Pattern Recognition 42 (2009) 283 – 292

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

A genetic programming framework for content-based image retrieval

Ricardo da S. Torres[a,*], Alexandre X. Falcão[a], Marcos A. Gonçalves[b], João P. Papa[a], Baoping Zhang[c], Weiguo Fan[c], Edward A. Fox[c]

[a]Institute of Computing, University of Campinas–UNICAMP, 13083-970 Campinas, SP, Brazil
[b]Department of Computer Science, Federal University of Minas Gerais, Belo Horizonte, MG, Brazil
[c]Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA

ARTICLE INFO

Article history:
Received 20 December 2007
Received in revised form 18 March 2008
Accepted 16 April 2008

Keywords:
Content-based image retrieval
Genetic programming
Shape descriptors
Image analysis

ABSTRACT

The effectiveness of content-based image retrieval (CBIR) systems can be improved by combining image features or by weighting image similarities, as computed from multiple feature vectors. However, feature combination do not make sense always and the combined similarity function can be more complex than weight-based functions to better satisfy the users' expectations. We address this problem by presenting a *Genetic Programming* framework to the design of combined similarity functions. Our method allows nonlinear combination of image similarities and is validated through several experiments, where the images are retrieved based on the shape of their objects. Experimental results demonstrate that the GP framework is suitable for the design of effective combinations functions.

1. Introduction

Advances in data storage and image acquisition technologies have allowed the creation of large image data sets. In order to deal with these data, it is necessary to develop appropriate information systems which can support different services. The focus of this paper is on content-based image retrieval (CBIR) systems [1]. Basically, CBIR systems try to retrieve images similar to a user-defined specification or pattern (e.g., shape sketch, image example). Their goal is to support image retrieval based on content properties (e.g., shape, texture, and color).

A *feature extraction algorithm* encodes image properties into a feature vector and a *similarity function* computes the similarity between two images as a function of the distance between their feature vectors. An image database can be indexed by using multiple pairs of feature extraction algorithms and similarity functions. We call each pair a *database descriptor*, because they tell how the images are distributed in the distance space. By replacing the similarity function, for example, we can make groups of relevant images more or less compact, and increase or decrease their separation [2]. These descriptors are commonly chosen in a domain-dependent fashion, and, generally, are combined in order to meet users' needs. For example, while one user may wish to retrieve images based on their color features, another one may wish to retrieve images according to their texture properties.

Feature vector and descriptor do not have the same meaning here. The importance of considering the pair, feature extraction algorithm and similarity function, as a descriptor should be better understood. In CBIR systems, it is common to find solutions that combine image features irrespective of the similarity functions [3]. However, these techniques do not make sense, for example, when the image content is a shape and the properties are curvature values along it and color/texture properties inside it. The similarity function usually has a crucial role in making the descriptor as invariant as possible to changes in image scale and rotation. This is true even when we consider only shape descriptors. It does not make sense, for example, to combine multiscale fractal dimensions [2] with bean angle statistics (BAS) [4] irrespective of their similarity functions. The importance of the similarity function coupled with the feature extraction algorithm is illustrated in Fig. 1. Precision–recall curves were computed from an MPEG-7 part B database [5] for four different descriptors. They provide different combinations of feature extraction algorithms that encode BAS [4] and segment saliences (SS) [6], with Euclidean metric and matching by optimum correspondent subsequence (OCS) [7] as similarity functions. We are not mixing properties, only replacing similarity functions, to show their role in the effectiveness of each descriptor. Both SS and BAS have been proposed with OCS. Fig. 1 shows that the configurations which use OCS yield the best effectiveness.

* Corresponding author. Tel.: +55 19 3521 5887.
E-mail addresses: rtorres@ic.unicamp.br (R.S. Torres), afalcao@ic.unicamp.br (A.X. Falcão), mgoncalv@dcc.ufmg.br (M.A. Gonçalves), jpaulo@ic.unicamp.br (J.P. Papa), bzhang@vt.edu (B. Zhang), wfan@vt.edu (W. Fan), fox@vt.edu (E.A. Fox).

0031-3203/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2008.04.010

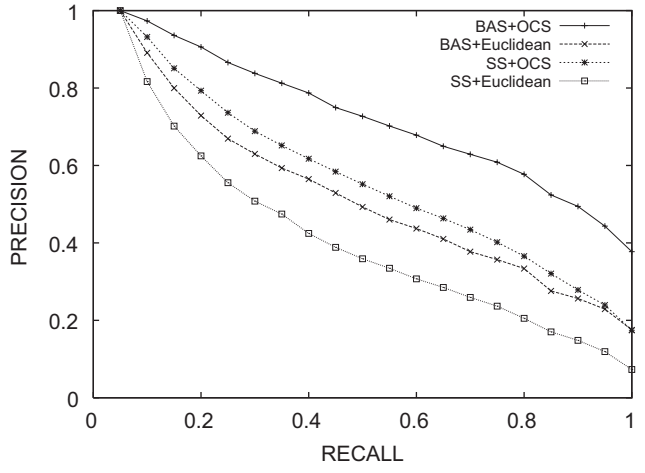

Fig. 1. Precision–recall curves for BAS and SS descriptors in MPEG-7 database using two different similarity functions.

At a higher level, we really wish to combine descriptors encoding several properties in order to address the semantic gap problem: it is not easy for a user to map her/his visual perception of an image into low level features. Without mixing distinct properties in a same feature vector, this combination could be done by weighting the similarity values resulting from different descriptors [8–10]. However, more complex functions than a linear combination are likely to provide more flexibility in matching the results with the users' expectations. We address the problem by presenting a *genetic programming* (GP) framework to the design of combined similarity functions. Our solution relies on the creation of a *composite descriptor*, which is simply the combination of pre-defined descriptors using the GP technique. We employ GP to combine the similarity values obtained from each descriptor, creating a more effective fused similarity function. As far as we know, this approach is original and opens a new and productive field for investigation (considering, for example, different applications, descriptors, and GP parameters).

Our motivation to choose GP stems from its success in many other machine learning applications [11–13]. Some works, for example, show that GP can provide better results for pattern recognition than classical techniques, such as Support Vector Machines [14]. Different from previous approaches based on *genetic algorithms* (GAs), which learn the weights of the linear combination function [15], our framework allows nonlinear combination of descriptors. It is validated through several experiments with two image collections under a wide range of conditions, where the images are retrieved based on the shape of their objects. These experiments demonstrate the effectiveness of the framework according to various evaluation criteria, including precision–recall curves, and using a GA-based approach (its natural competitor) as one of the baselines. Given that it is not based on feature combination, the framework is also suitable for information retrieval from multimodal queries, as for example by text, image, and audio.

The remainder of this paper is organized as follows. Section 2 gives the background information on GAs and GP. Section 3 introduces a generic model for CBIR which includes the notion of simple and composite descriptors. Section 4 presents a formal definition of the combination function discovery problem and describes our framework based on GP. Section 5 describes several experiments, which validate our approach, while Sections 6 and 7 discuss the main achieved results and related works, respectively. In Section 8 we conclude the paper, explaining implications of this study and presenting future research directions.

2. Background

2.1. Genetic programming

GAs [16] and GP [11] belong to a set of artificial intelligence problem-solving techniques based on the principles of biological inheritance and evolution. Each potential solution is called an individual (i.e., a chromosome) in a population. Both GA and GP work by iteratively applying genetic transformations, such as crossover and mutation, to a population of individuals to create more diverse and better performing individuals in subsequent generations. A fitness function is available to assign a fitness value for each individual.

The *main difference* between GA and GP relies on their internal representation—or data structure—of an individual. In general, GA applications represent each individual as a fixed-length bit string, like (1101110 …) or a fixed-length sequence of real numbers (1.2, 2.4, 4, …). In GP, on the other hand, more complex data structures are used (e.g., trees, linked lists, or stacks [17]). Fig. 2 shows an example of a tree representation of a GP individual.

Furthermore, the length of a GP data structure is not fixed, although it may be constrained by implementation to be within a certain size limit. Because of their intrinsic parallel search mechanism and powerful global exploration capability in a high-dimensional space, both GA and GP have been used to solve a wide range of hard optimization problems that oftentimes have no known optimum solutions.

2.2. GP components

In order to apply GP to solve a given problem, several key components of a GP system need to be defined. Table 1 lists these essential components along with their descriptions.

The entire combination discovery framework can be seen as an iterative process. Starting with a set of training images with known relevance judgments, GP first operates on a large population of random combination functions (individuals). These combination functions are then evaluated based on the relevance information from training images. If the stopping criteria is not met, it will go through the genetic transformation steps to create and evaluate the next generation population iteratively.

GP searches for good combination functions by evolving a population along several generations. Population individuals are modified by applying genetic transformations, such as *reproduction*, *mutation*, and *crossover*. The reproduction operator selects the best individuals and copies them to the next generation. The two main variation operators in GP are mutation and crossover. Mutation can be defined as random manipulation that operates on only one individual. This operator selects a point in the GP tree randomly and replaces the existing subtree at that point with a new randomly generated subtree [18]. The crossover operator combines the genetic material of two parents by swapping a subtree of one parent with a part of the other (see Fig. 3).

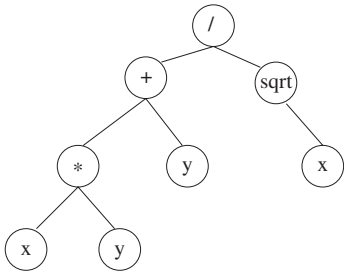

Fig. 2. A sample tree representation.

Table 1
Essential GP components [14]

Components	Meaning
Terminals	Leaf nodes in the tree structure (i.e., *x*, *y* as in Fig. 2).
Functions	Non-leaf nodes used to combine the leaf nodes. Commonly used numerical operations: +, −, ∗, /, log.
Fitness function	The objective function GP aims to optimize.
Reproduction	A genetic operator that copies the individuals with the best fitness values directly into the population for the next generation without going through the crossover operation.
Crossover	A genetic operator that exchanges subtrees from two parents to form two new children. Its aim is to improve the diversity as well as the genetic fitness of the population. This process is shown in Fig. 3.
Mutation	A genetic operator that replaces a selected individual's subtree, whose root is a picked mutation point, with a randomly generated subtree.

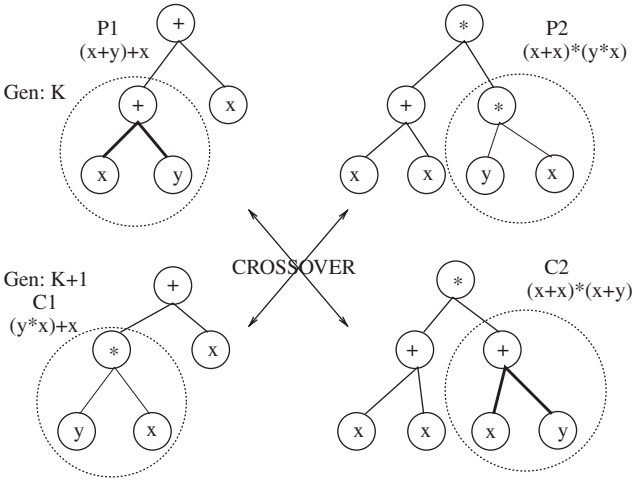

Fig. 3. A graphical illustration of the crossover operation [14].

3. CBIR model

In this section, we formalize how a CBIR system can be modeled.

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

- $D_I \subset \mathbb{Z}^2$ is a finite set of *pixels*, and
- $\vec{I} : D_I \rightarrow \mathrm{D}'$ 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, $\mathrm{D}' = \mathbb{R}^3$ 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 $(\varepsilon_D, \delta_D)$, where:

- $\varepsilon_D : \hat{I} \rightarrow \mathbb{R}^n$ 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* that computes the similarity between two images by taking into account 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: $\vec{v}_{\hat{I}} = (v_1, v_2, \ldots, v_n)$, where n is the dimension of the vector. Examples of possible feature vectors are the color histogram [19], the multi-scale fractal curve [2], and the set of Fourier coefficients [20]. They encode image properties, such as color, shape, and texture. Note that different types of feature vectors may require different similarity functions.

Fig. 4(a) 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.

Definition 4. A *composite descriptor* $\hat{D}$ is a pair $(\mathscr{D}, \delta_{\mathscr{D}})$ (see Fig. 4(b)), where:

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

Fig. 4(b) illustrates the use of a composite descriptor $\hat{D}$ to compute the distance between images $\hat{I}_A$ and $\hat{I}_B$.

4. GP framework for CBIR

The present framework uses GP to combine simple descriptors. This decision stemmed from three reasons: (i) the large size of the search space for combination functions; (ii) previous success of using GP in information retrieval; and (iii) no prior work on applying GP to image retrieval.

The corresponding CBIR system can be characterized as follows. For a given large image database and a given user-defined query pattern (e.g., a query image), the system retrieves a list of images from the database which are most "similar" to the query pattern, according to a set of image properties. These properties may take into account the shape, color, and/or texture of the image objects, and are represented by simple descriptors. These simple descriptors are combined using a composite descriptor $\hat{D}_{GP}$, where $\delta_{\mathscr{D}_{GP}}$ is a mathematical expression uniquely represented as an expression tree, whose non-leaf nodes are numerical operators (see Table 1) and the leaf node set is composed of the similarity values $d_i, i=1, 2, \ldots, k$. Fig. 5 shows a possible combination (obtained through the GP framework) of the similarity values d_1, d_2, and d_3 of three simple descriptors.

The overall retrieval framework can be divided into two different approaches based on whether or not it considers the use of validation sets in the similarity function discovery process.

The use of *validation sets* aims to avoid the effect of overtraining (overfitting) [14]. Overtraining can occur when the learned or evolved model fits the particulars of the training data overly well and consequently does not generalize to new unseen examples [21].

4.1. GP framework without validation sets

Algorithm 1 illustrates the GP-based retrieval framework without considering validation sets. Initially, the population starts with individuals created randomly (step 4). This population evolves generation by generation through genetic operations (step 5). A fitness function is used to assign the fitness value for each individual (step 5.1.1). This value is used to select the best individuals (step 5.2). Next, genetic operators are applied to this population aiming to create more diverse and better performing individuals (step 5.4).

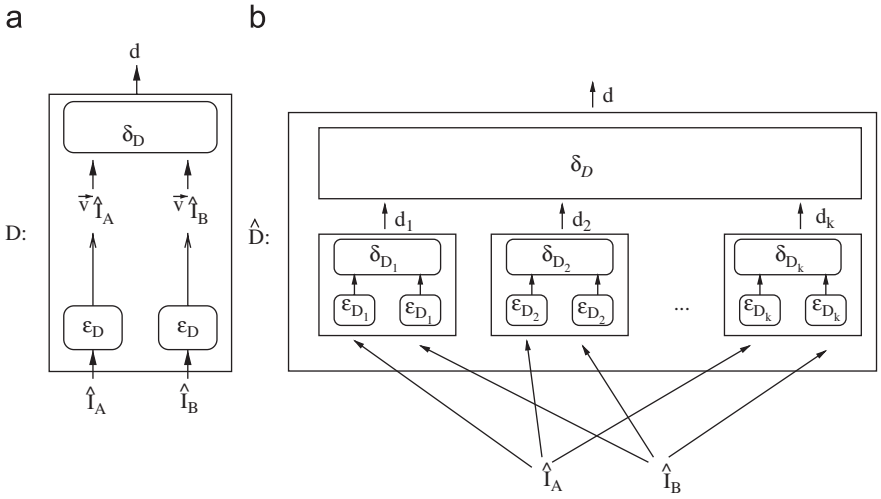

Fig. 4. (a) The use of a simple descriptor D for computing the similarity between images. (b) Composite descriptor.

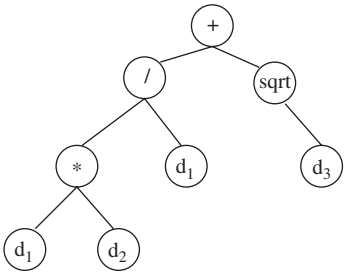

Fig. 5. Example of a GP-based similarity function represented in a tree.

The last step consists in determining the best individual to be applied to the test set (step 6). The commonest choice is the individual with the best performance in the training set (e.g., the first tree of the last generation).

Algorithm 1.
(1) Let T be a training set
(2) Let S be a set of pairs $(i, fitness_i)$, where i and $fitness_i$ are an individual and its fitness, respectively.
(3) $S \leftarrow \emptyset$
(4) $P \leftarrow$ Initial random population of individuals ("similarity trees")
(5) For each generation g of N_g generations do
  5.1. For each individual $i \in P$ do
  5.1.1. $fitness_i \leftarrow fitness(i, T)$
  5.2. Record the top N_{top} individuals and their fitness values in S_g
  5.3. $S \leftarrow S \cup S_g$
  5.4. Create a new population P by:
  5.4.1. Reproduction
  5.4.2. Crossover
  5.4.3. Mutation
(6) Apply the "best individual" in S on a test set of (query) images

4.2. GP framework with validation sets

The last step presented in the GP framework consists in determining the best individual to be applied to the test set. Since the natural choice would be the individual with best performance in the training set, it might not generalize due to overfitting during the learning phase [22]. In order to alleviate this problem, the best individuals over the generations are applied to a validation set. In that way, it is possible to select the individual that presents the best average performance in both sets: training and validation. Algorithm 2 presents the GP framework for image retrieval that considers the use of validation sets.

Note that, since the average does not ensure that the selected individual has a similar performance in both sets, it would be interesting to consider the standard deviation to correct such a bias. Formally, we apply the method described in Ref. [22] to determine the best individual: let $\bar{f}_i$ be the average performance of individual i in the training and validation sets, and $\sigma(f_i)$ be the corresponding standard deviation. The best individual is given by

$$\arg\max_i(\bar{f}_i - \sigma(f_i)) \quad (1)$$

Algorithm 2.
(1) Let T be a training set
(2) Let V be a validation set
(3) Let S be a set of pairs $(i, fitness_i)$, where i and $fitness_i$ are an individual and its fitness, respectively.
(4) $S \leftarrow \emptyset$
(5) $P \leftarrow$ Initial random population of individuals ("similarity trees")
(6) For each generation g of N_g generations do
  6.1 For each individual $i \in P$ do
  6.1.1. $fitness_i \leftarrow fitness(i, T)$
  6.2. Record the top N_{top} similarity trees and their fitness values in S_g
  6.3. $S \leftarrow S \cup S_g$
  6.4. Create a new population P by:
  6.4.1. Reproduction
  6.4.2. Crossover
  6.4.3. Mutation
(7) $F \leftarrow \emptyset$
(8) For each individual $i \in S$ do
  8.1. $F \leftarrow F \cup \{(i, fitness(i, V))\}$;
(9) BestIndividual $\leftarrow$ SelectionMethod(F, S)
(10) Apply the "best individual" on a test set of (query) images

The main difference between Algorithms 1 and 2 relies on the use of a validation set to identify appropriate individuals to be used on the test set. The individual selection method used in Algorithm 2

(step 8) considers the performance of individuals in the training set (stored in the set S) and in the validation set (stored in the set F), and selects the individual that satisfies Eq. (1).

5. Experiments

The experiments described below were carried out for shape-based descriptors. However, the proposed framework is generic and allows the combination of descriptors that encode different properties (i.e., color, texture, etc.).

5.1. Shape descriptors

Table 2 presents a brief overview of the shape descriptors used in our experiments. This list includes widely used descriptors for comparison purposes [20] and recently proposed ones [2,4]. Here, the GP framework is used to combine them in a suitable way, taking advantage of the fact that they encode different shape properties (frequency and spatial features, local and global information, etc.).

Many versions of these methods have been presented, but this work considers their conventional implementations.

5.2. GP system

The following is a detailed description of our implementation of the above framework.

- List of terminals: As pointed out in Section 4, our terminals are composed of the similarity functions defined by each descriptor presented in Section 5.1.
- Functions: The following functions were used in our implementation: $+, \times, /, sqrt$. Subtraction is not used, to avoid handling negative results. This function set is widely used in common GP experiments and is suitable to validate our ideas.
- Initial population generation: The initial set of trees, constrained to have a maximum depth of four levels, were generated by the *ramped half-and-half method* [11]. This method stipulates that half of the randomly generated trees *must* be generated by a random process which ensures all branches of the maximum initial depth. The remaining randomly generated trees require branches whose lengths do not exceed this depth. These constraints have been found to generate a good initial sample of trees [11]. Our experiments consider a population containing 600 individuals.
- Fitness functions: The fitness function plays a very important role in guiding GA/P to obtain the best solutions within a large search space. By considering our problems, a fitness function measures how effective a combination function represented by an individual tree is for ranking images. Good fitness functions will help GA/P to explore the search space more effectively and efficiently. Bad fitness functions, on the other hand, can easily make GA/P get trapped in a local optimum solution and lose the discovery power. The next paragraphs present a formal definition of the chosen fitness functions:
  *FFP*1 [13]: $F_{FFP1} = \sum_{i=1}^{|N|} r(\hat{I}_i) \times k_1 \times \ln^{-1}(i + k_2)$, where i is the image position after retrieval and $\hat{I}_i$ is the image at position i. $r(\hat{I}) \in \{0, 1\}$ is the relevance score assigned to an image, being 1 if the image is relevant and 0 otherwise. $|N|$ is the total number of retrieved images. k_1, k_2 are scaling factors. After exploratory analysis we set $k_1 = 6$ and $k_2 = 1.2$ in our experiments.
  *FFP*2 [13]: $F_{FFP2} = \sum_{i=1}^{|N|} r(\hat{I}_i) \times k_3 \times \log_{10}(1000/i)$, where k_3 is a scaling factor. We set $k_3 = 2$ in our experiments.
  *FFP*3 [13]: $F_{FFP3} = \sum_{i=1}^{|N|} r(\hat{I}_i) \times k_4^{-1} \times (e^{-k_5 \times \ln(i) + k_6} - k_7)$, where k_4, k_5, k_6, k_7 are scaling factors that are set to 3.65, 0.1, 4, and 27.32, respectively.
  *FFP*4 [13]: $F_{FFP4} = \sum_{i=1}^{|N|} r(\hat{I}_i) \times k_8 \times k_9^i$, where k_8 and k_9 are two scaling factors, which are set to 7 and 0.982, respectively.
  PAVG@10 [23]: $F_{PAVG@10} = \sum_{i=1}^{10} (r(\hat{I}_i) \times (\sum_{j=1}^{i} r(\hat{I}_j)/i))/TRel$, where *TRel* is the total number of relevant images in a collection.
  CHK [13]: $F_{CHK} = 1/|N| \sum_{i=1}^{|N|} (r(\hat{I}_i) \times \sum_{j=i}^{|N|} 1/j)$.
  LGM [13]: $F_{LGM} = (\sum_{i=1}^{|N|} (r_B(\hat{I}_i) \times 1/A((A-1)/A)^{i-1})) \times \sum_{i=1}^{|N|} r(\hat{I}_i)/|N|$, where $r_B(\hat{I}) \in \{1, -1\}$ is a function returning the relevance of image $\hat{I}$, being $+1$ if $\hat{I}$ is relevant, -1 otherwise. A is a user-defined parameter. We set A to 2.
  The fitness functions defined above were evaluated under the GP framework. PAVG@10, or average precision after 10 images are returned, is a common measure used in information retrieval evaluations [23]. Functions FFP1, FFP2, FFP3, FFP4, CHK, and LGM were used since they follow the principles of utility theory [13,24]. According to utility theory, there exists a utility function (a user's preference function) that assigns a utility value (the gained value from a user's perspective) for each item. These values vary from item to item. The item can be a book, a product, or an image, as in our case. In general, we assume *the utility of a relevant image decreases with its ranking order*. More formally, given a utility function $U(x)$, and two ranks x_1, x_2, with $x_1 < x_2$, according to this assumption, we expect the following condition to hold: $U(x_1) > U(x_2)$. The question is how to define the utility function. There are many possible functions that can be used to model this utility function satisfying the order-preserving condition given above. We decided to use FFP1, FFP2, FFP3, FFP4, CHK, and LGM, since most of them presented a good result in previous work on using GP for the ranking discovery problem [13].
- The GP operators:
  Reproduction. Reproduction copies the top $rate\_r \times P$ trees in the current generation to the next, directly without undergoing any genetic transformation. The reproduction rate, *rate_r*, is generally 0.1 or less, and P is the population size. In our case, $rate\_r = 0.05$.
  Crossover. Crossover ensures variety by creating trees that differ from their parents. For crossover, a method called *tournament selection* [11] is used. Tournament selection works by first selecting, with replacement, k (we use six) trees at random from the current generation. The two trees with the highest fitness values are paired and exchange subtrees.
  Mutation. In this case, an individual is selected, and a mutation point picked (a subtree of the individual). The subtree of the mutation point is deleted and replaced by a randomly generated subtree. Our experiments considered 25% as the percentage of individuals selected for mutation (the mutation rate).
- Stopping criterion: We stop the GP discovery process after 25 generations. We have observed that a period from 25 to 50 generations is sufficient to generate high-performing trees.

Table 2
Shape descriptors used in our experiments

Descriptor	Distance function
BAS40 [4]	OCS [7]
BAS60 [4]	OCS [7]
MS fractal dimension [2]	Euclidean distance
Fourier descriptors [20]	Euclidean distance
Moment invariants [20]	Euclidean distance

5.3. Image databases

Two different databases have been used to compare the proposed GP-based shape descriptors.

Fish shape database: This shape database contains 1000 images created by using 100 fish contours chosen randomly from the data set

available from www.ee.surrey.ac.uk/Research/VSSP/imagedb/demo.html. Since there is no semantic definition of classes for the fish contours in this database, we defined a class as consisting of 10 different manifestations of each contour by rotation and scaling. Then, the problem consists of 100 classes with 10 shapes each. In this case, each original image is considered as query image, and its manifestations are taken as relevant images.

Experiments using this database will assess the invariance of the GP-based descriptor regarding to rotation and scaling transformations.

MPEG-7 part B: This is the main part of the Core Experiment CE-Shape-1 [5]. The total number of images in the database is 1400: 70 classes of various shapes, each class with 20 images.

Two set of experiments were performed based on whether it considers or not the use of validation sets. The first set of experiments uses a two data-sets design in our experiments. In this case, we randomly split the data into training and test parts. The training set used a random 50% sample for each class. The second one includes validation set. In this case, the training, validation, and test sets used 30%, 20%, and 50% samples for each class, respectively. We considered the same test set for experiments with and without using validation sets.

6. Results

As mentioned earlier, the objective of an image retrieval system is to match database images to a user's query and place them in descending order of their predicted relevance (similarity).

6.1. Comparison criteria and baselines

We used precision after 10 images are returned as our comparison criteria.

The conducted experiments used two different samples (samples 1 and 2) for each data set, following the same distribution of training, validation, and test sets. The use of these samples aims to verify if the proposed approach is sample invariant.

We first evaluate the individual performance of each descriptor. Table 3 shows the average precision for each similarity evidence (shape descriptor). Note that the BAS60 shape descriptor presents the best result in both MPEG-7 and fish shapes collections.

6.2. GP results

The conducted experiments can be divided into two groups. The first session (Section 6.2.1) considers the BAS descriptors. In this case, it aims to verify if the GP framework is able to discover suitable similarity combination functions that outperform the baselines (each shape descriptor in isolation).

We also compare the effectiveness of our GP approach with a GA-based composite descriptor. The GA-based descriptor uses a fixed-length sequence of real numbers (weights) to indicate the importance of each descriptor. In this case, given a set of similarity functions δ_i of pre-defined descriptors, a GA-based similarity function is defined as $\delta_{GA}(\delta_1, \delta_2, \ldots, \delta_k) = w_1\delta_1 + w_2\delta_2 + \cdots + w_k\delta_k$, where w_i are weights defined by the GA framework. In our GA implementation, we considered a population of 100 individuals and 30 generations.

Table 3
Average precision after 10 images are returned, considering the evidences in isolation

Descriptor	MPEG-7 Precision@10		Fish shapes precision @10	
	Sample 1	Sample 2	Sample 1	Sample 2
BAS40	65.35	64.84	83.35	81.10
BAS60	**66.27**	**65.37**	**93.25**	**92.30**
MS fractal dimension	40.71	40.05	71.35	68.85
Fourier descriptors	20.25	20.44	24.20	23.75
Moment invariants	34.68	35.02	63.20	61.45

Table 4
Average precision after 10 images are returned, considering the GP-based descriptors using BAS40 and BAS60 descriptors for Approach 1

Descriptor	MPEG-7 precision@10		Fish shapes precision@10	
	Sample 1	Sample 2	Sample 1	Sample 2
BAS60	66.27	65.37	93.25	92.30
GP with PAVG@10	70.56 (6.47%)	69.21 (5.87%)	93.75 (0.54%)	92.75 (0.49%)
GP with FFP1	**70.92 (7.02%)**	69.59 (6.46%)	94.20 (1.02%)	93.30 (1.08%)
GP with FFP2	70.79 (6.82%)	69.76 (6.72%)	**94.30 (1.13%)**	**93.35 (1.14%)**
GP with FFP3	70.75 (6.76%)	69.44 (6.23%)	94.05 (0.86%)	93.30 (1.08%)
GP with FFP4	70.40 (6.23%)	68.97 (5.51%)	94.05 (0.86%)	93.30 (1.08%)
GP with CHK	70.73 (6.73%)	66.78 (2.16%)	94.20 (1.02%)	93.30 (1.08%)
GP with LGM	70.86 (6.93%)	**70.90 (8.46%)**	94.15 (0.97%)	93.20 (0.98%)
GA	69.37 (4.68%)	68.30 (4.48%)	93.40 (0.16%)	92.55 (0.27%)

Table 5
Average Precision after 10 images are returned, considering the GP-based descriptors using BAS40 and BAS60 descriptors for Approach 2

Descriptor	MPEG-7 precision@10		Fish shapes precision@10	
	Sample 1	Sample 2	Sample 1	Sample 2
BAS60	66.27	65.37	93.25	92.30
GP with PAVG@10	73.13 (10.35%)	72.04 (10.20%)	96.12 (3.07%)	93.04 (0.80%)
GP with FFP1	**73.36 (10.69%)**	72.40 (10.75%)	98.03 (5.12%)	96.04 (4.05%)
GP with FFP2	73.30 (10.60%)	72.30 (11.50%)	**98.03 (5.12%)**	**96.04 (4.05%)**
GP with FFP3	73.19 (10.44%)	72.08 (10.26%)	98.03 (5.12%)	96.04 (4.05%)
GP with FFP4	72.96 (9.94%)	71.30 (9.07%)	98.03 (5.12%)	95.95 (3.95%)
GP with CHK	70.08 (10.27%)	69.10 (5.71%)	98.03 (5.12%)	96.00 (4.00%)
GP with LGM	73.2 (10.45%)	**73.37 (12.24%)**	97.03 (4.05%)	95.30 (3.25%)
GA	69.37 (4.68%)	68.30 (4.48%)	93.40 (0.16%)	92.55 (0.27%)

BAS40 and BAS60 present an outstanding performance in both databases (see Table 3). Therefore, the second experiment session (Section 6.2.2) was carried out without considering these shape descriptors. The idea was to evaluate if the GP framework is able to discover suitable similarity functions even without considering good descriptors.

6.2.1. With BAS descriptors

Two experiments were carried out, named Approaches 1 and 2, which use Algorithms 1 and 2, respectively.

Tables 4 and 5 present the average precision of the GP-based descriptors, using different fitness functions. In this case, GP used the BAS descriptors.

With regard to the MPEG-7 collection, GP-based descriptors outperform the BAS60 baseline. For the first sample, FFP1 was the best fitness function, while LGM was the best for the second sample. Note also that GP presents a better result if compared to the GA-based descriptor, except for the CHK fitness function when applied to sample 2.

For the fish shapes collection, the BAS60 shape descriptor yields a high precision value, since the relevant image set is composed of similar images obtained by affine transformations (rotation and scaling). However, despite the high effectiveness of the baseline, the results based on the GP approach are better. For this collection, the best results were obtained for the FFP2 fitness function with regard to both samples (samples 1 and 2). It can be seen that the results obtained using the validation set, Approach 2, outperforms Approach 1 in both databases, due to the capacity of decreasing the effects of the overtraining problem.

Fig. 6 presents the best tree obtained by the GP framework, considering the FFP2 fitness function on sample 1 of the MPEG-7 collection. Note that the BAS60 descriptor appears in several nodes. This is an expected result since this is the best descriptor in isolation (see Table 3). Note also that this tree includes moment in-

variants and MS fractal dimension descriptors and does not consider the Fourier descriptor (the worst one in isolation—see Table 3).

```
(+ (+ (+ ContourMSFractal 0_0) 1)
  (+ (+ (+ ContourMSFractal 0_0)
      (+ (sqrt 0_5)
        (sqrt (* BAS60 0_5))))
    (* (* BAS60 BAS60)
      (* (* BAS60 BAS60)
        (* (* (* BAS60 BAS60)
            (* (+ (/ BAS60
                  (/ (+ ContourMSFractal MomentInvariants)
                    (sqrt (+ 0_0 BAS60))))
                (sqrt (+ BAS40 0_0))) BAS40))
          (+ (+ (sqrt (* (* BAS60 BAS60)
                (* (* (* BAS60 BAS60)
                    (+ (* BAS60 BAS60)
                      (* 1 BAS60))) BAS40)))
            (sqrt 0_0)) BAS60))))))
```

Fig. 6. Best GP tree using the FFP2 fitness function on sample 1 of the MPEG-7 collection.

Fig. 7 shows examples of similarity retrieval using three different query images (first column) and by taking into account the GP with FFP1, GA, and BAS60 (MPEG7 collection, sample 2). As we can see GP was able to return more similar images in the first positions, except for the last query. In this case the three methods return the same number (2) of relevant images among the first 10 position.

Fig. 8 presents the precision versus recall curves of the best GP-based descriptor, the GA-based descriptor, and the best evidence by taking into account the two samples of the MPEG-7 and fish shapes collections. Note that the GP-based descriptor has the best curve in all cases, except for sample 1 of the fish shapes data set. In this case, the GA-based descriptor outperforms the GP one for recall values lower than 0.47.

6.2.2. Without BAS descriptors

Tables 6 and 7 present the average precision of the GP framework without using BAS60 and BAS40 shape descriptors, considering the use of Algorithms 1 and 2 (with validation sets), respectively. As it

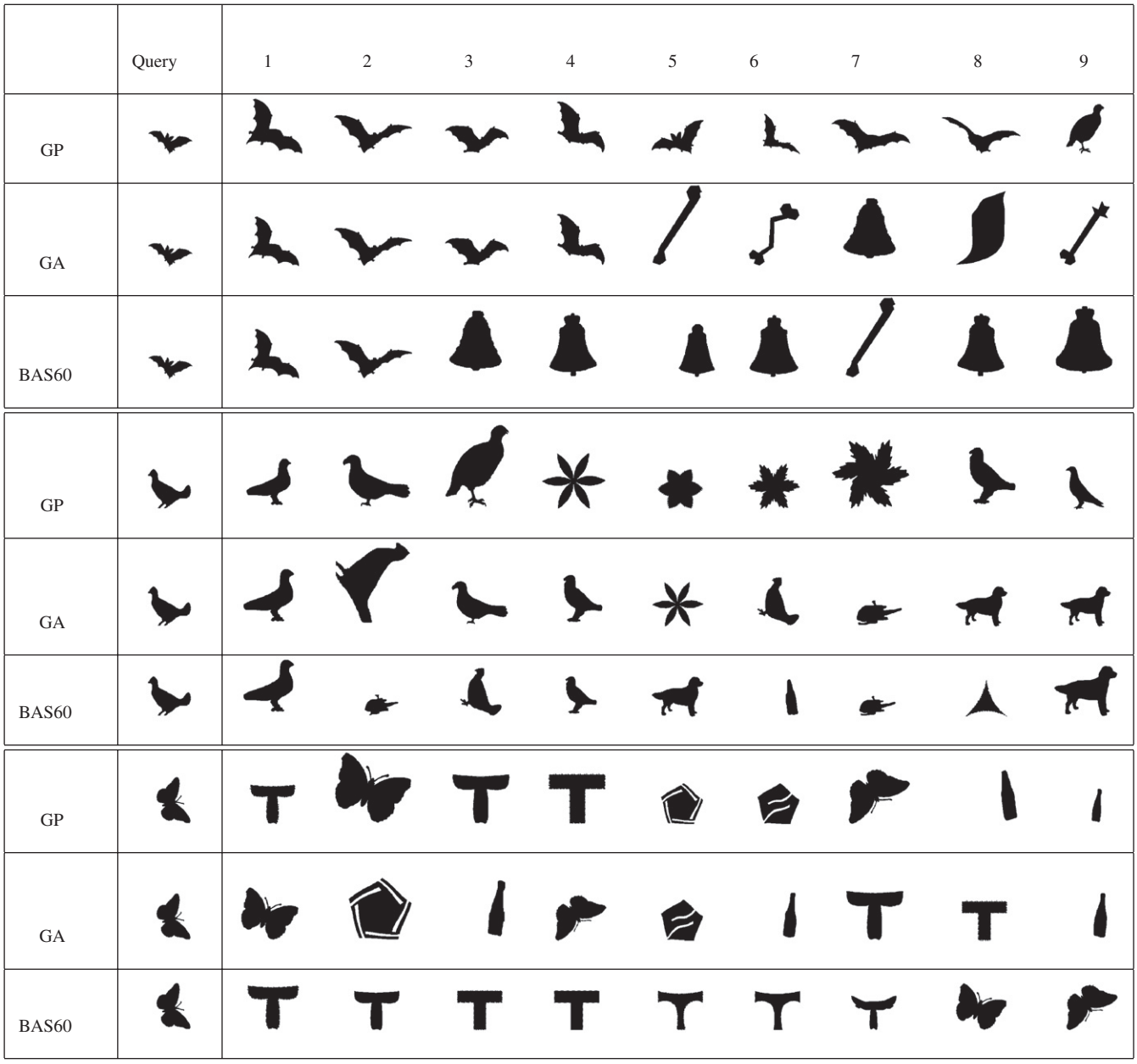

Fig. 7. Results of similarity retrieval for three query images (first column), using GP with FFP1, GA, and BAS60.

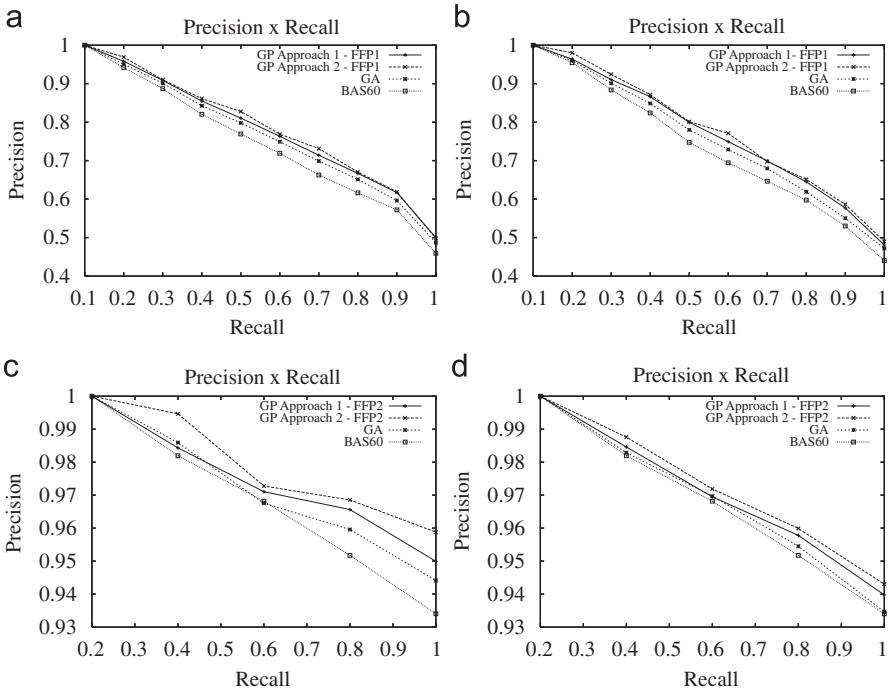

Fig. 8. Precision versus recall curves of the best GP descriptor, GA-based descriptor, and the best evidence. (a) MPEG-7—sample 1, (b) MPEG-7—sample 2, (c) fish shapes—sample 1, (d) fish shapes—sample 2.

Table 6
Average precision after 10 images are returned, considering the GP-based descriptors without BAS40 and BAS60 for Approach 1

Descriptor	MPEG-7 precision@10		Fish shapes precision@10	
	Sample 1	Sample 2	Sample 1	Sample 2
MS fractal dimension	40.71	40.05	71.35	68.85
GP with PAVG@10	46.13 (11.75%)	46.22 (15.41%)	76.70 (7.50%)	74.45 (8.13%)
GP with FFP1	46.25 (13.61%)	**46.32 (15.66%)**	77.65 (8.11%)	75.40 (9.51%)
GP with FFP2	46.63 (14.54%)	44.71 (11.64%)	76.95 (7.84%)	74.70 (8.49%)
GP with FFP3	46.73 (14.79%)	46.17 (15.28%)	76.95 (7.84%)	74.60 (8.35%)
GP with FFP4	46.46 (14.12%)	45.95 (14.73%)	76.90 (7.78%)	74.75 (8.57%)
GP with CHK	**46.94 (15.30%)**	46.14 (15.20%)	77.00 (7.92%)	74.70 (8.49%)
GP with LGM	46.13 (13.31%)	46.13 (15.18%)	**77.25 (8.27%)**	**76.25 (10.75%)**
GA	44.23 (8.64%)	42.45 (5.99%)	72.33 (1.37%)	71.55 (3.92%)

Table 7
Average precision after 10 images are returned, considering the GP-based descriptors without BAS40 and BAS60 for Approach 2

Descriptor	MPEG-7 precision @10		Fish shapes precision@10	
	Sample 1	Sample 2	Sample 1	Sample 2
MS fractal dimension	40.71	40.05	71.35	68.85
GP with PAVG@10	48.04 (18.00%)	48.04 (19.95%)	78.10 (9.46%)	77.03 (11.88%)
GP with FFP1	48.07 (18.07%)	48.06 (20.00%)	**79.17 (10.96%)**	77.10 (11.98%)
GP with FFP2	48.10 (18.15%)	47.37 (18.27%)	79.04 (10.77%)	77.16 (12.06%)
GP with FFP3	48.13 (18.22%)	47.86 (19.50%)	79.04 (10.77%)	77.30 (12.27%)
GP with FFP4	48.05 (18.02%)	47.70 (19.10%)	78.37 (9.83%)	77.43 (12.46%)
GP with CHK	**49.00 (20.36%)**	**48.20 (20.34%)**	78.95 (10.65%)	77.14 (12.04%)
GP with LGM	48.08 (18.10%)	47.90 (19.60%)	78.10 (9.45%)	**79.00 (14.74%)**
GA	47.54 (16.77%)	46.98 (17.30%)	76.54 (7.27%)	75.22 (9.25%)

can be seen, the GP framework was able to generate good similarity functions, even without using the descriptors with the best performances in isolation.

We performed a pair-wise *t*-test comparing the best GP framework with all baselines in Tables 4–7. The GP approach is statistically significant better than all the others, with $p < 0.05$, except for the GP-based descriptors using BAS40 and BAS60 descriptors for Approach 1 (Table 4). In this case, GP yields significantly better results considering $p < 0.1$.

It is worth mentioning that the training step took 30 min, on average, for the fish shapes data set (considering the two samples), running on a 3.2 GHz Pentium 4 with 2 G RAM. For the MPEG-7 data set, training took 40 min, on average.

7. Related works

7.1. Descriptors combination

In general, approaches for descriptors combination rely on assigning weights to indicate the importance of a descriptor [8–10,25]. Basically, the higher the weight the more important a descriptor is assumed to be.

The main drawback of these approaches is the fact that it is not easy to define good weight values for a given application, or even for a given user in advance. Therefore, several techniques (such as Refs. [26] and [27]) based on user feedback have been proposed to assist weight assignment for descriptors in retrieving images by content. In general, these methods are based on user judgments with regard to the relevance of previously returned images. Frome et al. [25] apply a maximal-margin formulation for learning linear combinations of elementary distances. This procedure, however, is quite different from our method: their approach learns from "triplets" of image: focal (training) image, an image labeled as "less similar" with regard to the focal image, and an image labeled as "more similar".

More recently, kernel-based approaches have been proposed for combining descriptors [28,29]. Even though they exploit image representation and distance measures, these approaches have not been applied for CBIR. In general, they try to determine appropriate kernel machines for available image classes, being, therefore, appropriate for image classification problems.

7.2. AI techniques in image processing

AI techniques, such as GA and GP, have been successfully used in several image processing applications: object recognition [30,31], object detection [12,32], image classification [33], etc.

Howard et al. [30] investigated the use of GP to support automatic ship detectors in SAR (synthetic aperture radar) imagery. They use pixel statistics associated with pixel windows as terminals. Unfortunately, they do not compare their method with any other approach. Bhanu and Lin [12] applied GP to combine image processing operations for object detection. In their framework, composite operators are represented by binary trees where internal nodes represent the pre-specified primitive operators and the leaf nodes represent the original image or primitive (pre-defined) image features. They also worked on selecting appropriate features for target detection using GA [32]. A similar approach based on GA was used by Sun et al. [31] to select features for object detection. In image classification, Agnelli et al. [33] used the GP-based framework to find out the best combination of image scalar features. They used a small database (102 images) for validation and did not compare their GP-based method with any other evolutionary approach.

8. Conclusions

We considered the problem of combining simple descriptors for content-based image retrieval. Our solution uses genetic programming (GP) to discover an effective combination function. The proposed framework was validated for shape-based image retrieval, through several experiments involving two image databases, and many simple descriptors and fitness functions.

We conclude that the new framework is flexible and powerful for the design of effective combination functions. The effectiveness results demonstrate that the GP framework can find better similarity functions than the ones obtained from the individual descriptors. Our experiments also demonstrate that the GP framework yields better results than the GA approach. In fact, even compared to outstanding baselines (BAS60 on fish shapes data set), GP was able to find out a better result.

We also evaluated a set of fitness functions based on utility theory to find the best combination function for the image search problem. The experiments showed that several of the used fitness functions are very effective in guiding the GP search. Among the various fitness functions we tested, FFP1, FFP2, and LGM are the ones we recommend for the image retrieval problem.

The GP framework for the image retrieval problem is considered "global", as it tries to find the best descriptor combination (represented as just one tree), which optimizes the number of relevant images returned. "Local" strategies, which are suitable to determine the best descriptor combination for a given class, would be useful in classification problems (e.g., Refs. [14,28]). Future work addresses this research topic. We also plan to devise an automatic mechanism to incorporate the GP-based descriptors in search engines. Another important issue that is being investigated is concerned with the development of relevance feedback (RF) approaches based on the GP framework proposed in this paper [34]. RF methods take advantage of the interactions between the user and the image search engine to increase the number of returned relevant images in a given query session [15,35–39].

Acknowledgments

This work was supported by FAPESP, CNPq, CAPES, FAPEMIG, and Microsoft Research.

References

[1] R.S. Torres, A.X. Falcão, Content-based image retrieval: theory and applications, Rev. Inf. Teór. Apl. 13 (2) (2006) 161–185.
[2] R.S. Torres, A.X. Falcão, L. da F. Costa, A graph-based approach for multiscale shape analysis, Pattern Recognition 37 (6) (2004) 1163–1174.
[3] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, R. Jain, Content-based image retrieval at the end of the years, IEEE TPAMI 22 (12) (2000) 1349–1380.
[4] N. Arica, F.T.Y. Vural, BAS: a perceptual shape descriptor based on the beam angle statistics, Pattern Recognition Lett. 24 (9–10) (2003) 1627–1639.
[5] L.J. Latecki, R. Lakamper, Shape similarity measure based on correspondence of visual parts, IEEE TPAMI 22 (10) (2000) 1185–1190.
[6] R.S. Torres, A.X. Falcão, Contour salience descriptors for effective image retrieval and analysis, Image Vision Comput. 25 (1) (2007) 3–13.
[7] Y.P. Wang, T. Pavlids, Optimal correspondence of string subsequences, IEEE TPAMI 12 (11) (1990) 1080–1087.
[8] K. Porkaew, S. Mehrotra, M. Ortega, K. Chakrabarti, Similarity search using multiple examples in MARS, in: Visual Information and Information Systems, Lecture Notes in Computer Science, vol. 1614, Springer, Amsterdam, 1999, pp. 68–75.
[9] M.S. Lew (Ed.), Principles of Visual Information Retrieval—Advances in Pattern Recognition, Springer, London/Berlin/Heidelberg, 2001.
[10] H. Shao, J.-W. Zhang, W.C. Cui, H. Zhao, Automatic feature weight assignment based on genetic algorithm for image retrieval, in: IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, 2003, pp. 731–735.
[11] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, 1992.
[12] B. Bhanu, Y. Lin, Object detection in multi-modal images using genetic programming, Appl. Soft Comput. 4 (2) (2004) 175–201.
[13] W. Fan, E.A. Fox, P. Pathak, H. Wu, The effects of fitness functions on genetic programming-based ranking discovery for web search, JASIST 55 (7) (2004) 628–636.
[14] B. Zhang, Intelligent fusion of structural and citation-based evidence for text classification, Ph.D. Thesis, Department of Computer Science, Virginia Polytechnic Institute and State University, 2006.
[15] Z. Stejić, Y. Takama, K. Hirota, Mathematical aggregation operators in image retrieval: effect on retrieval performance and role in relevance feedback, Signal Processing 85 (2) (2005) 297–324.
[16] J.H. Holland, Adaptation in Natural and Artificial Systems, MIT Press, Cambridge, MA, 1992.
[17] W.B. Langdon, Data Structures and Genetic Programming: Genetic Programming+Data Structures=Automatic Programming!, Kluwer Academic Publishers, Dordrecht, 1998.
[18] W. Banzhaf, P. Nordin, R.E. Keller, F.D. Francone, Genetic Programming—An Introduction: On the Automatic Evolution of Computer Programs and its Applications, Morgan Kaufmann, San Francisco, CA, 1998.
[19] M. Swain, D. Ballard, Color Indexing, Int. J. Computer Vision 7 (1) (1991) 11–32.
[20] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Addison-Wesley, Reading, MA, 1992.
[21] M. Seshadri, Comprehensibility, overfitting and co-evolution in genetic programming for technical trading rules, Ph.D. Thesis, Worcester Polytechnic Institute, 2003.
[22] A. Lacerda, M. Cristo, M.A. Goncalves, W. Fan, N. Ziviani, B. Ribeiro-Neto, Learning to advertise, in: ACM SIGIR, 2006, pp. 549–556.
[23] R. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval, Addison-Wesley Longman Publishing Co. Inc., Boston, MA, 1999.
[24] P.C. Fishburn, Non-Linear Preference and Utility Theory, John Hopkins University Press, Baltimore, 1988.
[25] A. Frome, Y. Singer, J. Malik, Image retrieval and classification using distance functions, in: NIPS, 2006.
[26] S.D. MacArthur, C.E. Brodley, A.C. Kak, L.S. Broderick, Interactive content-based image retrieval using relevance feedback, Comput. Vision Image Understanding 88 (2) (2002) 55–75.
[27] Y. Rui, T.S. Huang, M. Ortega, S. Mehrotra, A power tool in interactive content-based image retrieval, IEEE Tran. Circuits Syst. Video Technol. 8 (5) (1998) 644–655.
[28] Y.-Y. Lin, T.-L. Liu, C.-S. Fuh, Local ensemble kernel learning for object category recognition, in: CVPR, 2007, pp. 1–8.
[29] H. Zhang, A. Berg, M. Maire, J. Malik, Discriminative nearest neighbor classification for visual category recognition, in: CVPR, 2006, pp. 2126–2136.
[30] D. Howard, S.C. Roberts, R. Brankin, Target detection in SAR imagery by genetic programming, Adv. Eng. Software 30 (5) (1999) 303–311.
[31] Z. Sun, G. Bebis, R. Miller, Object detection using feature subset selection, Pattern Recognition 37 (11) (2004) 2165–2176.
[32] B. Bhanu, T. Lin, Genetic algorithm based feature selection in SAR images, Image Vision Comput. 21 (7) (2003) 591–608.
[33] D. Agnelli, A. Bollini, L. Lombardi, Image classification: an evolutionary approach, Pattern Recognition Lett. 26 (1–3) (2002) 303–309.
[34] C.D. Ferreira, R.S. Torres, Image retrieval with relevance feedback based on genetic programming, Technical Report IC-07-05, Institute of Computing, State University of Campinas, Feburary 2007.
[35] D. Tao, X. Tang, X. Li, Which components are important for interactive image searching, IEEE Trans. Circuits Syst. Video Technol. 18 (1) (2008) 3–11.

[36] D. Tao, X. Tang, X. Li, X. Wu, Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval, IEEE TPAMI 28 (7) (2006) 1088–1099.
[37] D. Tao, X. Li, S.J. Maybank, Negative samples analysis in relevance feedback, IEEE Trans. on Knowl. Data Eng. 19 (4), 568–580.
[38] D. Tao, X. Tang, X. Li, Y. Rui, Direct Kernel based discriminant analysis: a new content-based image retrieval relevance feedback algorithm, IEEE Trans. Multimedia 8 (4) (2006) 716–727.
[39] J. Li, N. Allinson, D. Tao, X. Li, Multitraining support vector machine for image retrieval, IEEE TIP 15 (11) (2006) 3597–3601.

About the Author—RICARDO DA SILVA TORRES received his B.Sc. in Computer Engineering from the University of Campinas, Brazil, in 2000. He got his doctorate in Computer Science from the same university in 2004. He has been Professor at Institute of Computing, University of Campinas, since 2005. His research interests include image analysis, content-based image retrieval, image databases, digital libraries, and geographic information systems.

About the Author—ALEXANDRE X. FALCÃO received his B.Sc. in Electrical Engineering (1988) from the University of Pernambuco, PE, Brazil. He has worked in image processing and analysis since 1991. In 1993, he received his M.Sc. in Electrical Engineering from the University of Campinas, SP, Brazil. During 1994-1996, he worked at the University of Pennsylvania, PA, USA, on interactive image segmentation for his doctorate. He got his doctorate in Electrical Engineering from the University of Campinas in 1996. In 1997, he developed video quality evaluation methods for Globo TV, RJ, Brazil. He has been Professor at the Institute of Computing, University of Campinas, since 1998, and his research interests include image segmentation and analysis, volume visualization, content-based image retrieval, mathematical morphology, digital TV, medical imaging applications and pattern recognition.

About the Author—MARCOS ANDRÉ GONÇALVES concluded his doctoral degree in Computer Science at Virginia Tech in 2004. He earned a Master degree from University of Campinas (UNICAMP) in 1997 and a Bachelor degree from the Federal University of Ceará (UFC) in 1995, both in Computer Science. He has published 6 book chapters, 16 journal papers, and more than 60 conference/workshop papers in the digital library, databases, and information retrieval fields.

About the Author—JOÃO PAULO PAPA received his B.Sc. in Information Systems from the State University of São Paulo, SP, Brazil. He has worked in image processing since 1999. In 2005, he received his M.Sc. in Computer Science from the Federal University of São Carlos, SP, Brazil. He has been a full Ph.D. student from University of Campinas since 2005, and his research interests include image restoration, pattern recognition and image processing.

About the Author—BAOPING ZHANG is a Software Engineer at Microsoft Corporation. She was previously a member of the Digital Library Research Laboratory at Virginia Tech. She has a Ph.D. in Computer Science from Virginia Tech, and she has worked on text classification and genetic programming.

About the Author—WEIGUO FAN is an associate professor of information systems and computer science at Virginia Tech. He received his Ph.D. in Information Systems from the University of Michigan, Ann Arbor, in July 2002, his M.Sc. in Computer Science from the National University of Singapore in 1997, and his B.E. in Information and Control Engineering from the Xi'an Jiaotong University, PR China, in 1995. His research interests focus on the design and development of novel information technologies — information retrieval, data mining, text/web mining, personalization and knowledge management techniques — to support better business information management and decision making. He has worked on the development of adaptive and intelligent text mining and web mining techniques for more advanced business intelligence applications, such as search engine ranking function discovery and optimization, text summarization, Web-based information extraction and question answering. He has published more than 80 refereed journal and conference papers. His research has appeared in many prestigious information technology journals such as IEEE Transactions on Knowledge and Data Engineering, IEEE Intelligent Systems, Information Systems, Decision Support Systems, ACM Transactions on Internet Technology, Pattern Recognition, etc., and in many leading information technology conferences such as SIGIR, WWW, CIKM, HLT, etc. His text mining research has been cited more than 500 times according to Google Scholar. His research is currently funded by four NSF grants and one PWC grant.

About the Author—EDWARD A. FOX holds a Ph.D. and M.S. in Computer Science from Cornell University, and a B.S. from M.I.T. Since 1983 he has been at Virginia Polytechnic Institute and State University (VPI&SU or Virginia Tech), where he serves as Professor of Computer Science. He directs the Digital Library Research Laboratory and the Networked Digital Library of Theses and Dissertations. He has been (co)PI on over 100 research and development projects. In addition to his courses at Virginia Tech, Dr. Fox has taught about 70 tutorials in over 25 countries. He has given over 60 keynote/banquet/international invited/distinguished speaker presentations, over 140 refereed conference/workshop papers, and over 250 additional presentations.