INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Image Retrieval with Relevance Feedback based on Genetic Programming

Cristiano D. Ferreira *Ricardo da S. Torres*

Technical Report - IC-07-005 - Relatório Técnico

February - 2007 - Fevereiro

The contents of this report are the sole responsibility of the authors.
O conteúdo do presente relatório é de única responsabilidade dos autores.

Image Retrieval with Relevance Feedback based on Genetic Programming

Cristiano D. Ferreira* Ricardo da S. Torres*

February, 2007

Abstract

In the last years, large digital image collections are generated, manipulated, and stored in databases. In this scenery, it is very important to develop mechanisms to provide automatic means to retrieve images in an efficient and effective way. However, the subjectivity of the user perception of an image usually hampers a fully automatic search and retrieval. Relevance Feedback is one of the commonest approaches to overcome this difficult.

In this paper, a new content-based image retrieval framework with relevance feedback is proposed. This framework uses Genetic Programming (GP) to learn the user needs. The objective of this learning method is to find a function that combines different values of similarity, from distinct descriptors, and best encodes the user perception of image similarity. Several experiments are performed to validate the proposed method, aiming to compare our work with other relevance feedback techniques. The experiment results show that the proposed method outperforms all of them.

1 Introduction

Large image collections have been created and manipulated in managed retrieval applications. These collections are employed in many areas, such as digital libraries, medicine, remote sensing, and others [5, 17, 21]. Given these collections size, it is essential to provide efficient and effective means to retrieve images.

This is the objective of the so-called *contend-based image retrieval (CBIR) systems* [14, 23, 26]. In these systems, the searching process consists in, for a given image, computing the most similar images stored in the database. The searching process rely on the use of image *descriptors*. A descriptor can be characterized by the *feature vector extraction* and the *similarity computation*. The feature vectors encode image properties, like color, texture and shape. Therefore, the similarity between two images is computed as a function of their feature vectors distance.

Different descriptors usually encode distinct image features. However, it is common to retrieve images based on more than one feature. Accordingly, descriptors are combined to

*Institute of Computing, University of Campinas, 13081-970 Campinas, SP

supply this need. Most of the strategies used to perform this combination are based on the assignment of weights to descriptors [9, 32]. These weights try to assess the importance of each descriptor in the composition. There are also other techniques which try to find more complex similarity combination functions for the descriptors, such as [7, 8].

In these approaches, which the descriptors are statically combined [7–9, 32], once the descriptors composition is defined, this composition is fixed and used in all the retrieval sessions. Nevertheless, different people can have distinct visual perception of the same images. Thus, a fixed combination of descriptors may not characterize, simultaneously, the important visual elements for different users in a query. Motivated by this limitation, *relevance feedback* approaches were incorporated into CBIR systems [4, 16, 22, 27]. This technique makes possible the user interaction with the retrieval systems.

Basically, the relevance feedback-based image retrieval process consists in three steps: (i) showing a small number of retrieved images to user; (ii) indication of relevant or irrelevant images by the user; (iii) finally, learning the user needs from his/her feedback, and selecting a new set of images to be shown. This procedure is repeated until a satisfactory result is reached.

An important element of the relevance feedback technique is the learning process. Most of the relevance feedback methods designed to CBIR systems try to learn the user needs by achieving a function to compute the relevance degree of each image, without employing the similarity measure designed by the descriptors [4, 16, 25]. However, the effectiveness of a descriptor does not rely only on the quality of the extracted feature vectors but also depends on the similarity function used. Then, the effectiveness of a descriptor may decrease if the similarity metric designed for the descriptor is not used.

In this paper a new relevance feedback-based CBIR method is proposed. This method adopts a genetic programming approach to learn user preferences in a query session. Genetic programming [18] is a Machine Learning technique used in many applications, such as data mining, signal processing, and regression [2, 11, 33]. This technique is based on the evolution theory to find optimal solutions. It is a form of evolutionary algorithm [1] which is distinguished from the others mainly by the individual representation. In our method, we aim to find a function that combines the similarity values computed by different descriptors, encoding the user needs. Therefore, the similarity measure defined by the descriptors is preserved. The proposed method performance is compared with other relevance feedback techniques for image retrieval [25, 27]. The experiment results show that the proposed method outperforms all of them.

This paper is organized as follows. Section 2 discusses related works. Section 3 describes the CBIR model used (Section 3.1) and gives a brief overview of the Genetic Programming basic concepts. (Section 3.2). Section 4 details the GP-based framework proposed in this paper. Experiments are reported in section 5. Finally, conclusions and future works are presented in section 6.

2 Related work

Relevance feedback (RF) [4, 16, 22, 27, 31] is a technique initially proposed for document retrieval that has been used with great success for human-computer interaction in CBIR. RF addresses two questions referring to CBIR process. The first one is the semantic gap between high level visual properties of images and low level features used to describe them. Usually, it is not easy for a user to map her visual perception of an image into low level features as color and shape. Another issue is concerned with the subjectivity of the image perception. Different people, or even same person in distinct circumstances, can have distinct visual perceptions of the same image.

One of the first relevance feedback-based CBIR method was proposed in [27]. In this work, the learning process is based on assigning weights to each descriptor (*interweight*), and also to each feature vector bin, that is, to each position in this vector (*intraweight*). The learning algorithm heuristically estimates the weight values that best encodes the user needs in the retrieval process. In [25], the weight assignment is again employed. However, an optimization framework is applied to estimate the weights. This framework is based on the minimization of the Generalized Euclidean Distance. These methods [25, 27] are used as baselines in our experiments.

Another pioneer work in the area is the *PicHunter* system, presented in [4]. The PicHunter uses a Bayesian framework for learning process. This mechanism tries to predict the image closer to the user needs. In [10], another approach for *relevance feedback* using Bayesian inference is proposed: the *rich get richer (RGR)*. This method considers the consistency among successive user feedbacks provided in the learning process.

In [3], the query pattern is a set of images, instead of a single one. Furthermore, the retrieval process is divided into two phases, exploration and exploitation. The exploration phase uses a simulated annealing based method and the user feedback for query pattern definition. The exploitation phase uses weights associated to each descriptor to retrieve images.

Another learning technique commonly used in RF is *Support Vector Machine (SVM)*. Basically, the goal of SVM-based methods is to find a hyperplane which separates both relevant and irrelevant images in a feature space. In the work proposed in [16], SVM is used as learning method. The experiments showed that a minimum number of positive and negative examples, heuristically chosen as four, is necessary to guarantee the learning. In [31], Tong et al. propose the use of a *support vector machine active learning* method to separate relevant images from the others. On each iteration, the images closer to the separation hyperplane, the most ambiguous ones, are displayed to the user. To the end of the process, the most distant images from hyperplane are shown.

A genetic algorithm (GA)-based relevance feedback method is proposed in [29]. Local similarity pattern (LSP) is used in the retrieval process. LSP is defined as a structure containing R and F_R, where R is a set with $N \times N$ regions obtained by the image uniform partitioning, and F_R is a set of image features that are extracted from each region and used for similarity computation. GA and relevance feedback are used to determine the feature that best describes each LSP region.

In the aforementioned methods, the learning process is based on either assigning weights to similarity values determined by different descriptors [3, 25, 27, 29] or finding a function to compute the relevance degree of each image [4, 10, 16, 25, 31].

The first group allows only a linear combination of the similarity values. However, a more complex relation among the similarity values could be necessary to express the user needs.

The second group ignores the similarity function defined by the used descriptors. Nevertheless, the effectiveness of a descriptor is not only due to the feature vector codification, but also depends on the similarity function defined by this descriptor. The effectiveness of SVM-based methods only depends of the discriminatory power of the feature vectors. Other approaches define specific similarity functions (e.g., Generalized Euclidean distance [25]) to be employed in the search process. In all these cases, the similarity functions defined by the descriptors are not employed.

In the method proposed in this paper, the similarity functions defined for all available descriptors are used. Furthermore, the proposed GP framework allows a more complex combination of the similarity values than linear combination. Genetic programming is used to obtain this function.

To the best of our knowledge, the GP technique was never used in relevance feedback for CBIR. This approach, however, has been used in Information Retrieval [12, 19]. In [19], a GP-based framework is proposed to associate advertisement to web pages based on these page contents. Fan et al [12] presents a GP-based framework to obtain ranking functions for document retrieval.

In [7, 8], a GP framework is used in the CBIR domain. The objective of this framework is to find a function which combines distance values calculated from different descriptors. This approach does not use relevance feedback.

3 Background

This section presents the CBIR model adopted in our work and a brief overview of the Genetic Programming basic concepts.

3.1 CBIR model

This paper uses the CBIR model proposed in [7, 8], described in the following.

Definition 1 *An* **image** $\hat{I}$ *is a pair* $(D_I, \vec{I})$*, where:*

- D_I *is a finite set of* pixels *(points in* $\mathbb{Z}^2$*, that is,* $D_I \subset \mathbb{Z}^2$*), and*
- $\vec{I} : D_I \rightarrow \mathsf{D}'$ *is a function that assigns to each pixel* p *in* D_I *a vector* $\vec{I}(p)$ *of values in some arbitrary space* D' *(for example,* $\mathsf{D}' = \mathbb{R}^3$ *when a color in the RGB system is assigned to a pixel).*

Definition 2 *A* ***simple descriptor*** *(briefly,* ***descriptor****)* D *is defined as a pair* (ϵ_D, δ_D)*, where:*

- $\epsilon_D : \hat{I} \rightarrow \mathbb{R}^n$ *is a function, which extracts a* feature vector $\vec{v}_{\hat{I}}$ *from an* image $\hat{I}$.
- $\delta_D : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ *is a* similarity function *(e.g., based on a distance metric) that computes the similarity between two images as the inverse of the distance between their corresponding* feature vectors.

Definition 3 *A **feature vector** $\vec{v}_{\hat{I}}$ of an image $\hat{I}$ is a point in $\mathbb{R}^n$ space: $\vec{v}_{\hat{I}} = (v_1, v_2, ..., v_n)$, where n is the dimension of the vector. Examples of possible feature vectors are the color histogram [30], the multiscale fractal curve [6], the set of Fourier coefficients [24]. They essentially encode image properties, such as color, shape, and texture. Note that different types of feature vectors may require different similarity functions.*

Figure 1 illustrates the use of a simple descriptor D to compute the similarity between two images $\hat{I}_A$ and $\hat{I}_B$. First, the extraction algorithm ϵ_D is used to compute the feature vectors $\vec{v}_{\hat{I}_A}$ and $\vec{v}_{\hat{I}_B}$ associated with the images. Next, the similarity function δ_D is used to determine the similarity value d between the images.

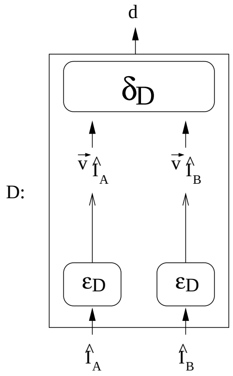


Figure 1: The use of a simple descriptor D for computing the similarity between two images.

Definition 4 *A **composite descriptor** $\hat{D}$ is a pair $(\mathcal{D}, \delta_{\mathcal{D}})$ (see Figure 2), where:*

- *$\mathcal{D} = \{D_1, D_2, \ldots, D_k\}$ is a set of k pre-defined simple descriptors.*
- *$\delta_{\mathcal{D}}$ is a similarity combination function which combines the similarity values d_i obtained from each descriptor $D_i \in \mathcal{D}$, $i = 1, 2, \ldots, k$.*

3.2 Genetic Programming

Genetic programming (GP) [18], such as other evolutionary computation algorithms, is an artificial intelligence problem-solving technique based on the principles of biological inheritance and evolution. In GP approach, the individuals represent programs that undergo evolution. The *fitness* evaluation consists in executing these programs, and measuring their

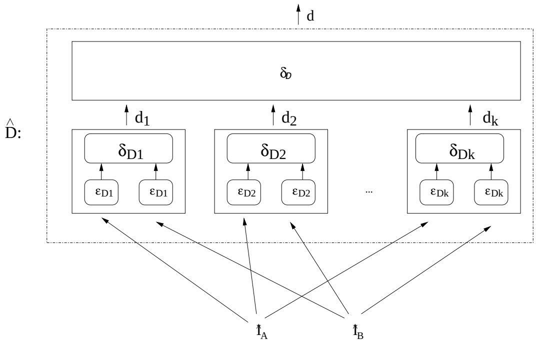


Figure 2: Composite descriptor.

Algorithm 1 Basic GP evolution algorithm for N generations.

```
1 Generate a initial population of individuals
2 for N generations do
3     Calculate the fitness of each individual
4     Select the individuals to genetic operations
5     Apply reproduction
6     Apply crossover
7     Apply mutation
8 end for
```

degrees of evolution. Genetic programming, then, involves an evolution-directed search in the space of possible computer programs that best solve a given problem.

A basic evolution algorithm used in genetic programming is described in Algorithm 1. At beginning of the evolution, an initial population of individuals is created (line 1). Next, a loop of successive steps are performed to evolve these individuals: the fitness calculation of each individual (line 3), the selection of the individuals (line 4), based on their fitness, to breed a new population by applying genetic operators (lines 5–7). In the following, these steps are presented in more details.

Usually, a GP individual represents a program and is encoded in a tree. In this kind of enconding, an individual contains two kinds of nodes, *terminals* (leaf nodes) and *functions* (intern nodes). Terminals are usually programs inputs, although they may also be constants. Functions take inputs and produces outputs. A function input can be either a terminal or the output of another function.

The fitness of an individual is determined by its effectiveness in producing the correct outputs for all cases in a *training set*. The training set is a set containing inputs and their correspondent previously known outputs.

To evolve the population, and optimize the desired objectives, it is necessary to choose the correct individuals to be subject to genetic operators. Thus, *selection operators* are employed to select the individuals, usually, based on their fitness. Examples of selection method are *roulette wheel*, *tournament* and *rank-based* selections [1].

Genetic operators introduce variability in the individuals and make evolution possible, which may produce better individuals in posterior generations. The *crossover* operator exchanges sub-trees from a pair of individuals, generating two others. *Mutation* operator replaces a randomly chosen sub-tree from an individual by a sub-tree randomly generated. The *reproduction* operator simply copies individuals and insert them in the next generation.

4 GP-based relevance feedback framework

This section presents the proposed GP-based CBIR framework with relevance feedback. In this method, a composite descriptor $\hat{D} = (\mathcal{D}, \delta_{\mathcal{D}})$ (see Section 3.1) is employed to rank N database images defined as $DB = \{db_1, db_2, \ldots, db_N\}$. The set of K simple descriptors of $\hat{D}$ is represented by $\mathcal{D} = \{D_1, D_2, \ldots, D_K\}$. The similarity between two images I_j and I_k, computed by D_i, is represented by $d_{iI_jI_k}$. All similarities $d_{iI_jI_k}$ are normalized between 0 and 1. A Gaussian normalization [27] can be employed to normalize these values.

Let L be a number of images displayed on each iteration. Let Q be the query pattern $Q = \{q_1, q_2, \ldots, q_M\}$, where M is the number of elements in Q, formed by the query image q_1 and all images defined as relevant during a retrieval session.

Observe that the query pattern used in our method is composed not only by the query image but also by all images labeled as relevant over all iterations. We expect to improve the retrieval process by considering the entire information obtained from the user about her needs.

Algorithm 2 presents an overview of the retrieval process proposed in this paper. The user interactions are indicated in italic. At the beginning of the retrieval process, the user indicates the query image q_1 (line 1). Based on this image, a initial set of images is selected to be shown to the user (line 2). Thus, the user is able to indicate the relevant images, from this initial set, starting the relevance feedback iterations. Each iteration involves the following steps: user indication of relevant images (line 4); the update of the query pattern (line 5); the learning of the user preference by using GP (line 6); database images ranking (line 7); and the exhibition of the most similar images (line 8).

Algorithm 2 The proposed GP-based relevance feedback process.

```
1 User indication of query image $q_1$
2 Show the initial set of images
3 while the user is not satisfied do
4     User indication of the relevant images
5     Update query pattern $Q$
6     Apply GP to find the best individuals (similarity composition functions) – see
      Algorithm 1
7     Rank the database images
8     Show the $L$ most similar images
9 end while
```

The selection of the initial image set, the use of GP to find the best similarity composition functions and the algorithm to rank database images are presented in details in the following subsections.

4.1 Selecting the initial image set

The initial set of images showed to the user is defined by ranking the database images db_i according to their similarity to the query image q_1. This process is performed in two steps. Firstly, each simple descriptor $D_j \in \mathcal{D}$ is used to compute the similarity $d_{jq_1db_i}$. Next, the arithmetic mean is used to combine all these similarity values, that is

$$\delta_{MEAN}(q_1, db_i) = \frac{\sum\limits_{j=1}^{K} d_{jq_1db_i}}{K} \tag{1}$$

This combination uses all descriptors available and assigns the same degree of importance to all of them.

Hence, the L first images are exhibited to the user. The user, then, identifies the set $R = \{R_1, R_2, \ldots, R_P\}$ of P relevant images, and all images $\{R_i | R_i \notin Q\}$ are inserted into the query pattern Q.

4.2 Finding the best similarity combinations – The GP framework

As aforementioned, the goal of our learning mechanism is to find the similarity combination functions that best encode the user needs. We employ GP to find these combinations. As presented in Section 3.2, the GP technique requires the definition of several components, such as selection method, genetic operators, etc. In this section, the two main GP elements of our GP-based learning process, the individual definition and the fitness computation, are discussed in details.

4.2.1 Individual definition

In our method, each GP individual represents a candidate function $\delta_{\mathcal{D}}$, that is, a similarity combination function. This is encoded in a tree structure, as proposed in [7, 8]. Intern nodes contain arithmetic operators. Leaf nodes have similarities values $d_{iI_jI_k}$, where $1 \leq i \leq K$. Figure 3 shows an example of an individual. The individual in this figure represents the function $f(d_{1I_jI_k}, d_{2I_jI_k}, d_{3I_jI_k}) = \frac{d_{1I_jI_k} * d_{2I_jI_k}}{d_{1I_jI_k}} + \sqrt{d_{3I_jI_k}}$. This figure considers the use of three distinct descriptors and the set of operators $\{+, /, \log, sqrt\}$ as intern nodes. This individual representation is very suitable for the proposed GP search engine, since it directly encodes the candidates for the $\delta_{\mathcal{D}}$ function.

4.2.2 Individual fitness computation

The goal of the proposed fitness computation process is to assign the highest fitness values to the individuals that best encodes the user preferences. In our approach, the fitness

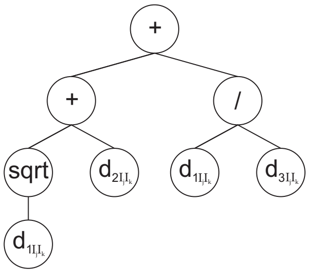


Figure 3: Example of an GP individual.

computation is based on the ranking of the database images defined by each individual. Individuals which rank relevant images at the first positions must receive a high fitness value. The proposed fitness computation process is based on this objective criterion.

Training set definition. As aforementioned, the fitness of an individual is computed based on its performance in a training set. In the proposed method, the training set is defined as the following.

Definition 5 *The **training set** can be defined as a pair $\mathcal{T} = (T, r)$ where:*

- *the* training images *set $T = \{t_1, t_2, \ldots, t_{N_T}\}$ is a set of N_T distinct images.*
- *$r : T \rightarrow \mathbb{R}$ is a function that indicates the user feedback for each image in T.*

For instance, $r(t_i)$, where $t_i \in T$, can be defined as

$$r(t_i) = \begin{cases} 1, & \text{if } t_i \text{ is relevant.} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

It is important that N_T is small ($N_T << N$) to allow a fast computation of each individual fitness. On the other hand, T must also contain a number of images sufficient to represent both, the entire database and the user needs, allowing a suitable evaluation of the individuals. In our approach, T is composed by the last L images exhibited to the user and other $N_T - L$, randomly chosen from the database. To answer the computation time issue, experiments (see Section 5) show that values between 0.5% to 5% of the database size can be suitable choices for N_T. Note that this training set composition represents the user needs, by considering the last exhibited images, and the entire database, by randomly choosing database images.

Fitness computation. The fitness of an individual δ_i is computed based on the similarity between the query pattern and all images from the training set. The fitness computation process is divided into three phases. On the first phase, M ranked lists are computed, each one considering the similarity, according to δ_i, among all training set images and each image in the query pattern. On the second phase, these rankings are evaluated. Finally, on the third phase, the final individual fitness is computed. Figure 4 illustrates this process applied to an individual δ_i.

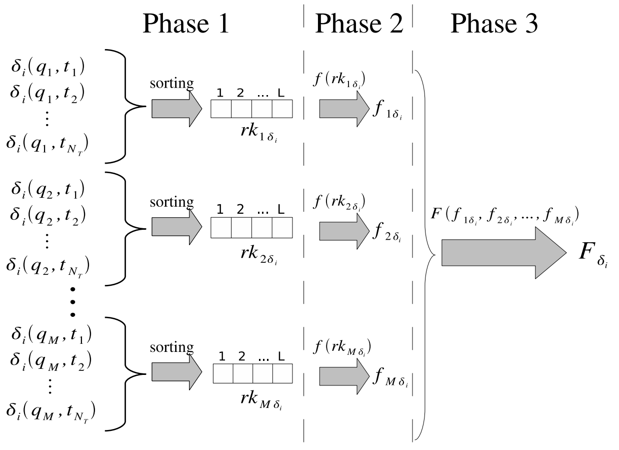


Figure 4: Fitness computation of an individual δ_i.

Phase 1. As can be seen in Figure 4, on the first phase, for each query pattern image q_j, the training images $t_k \in T$ are sorted according to their similarity $(\delta_i(q_j, t_k))$. The L first images define a ranked list $rk_{j\delta_i}$. Thus, M ranked lists are computed, one with regard query pattern image q_j.

Phase 2. Once these rankings are obtained, the second phase starts. The goal of this phase is to evaluate each single ranked list $rk_{j\delta_i}$ generated in Phase 1. This evaluation consists in assigning high values to ranked lists, in which relevant images present in the training set are ranked in the first positions. This evaluation is accomplished by applying an evaluation function $f(rk_{j\delta_i})$ that considers the rank position of the relevant images in $rk_{j\delta_i}$.

In our approach the function $f(rk_{j\delta_i})$ follows the *utility theory* principles [13]. According to this theory, there is an *utility function* which assigns a value to an item, regarding user preference. Usually, it assumes that the utility of an item decreases according it position in a certain ranking [11]. Formally, given two items It_i and It_{i+1}, where i is a position in a ranking, the following condition must be satisfied by a utility function $U(x)$: $U(It_i) > U(It_{i+1})$. In this paper, each item is an image.

An example of $f(rk_{j\delta_i})$ evaluation function is

$$f(rk_{j\delta_i}) = \sum_{l=1}^{L} r(rk_{j\delta_i}[l]) \times k_1 \times \log_{10}(1000/l) \text{ [11]} \tag{3}$$

where k_1 is a constant experimentally defined in [11] as 2, $rk_{j\delta_i}[l]$ is the l^{th} image in the ranking $rk_{j\delta_i}$ and $r(rk_{j\delta_i}) = 1$, if $rk_{j\delta_i}[l] \in R$ or $r(rk_{j\delta_i}) = 0$ otherwise.

Hence, applying $f(rk_{j\delta_i})$ to each ranking $rk_{j\delta_i}$ defines M values $f_{1\delta_i}, f_{2\delta_i}, \ldots, f_{M\delta_i}$.

Phase 3. On the third phase, the final fitness F_{δ_i} of the individual δ_i is computed as the average of the values $f_{j\delta_i}$, that is

$$F(f_{1\delta_i}, f_{2\delta_i}, \ldots, f_{M\delta_i}) = \frac{\sum\limits_{j=1}^{M} f_{j\delta_i}}{M} \tag{4}$$

Algorithm 3 summarizes the fitness computation process. Lines 3–7 refer to the first phase of the fitness computation: the ranked lists obtention. Phase 2, the rankings evaluation process, corresponds to line 8. At last, the final fitness value of the individual is computed in line 10 (phase 3).

Algorithm 3 The fitness computation process.

```
1  Input:  individual $\delta_i$, query pattern $Q$, training set $\mathcal{T}$.
2  Output:  the fitness of the individual $\delta_i$.
3  for all $q_j \in Q$ do
4      for all $t_k \in T$ do
5          $rk_{j\delta_i}[k] \leftarrow \delta_i(q_j, t_k)$
6      end for
7      Sort $rk_{j\delta_i}$
8      $f_{j\delta_i} \leftarrow f(rk_{j\delta_i})$
9  end for
10 $F_{\delta_i} \leftarrow \frac{\sum_{j=1}^{M} f_{j\delta_i}}{M}$
11 return $F_{\delta_i}$
```

An important aspect of this process is the incorporation of all the knowledge accumulated over the iterations in the learning process by using all known relevant images in the fitness computation. This solution addresses the common problem of relevance feedback techniques, related to small training set provided to the learning process [34].

4.3 Ranking database images

Once computed the fitness of the individuals, it is possible to define the best individual that will be used to rank the database images. However, it is possible that more than one individual has a high fitness. Actually, if the query pattern size M is small, there is a highly probability that many individuals have a good fitness. Our strategy tries to improve the database images ranking by combining the ranked lists obtained from these "good" individuals. This combination is achieved by applying a *voting scheme*. Let δ_{best} be the best individual obtained from GP (see Section 4.2) in the current iteration. The set S of individuals selected to vote is defined as

$$S = \{\delta_i | \frac{F_{\delta_i}}{F_{\delta_{best}}} \geq \alpha\} \tag{5}$$

where $\alpha \in [0, 1]$ (e.g., $\alpha = 0.95$). The α value is called *voting selection ratio threshold.*

In the voting scheme, all selected individuals vote for β (e.g., $\beta = L$) candidate images. The most voted images are showed to the user. Algorithm 4 presents this process in details.

Algorithm 4 The voting scheme used to rank the database images.

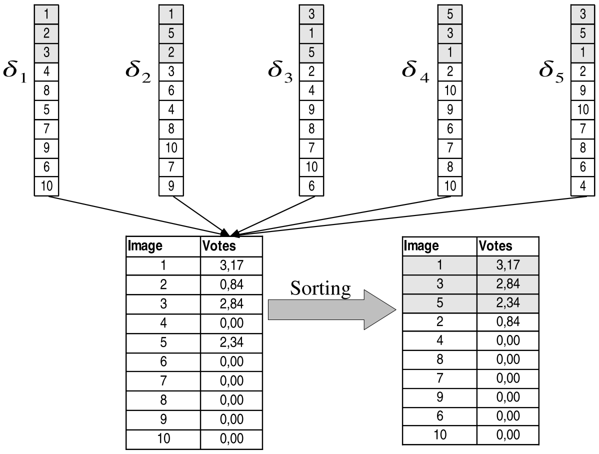
```
1  Input:  Set $S$ of selected individuals, database $DB$, query pattern $Q$, number of
       displayed images $L$.
2  Output:  ranked list of images to be displayed.
3  for all $\delta_i \in S$ do
4      for all $db_j \in DB$ do
5          $rk_i[j] \leftarrow Sim_{\delta_i}(Q, db_j)$
6      end for
7      Sort $rk_i[j]$
8      for $j \leftarrow 1$ to $\beta$ do
9          $votes[rk_i[j]] \leftarrow votes[rk_i[j]] + 1/j$
10     end for
11 end for
12 Sort $DB$ regarding $votes$
13 return the $L$ most voted images
```

Firstly, the database images are sorted by using each selected individual δ_i, regarding the similarity $Sim_{\delta_i}(Q, db_j)$, between each image db_j and the query pattern Q (lines 3–7). Thus, there is a ranking of images associated to each selected individual $\delta_i \in S$. The similarity function $Sim_{\delta_i}(Q, db_j)$ is defined as the greatest value among $\{\delta_i(q_k, db_j) | 1 \leq k \leq M\}$.

Observe that all images of the query pattern are used to rank the database. Therefore, our approach considers that not only the database images similar to the query image are good candidates to be relevant to the user, but also, those similar to any image belonging to the query pattern.

Each image on the first β (e.g., $\beta = L$) positions in each ranking receives a vote inversely proportional to its position (lines 8–10). For instance, the first image receives a vote equal to 1, the second, 1/2, the third, 1/3 and so on. Then, the database images are sorted according to the sum of their votes (line 12). Finally, the L most voted images are selected to be shown to the user.

Figure 5 exemplifies this voting scheme. In this example, there are ten images in the database ($DB = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$), five individuals selected to vote ($S = \{\delta_1, \delta_2, \delta_3, \delta_4, \delta_5\}$) and $\beta = L = 3$. In the ranked list defined by individual δ_1, the images $1, 2$ and 3 appear, in this order, in the first three positions. Then, image 1 receives a vote equal to 1, image 2, in turn, receives 1/2, while 1/3 is added to the image 3 votes. This process is repeated for the ranked lists defined by the other individuals. The table in the bottom left position of this figure shows the total votes of each candidate image. Finally, the bottom right table shows the database images sorted with regard their votes. The first three images ($1, 3$, and 5) would be showed to the user.



δ_1	δ_2	δ_3	δ_4	δ_5
1	1	3	5	3
2	5	1	3	5
3	2	5	1	1
4	3	2	2	2
8	6	4	10	9
5	4	9	9	10
7	8	8	6	7
9	10	7	7	8
6	7	10	8	6
10	9	6	10	4

Image	Votes
1	3,17
2	0,84
3	2,84
4	0,00
5	2,34
6	0,00
7	0,00
8	0,00
9	0,00
10	0,00

Image	Votes
1	3,17
3	2,84
5	2,34
2	0,84
4	0,00
8	0,00
7	0,00
9	0,00
6	0,00
10	0,00

Figure 5: Example of the application of the voting scheme.

5 Experiments

This section describes in details the experiments performed to validate our framework.

5.1 Image descriptors

The proposed framework was presented in a generic way, since there is no restrictions about descriptors that can be used to characterize the images. Color, shape, and texture based descriptors are the commonest ones. Specifically, in the experiments reported in this paper, only shape descriptors were used. They are described in the following.

Moment Invariants: For Moment Invariants, each object has been represented by a 14 dimensional feature vector, including two sets of normalized Moment Invariants [15], one from the object boundary and another from a solid silhouette. The Euclidean distance was used as similarity measure.

Fourier Descriptors: We have implemented the method described in [15, 24] to represent a shape with Fourier Descriptors applied to a contour. Each original object and its transformed versions are represented by the most significant 126 components. Again, the Euclidean distance was used as similarity function.

Contour Multiscale Fractal Dimension or shortly, **MS Fractal Dimension**: We have implemented the method described in [6] (with degree of the multiscale fractal polynomial equal to 25 and generating a 25-bin representation) to extract multiscale fractal values for a contour. Again, the Euclidean distance was used to measure the similarity between two multiscale fractal dimension representations.

5.2 Image Database

Two image databases were used in the experiments and they are presented in the following.

Table 1: Configuration parameters.

Parmeter	**Fish contours DB**	**MPEG 7 DB**
Population size	30	60
Maximum number of generations	10	10
Maximum tree depth	10	15
Function set	$+, \times, /$ (*protected*)	$+, \times, /$ (*protected*)
Terminal set	similarity by simple descriptors (Section 5.1)	similarity by simple descriptors (Section 5.1)
Initialization method	half and half	half and half
Initial ramp	$2-6$	$2-6$
Crossover rate	0.85	0.80
Mutation rate	0.15	0.20
Selection method	tournament (size 2)	tournament (size 2)
Fitness function	FFP2 (Eq. 3)	FFP2 (Eq. 3)
Training set size	55 (0.5% of DB size)	70 (5% of DB size)
Voting selection ratio threshold	0.95	1.00

The first database is composed of eleven hundred fish contour images. These images were obtained from a original set of 1100 images available from [28]. From each image contour, rotation and scaling were applied, originating nine new images. Thus, the final database contain 11000 images distributed in 1100 class. In the experiments, 1100 images are randomly chosen as query, each one from a distinct class.

The experiment with fish contour images will assess the performance of the proposed method in a database with many images and a few number of relevant ones for each query.

The other database is the MPEG-7 Part B. This is the main part of the Core Experiment CE-Shape-1 [20]. The total number of images in the database is 1400: 70 classes of various shapes, each class with 20 images. In the experiments, all images from this database are chosen as query. With this database is possible to appraise the GP-based approach when there are many relevant images to be found.

5.3 GP Implementation

We implement a CBIR system with the minimal requirements to validate our method. The user behavior was simulated by computer. In each iteration, all images belonging to the same class of the query are labeled as relevant.

The configuration parameters used in framework implementation are shown in Table 1.

These parameters were determined empirically through several experiments. As can be seen in this table, only crossover and mutation operators were used in search process. Both uses 2-tournament as selection method. Due to the small population size, the use of reproduction operator makes the population diversity fall down quickly. Thus, this operator was not employed. The protected division used in function set returns 1 if divisor value was zero. The maximum number of generations adopted was 10, but if a individual had normalized fitness value (between 0 and 1) equal to 1 before the last generation, the GP run is finished earlier. The used fitness function FFP2 (Equation 3) is presented in [11].

5.4 Performance measures

We use two different kinds of curves, *precision-recall* and *recall-iterations*, and a table of recall values to evaluate performance in the experiments.

Precision-Recall curve is a common performance evaluation criterion used in information retrieval systems that have been employed to evaluate CBIR systems. Precision $Pr(q)$ can be defined as the number of retrieved relevant images $R(q)$ over the total number of retrieved images $N(q)$ for a given query q, that is $Pr(q) = \frac{R(q)}{N(q)}$ and Recall $Re(q)$ is the number of retrieved relevant images $R(q)$ over the total number of relevant images present in the database $M(q)$ for a given query q, that is $Re(q) = \frac{R(q)}{M(q)}$.

It is desired that both precision and recall be high. However, these measures are conflicting. To increase the number of retrieved relevant images (and the recall), generally it is needed to increase the number of returned images, and consequently, the precision decreases. Thus, precision-recall curve is usually employed to show this relation and so characterize the performance of a CBIR technique.

Another evaluation criterion used is recall-iteration curve. The recall value is measured with regard to the number L of displayed images on each iteration. Thus, the recall-iteration curve shows how the number of displayed relevant images increases over the iterations.

A table of recall values is also presented. In this table, again, the recall value is measured regarding the number L of displayed image on each iteration.

5.5 Experiment design

Experiments considered 10 iterations for each query. On each iteration, 40 images were displayed. The first set of images displayed, for a given query, are based on the average of the similarity values measured by each employed descriptor. We refer to our approach as GP.

For each database, the experiments are divided into two parts. In the first, we evaluate the performance of our method under different values of training set size and voting selection ratio. This experiments aim to evaluate how these parameters affect the proposed method. In the other part of the experiments, the proposed method is compared with two other relevance feedback techniques, that we call Rui98 [27] and Rui2000 [25] (see Section 2) with regard their effectiveness.

In the first part, tables with recall values are shown. These recall values are computed as the average of recall values for all queries submited to the system, considering last the iteration.

In the second part, both precision-recall and recall-iterations curves are presented. The precision-recall curve exhibited is computed as the average of precision-recall curves computed for all queries submitted to the system. Only the precision-recall values on the last

Table 2: Recall values for different training set sizes in fish contour database.

Training set size (% DB size)	55(0.5)	110(1.0)	165(1.5)	220(2.0)
Recall at 40 (%)	**95.44**	94.27	94.00	94.11

Table 3: Recall values for different training set sizes in MPEG7 database.

Training set size (% DB size)	70(5)	140(10)	210(15)	280(20)
Recall at 40 (%)	**77.14**	76.21	76.20	76.04

iteration were considered. The recall-iterations curve shows the average of recall values on each iteration for all queries.

5.6 Results

Tables 2 and 3 show the recall values for different training set sizes in the fish contour and in the MPEG7 databases, respectively. In both databases, the best performances were obtained with the smallest training set sizes tested. Table 4 presents the recall values for different voting selection ratio thresholds in the fish contour and in the MPEG7 databases. The best results were obtained for the threshold values 0.95 and 1.00 for the fish contour and MPEG7 databases, respectively. All the best parameter values discovered in the first part were employed in the second part of the experiments.

Figures 6 and 7 show precision-recall curves comparing the relevance feedback techniques performance in the fish contour and in the MPEG7 databases, respectively. The proposed method outperforms Rui98 and Rui2000 in both cases. In the fish database, GP and Rui98 presents close precision values until a recall of 0.5. From this point on, GP precision values are higher than Rui98 ones. In the MPEG7 database, unlike the first case, Rui2000 presents a better results than Rui98. Again, GP yelds the highest precision values, but now, for all recall values.

Figures 8 and 9 shows the recall-iteration curves of the relevance feedback techniques

Table 4: Recall values for different voting selection ratio thresholds in fish contour database.

Voting selection ratio thresholds	**Fish DB Recall (%)**	**MPEG7 DB Recall (%)**
0.80	95.35	77.11
0.85	95.31	77.12
0.90	95.39	77.15
0.95	**95.44**	77.14
1.00	94.51	**78.19**

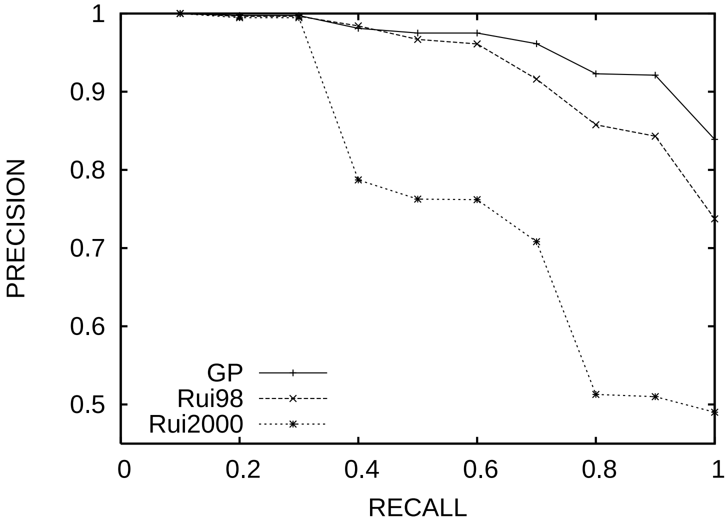


Figure 6: Precision-recall curves showing relevance feedback techniques performance in the fish contour database.

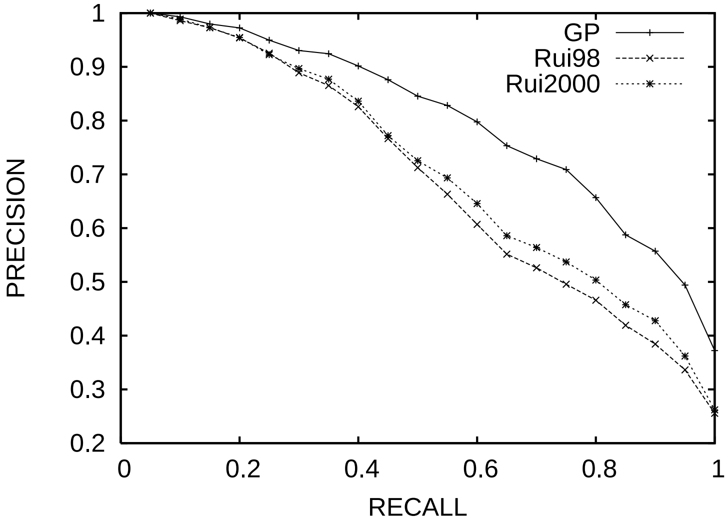


Figure 7: Precision-recall curves showing relevance feedback techniques performance in the MPEG7 database.

for fish contour and MPEG7 databases, respectively. Again, GP outperforms the other methods in both cases. In the first iteration of fish contour database, Rui98 presents a better recall value than GP. However, in subsequent iterations, GP recall values increase faster than Rui98 ones. Observe that for the fish contour database the recall value of GP does not increase too much. Nevertheless, for the MPEG7 database, GP presents a high recall increasing ratio. The behavior in the first case happens due to the high recall values reached already in early iterations. Thus, it is difficult to improve the first good recall values. On the other side, in the MPEG7 database, GP needs more query refinement to obtain better results. In this case, GP still increases the number of relevant images with the new feedbacks provided over the iterations.

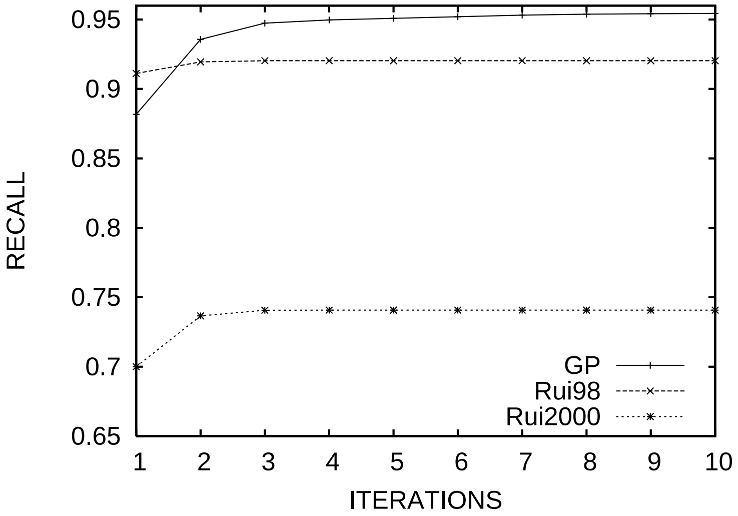


Figure 8: Recall-iterations curves showing relevance feedback techniques performance in the fish contour database.

6 Conclusions

We presented a novel relevance feedback-based CBIR framework. This method uses genetic programming to learn the user preferences and allows a complex combination of the similarity values, using the similarity functions defined for all available descriptors. The query pattern is composed by all relevant images and a voting scheme is used to rank the database images on each iteration. Experiments show that the proposed method improves the retrieval effectiveness finding a good composition of descriptors. Furthermore, our method outperformed two other relevance feedback techniques [25, 27].

One of the next steps of our work is to validate our framework considering other databases. We also plan to compare our method against other relevance feedback techniques [3, 31]. Other future work consists in improving the scalability of our technique, trying to reduce the query pattern size, without reducing the effectiveness of the retrieval process. Another important issue is related to the incorporation of negative feedbacks. This kind of feedback could provide more information to the searching process and, then, could improve the retrieval effectiveness.

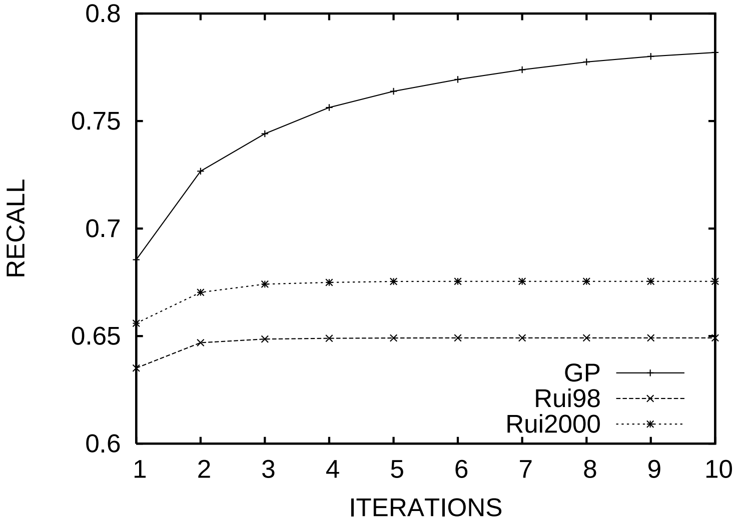


Figure 9: Recall-iterations curves showing relevance feedback techniques performance in the MPEG7 database.

7 Acknowledgments

This work was supported by CNPq, FAEPEX, FAPESP, Microsoft eScience project and WebMaps project.

References

[1] T. Bäck, D. B. Fogel, and Z. Michalewicz. *Evolutionary Computation 1 Basics Algorithms and Operators.* Institute of Physics Publishing, 2002.

[2] B. Bhanu and Y. Lin. Object Detection in Multi-Modal Images Using Genetic Programming. *Applied Soft Computing*, 4(2):175–201, May 2004.

[3] M. Cord, J. Fournier, and S. Philipp-Foliguet. Exploration and search-by-similarity in cbir. In *XVI Brazilian Symposium on Computer Graphics and Image Processing*, pages 175–182, 2003.

[4] I. J. Cox, M. L. Miller, T. P. Minka, T. V. Papathomas, and P. N. Yianilos. The Bayesian Image Retrieval System, PicHunter: Theory, Implementation, and Psychophysical Experiments. *IEEE Transactions on Image Processing*, 9(1):20–37, January 2000.

[5] R. da S. Torres. *Integrating Image and Spatial Data for Biodiversity Information Management.* PhD thesis, Institute of Computing, University of Campinas, 2004.

[6] R. da S. Torres, A. X. Falcão, and L. da F. Costa. A Graph-based Approach for Multiscale Shape Analysis. *Pattern Recognition*, 37(6):1163–1174, June 2004.

[7] R. da S. Torres, A. X. Falcão, M. A. Gonçalves, B. Zhang, W. Fan, and E. A. Fox. A New Framework to Combine Descriptors for Content-based Image Retrieval. Technical Report IC-05-21, Institute of Computing, State University of Campinas, Campinas, Brazil, 2005.

[8] R. da S. Torres, A. X. Falcão, B. Zhang, W. Fan, E. A. Fox, M. A. Gonçalves, and P. Calado. A new framework to combine descriptors for content-based image retrieval. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 335–336. ACM Press, 2005.

[9] R. Dorairaj and K. Namuduri. Compact combination of MPEG-7 color and texture descriptors for image retrieval. In *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 387–391, November 2004.

[10] L. Duan, W. Gao, W. Zeng, and D. Zhao. Adaptive relevance feedback based on Bayesian inference for image retrieval. *Signal Processing*, 85(2):395–399, February 2005.

[11] W. Fan, E. A. Fox, P. Pathak, and H. Wu. The Effects of Fitness Functions on Genetic Programming-Based Ranking Discovery for Web Search. *Journal of the American Society for Information Science and Technology*, 55(7):628–636, 2004.

[12] W. Fan, M. D. Gordon, and P. Pathak. A generic ranking function discovery framework by genetic programming for information retrieval. *Information Processing & Management*, 40(4):587–602, July 2004.

[13] P. C. Fishburn. *Non-Linear Preference and Utility Theory.* Johns Hopkins University Press, Baltimore, 1988.

[14] M. Flickner, H. Sawhney, W. Niblack, Q. H. J. Ashley, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by Image and Video Content: the QBIC System. *IEEE Computer*, 28(9):23–32, Sep 1995.

[15] R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Addison-Wesley, Reading, MA, USA, 1992.

[16] P. Hong, Q. Tian, and T. S. Huang. Incorporate support vector machines to content-based image retrieval with relevant feedback. In *Proceedings of the 7th IEEE International Conference on Image Processing*, pages 750–753, 2000.

[17] A. Kak and C. Pavlopoulou. Content-based image retrieval from large medical databases. *3D Data Processing Visualization and Transmission*, pages 138–147, June 2002.

[18] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, USA, 1992.

[19] A. Lacerda, M. Cristo, M. A. Gonçalves, W. Fan, N. Ziviani, and B. Ribeiro-Neto. Learning to advertise. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 549–556, 2006.

[20] L. J. Latecki and R. Lakamper. Shape Similarity Measure Based on Correspondence of Visual Parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1185–1190, 2000.

[21] M. S. Lew, editor. *Principles of Visual Information Retrieval – Advances in Pattern Recognition.* Springer-Verlag, London Berlin Heidelberg, 2001.

[22] B. Li and S. Yuan. A novel relevance feedback method in content-based image retrieval. In *Proceedings of International Conference on Information Technology: Coding an Computing*, pages 120–123, 2004.

[23] V. E. Ogle and M. Stonebraker. Chabot: Retrieval from Relational Database of Images. IEEE *Computer*, 28(9):40–48, Sep 1995.

[24] E. Persoon and K. Fu. Shape Discrimination Using Fourier Descriptors. *IEEE Transanctions on Systems, Man, and Cybernetics*, 7(3):170– 178, 1977.

[25] Y. Rui and T. Huang. Optimizing learning in image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 236–245, 2000.

[26] Y. Rui, T. S. Huang, and S. F. Chang. Image Retrieval: Current Techniques, Promising Directions, and Open Issues. *Journal of Communications and Image Representation*, 10(1):39–62, March 1999.

[27] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra. Relevance Feedback: A Power Tool for Interactive Content-Based Image Retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):644–655, 1998.

[28] ShapeDB. www.ee.surrey.ac.uk/research/vssp/imagedb/demo.html, December 2006.

[29] Z. Stejić, Y. Takama, and K. Genetic algorithm-based relevance feedback for image retrieval using local similarity patterns. *Information Processing and Management*, 39(1):1–23, 2003.

[30] M. Swain and D. Ballard. Color Indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[31] S. Tong and E. Y. Chang. Support vector machine active learning for image retrieval. In *Proceedings of 9th ACM international conference on Multimedia*, pages 107–118, New York, NY, USA, 2001. ACM Press.

[32] A. Vadivel, A. Majumdar, and S. Sural. Characteristics of weighted feature vector in content-based image retrieval applications. *Intelligent Sensing and Information Processing*, pages 127–132, 2004.

[33] B. Zhang, M. A. Gonçalves, W. Fan, Y. Chen, E. A. Fox, P. Calado, and M. Cristo. Combining structural and citation-based evidence for text classification. In *Proceedings of the 13th ACM Conference on Information and Knowledge Management*, pages 162–163, 2004.

[34] X. S. Zhou and T. S. Huang. Relevance feedback in image retrieval: A comprehensive review. *Multimedia System*, 8(6):536–544, 2003.