The PIBE Personalizable Image Browsing Engine*

Ilaria Bartolini
DEIS - IEIIT-BO/CNR
University of Bologna
Viale Risorgimento, 2 - 40136
Bologna, Italy
ibartolini@deis.unibo.it

Paolo Ciaccia
DEIS - IEIIT-BO/CNR
University of Bologna
Viale Risorgimento, 2 - 40136
Bologna, Italy
pciaccia@deis.unibo.it

Marco Patella
DEIS - IEIIT-BO/CNR
University of Bologna
Viale Risorgimento, 2 - 40136
Bologna, Italy
mpatella@deis.unibo.it

ABSTRACT

In this paper we describe PIBE, a new *Personalizable Image Browsing Engine* that allows an effective visual exploration of large image collections combining computer vision and database techniques. The principal features of PIBE include the possibility of modifying the browsing structure by means of graphical *personalization actions* provided by the visual interface, and of persistently storing such customizations for subsequent browsing sections. The PIBE hierarchical browsing structure, called *Browsing Tree*, can be *locally* customized, thus avoiding global reorganizations, which are clearly unfeasible with large collections. Indeed, each node of the Browsing Tree has associated a cluster of images and a specific dissimilarity function. We present the basic principles of the PIBE engine, and report some experimental results showing the effectiveness and the efficiency of the browsing and personalization functionalities provided.

1. INTRODUCTION

Exploration of large image collections is a complex and often tedious task. This is a direct consequence of the well-known difficulty one encounters when trying to precisely characterize the actual content of images and to define suitable criteria for comparing such contents. Furthermore, even specifying what one is actually looking for in an image database (DB) is not a trivial problem [11].

In order to address this issue, researchers from computer vision community have proposed, over the years, a plethora of image retrieval systems that represent each image as a set of low-level features, such as color, texture, and shape, and typically provide the user with a *query-by-example* search modality to explore the DB contents. Accordingly, the system returns to the user those images that are more similar to the provided query image, and these result images can be in turn used as starting points for further searches. To improve the effectiveness of results, it is nowadays well recognized that the (dis)similarity criterion used to compare images needs to be dynamically adapted, so as to properly capture user preferences. In particular, *relevance feedback* techniques can be used to this purpose [3], even if it has to be understood that this comes at the price of a reduced search efficiency [2].

An effective interaction with large image DB's cannot be based only on querying, rather it has also to include advanced *browsing* facilities [11]. These are needed, say, to determine a "good" starting point for searching (e.g., a suitable query image might not be available at the beginning of a user session), to get an overall view of the DB contents (e.g., the user does not know yet what she is looking for), and so on.

Although several *browsing systems* have been presented in the literature (e.g., [1, 4, 5, 8, 11, 13]), they suffer some major problems that limit their applicability to large image DB's. In particular:

1. With some notable exception, most image browsers are based on a *static browsing structure*, that is, the organization of images is based on a fixed (usually hierarchical) layout that cannot be altered by the user. This means that no *personalization* at all is possible.

2. Browsing systems that allow the user to modify the browsing structure through interaction typically do so by reorganizing the *whole* DB or, at least, a large part of it. Clearly, this becomes unfeasible with very large image collections.

3. All the browsing systems for which personalization is an issue do not consider the possibility to make persistent the so-modified browsing structure. Thus, each time the user starts a new session she is faced with the original (default) image DB organization.

Motivated by such observations, in this paper we propose PIBE (*Personalizable Image Browsing Engine*), an adaptive image browsing engine aiming to overcome above limitations. In particular, PIBE provides the user with a customizable hierarchical browsing structure (called the *Browsing Tree*), whose changes persist across different sessions, and with a set of graphical *personalization actions* that can be used to modify the Browsing Tree. The effects of such actions are defined so as to guarantee that only a "local" reorganization of the image DB is required. This is possible since PIBE maintains specific similarity criteria for each portion (sub-tree) of the Browsing Tree.

*Part of this work was supported by the PANDA IST Thematic Network.

CVDB '04 Paris, France

Figure 1 provides an overall view of the PIBE architecture. In particular, PIBE builds *off-line* a default Browsing Tree (BT in the figure) over the image features. At run time the user interacts with the system by means of a graphical user interface (GUI), which, among others, can accept specific requests to modify the browsing structure. These are actually managed by the *Browsing Processor* component. Finally, the *Visualizer* component is in charge of displaying the content of the (possibly modified) Browsing Tree at several abstraction levels. In particular, the Visualizer implements spatial visualization techniques that are able to arrange currently browsed images on a 2-D display by respecting their mutual dissimilarities (see Figure 3).

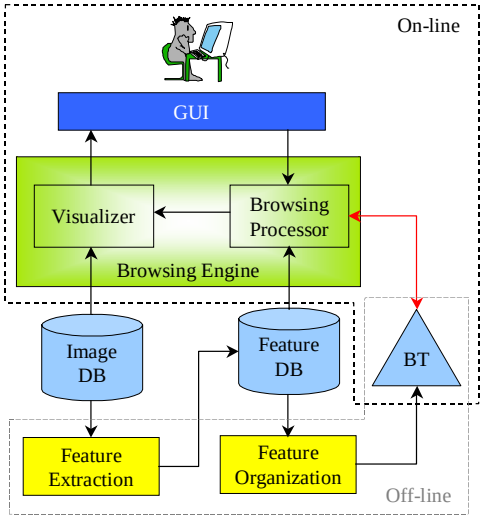


Figure 1: The PIBE architecture (BT = Browsing Tree).

The rest of the paper is organized as follows. In Section 2 we describe the state-of-the-art. Section 3 reports the main facilities that distinguish PIBE from other browsing systems, describing in details each component of its architecture. In Section 4 we report implementation notes and present some experimental results showing the effectiveness and the efficiency of PIBE. In Section 5 we conclude, drafting possible directions for future work.

2. RELATED WORK

The main peculiarity that distinguishes browsing systems presented in literature in the last few years is the information they consider to evaluate the similarity between images. A first case is represented by those systems in which the similarity is assessed in terms of *automatically* extracted low-level features (e.g., [1, 4, 8]). A second class of systems, on the other hand, considers either high-level semantic concepts, *manually* associated to each object, or some extra information, as, for example, the shoot time recorded for each photo by modern digital cameras (e.g., [5, 13]). Finally, only a limited number of systems combine information considered by the previous cases (see e.g., [11]). PIBE is inspired by state-of-the-art browsing systems that belong the first class and that, in some way, integrate user-system interaction mechanisms. Even if such systems are not purely browsing systems, but also include image retrieval functionalities, they represent a good starting point in defining a customizable browsing system like PIBE, and for this reason in the following we briefly review their main features.

The well-known image database system El Niño [11] provides the user with a set of instruments which allow to explore the image DB in a way that reflects her preferences. In details, during each session, the user gives some relevance feedback on an initial (random) configuration of images displayed on the screen. The user can also decide which is the more appropriate location for each image, by dragging it with the mouse to the desired position on the display. Depending on the user personalization, the system reorganizes the whole image DB, showing the updated configuration to the user. However, El Niño suffers a main limit: The browsing structure is represented by means of a flat organization (instead of a hierarchical one). Thus, the reorganization of the image DB is a costly operation, because the whole DB is considered. Moreover, browsing heavily depends on the initial random configuration that does not guarantee the visualization of images of interest for the user.

Another image retrieval system that provides some adaptive browsing facilities is PicSOM [8]. PicSOM uses *Self-Organizing Maps* (SOM's) as a way to cluster and visualize the images in a 2-D grid, and defines a hierarchical browsing structure based on SOM's (i.e., the *Tree Structured Self-Organizing Map*, TS-SOM). Using PicSOM, the user can browse the different levels of the browsing structure and modify the organization of the images by providing a relevance feedback on the displayed images. Depending on the TS-SOM, PicSOM is able to perform a reorganization (convolution) of the tree maps without considering the whole DB. However, the browsing functionality is not intuitive for the user: It is based on maps represented by colored regions (i.e., dark/light regions correspond to positive/negative values of the convolution of a map) that do not give the user any idea about the actual images present in the involved region (thus, the user browses the structure without knowing where she is going).

The work presented in [4] proposes an approach, called *active browsing*, that integrates relevance feedback into the browsing environment to update the image DB organization. This is done by defining a browsing structure (called *similarity pyramid*) by means of a hierarchical clustering algorithm. At a coarse level, the similarity pyramid allows to get a view of the image DB as a set of clusters represented by some representative images. On the other hand, in a finer layer the user can see the specific images within each cluster. However, the reorganization of the similarity pyramid is a costly operation: Given the set of relevant images selected by the user, the system defines a threshold to prune from the structure the images that are far from the selected ones. Then, it reorganizes the pyramid in order to have similar images close in the image layout. It has to be noted that, if the set of images selected by the user are scattered within the structure, the pruning phase will eliminate only a few images and the update will have to consider a large portion of the image DB.

3. PRINCIPLES OF PIBE

The basic idea of our browsing system is to customize the default Browsing Tree (BT) depending on the users interactions. Before reporting details of how PIBE builds and updates the BT, it is important to highlight the three main

ingredients behind the browsing structure, namely: Image descriptors, (dis)similarity functions, and clustering algorithms. In doing so we abstract from the differences arising from specific implementation choices and just concentrate on implementation-invariant functionalities.

Image visual descriptors: Each image is represented by content-based descriptors representing the image in terms of low-level features such as color, texture, shape, etc. In PIBE, the so-obtained descriptors are only requested to be representable as points (*feature vectors*) in a N-dimensional space. Usually, the more the descriptors are robust in representing the image content, the more the dimensionality of the associated space and the complexity of the algorithms applied for their extraction increase. A simple example of feature vectors are color histograms, where one has the degree of freedom of choosing a suitable numbers of bins (e.g., $N = 32, 64, 128, \ldots$).

Classes of similarity functions: When comparing two images, PIBE needs a (dis)similarity function able to establish in which measure their descriptors are close. In order to support personalization actions on the BT, the (dis)similarity function has to be part of a *parameterized* class of functions. Relevant examples of such classes include L_p norms (e.g., $p = 2$ yields the Euclidean distance), weighted Euclidean distances (where the parameter is the N-dimensional weight vector), and the class of quadratic distances (parameterized by an $N \times N$ matrix of weights). Note that there is a natural trade-off between the complexity of a distance function and its capability of adequately reflecting the user-perceived image similarity.

Clustering algorithms: For large image DB's, the default BT is automatically built using some clustering algorithm on the images' descriptors. Further, a default dissimilarity function is used to compare the feature vectors. The same applies when the BT is customized through some personalization actions. As it will be clear in Section 3.4, in such cases a partial *re-clustering* has to be performed, now with a *personalized* dissimilarity function.

Since the BT is a hierarchical structure, a natural choice for deriving/updating it would be to use a *hierarchical* clustering algorithm [7]. Alternatively, one could use as well a *partitioning* (flat) algorithm, such as *k-means* [7], by recursively applying it down to the desired granularity level. Aspects to be considered in choosing a specific clustering algorithm are time complexity, the ability of discovering arbitrarily shaped clusters, and the sensitivity to noise and outliers.

Although PIBE is parametric with respect to the particular choices adopted for the above points, it is clear that each instantiation should satisfy a number of requirements in order to obtain satisfactory levels of efficiency and effectiveness. First, the combination of choices (for images' descriptors, class of distance functions, and clustering algorithm) should guarantee an overall *scalability* with respect to the cardinality of the image DB. Second, because of visualization constraints, a fundamental requirement for the BT consists in guaranteeing a *suitable cardinality* for each cluster: Clusters too small have poor significance, whereas clusters too large are not effectively visualizable.

In the following we illustrate in detail the Browsing Tree, motivating the particular choices we have adopted for the above described points, and showing how the structure can be customized by the user. In particular, Section 3.1 describes how PIBE builds the default BT. Section 3.2 depicts the set of provided browsing skills. Finally, in Sections 3.3 and 3.4 we accurately illustrate the adaptive *personalization actions* the user can perform to persistently modify the BT.

3.1 The Browsing Tree

In the current implementation of PIBE, we extract from each image a 32-D HSV color histogram and compute the dissimilarity of two histograms $\mathbf{p}$ and $\mathbf{q}$ as a weighted Euclidean distance, that is:

$$d(\mathbf{p}, \mathbf{q}; \mathbf{w}) = \left(\sum_{i=1}^{32} w_i (p_i - q_i)^2 \right)^{1/2} \quad (1)$$

where $\mathbf{w}$ is a vector of weights, whose components are determined as explained in the following.

The choice of using such simple image descriptors as the color histograms, together with the class of weighted Euclidean distances, is mainly motivated by the observation that their combination represents a good compromise between quality of results and complexity of computation when user interaction has to be taken into account [3]. The alternative of using, say, quadratic distance functions, which are indeed able to take into account the correlation between colors, would indeed lead to prohibitive (i.e., quadratic) costs for distance computation. This would negatively affect both the spatial visualization algorithm (see Section 3.2) and the (re-)clustering of images.

The hierarchical structure of the default BT is obtained by first applying the k-means algorithm to the whole image DB. Each of the so-obtained k clusters is then recursively partitioned into k sub-clusters, and so on until less than k images are left in a cluster. During this phase, the weights in Eq. 1 all assume their default value, $w_i = 1$.

Each node of the resulting k-ary tree, which corresponds to a cluster C_j of images, is then enriched with the following information. First, the *centroid*, $c(C_j)$, i.e., the point obtained by averaging all the feature vectors in C_j, is determined. Second, the *local* weight vector $\mathbf{w}_j$ is computed. In particular, following [6], the weight $w_{j,i}$ is:

$$w_{j,i} \propto \frac{1}{\sigma_{j,i}^2} \quad (2)$$

where $\sigma_{j,i}^2$ is the variance of feature vectors in cluster C_j along the i-th coordinate. More precisely, for each cluster C_j we maintain its cardinality, $|Cj|$, and the values of $\sum_{\mathbf{p} \in C_j} \mathbf{p}$ and $\sum_{\mathbf{p} \in C_j} \mathbf{p}^2$. From these statistics the centroid and the variance along each coordinate are immediately derived.

Finally, the *representative image*, $p(C_j)$, of C_j is defined as the image in C_j that is closest to $c(C_j)$, that is:

$$p(C_j) = \text{argmin}_{\mathbf{p}} \{ d(\mathbf{p}, c(C_j); \mathbf{w}_j), \mathbf{p} \in C_j \} \quad (3)$$

The role of $p(C_j)$ is to represent cluster C_j for visualization purposes. To this end, $p(C_j)$ is stored in the parent node of C_j (thus, each node contains the representative images of all its child sub-clusters).

To give an example of a BT, Figure 2 plots a portion of tree with $k = 3$. The root node (cluster C_0 in the figure) represents the whole image DB, which is partitioned into 3 first-level clusters (namely, C_1, C_2, and C_3). Thus, at the top-level of the BT the user can see the 3 representative images associated to such clusters. Similarly, each internal node contains 3 representative images, whereas leaf nodes contain individual images.

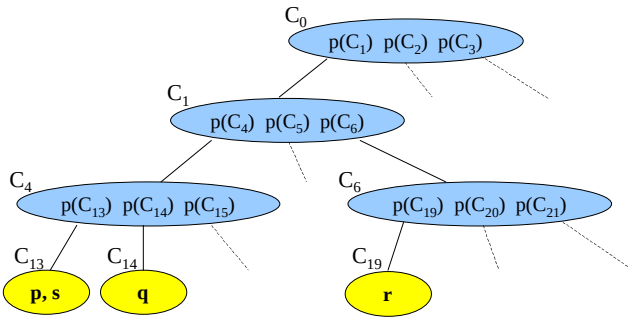


Figure 2: Example of Browsing Tree.

The fan-out of tree nodes, k, is an input parameter: Intuitively, high values for k give a more detailed view of the DB content. However, the higher the value of k, the higher the cost for the BT construction and reorganization. Indeed, since the time complexity of k-means is $O(kn)$ (n = number of images to be (re-)clustered), the (re-)clustering cost is derived to be $O(kn \log_k n)$, thus increasing with k. This requires to trade-off the two requirements, as we discuss in Section 4.

3.2 Browsing

As described in Figure 1, the *Browsing Engine* is the basic component of the PIBE architecture. It consists of the *Visualizer* and the *Browsing Processor* that together implement the navigation mechanism. The Browsing Processor is also in charge of the management of BT updates.

To show the content of the image DB, the Visualizer adopts a spatial visualization approach, where high dimensional feature vectors are mapped on the 2-D screen by grouping together similar images, rather than presenting them in a sequential way [10]. In particular, among the available techniques in PIBE we have chosen to adopt *Multidimensional Scaling* (MDS) [12]. For an example, see Figure 3, where the display of $k = 8$ representative images is reported using the PIBE GUI (for the sake of clarity, in the following we will show only the image view instead of the whole user interface).

To generate the image layout, MDS needs as input all the dissimilarities between the involved images (i.e., representative images for internal nodes of the BT, or individual images for leaf nodes). For each pair of images $\mathbf{p}$ and $\mathbf{q}$ to be displayed, a *specific* weighted Euclidean distance needs to be used.[1] Remind that, at any time, the BT maintains a local distance function for each cluster. Thus, $\mathbf{p}$ and $\mathbf{q}$ should be compared using the distance function of the *most specific cluster* containing both of them. Formally, let $LCA(\mathbf{p}, \mathbf{q})$ be the *least common ancestor* of $\mathbf{p}$ and $\mathbf{q}$ in the BT. Then, $\mathbf{w}_{LCA(\mathbf{p},\mathbf{q})}$ is used to compare the two images. For the example of Figure 2, to compare images $\mathbf{p}$ and $\mathbf{s}$ we use $\mathbf{w}_{13}$, whereas for the comparison of images $p(C_{14})$ and $p(C_{19})$ we would use $\mathbf{w}_1$.

[1]For simplicity of explanation, in the following $\mathbf{p}$ will denote, with a slight abuse of notation, both an image or a representative image.

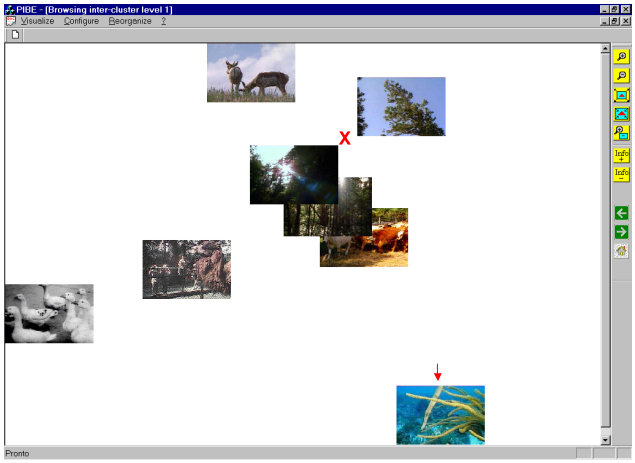


Figure 3: Example of spatial visualization with the PIBE GUI ($k = 8$).

PIBE provides two browsing modalities to explore the DB content:

Vertical: The user selects an image on the display (by *clicking on* it with the left mouse button) and *zooms in* the cluster content (i.e., the k images representing the child clusters are shown). This modality is the traditional top-down way of browsing a hierarchical structure. PIBE also gives the user the opportunity to view the content of an entire subtree. To do this, the user clicks on a cluster image with the right mouse button and provides a maximum number m of images to be displayed (in order to avoid cluttering the screen if the subtree contains a large number of images). The Browsing Processor will then select the m images in the subtree which are most similar (according to the local distance function) to the cluster representative. Figure 4 shows the result of a vertical exploration, where the user has selected the image pointed by the red arrow in Figure 3 (in the example, $k = 8$).

Horizontal: Vertical browsing only allows a top-down exploration of the DB content: If the user is interested in something that is similar to two or more cluster representative images, she has to visit all the relevant tree branches one at a time. To overcome such limitations, PIBE includes a novel browsing modality, called *horizontal* (or *continuous*), that allows the user to also explore the regions of the space where no representative image is present. These regions are also called the *empty space* of the BT.

When the user clicks with the right mouse button on a point in the empty space, she is required to enter the maximum number m of images to be displayed. The Browsing Processor then computes the distances between the selected point and the centroids of clusters at the next BT level, and returns to the Visualizer a

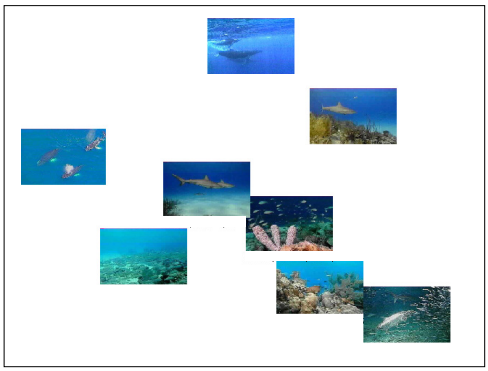


Figure 4: Results of vertical exploration on the example of Figure 3 ($k = 8$).

set of m images, selected from closest clusters first. For the example of Figure 3 ($k = 8$), if the user clicks on the red cross and selects $m = 20$, and supposing that all child clusters are not leaf nodes, the result images would be obtained as follows: 8 from the cluster closest to the selected point, 8 from the second closest cluster, and $20 - (8 + 8) = 4$ from the third closest cluster (results are shown in Figure 5).

Note that if the user executes in sequence h steps of horizontal browsing, the displayed (representative) images are guaranteed to belong to clusters whose least common ancestor is no more than $h+1$ levels up in the BT. In particular, with just 1 step we are guaranteed to see images that all have a same grand-parent node. This, as it will be explained in Section 3.4, allows to bound the complexity of BT updating.

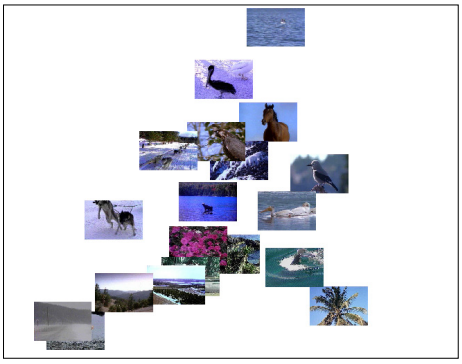


Figure 5: Results of horizontal exploration ($m = 20$) on the example of Figure 3 ($k = 8$).

3.3 Personalization Facilities

State-of-the-art browsing systems that allow the user to personalize the browsing structure usually only consider a set of relevant examples, selected by the user, to modify the structure. If the selected objects are included in different portions of the structure, almost all the entire browsing structure needs to be reorganized [4, 11, 8].

In PIBE we propose novel personalization modalities able to guarantee a *local* reorganization of the BT. In particular, PIBE allows the user to move a cluster representative image $p(C_s)$ (where s stands for "source") using two different *actions*:

1. move $p(C_s)$ on the representative image, $p(C_t)$, of a "target" cluster C_t as a single image (by dragging it with the left mouse button). This is called a *"moving image"* action;

2. move $p(C_s)$ on $p(C_t)$ as representative of the whole cluster C_s (using the right button). This is called a *"moving cluster"* action.

Both actions are considered by PIBE as *asymmetric* operations. Thus, the leading character of the personalization process is the target cluster C_t.

The semantics behind a "moving image" action is to move image $p(C_s)$ into cluster C_t, deleting it from cluster C_s.[2] In this case, the user wants to tell the system that image $p(C_s)$ is similar to image $p(C_t)$. After such user action, $\mathbf{w}_t$ is updated accordingly, since now it is $C_t = C_t \cup \{p(C_s)\}$. By using the so-modified weights, in following browsing sessions the user will see image $p(C_s)$ close to images of C_t.

On the other hand, with a "moving cluster" action the user adds *all* images of C_s to C_t, thus $C_t = C_t \cup C_s$. Thus, she means that all images of C_s, including $p(C_s)$, are similar to $p(C_t)$. Also in this case, the Browsing Processor moves C_s into C_t in the BT and recomputes $\mathbf{w}_t$. Note that the "moving cluster" action can be obtained by repeating the "moving image" action for all images of C_s. Thus, it represents a faster, even if coarser, way to adapt the BT.

Since the actual outcome of a personalization action on the tree is not always easy to predict, PIBE also provides an useful *preview* functionality that allows the user to see the resulting BT before its persistent update. In this way, if the user is not satisfied with the result of an action, she can undo it and return to her current tree.

3.4 Updating the Browsing Tree

In this section we detail how the Browsing Processor updates the BT and provide some more details about the complexity of personalization actions.

Without loss of generality, let us consider the more demanding case where the user moves cluster C_s to cluster C_t (i.e., a "moving cluster" action). The Browsing Processor first recomputes $\mathbf{w}_t$. Then, the moved cluster C_s is deleted from the BT and, using the new weights $\mathbf{w}_t$, the k-means algorithm is recursively applied on the updated $C_t = C_t \cup C_s$, thus obtaining a new sub-tree. Overall, the complexity of restructuring C_t is $O(k|C_t| \log_k |C_t|)$. Although in the worst case, arising when we move a cluster at the first level of the BT, this can lead to high re-clustering costs, in the most common situations, where the user acts on lower levels of the BT, the complexity substantially reduces. This follows from the simple observation that the average cluster size decrease exponentially with the level of the cluster itself in the BT.[3]

Besides restructuring the target cluster C_t, the Browsing Processor needs also to adjust the representative images of clusters that contained the moved (deleted) cluster C_s.

[2]Clearly, a new representative image for cluster C_s will be computed.

[3]More precisely, for a cluster at level l, the root being at level $l = 0$, the average size is n/k^l.

Consider first the case where the personalization action has been executed after an ordinary vertical browsing step. In this case, we should determine the new representative image for the parent cluster, call it C_p, of C_t and C_s. However, since the personalization action has taken place *within* C_p, and the local distance of C_p does not change, it is immediate to see that $p(C_p)$ will not change as well.

Let us now consider the case of a personalization action that is executed after one horizontal browsing step. In this case, as explained in Section 3.2, C_s and C_t are guaranteed to have the same grand-parent, C_{gp}, thus the personalization action stays local to C_{gp}. It follows that the statistics of C_{gp} do not change at all. However, we need to update the statistics of the parent clusters of C_s and C_t. This generalizes to the case of h consecutive horizontal browsing steps, for which the statistics of at most $2h$ clusters need to be updated. Although this might seem computationally demanding, consider that, given the statistics stored within each node of the BT, the new statistics for each cluster are immediately derived from those of the child sub-clusters. The only somewhat challenging task is to determine the new representative images. Although in the current implementation of PIBE this is done going through all images in the cluster in a sequential way, an index-based solution is under development. The idea is to index the whole image DB and to select as representative image the one closest, according to the local distance function, to the cluster centroid. Although this does not guarantee that the representative image belongs to the cluster, we expect that, because of the use of local distance functions, this will be rather an exception than the rule.

To give an intuitive idea of the effects of PIBE actions, Figure 6 shows a default BT. In particular, Figure 6 (b) shows the $m = 20$ images closest to the representative image of the cluster, call it C_t, pointed to by the arrow in Figure 6 (a). From Figure 6 (b), it can be seen that images mainly share a similar color distribution, which only partially captures actual images' content.

Figure 7 shows how the BT changes when the user executes some personalization actions aiming to move images depicting "trees" into cluster C_t . In particular, Figure 7 (a) shows the new representative image of C_t (the one pointed to by the arrow). As expected, also representative images of some sibling clusters (at least the ones involved as sources in a "moving image" action) are changed. In Figure 7 (b) the $m = 20$ images closest to the new representative image of C_t are shown. It can be seen that in the so-modified C_t, the displayed images indeed provide a more variegated (thus, complete) view of trees.

4. IMPLEMENTATION OF PIBE

We have implemented PIBE in Visual C++ under a Pentium II 450 MHz PC equipped with 256MB of memory running Windows NT, and tested it with over 2000 real-life images extracted from the IMSI collection.[4] The BT was stored in secondary memory to guarantee the persistence of personalization actions.

In the present version, each image is represented in the HSV color space as a 32-D histogram, obtained by quantizing the *Hue* and the *Saturation* components in 8 and 4 equally-spaced intervals, respectively. As to the value of the parameter k, which determines both the arity of the BT and the number of images displayed in the PIBE GUI, in our experiments we found that a value in the interval $[8, 12]$ is appropriate. In particular, to this end, we used the *Davies-Bouldine* cluster validity index, that is particularly suitable to identify the k value (within an interval) that generates compact and well-separated clusters [7]. This allows to avoid the cluttering of too many images on the screen during the exploration of the tree structure, particularly at higher levels of the hierarchy, and to obtain quick response times for both the MDS algorithm (which is quadratic in the number of images to be displayed) and the update of the BT. This is also shown in Figures 6 (a) and 7 (a). In particular, for such visual results we used $k = 11$, the same value selected for the rest of the experimental evaluation. With this setup, the BT needs about 120 seconds to be built and requires 3.29MB to be stored.

4.1 User Interface

Besides browsing and personalization actions that are performed using the mouse buttons on the main window, the user interface of PIBE includes a set of buttons (see Figure 8) that allow the user to customize her current view of the image DB. The first five buttons allow the user to zoom in/out the current vista, to increase/decrease the size of image thumbnails and to combine the zooming in with the increment of image size, in order to increase the level of detail of the display. The following two buttons allow the user to display or to delete additional information related to each cluster that is very useful for experimental purposes. In details, it is possible to show the cluster identifier, the name of the representative image, the number of images belonging to the cluster, and the identifier of the parent cluster.

Figure 8: The PIBE toolbar.

Finally, the three buttons at the bottom let the user traverse the BT, by navigating backward (to the parent node) and forward (to the last visited child of the current node), and to return to the root node of the BT.

4.2 Experimental Results

We tested the performance of PIBE in order to evaluate if the system actually reflects the user's needs. To this end we defined appropriate indices able to quantify the effectiveness of our system and to evaluate its efficiency.

The setup for the experiments was as follows. From the whole dataset we selected a number of images of interest defining the set G of *target* objects. In particular, in the experiments we show G consists of 51 "fish" images. We consider results obtained by two distinct scenarios:

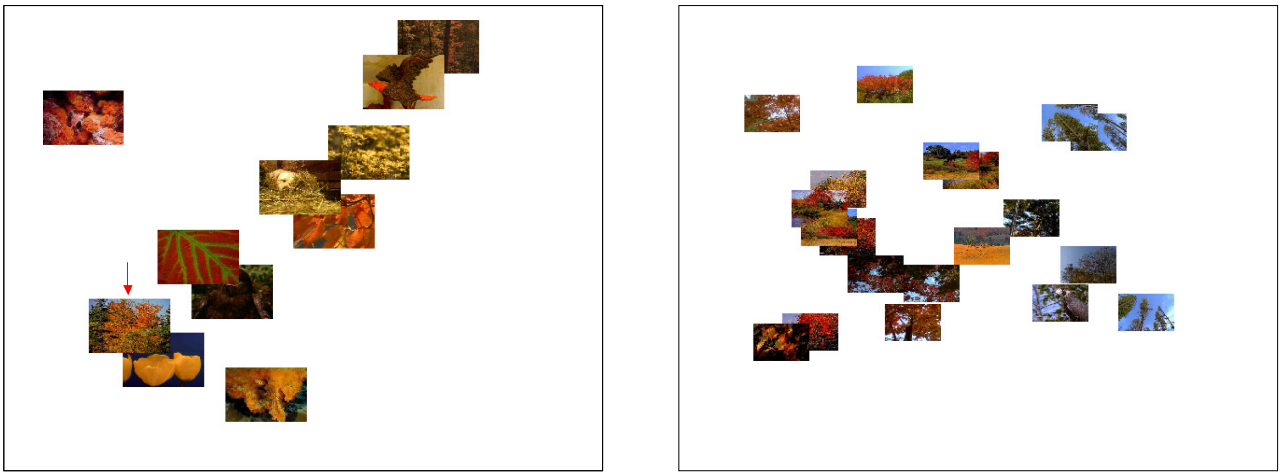
- Default: This is the strategy that uses the default BT (in this case, the 51 "fish" images are scattered through 24 distinct leaf nodes);

- PIBE: In this case, we consider a customized version of the BT, derived from the default tree by applying some personalization actions with the aim to insert the images of G into a single target cluster.

[4]IMSI MasterPhotos 50,000: `http://www.imsisoft.com`.

(a) (b)

Figure 6: Spatial visualization with the default Browsing Tree ($k = 11$): (a) images in a node of the tree; (b) the $m = 20$ images closest to the image pointed to by the red arrow.



(a) (b)

Figure 7: Spatial visualization with a personalized Browsing Tree ($k = 11$): (a) images in a node of the tree; (b) the $m = 20$ images closest to the image pointed to by the red arrow.

For the PIBE case, we envision two different approaches of customization that we refer to as Top_down and Bottom_up. With the first approach, the user first looks for a high-level target cluster, containing a large number of target images. Starting from this node, with only a few actions, the user should be able to find all images in G, since the "moving cluster" action concerns high-level, thus large, clusters. This way, however, irrelevant images (i.e., images not in G) are also moved into the target cluster. In the Bottom_up strategy, on the other hand, the user selects, as target cluster, the leaf node that contains the highest number of target images. Working at the leaf level, the user will need a higher number of actions to achieve her goal, because she will merge small, but more precise, clusters. The Bottom_up strategy is thus better suited to achieve a higher quality of the customized BT, even if the Top_down approach will be much more faster in terms of number of user actions.

Effectiveness: To measure the effectiveness of PIBE we consider the classical *precision* measure. For a given cluster C, the precision P is defined as the number of target objects in C over the cardinality of C. Formally, $P = |C \cap G|/|C|$. Tables 1 and 2 show precision values of the target cluster for the Top_down and Bottom_up strategies, respectively. More in details, each row of the tables reports the number of actions performed by the user during the personalization process, the number of images contained in the target cluster, the number of target images in the target cluster, and the precision values.

n. actions	n. images	n. relevant images	P (%)
0	34	23	67.65
1	62	48	77.42
2	80	51	63.75

Table 1: Precision vs. number of user actions (Top_down approach).

n. actions	n. images	n. relevant images	P (%)
0	2	2	100
10	22	22	100
20	34	34	100
30	52	51	98.08

Table 2: Precision vs. number of user actions (Bottom_up approach).

In both tables, the first row (n. actions=0) represents the Default case. For clarity of explanation, let us consider the Top_down approach (Table 1): The user selects as target a second-level cluster with 34 images, only 23 of which are relevant (initial precision is 23/34 = 67.65%). After two actions only, the user is able to obtain a single cluster containing all the 51 relevant images with a final precision of 51/80 = 63.75%. Table 2 shows results for the Bottom_up strategy: The user starts with a leaf node (i.e., a third-level cluster) with 2 images with a precision of 100%. In this case, the user needs 30 actions to achieve her goal, obtaining a cluster of 52 images with a precision of 51/52 = 98.08%. It has to be noted that in both cases the obtained BT's are quite successful in separating target and irrelevant images. In fact, precision values, not included here for lack of space, for children nodes containing target images are always close to 100%. PIBE is therefore able to capture user's goals of clustering together target images, taking them away from other images. Concerning the average costs of the personalization actions, these are 2.3 and 1.7 seconds for the "moving cluster" and the "moving image" actions, respectively. Such values are however pessimistic, since in our prototype implementation images are stored on a CD-ROM, whose access time is a major factor in determining above costs, and the choice of representative images is done sequentially for each cluster (see Section 3.4).

Efficiency: To evaluate the efficiency of PIBE we measure the time of a browsing session, represented by the number of the mouse clicks needed to reach all target images by means of the PIBE GUI. For fairness, we consider that an image is reached when the leaf node containing it is accessed. Thus, we compute the saved time (denoted by ST) as $ST = (1 - \frac{T_{PIBE}}{T_{Default}}) * 100$, where T_{PIBE} is the time needed using a personalized BT, whereas $T_{Default}$ is that required by a Default tree. Table 3 shows ST values (in percentage) for both the Top_down and Bottom_up strategies.

strategy	$T_{Default}$	T_{PIBE}	ST (%)
Top_down	55	29	47.27
Bottom_up	55	20	63.64

Table 3: Time saved by PIBE.

Using the Default tree, the user needs 55 clicks to reach all target images, that is, the 24 leaf nodes containing them. On the other hand, using PIBE the user is able to reach all relevant images in 29 (Top_down) and 20 (Bottom_up) clicks.

Finally, it has to be noted that, by exploring the whole content of a cluster (see Section 3.2), in the case of the Top_down approach it is possible to visualize all target images with just 2 clicks (3 clicks for the Bottom_up strategy), thus saving 96.36% (resp. 94.54%) of browsing time and still obtaining high precision values, as shown in Table 1 (resp. Table 2). This result is made possible by the combination of the personalization actions and the visualization facilities provided by PIBE.

5. CONCLUSIONS

In this paper we have presented a personalizable image browsing system called PIBE, which provides the user with a novel set of browsing and adaptive facilities allowing to customize the Browsing Tree in an effective and efficient way. Thanks to the intrinsic properties of the Browsing Tree, namely its hierarchical nature and the association of local similarity measures to each node/cluster, PIBE is able to guarantee a persistent reorganization of the structure with just local modifications.

We are investigating several issues in order to improve the effectiveness and efficiency of PIBE. Among these, we are studying how to obtain a Browsing Tree where each cluster can have a variable number of child sub-clusters (thus, making k a local value). We are also considering the integration of "split" operations, that could be helpful to eliminate irrelevant images from a cluster. Concerning efficiency, we are adding an index-based support for the fast determination of representative images.

In its actual implementation, PIBE does not take into account textual descriptions, e.g., sets of keywords, that might be linked with each image, but only relies on low-level image descriptors. We plan to integrate *semantic* concepts extracted by such textual descriptions to improve the exploration capabilities of users: To this end, we are considering the use of lexical ontologies, like WordNet [9].

6. REFERENCES

[1] I. Bartolini and P. Ciaccia. MuSIQUE: A multi-system image querying user interface. In *SEBD 2003*, pp. 437–448, Cetraro, Italy, June 2003.

[2] I. Bartolini, P. Ciaccia, and F. Waas. FeedbackBypass: A new approach to interactive similarity query processing. In *VLDB'01*, pp. 201–210, Rome, Italy, Sept. 2001.

[3] K. Chakrabarti, M. Ortega, S. Mehrotra, and K. Porkaew. Evaluating refined queries in top-k retrieval systems. *IEEE TKDE*, 16(2):256–270, 2004.

[4] J. Chen, C. Bouman, and J. Dalton. Active browsing using similarity pyramids. In *SPIE 1999*, pp. 144–154, San Jose, CA, 1999.

[5] A. Graham, H. Garcia-Molina, A. Paepcke, and T. Winograd. Time as essence for photo browsing through personal digital libraries. In *JCDL'02*, pp. 326–335, Portland, Oregon, USA, June 2002.

[6] Y. Ishikawa, R. Subramanya, and C. Faloutsos. MindReader: Querying databases through multiple examples. In *VLDB'98*, pp. 218–227, New York City, NY, USA, Aug. 1998.

[7] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, New Jersey, 1988.

[8] J. Laaksonen, M. Koskela, S. Laakso, and E. Oja. Self-organising maps as a relevance feedback technique in content-based image retrieval. *Pattern Analysis and Applications*, 2(4):140–152, 2000.

[9] G. A. Miller. WordNet: A lexical database for english. *CACM*, 38(11):39–41, 1995.

[10] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood. Does organisation by similarity assist image browsing? In *CHI'01*, pp. 190–197, Seattle, WA, USA, Apr. 2001.

[11] S. Santini and R. Jain. Integrated browsing and querying for image databases. *IEEE Multimedia*, 7(3):26–39, 2000.

[12] W. S. Torgerson. *Theory and Methods of Scaling*. John Wiley and Sons, New York, NY, 1958.

[13] M. Wallace, K. Karpouzis, G. Stamou, G. Moschovitis, S. Kollias, and C. Schizas. The electronic road: Personalized content browsing. *IEEE Multimedia*, 10(3):49–59, 2003.