoshenko et al. [7] have proposed a technique to partition the image into disjoint subsets. Their approach splits each query into representative subclasses and finds the most similar stored subclasses to each part of the query. Notwithstanding, the image classes need to represent a structural hierarchy, for instance the relationship present in images of a car and its parts.

The main difference among previous approaches and ours is that we combine both agglomerative and divisive strategies in order to obtain a hierarchical clustering structure, and that it does not depend on the image metadata but only on the pre-defined similarity function.

3 Image descriptors

As mentioned earlier, CBIR relies on representing the images by some ideally compact and possibly application-dependent metadata. The commonest metadata used in this domain is based on the color feature, more specifically, its distribution over the images. Next we review a few representative color-based image descriptors that have been used elsewhere and which we use to illustrate the flexibility of our framework.

3.1 Global Color Histogram (GCH) [13]

The simplest approach to encode the information present in an image is the Global Color Histogram (GCH). A GCH is an ordered set of values, one for each distinct color, representing the probability of a pixel being of that color. This approach uses uniform quantization and normalization to reduce the number of distinct colors and to avoid scaling bias. The L_1 (City-block) or L_2 (Euclidean) are the most used metrics for histogram comparison.

3.2 Color Coherence Vectors (CCVs) [8]

Zabih et al. have presented an approach to compare images based on color coherence vectors. They define color's coherence as the degree to which pixels of that color are members of large similarly-colored regions. They refer to these significant regions as coherent regions. Coherent pixels are part of some sizable contiguous region (connected components), while incoherent pixels are not.

3.3 Border/Interior Classification (BIC) [12]

Stehling et al. have presented the border/interior pixel classification (BIC). The approach relies on the RGB color-space uniformly quantized in $4 \times 4 \times 4 = 64$

colors. After the quantization, the image pixels are classified as *border* or *interior*. A pixel is classified as *interior* if its 4-neighbors (top, bottom, left, and right) have the same quantized color. Otherwise, it is classified as *border*. After the image pixels are classified, two color histograms are computed: one for border pixels and another for interior pixels.

4 DAHC

The *Divisive-Agglomerative Hierarchical Clustering* (DAHC) is a hybrid clustering technique that relies on the combination of features from both divisive and agglomerative clustering paradigms. This combination yields good-quality clustering solutions with fewer computational operations. By constructing a tree structure as a result of the clustering task, the method allows a substantial reduction in the query processing time. This is possible because instead of performing a linear scan of the data set, DAHC compares the query element only to representative elements of clusters. Hence, a large number of otherwise potential candidate answers need not be inspected at all.

Before we present the proposed technique, we introduce the notation used in the same in Table 1. Figure 1 shows an illustration of DAHC. At the beginning,

Table 1. Symbol definitions.

Symbol	**Meaning**
c, c_{rep}, c_{child}	A cluster; a representative element of the cluster c; and a reference to the lower level c in a hierarchy of clusters
C_i	Set of clusters at the i^{th} level
k	The number of clusters for each clustering task
$f \in [0, 1)$	The factor of re-clustering
E	Set of elements under analysis at i^{th} level
D	A metric measuring the dissimilarities among elements in E
z	The number of levels in a given hierarchy

we have E elements to be clustered. After the first iteration, we have the set of clusters C_1 at the level 1 of the hierarchy with k clusters $c_1 \dots c_k$. Note that *any* clustering technique that can be parameterized to produce a given number of clusters (k in this case) can be used in this step. Next, for each cluster c_i in C_1, we find the $\lfloor f \times k \rfloor$ closest clusters to c_i and combine them, creating a new set of clusters E_{c_i}. This is the agglomerative step of our approach. We assume that, by construction, each cluster has a representative image, thus, the notion of closeness between clusters reduces to closeness between their representative images. (The similarity function defined by a given image descriptor is used to yield the notion of closeness.) For each of these clusters E_{c_i}, the same steps above are repeated generating the next level in the hierarchy of clusters with nodes $c_{11} \dots c_{1k} \dots c_{k1} \dots c_{kk}$. We iterate this process while $|E_{c_i}| > k$. Note that k is an important parameter as it allows one to be more or less aggressive in the

agglomerative step, and it has a direct impact on the shape of the resulting tree. Algorithm 1 formalizes the DAHC approach.

Sometimes, depending on the data set, descriptor used, or even the technique to find the cluster representatives, a hierarchical tree can be non-balanced. The f and k factors together reduce the impact of a possible degenerate DAHC's tree. Greater values of k lead to lower and wider trees. Greater values of f lead to deeper trees given that there will be more elements to cluster providing more child nodes. In general, the DAHC height does not impact so much on the retrieval time. This is because for each level, only the representatives need to be compared to the query element. As the width is also not a problem given that only some branches are chosen during a retrieval, the possible DAHC's tree unbalancing is not problematic.

It should be clear now that DAHC employs both the divisive (in the first step) and agglomerative approaches (in the second step), hence the hybrid nature of our proposal. Furthermore, *any* similarity function supported by the chosen clustering technique can be used (CLUSTER(k,E,D) function), which leads to the flexibility of our proposal.

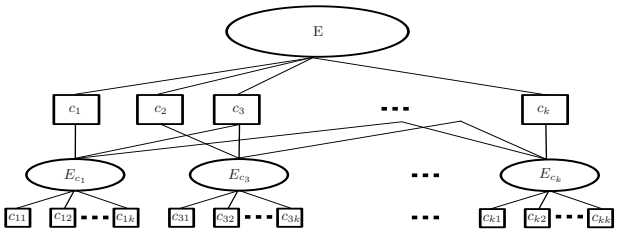


Fig. 1. A representation of DAHC.

The function CLUSTER(k, E, D) can use any partitional clustering method such as K-means or K-medoids [5]. The partitional algorithm is responsible for finding the representative elements (c_{rep}) within each level and at each cluster. In our implementation, we use the K-medoids as it is independent of the metric space. K-medoids only needs a dissimilarity matrix between elements while K-means requires an Euclidean-space dissimilarity metric. The metric-space independence make DAHC truly flexible, and potentially applicable to other domains besides CBIR.

Figure 2 illustrates a simple hierarchical structure obtained when using the DAHC technique on a small sample of images. The figure shows all nodes in the first level but the child nodes of only one single first-level node, namely c_5 due to space constraints.

Processing a query using the DAHC's structure is rather simple. Given a query image, we need to choose, at each level, the sub-tree that will be traversed. This is done by comparing each cluster representative to the query and choosing

Algorithm 1 DAHC

Require: The number of clusters k, the re-clustering factor $f \in [0, 1)$,the set of elements E, and a metric D;

```
1: function DAHC(k, f, E, D)
2:     C ← Cluster(k, E, D)                          ▷ Divisive step
3:     for each c ∈ C do
4:         C* ← ⌊f × k⌋ closest clusters of c ∈ C
5:         E* ← {}
6:         for each c* ∈ C* do                       ▷ Agglomerative step
7:             E* ← E* ∪ c
8:         end for each
9:         if cardinality(E*) > k then               ▷ Deepening
10:            c_child ← DAHC(k, f, E*, D)
11:        end if
12:    end for each
13: end function
```

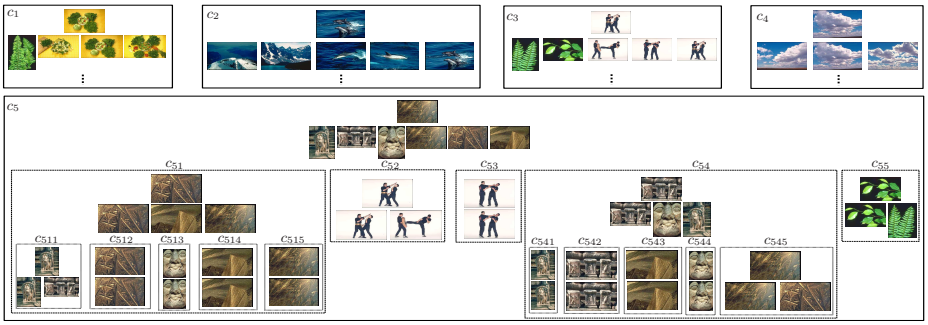


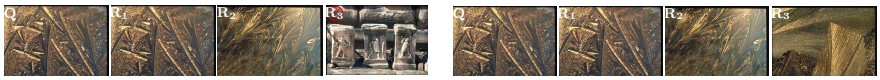
Fig. 2. DAHC sample for 24 images and 8 classes.

the one with highest level of similarity. (Recall that we assume that the clustering technique used in the function Cluster(k, E, D) will determine representative images for each cluster.) Note that again, our approach allows for flexibility as the choice of similarity function is orthogonal to the proposal itself. Once the lowest level (leaf) of the tree is reached, we sort the cluster representatives. Finally, we gather the elements in each cluster using the sorted representatives until we complete the minimum required number of elements in the retrieval.

To illustrate the method's effectiveness, we create a simple example with 24 images of 8 classes (3 images per class) using the Corel Photo Gallery. Figure 3(a) depicts a query image and its top-3 results using the BIC descriptor — the most effective one used in our experiments — and a linear scan. One can clearly see that the rightmost image in part (a) of the figure is not similar to the query.

Figure 2 shows the DAHC results for the images in our example. For instance, the wrong result R_3 for the query Q in Figure 3(a) is inserted in the cluster c_{51}. However, in the next level it is put in a separate cluster (C_{511}), whose representative is not similar to Q. Consequently that cluster is not investigated

and R_3 is not retrieved. It should be clear to see that using a similar reasoning many other sub-trees are not traversed, thus speeding up query processing time.



(a) (b)

Fig. 3. Top 3 results using (a) linear scan and (b) DAHC using the sample image set in Figure 2 and the letfmost image as the query.

We conclude by stating the following theorem, which leads to a corollary regarding the finiteness of the DAHC construction.

Theorem *For any given branch in the DAHC tree, its leaf clusters, at some point, will have less than M elements for any value of $M > 1$.*

Proof. We have to show that there is a finite value z such that the number of elements clustered at level z for a given branch, $|E_z|$, is less than M, where $M > 1$. For our purposes, this means that, at some level of the tree, there will be less than the M elements required to trigger a new clustering task within that branch.

The creation of new clusters and hence the deepening from a given branch is controlled by the factor of re-clustering f. Analyzing the hierarchy and given that $f < 1$, we have $|c_j| < |E_i|$, i.e., the size of each generated cluster is always less than $|E|$ and where $|\cdot|$ denotes the number of elements.

Furthermore, we see that $\sum_{j=1}^{k} |c_j| = |E_i|$, i.e., the sum of the number of elements of all clusters in a tree branch is always equal to $|E_i|$. Therefore, we generate the next level of the hierarchy for the cluster c_i selecting $\lfloor f \times k \rfloor$ closest clusters to the cluster c_j. The number of selected clusters for a branch in the next level, $i+1$, is always less than k given that $f \in [0, 1)$, we have $\lfloor f \times k \rfloor < k$. Thus

$$\sum_{j=1}^{\lfloor f \times k \rfloor} |c_j| < |E_i| \tag{1}$$

i.e., the sum of the elements for all selected clusters in the re-clustering stage for branch c_j is always less than the total number of elements to be clustered in this level E. However, the number of elements to be clustered in the next level branch is given by

$$|E_{i+1}| = \sum_{i=j}^{\lfloor f \times k \rfloor} |c_j|. \tag{2}$$

From Equations 1 and 2, we have that $|E_{i+1}| < |E_i|$. From this equation and the fact that at each new level $|E_{i+1}|$ is, at least, one element less than in the previous level, we have that there is a level z such that $|E_z| \leq M$ for any $M > 1$ which proves that the DAHC branch converges.

Corollary *DAHC method always converges in the number of clusters (width) and in the number of levels (depth).*

Proof. To prove DAHC convergence, we have to show that for every given cluster of DAHC tree, the number of elements to be clustered will eventually be less than the minimum required number of elements to trigger a new clustering task. Hence, we have two possibilities: (1) the width convergence and (2) the depth convergence.

DAHC width is controlled by the function $\textsc{Cluster}(k, E, D)$. As any partitional clustering technique can be used in this step, if such approach converges than DAHC converges in width. Partitional clustering techniques assume: (a) $k > 1$ (i.e., the clustering only makes sense if there are, at least, two clusters); (b) non-empty clusters; and (c) non-overlapping clusters (hard assignment)

They always converge to the solution either by stability or by a fixed number of iterations [5]. As a consequence, DAHC always converge in width.

The depth convergence is readily derived from Theorem 1. From that theorem, we have that for any given cluster in the DAHC tree, no matter its branches depth, it converges. As the number of clusters is limited for each level (width limit in f and k) it follows that the DAHC is finite, as we wanted to prove.

5 Experiments and Results

In this work, we have used the *query-by-example* (QBE) paradigm [4]. In QBE, we provide a query image to the system and we expect in return images that are similar to the given query image.

In our experiments, we use the set of image descriptors described in Section 3. We compare DAHC to (1) a linear scan of the data set with no clustering, (2) a partitional clustering technique (denoted by PC), and also (3) a divisive hierarchical clustering technique (denoted by DHC). Note that the last one also results in a tree that can be traversed at query time, similarly to what it is done within the DAHC. Furthermore, all these four solutions can be equipped with the similarity function provided along with the image descriptors, yielding 12 different combinations of image descriptors and cluster-and-search approaches.

We used two data sets. The first one comprises 1,624 images from Corel Photo Gallery[3] reported in [12]. This database contains 50 image categories and is referred to as the Corel Relevant sets (RRSets). The second set is from the FreeFoto collection[4]. It comprises 3,462 natural images divided into nine classes.

To assess the system effectiveness, we divide an image database into training and testing sets. We perform 5-fold cross-validation in the evaluation process. We repeat this process 10 times and provide average results. In the clustering tasks, we use K-medoids technique [5]. In the cross-validation, we use the training elements to build the tree and the testing elements to perform the retrievals.

We use the average *Precision* [4] metric to assess the retrieval effectiveness. *Precision* is the ratio of the number of relevant images retrieved to the total number of irrelevant and relevant images retrieved.

[3] `http://www.cs.ualberta.ca/~mn/BIC/queries.html`

[4] `http://www.liv.ic.unicamp.br/~undersun/pub/communications.html`

In the experiments, we have calculated the average *Precision* using the top 30 retrieved images. This represents the number of relevant images in the top 30 resulting images for each query. This value is an estimation of the number of retrieved images an user would accept to inspect in order to determine their relevance to his/her needs and it was previously reported in [12].

In this section, we present results for our method and provide comparisons to the state-of-the-art approaches. PC stands for *Partitional Clustering* (simple K-medoids), DHC stands for *Divisive Hierarchical Clustering* and DAHC-f is our *Divisive-Agglomerative Hierarchical Clustering* with factor of re-clustering f. In the experiments, we use values of k that are multiples of 5, and we report results using $f \in \{0.05, 0.1, 0.2\}$. High values of f lead to high overload in the offline creation of the hierarchical structure. We provide results for GCH, CCV, and BIC image descriptors. We show effectiveness loss and performance gain results with respect to the linear search over the entire data sets using each of the image descriptors presented earlier.

Tables 1(a) and 1(b) show the reference values for the average precision and average number of image comparisons to obtain the top 30 images for each query. Figures 4 to 6 show the effectiveness (small) loss and performance gain results for DAHC with respect to linear scan, partitional clustering, and divisive hierarchical clustering approaches.

(a)

	BIC	GCH	CCV
Corel RRSets	53.0%	41.8%	40.7%
FreeFoto	68.1%	55.3%	59.9%

(b)

Corel RRSets	*FreeFoto*
$\approx 21,000$	$\approx 48,000$

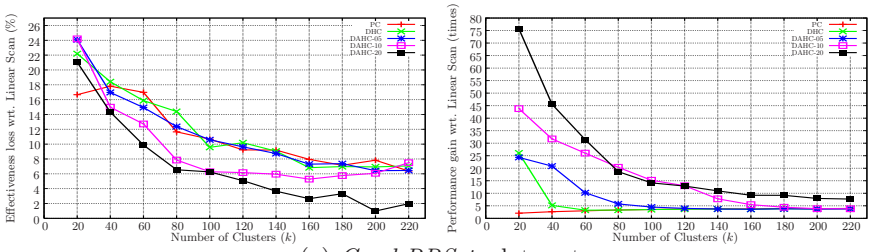
Table 2. (a) Average p_{top30} using linear scan for each image descriptor. (b) Number of image comparisons using linear scan (which is independent of the image descriptors)

Clearly, DAHC is more efficient and more effective than using either a divisive-hierarchical-only or partitional-only clustering approach, and (2) it improves search time by up to two orders of magnitude while incurring in small loss of effectiveness (typically 5-15%) regardless the database and the image descriptors when compared to the full linear scan of the data sets.

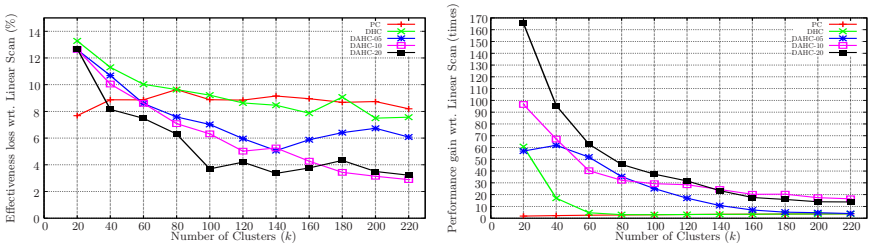
For instance, for the *Corel RRSets* database and using the BIC image descriptor (Figure 6), DAHC, with factor of re-clustering $f = 0.2$ and $k = 140$ clusters, yields about 2.4% of effectiveness loss with respect to the full linear scan of the database. However, this is a small loss when compared to its efficiency gain. For this same configuration, DAHC is 20 times faster than the full linear scan of the database and about 10 to 15 times faster than partitional clustering or divisive hierarchical clustering approaches.

We have found that there is a trade-off between f and k in order to produce good results (both in efficiency and effectiveness). On one hand, if we increase k, i.e., the number of clusters, we increase the number of required operations to perform a retrieval given that we have more representative elements to take

into account. On the other hand, if we increase the value of f, i.e., the factor of re-clustering, we smooth the re-clustering stages and, consequently, improve the overall effectiveness results. Nevertheless, if the value of f becomes higher, we increase the offline overload when creating the hierarchical structure. Throughout experiments, we have found that $f = 0.2$ is a good trade-off for offline efficiency and online effectiveness. The $f = 0.2$ value means that for each clustering task, we have a factor of re-clustering of 20% of the elements under consideration. In each stage, $70 < k < 140$ is a good choice for k.



(a) *Corel RRSets* data set.



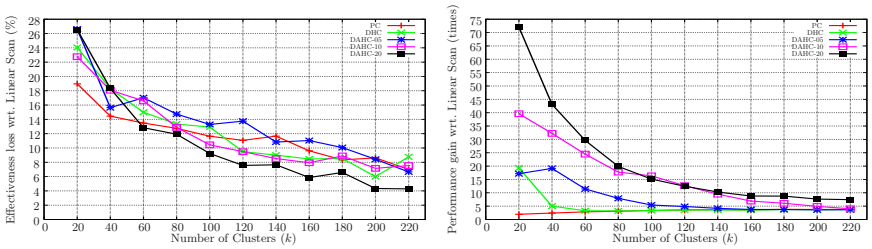
(b) *FreeFoto* data set.

Fig. 4. Effectiveness loss (left) and performance gain (right) for GCH descriptor with respect to linear scan.

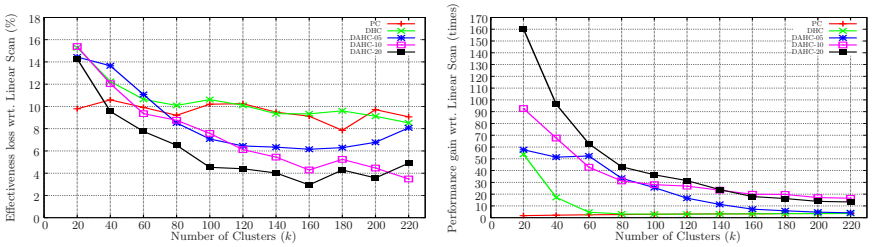
6 Conclusions

In this paper, we have presented a new flexible cluster-and-search approach for Content-Based Image Retrieval that is able to reuse any previously proposed image descriptor as long as a suitable similarity function is provided. For that, we have proposed a *Divisive-Agglomerative Hierarchical Clustering* approach (DAHC) that organizes the clusters in a tree that can be then traversed efficiently using the similarity function associated with the chosen image descriptors.

We have provided several experiments showing that our technique is suitable for CBIR and that it reduces the number of required operations to perform a retrieval and still provides good effectiveness when compared to partitional and divisive hierarchical clustering approaches and also to the full linear scan of



(a) *Corel RRSets* data set.



(b) *FreeFoto* data set.

Fig. 5. Effectiveness loss (left) and performance gain (right) for CCV descriptor with respect to linear scan.

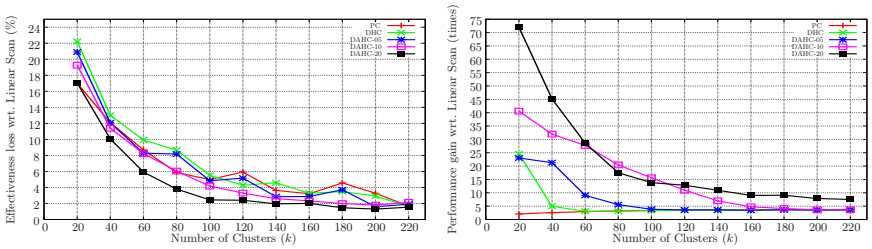
the data set. The effectiveness small losses are acceptable in practical situations given the orders of magnitude reduction in the number of required operations in each retrieval task.

DAHC relies on the choice of two factors: the number of clusters k and the re-clustering factor f. On one hand, if we increase k, we improve the effectiveness. However, high values of k lead to more required operations in order to perform a retrieval. On the other hand, high values of f improve the re-clustering stage and the online efficiency. Notwithstanding, the greater f the greater the overload in the offline creation of the hierarchical structure of the database. In addition, we have provided a formal proof of DAHC's convergence.

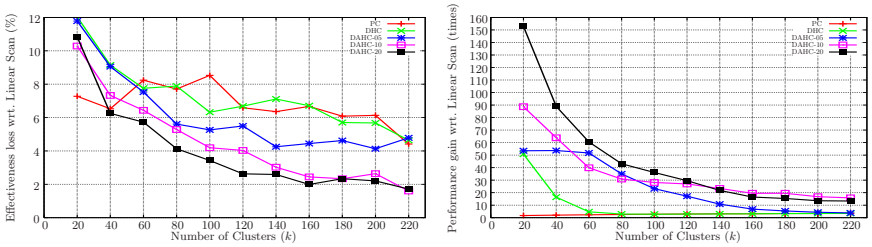
Finally, although unsubstantiated here, it is conceptually possible that DAHC can be used within other domains, such as textual information retrieval. Our future work include the application of our method for text retrieval and indexing using the state-of-the-art text descriptors in the literature. Furthermore, we intend to validate the method on a web-scale CBIR environment such as one containing several thousands of images.

References

1. S. Antani, R. Long, and G. Thoma. Content-based image retrieval for large biomedical image archives. In *MEDINFO*, 2004.
2. R. Baeza-Yates. *Clustering and Information Retrieval*. Kluwer, 1 edition, 2003.
3. S. Bhatia. Hierarchical clustering for image databases. In *Intl. Conference on Electro Information Technology*, pages 6–12, 2005.



(a) *Corel RRSets* data set.



(b) *FreeFoto* data set.

Fig. 6. Effectiveness loss (left) and performance gain (right) for BIC descriptor with respect to linear scan.

4. A. D. Bimbo. *Visual Information Retrieval*. Morgan Kaufmann, San Francisco, CA, USA, 1 edition, 1999.
5. C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 1 edition, 2006.
6. J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco, CA, USA, 1 edition, 2005.
7. D. Kinoshenko, V. Mashtalir, and E. Yegorova. *Machine Learning and Data Mining in Pattern Recognition*, chapter Hierarchical Partitions for Content Image Retrieval from Large-Scale Database. Springer, 2005.
8. G. Pass, R. Zabih, and J. Miller. Comparing images using color coherence vectors. In *ACMMM*, 1997.
9. J. Seo and B. Shneiderman. *Interactive Exploration of Multidimensional Microarray Data: Scatterplot Ordering, Gene Ontology Browser, and Profile Search*. Phd thesis, University of Maryland, College Park, 2003.
10. M.-L. Shyu, S.-C. Chen, M. Chen, and C. Zhang. A unified framework for image database clustering and CBIR. In *MMDBS*, pages 19–27, 2004.
11. R. Stehling, M. Nascimento, and A. Falcão. An adaptive and efficient clustering-based approach for CBIR in image databases. In *IDEAS*, pages 356–365, 2001.
12. R. Stehling, M. Nascimento, and A. Falcão. A compact and efficient image retrieval approach based on border/interior pixel classification. In *CIKM*, pages 102–109, 2002.
13. M. J. Swain and D. H. Ballard. Color indexing. *IJCV*, 7(1):11–32, 1991.
14. C. Thies, A. Malik, D. Keysers, M. Kohnen, B. Fischer, and T. Lehmann. Hierarchical feature clustering for CBIR in medical image databases. In *Medical Imaging*, pages 598–608, 2003.