Incorporating multiple distance spaces in optimum-path forest classification to improve feedback-based learning

André Tavares da Silva, Jeferson Alex dos Santos, Alexandre Xavier Falcão, Ricardo da S. Torres, Léo Pini Magalhães

Abstract

In content-based image retrieval (CBIR) using feedback-based learning, the user marks the relevance of returned images and the system learns how to return more relevant images in a next iteration. In this learning process, image comparison may be based on distinct distance spaces due to multiple visual content representations. This work improves the retrieval process by incorporating multiple distance spaces in a recent method based on optimum-path forest (OPF) classification. For a given training set with relevant and irrelevant images, an optimization algorithm finds the best distance function to compare images as a combination of their distances according to different representations. Two optimization techniques are evaluated: a multi-scale parameter search (MSPS), never used before for CBIR, and a genetic programming (GP) algorithm. The combined distance function is used to project an OPF classifier and to rank images classified as relevant for the next iteration. The ranking process takes into account relevant and irrelevant representatives, previously found by the OPF classifier. Experiments show the advantages in effectiveness of the proposed approach with both optimization techniques over the same approach with single distance space and over another state-of-the-art method based on multiple distance spaces.

1. Introduction

Large image collections have increased the demand for efficient and effective information retrieval methods based on the visual content of the images. The simplest visual content representation is a feature vector, which encodes color, texture, and/or shape measures of an image. The similarity between images can be measured by the distance between their feature vectors. More complex representations are very specific for shape or texture or color, requiring special distance functions [1–5]. In this case, color, texture, and shape features are not simple measures that can be concatenated into a single feature vector. Each pair, feature extraction function and distance function, must be interpreted as an independent entity referred to as a simple descriptor. Therefore, the use of multiple simple descriptors creates the problem of dealing with multiple distance spaces. Solutions to this problem find the best combination of distance values from simple descriptors, resulting in a composite descriptor [6].

For a given query image, a content-based image retrieval (CBIR) system can rank the most relevant images based on their distances to the query image. However, a semantic gap usually occurs between this result and the user's expectation, due to the absence of relevant information in those simple/composite descriptors. In order to reduce the semantic gap problem, feedback-based learning approaches have been investigated. In these methods, the user usually indicates which images are relevant (irrelevant) within a small set of returned images and the CBIR system learns how to better rank and return more relevant images in a next iteration. This search process can be repeated until the user is satisfied. The learning process is usually focused on the selection of a more effective simple descriptor [7,8], by changing features and/or distance function, on a better composite descriptor [9,10], on changing the query strategy [11,12], or on training a more accurate pattern classifier [13–16].

In this paper we propose a method that simultaneously improves the composite descriptor, the query strategy, and the pattern classifier during the feedback-based learning process. Our method extends our previous work [17,16] by incorporating pimization techniques to compute composite descriptors. Two

This paper has been recommended for acceptance by C.-S. Li.

Corresponding author. Fax: +55 19 3521 3845.
E-mail addresses: atavare@dca.fee.unicamp.br (A.T. da Silva), jsantos@ic.unicamp.br (J.A. dos Santos), afalcac@unicamp.br (A.X. Falcão), rtorres@ic.unicamp.br (R. da S. Torres), leopini@fee.unicamp.br (L.P. Magalhães).

Article history:
Received 21 December 2010
Accepted 6 December 2011
Available online 14 December 2011

Keywords:
Content-based image retrieval
Optimum-path forest classifiers
Composite descriptor
Genetic programming
Multi-scale parameter search
Image pattern analysis

Available online 14 December 2011
Accepted 6 December 2011
Received 21 December 2010
Article history:

Article info

Computer Vision and Image Understanding 116 (2012) 510–523

Crown Copyright © 2011 Published by Elsevier Inc. All rights reserved.

1077-3142/$ - see front matter Crown Copyright © 2011 Published by Elsevier Inc. All rights reserved.
techniques are evaluated in this context: a multi-scale parameter search (MSPS) [18], never used for CBIR, and a genetic programming (GP) algorithm [6]. The MSPS method is a recent optimization approach under development. This paper presents a slightly different implementation of the algorithm in [18], which was better suited for distance combination. The choice of GP is also motivated by the successful recent works [9,10,19]. Santos et al. [9,19], for example, showed that the GP-based method improves the retrieval effectiveness, outperforming methods based on genetic algorithm.

Support vector machines (SVMs) became a popular classifier in feedback-based learning [13,14,20,15]. Our method is based on the optimum-path forest (OPF) classifier [21], because we have recently demonstrated its gain in effectiveness and efficiency over SVMs for CBIR [17,16]. The OPF classifier uses the Image Foresting Transform algorithm [22], which is a generalization of Dijkstra’s algorithm for multiple sources and more general path-cost functions in a graph. Its training set is interpreted as a complete graph weighted on the arcs by the combined distance values between nodes. Therefore, the composite descriptor should produce higher arc weights between relevant and irrelevant images, and lower arc weights within each class. The path-cost function assigns to any path its maximum arc weight, but the paths are constrained to start in a special set of representative images (called prototypes) from both classes. The prototypes compete among themselves and each prototype conquers its most closely connected images by paths of minimum cost, partitioning the graph into an optimum-path forest (classifier). The classification of database images based on OPF follows the same rule, by considering minimum-cost paths from the prototypes to the new image and assigning it to the class of the most closely connected prototype. The choice of prototypes is a key aspect in the OPF approach. First, they must “defend” their classes in order to avoid paths from prototypes of distinct classes. Since the method is choosing paths, whose maximum arc weight is minimum, the best prototypes will be the closest images between the relevant and irrelevant classes [21]. Second, the prototypes are used as new query points, changing the ranking process to sort images based on a normalized average distance to the relevant and irrelevant prototypes.

This paper is organized as follows. Section 2 provides a survey of related works and Section 3 reviews the original CBIR approach based on relevance feedback and OPF classification [17]. The method is initially described for a simple descriptor and then the optimization techniques that integrate multiple distance spaces are presented in Section 4 for composite descriptors. Exhaustive experiments involving several datasets, two state-of-the-art approaches [17,10], various simple descriptors, and the composite descriptors from the MSPS and the GP techniques are presented in Section 5. Finally, Section 6 states conclusion and discusses future work.

2. Related work

Over the past decades, several softwares for content-based image retrieval (CBIR) have been developed, such as IBM QBIC [23], UIUC MARS [24], PicHunter [25], TinEye [26], and Windsurf [27]. However, CBIR is still an open problem, as demonstrated by the ImageCLEF1 and PASCAL VOC2 challenges.

The research on CBIR initially received great influence from text retrieval [28], taking advantage of the experience reported in journals and conferences, such as TREC (Text REtrieval Conference). One of the main contributions was the Precision × Recall curve [29] that has been extensively used to evaluate the effectiveness of CBIR systems. Another contribution is also the idea of feedback-based learning, which considerably advanced the human-computer interaction process in CBIR systems [30,15,10,16]. We may divide the efforts in CBIR into two main goals of improvement: efficiency and effectiveness. Efficiency gain usually involves indexing structures [31–33], to access images in a more efficient way, and better scalability to huge image collections [34,35], by reducing the number of dimensions of the search space before indexing the data. Techniques such as PCA [36] and CSVD [37], for instance, can be used for this purpose. Effectiveness gains are essentially obtained by reducing the semantic gap problem.

Our method can take advantage of techniques that improve efficiency, but the focus of this work is on effectiveness gain. In this context, the initial studies proposed simple descriptors [38,13–41], with local [42,41] and/or global [43,44,1–3] image features, and strategies to change the query during the search, by moving the query point in the feature space [11] and by creating multiple query points [45,46,12]. However, multiple image representations may require distinct distance functions [1–3,47,4,5]. In this context, the simplest approach is to compute a composite descriptor as a fixed combination of the distance values from multiple representations [48].

Many recent works, however, have focused on feedback-based learning to either elaborate composite descriptors that improve rank- ing of relevant images [6,49–51,10] or to design pattern classifiers [15,52,17,16] that select images from the database as the best candidates to be relevant images. These methods take advantage of the user feedback about the relevance of the images in order to improve the composite descriptor or the pattern classifier at each iteration. Classifier-based methods can also be divided into those that return the most relevant images at each iteration [53,52,17] and those that return the most informative images (i.e., images that are difficult to be classified) during a few iterations [25,13–16], postponing the most relevant ones to the last iteration. While the strategy of the former approaches is greedy, aiming the most relevant images at each iteration, the strategy of the latter methods is planned, in the sense that the user has to decide in which iteration the system will return the most relevant images [16]. Planned approaches are widely known as active learning methods. However, active learning could have a more general interpretation, being accepted as a synonymous for feedback-based learning, given that the learner (the classifier) is active in the learning process, by trimming images from the database to create a set of candidate ones for ranking.

In this work we are exploiting the use of composite descriptor and pattern classifier at the same time. Methods based on composite descriptors tend to be more effective than approaches based on simple descriptors in most applications [49,50,10]. In addition, other works have demonstrated that the classification of the relevant candidates before ranking is a good strategy to increase the effectiveness of the CBIR system [15,17]. We have proposed in [17] a greedy approach based on optimum-path forest (OPF) classifier and demonstrated its gain in effectiveness with respect to the first active learning method based on support vector machines (SVMs) [13,14]. It is important to note that it is not possible to combine distance functions with SVMs, because this classifier is based on support (feature) vectors and they will not be the same for distinct data representations (samples using different features and distance functions). For this reason, SVM-based methods typically combine features by fusing or concatenating the vectors [54,55,15,56]. Due to the large size of the resulting vectors, some studies also suggest the reduction of dimensionality by using techniques such as PCA [36,56]. Other works try to reduce the number of vectors by feature selection techniques [54,57,8,7]. Some works also allow the user to assign a finer gradation of relevancy, such as somewhat relevant, not sure, and somewhat irrelevant [58,59]. The OPF classifier is a multi-class classifier and can handle multiple gradation levels. Although we will present the method for the case of relevant/irrelevant images, its extension to multiple grades is straightforward.

---

2 http://pascal-clues.soton.ac.uk/challenges/VOC/ (As of 10/10/2011).
3. Relevance feedback with optimum-path forest

Fig. 1 shows the overview of our approach for feedback-based learning using the OPF classifier and a simple descriptor \( D = \{v, d\} \), being \( v \) the feature extraction function and \( d(s, t) \) the distance function between two image representations \([\text{e.g., } d(s, t) = \| \tilde{v}(t) - \tilde{v}(s) \|]\) [60]. Its extension to composite descriptor is explained in Section 4. At the first iteration, we have a simple search by similarity where, for a given image database \( X \), the user specifies a query image \( q \) and the system simply ranks the \( N \) closest images \( t \in X \) in the increasing order of \( d(q, t) \). We aim to return a list \( X' \) with the \( M \) most relevant images in \( X \) with respect to that query image \( q \). However, due to the limitation of the descriptor \((v, d)\) in representing the user’s expectation (semantic gap), the list \( X' \) contains relevant and irrelevant images according to the user’s opinion. To reduce the semantic gap problem, the system asks for the user’s feedback about the relevance of the returned images during a few iterations. The user indicates which images are relevant (or irrelevant) in \( X' \), forming a labeled training set \( T \) which gains new elements at each iteration of relevance feedback by \( T \leftarrow T \cup X' \).

Our feedback-based learning process, named \( \text{OPF}_{\text{BF}} \), uses the set \( T \) to project a new OPF classifier at each iteration of the relevance feedback loop. This process consists of first estimating prototypes as the closest images from distinct classes in \( T \), forming the sets \( S_T \subseteq T \) and \( S_I \subseteq T \) of relevant and irrelevant prototypes, respectively. The training process considers a complete graph, whose nodes are all elements in \( T \) and arcs \( (s, t) \) between images \( s \) and \( t \) are weighted by \( d(s, t) \). Every path in the graph has a cost and minimum-cost paths are computed from \( S_T \cup S_I \) to each node \( t \in T \), such that the classifier is an optimum-path forest rooted in \( S_T \cap S_I \) (Section 3.1). In this forest, the nodes \( t \in T \setminus (S_T \cup S_I) \) are conquered and labeled (in the same class as relevant/irrelevant) by the prototype in \( S_T \cup S_I \), which offers the minimum-cost path with terminus \( t \). Afterward, this classifier evaluates the images in \( Z \setminus T \), by computing the cost of the optimum path from \( S_T \cup S_I \) to each node \( t \in Z \setminus T \) in an incremental way, and inserts \( t \) in a reduced set \( Y \subseteq Z \setminus T \) of relevant candidates whenever the optimum path is rooted in \( S_T \).

The system then returns a new list \( X' \subseteq Y \) with the \( N \) closest images in the increasing order of a normalized mean distance \( d(t, S_T, S_I) \) between \( t \) and the two sets of prototypes (Eq. (1)) [17,16]. Therefore, each iteration may produce distinct sets of prototypes, changing the query/ranking strategy.

\[
d(t, S_T, S_I) = \frac{d(t, s_T)}{d(t, S_T) + d(t, S_I)},
\]

\[
d(t, S_T) = \frac{1}{|S_T|} \sum_{s \in S_T} d(s, t),
\]

\[
d(t, S_I) = \frac{1}{|S_I|} \sum_{s \in S_I} d(s, t).
\]

Again, the user may indicate relevant and irrelevant images in \( X' \) and a new training set is created by \( T \leftarrow T \cup X' \) to redesign an improved classifier. This process is repeated until the user is satisfied.

Next sections detail the optimum-path forest training and classification processes and the feedback-based learning algorithm, respectively.

### 3.1. Training and classification by optimum-path forest

The OPF classifier aims at exploiting possible connectivities between samples and representative elements (prototypes) of each class in a given distance space to improve classification. Its decisions are not usually taken based on direct distances between samples, but on optimal sequences of distance values between samples that are most closely connected to some prototype. For a given training set \( T \), samples may be connected based on different adjacency rules. In this work we adopted a complete graph, as shown in Fig. 2a, whose nodes are all images in \( T \). In order to decide the class of any sample \( t \), we are interested in finding the minimum-cost path from a set of prototypes. Therefore, a path \( \pi_t \) in this graph with terminus \( t \) is a sequence \( (t_1, t_2, \ldots, t_n = t) \) of distinct nodes, such that the root \( R(\pi_t) = t_1 \) is a prototype. The value \( c(\pi_t) \) of a path is defined as the maximum arc weight \( d(t_i, t_{i+1}) \) along it:

\[
c(\pi_t) = \max_{i=1, \ldots, n-1} \{d(t_i, t_{i+1})\}.
\]

By definition, any other path whose root is not a prototype will be assigned to infinity cost and a trivial path formed by a single node (prototype) will have cost equal to zero. A path \( \pi_t \) will be optimum whenever \( c(\pi_t) \leq c(\tau_t) \) for any other path \( \tau_t \) with the same terminus \( t \), irrespective to their root nodes. Thus, the prototypes compete among themselves and the OPF classifier always decides the class of \( t \) as being the one of its most closely connected prototype, according to Eq. (4). For given sets \( S_T \) and \( S_I \) of relevant and irrelevant prototypes, this competition creates a minimum-cost path forest (optimum partition) rooted at \( S_T \cup S_I \) (Fig. 2b).

Therefore, training consists of finding the prototypes from all classes and computing an optimum-path forest (classifier) in the training set \( T \). The minimization of the path costs \( c(\pi_t) \) to each training node \( t \) tends to choose the closest nodes to form the links along the optimum paths. By choosing prototypes as the closest samples between distinct classes (images with black borders in Fig. 2b), they will protect other samples of the same class to be conquered by prototypes from other classes. Let \( \tilde{I}(t) \) be the class of image \( t \in T \), which may be relevant or irrelevant. The prototypes are obtained by computing a minimum spanning tree (MST) in the complete graph with nodes in \( T \) [61]. For each arc \((s, t)\) in the MST, if \( \tilde{s}(s) \neq \tilde{I}(t) \) then \( s \) and \( t \) are marked as prototypes. If \( \tilde{s}(s) \) is relevant and \( \tilde{I}(t) \) is irrelevant, then \( s \) is inserted in \( S_T \) and \( t \) is inserted in \( S_I \). Similarly, it is done...
in the other way around. The optimum-path forest $P$ in Fig. 2b is then an acyclic graph which stores all optimum paths in a predecessor map, i.e., a function which assigns to every node $t \in T \setminus S_R \cup S_I$ its predecessor node $P(t)$ in the optimum path from $S_R \cup S_P$. This forest can be computed by Algorithm 1, customized for path-cost function $c$ [22,21]. This algorithm outputs for each node $t \in T$, the minimum-path cost $C(t)$ of the optimum path with terminus $t$, its root node $R(t) \in S_R \cup S_I$, and the predecessor $P(t)$ in that optimum path. A list $T'$ of the training nodes in the increasing order of cost $C(t)$ is also output to speedup classification.

Algorithm 1. OPF Algorithm

\begin{algorithm}
\begin{algorithmic}
\State \textbf{Input:} Training set $T$, the sets of prototypes $S_R \subset T$ and $S_I \subset T$, and a descriptor $(x,d)$.
\State \textbf{Output:} Optimum-path forest $P$ (predecessor map), path-cost map $C$, root map $R$, and the ordered list $T'$ of training nodes.
\State \textbf{Auxiliary:} Priority queue $Q$ and cost variable $\text{cst}$.
\For {each $s$ $\in T \setminus S_R \cup S_I$}
\State Set $C(s) = +\infty$
\EndFor
\For {each $s$ $\in S_R \cup S_I$}
\State Set $C(s) = 0, P(s) = \text{nil}$, $R(s) = s$, and insert $s$ into $Q$.
\EndFor
\While{$Q$ is not empty}
\State Remove from $Q$ a node $s$ with minimum $C(s)$ and insert $s$ in $T'$.
\For {each $t$ $\in T'$ such that $C(t) > C(s)$}
\State Compute $\text{cst} = \max\{C(s), d(s,t)\}$.
\If {$\text{cst} < C(t)$}
\State $P(t) \leftarrow s$, $R(t) \leftarrow R(s)$ and $C(t) \leftarrow \text{cst}$.
\Else
\State $C(t) = +\infty$
\EndIf
\State Insert $t$ in $Q$.
\EndFor
\State Update the position of $t$ in $Q$.
\EndWhile
\end{algorithmic}
\end{algorithm}

The algorithm follows the dynamic programming scheme [22]. Lines 1–7 initialize the cost, predecessor, and root maps, forcing the optimum paths to start in $S_R \cup S_I$, and insert the roots in $Q$. The main loop computes an optimum path from $S_R \cup S_I$ to every node $s \in T$ in a non-decreasing order of cost (Lines 8–21). At each iteration, a path of minimum cost $C(s)$ is obtained in $P$ when we remove its last node $s$ from $Q$, preserving its order in $T'$ (Line 9). The rest of the lines evaluate if the path that reaches an adjacent node $t \neq s$ through $s$ is cheaper than the current path with terminus $t$ and update $Q, C(t), R(t),$ and $P(t)$ accordingly. The test $C(t) > C(s)$ in Line 10 avoids to visit nodes $t$ that will never be updated from nodes $s$. The test $C(t) = +\infty$ in Line 14 identifies when a node $t$ is being visited for the first time.

In the classification phase (Fig. 2c), for every sample $t \in Z \setminus T$, the optimum path with terminus $t$ can be easily identified by finding which training node $s' \in T'$ provides the minimum value in

$$C(t) = \min\{\max\{C(s), d(s,t)\}\}. \quad (5)$$

Node $s'$ is the predecessor $P(t)$ in the optimum path with terminus $t$ and the image $t$ is classified (Fig. 2d) as $\mathcal{I}(R(s'))$.

The role of $T'$, sorted by the cost $C(s)$ computed by Algorithm 1, is to speed up the evaluation of Eq. (5). The search for the minimum value can be halted when the cost for a node $p$ whose position in $T'$ exceeds the position of $s$ is greater then the cost computed previously ($C(p) > \max\{C(s), d(s,t)\}$) [62]. Thus, it is not necessary to compare each sample $t \in Z \setminus T$ with all elements of the set $T$.

3.2. Feedback-based learning algorithm

Algorithm 2 presents the feedback-based learning using OPF classification [17] (as illustrated in Fig. 1). It returns a list $R \subset Z$ of images in the decreasing order of relevance.

In Line 1, the output list $R$ and the training set $T$ are initialized to empty sets. For a given query image $q$ and image database $Z$, the algorithm first returns a list $X$ with the $N$ closest images $t \in Z$ in the increasing order of $d(q,t)$ (Line 2). The learning process is executed in the main loop (Lines 3–9). In Line 4, the user marks the relevant images in $X$, and the remaining images are considered as irrelevant, or vice versa. This essentially creates a training set $T = T \cup X'$ where each image $t \in T$ has a relevance label $\mathcal{I}(t)$. The relevant images of $X$ are inserted in the output list $R$ (Line 5). The OPF training is computed in Line 6 (Algorithm 1). It results sets $S_R \subset T$ and $S_I \subset T$ of relevant and irrelevant prototypes; the OPF attributes, predecessor $P(t)$, cost $C(t)$, and root $R(t)$ for each $t \in T$; and set $T'$ of training images in the increasing order of optimum cost $C(t)$ for faster classification. In Line 7, OPF is used to classify images in $Z \setminus T$ using Eq. (5), forming a set $Y$ with the images classified as relevant. A new list $X'$ with the $N$ closest images $t \in Y$ is created in the increasing order of a normalized mean distance $d(t, S_R, S_I)$ between $t$ and the two sets of prototypes (Eq. (4)).

Algorithm 2. OPF$_{RS}$ Algorithm

\begin{algorithm}
\begin{algorithmic}
\State \textbf{Input:} A query image $q$, a descriptor $(x,d)$, the number $N$ of returned images per iteration, and an image database $Z$.
\State \textbf{Output:} A list $R \subset Z$ of retrieved images in the decreasing order of relevance.
\State \textbf{Auxiliary Set} $T$ of training images, maps $(R,P,C,T')$ of the OPF classifier, sets $S_R \subset T$ and $S_I \subset T$ of prototypes, set $Y \subset Z \setminus T$ of images classified as relevant, and ordered list $X' \subset Y$ of $N$ returned images per iteration.
\State Set $R \leftarrow \emptyset$ and $T \leftarrow \emptyset$.
\State Compute a list $X$ with the $N$ closest images $t \in Z$ in the increasing order of $d(q,t)$.
\While {the user is not satisfied}
\State The user marks relevant (irrelevant) images, forming a training set $T = T \cup X'$ where each image has a relevance label $\mathcal{I}(t)$.
\State Compute sets $S_R$ and $S_I$ of prototypes in $T$ (Section 3.1) and execute $(R,P,C,T') \leftarrow$ OPF$(T, S_R, S_I, \mathcal{I}, v, d)$ (Algorithm 1).
\State Classify images in $Z \setminus T$ using $(R,P,C,T')$ and $\mathcal{I}(t)$ for $t \in S_R \cup S_I$ (Eq. (5)), and create set $Y$ with the relevant candidates.
\State Compute a list $X'$ with the $N$ closest images $t \in Y$ in the increasing order of $d(t, S_R, S_I)$ (Eq. (1))
\EndWhile
\Return $R \leftarrow R \cup X'$.
\end{algorithmic}
\end{algorithm}

The main loop repeats until the user is satisfied and the retrieved images in $R$ are all relevant images together with the last set $X'$ in the increasing order of $d(t, S_R, S_I)$.
4. Extension to multiple descriptors

Now consider a set $D = \{D_1, D_2, ..., D_n\}$ of simple descriptors such that a distance $d_i(s, t)$, $i = 1, 2, ..., n$, between any given pair of images $s, t \in \mathcal{Z}$ can be computed as a function of their respective representations $\tilde{v}_i(s)$ and $\tilde{v}_i(t)$.

We wish to find the best combination of simple descriptors in order to retrieve relevant images from $\mathcal{Z}$ with respect to a given query image and user. This combination is found through a composite descriptor [60] that integrates the respective distance spaces. That is, a composite descriptor $D^\ast$ is a pair $(\mathcal{D}, d^\ast_D)$, where $d^\ast_D$ is a function that best combines the distance values computed by each descriptor into a final distance value $d^\ast(s, t)$ (Fig. 3).

Observe that the distance values from different descriptors may diverge significantly in scale. Thus, it is important that all values be normalized between 0 and 1. A Gaussian normalization [38] can be employed for that.

At each iteration of the feedback-based learning process, a training set $T$ with relevant and irrelevant images is used to choose the best combination function using some optimization algorithm. For a given $T$ and set $\mathcal{D}$ of descriptors, the optimization process essentially evaluates candidates $d^\ast_D$ according to some criterion function. The criterion function should measure the ability of the system in ranking the most relevant images first. We measure this property by averaging the FFP4 [63] values (Eq. (6)) with respect to each relevant image in $T_R$, rather than using only the query image. As introduced in a previous work [6], this approach has proven to be more effective than the simplest method based on the query image only. Let $T_R$ be the subset of relevant images in $T$ and $T_u$ be a set of training images $t \in T$ in their increasing order of distance $\delta_D(t, u)$ with respect to a given relevant image $u \in T_R$, our criterion function $F(T, \mathcal{D}, d^\ast_D)$ is defined as

$$F(T, \mathcal{D}, d^\ast_D) = \frac{1}{|T_R|} \sum_{u \in T_R} \sum_{k=1}^{|T_u|} \lambda_k 0.982^k,$$

where $\lambda_k \in \{0, 1\}$ indicates the user’s opinion about the relevance of each image at every position $k$ in $T_u$, as either relevant ($\lambda_k = 1$) or irrelevant ($\lambda_k = 0$). The constant 0.982 was experimentally determined in [63].

In Algorithm 2, the complete graph used to compute the OPF classifier has its arcs $(s, t)$ weighted by the best combined distance $d^\ast(s, t)$ that results from the optimization process (Fig. 3).
The combination function $\delta D^*$ is also used to compute the normalized distance $\delta(t, S_t, S_i)$ in that algorithm. Fig. 4 details the learning and retrieval process from our feedback-based learning approach. First a composite descriptor is defined using combination techniques such as MSPS or GP that will be used by the OPF classifier.

In the following sections we present two optimization techniques to compute composite descriptors: the multi-scale parameter search (MSPS) and a genetic programming algorithm (GP).

4.1. Composite descriptors using multi-scale parameter search

In the multi-scale parameter search (MSPS), the combination function $\delta D$ is represented by a same equation and the parameters of this function must be optimized [18]. In this paper, this equation is defined by

$$\delta D(s, t) = \sum_{i=1}^{n} d_i(s, t)$$  

(7)

where $\theta = (\theta_1, \theta_2, \ldots, \theta_n)$ is its parameter vector, being $0 \leq \theta_i \leq 2$. The parameters may give distinct non-linear emphases to their corresponding descriptors. Given that the values of $F(T, D, \delta D)$ during optimization will depend only on the choice of $\theta$, we will denote $F(T, D, \delta D)$ by $F(T, D, \theta)$ for clarity. From any given state $\theta = (\theta_1, \theta_2, \ldots, \theta_n)$, the idea is to find the best displacement vector $\Lambda^*$, by exploiting multiple scales of the parameter space, and update the parameter vector into a next value $\theta \rightarrow \theta + \Lambda^*$, repeating this process until a maximum $F(T, D, \theta^*)$. In order to avoid local maximum, the method perturbs $\theta$ on each parameter axis $i = 1, 2, \ldots, n$ and with $m$ displacement scales $j = 1, 2, \ldots, m$ along each axis. At each iteration, it evaluates $F(T, D, \theta + \Lambda)$ for the displacement vectors $\Lambda$ that result from all perturbations on each axis, separately, all scales, and the resulting vectors from each scale. This method works as follows.

Let $0 < \Lambda_j \leq 1$ be a positive displacement along the parameter axis $i$ for a scale $j$. The method takes into account the following displacements:

- the best perturbation along each parameter axis $i$, $\Lambda_j = (0, \ldots, \Lambda_j, \ldots, 0)$ for $\Lambda_j \in (\Lambda_j, 0, -\Lambda_j)$, such that $F(T, D, \theta + \Lambda_j) = \max \{F(T, D, \theta + \Lambda_j, 1), F(T, D, \theta), F(T, D, \theta - \Lambda_j)\}$

$$F(T, D, \theta + \Lambda_j) = \max \{F(T, D, \theta + \Lambda_j), F(T, D, \theta), F(T, D, \theta - \Lambda_j)\}$$  

(8)

- and the resulting vectors $\Lambda = \sum_{j=1}^{m} \Lambda_j j = 1, 2, \ldots, m$.

Hence, the choice of $\Lambda^*$ can be expressed by:

$$F(T, D, \theta + \Lambda^*) = \max \begin{cases} F(T, D, \theta + \Lambda_j) & \text{for } i = 1, 2, \ldots, n \\ F(T, D, \theta + \Lambda_j) & \text{for } j = 1, 2, \ldots, m. \end{cases}$$

(9)

Algorithm 3. MSPS Algorithm

Input: Training set $T$, set $D$ of simple descriptors, and displacements $\Lambda_i$, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$.
Output: The best parameter vector $\theta^*$.

Auxiliary: Displacement vectors $\Lambda, \Lambda^*$ and $\Lambda$, parameter vectors $\theta^*$ and $\theta$, and variables $i, j, V_0, V^*, V$, and $\Lambda^*$.

1. $\theta \leftarrow (1, \ldots , 1)$
2. $V^* \leftarrow F(T, D, \theta)$ and $\theta^* \leftarrow \theta$
3. repeat
4. $V_0 \leftarrow V^*$ and $\theta \leftarrow \theta^*$
5. for $j = 1$ to $m$
6. $\Lambda \leftarrow (0, \ldots , 0)$.
7. for $i = 1$ to $n$
8. $\Lambda^* \leftarrow (0, \ldots , 0)$.
9. $V^* \leftarrow F(T, D, \theta + \Lambda) + V^* = F(T, D, \theta - \Lambda)$;
10. if $V^* > V$ then $V \leftarrow V^*$ and $\Lambda^* = -\Lambda$;
11. if $V > V^*$ then $V \leftarrow V^*$ and $\Lambda^* = -\Lambda$;
12. if $V > V^*$ then $\theta \rightarrow \theta + \Lambda^*$ and $V^* = V$;
13. $\Lambda \leftarrow \Lambda + \Lambda^*$;
14. end
15. $V \leftarrow F(T, D, \theta + \Lambda)$;
16. if $V > V^*$ then $V^* \leftarrow V$ and $\theta^* \leftarrow \theta + \Lambda$;
17. end
18. until $V^* > V_0$
19. Return $\theta^*$.

Algorithm 3 illustrates the computation of the best parameter vector $\theta^*$ in Eq. (7). In this work, we fixed the displacements $\Lambda_i$ as $0.001, 0.01, 0.12, 0.45$, and $1.0$ along $m = 5$ scales for every parameter $i = 1, 2, \ldots, n$. These displacements could have also be created by some increasing function. In Line 1, vector $\theta$ is initialized with the same weight to all descriptors. Lines 4–16 search for the $\theta^*$ that maximizes Eq. (6). If there is a $\Lambda$ (Line 8), tested for positive and negative displacements, that increases the current criterion value, it is stored in $\Lambda^*$ (Lines 10–11). Lines 10 and 11 aim at comparing $V_0$ with $V^*$ and $V$. If both $V^*$ and $V$ are greater than $V$, Line 10 will substitute $V$ by $V^*$ and so Line 11 will be comparing $V^*$ with $V$ in order to choose the highest one. The best displacement vector, $\Lambda^*$, is used in Lines 12 and 13 to update the parameter vector and to compose the combination of the best displacements for each scale, respectively. Lines 15 and 16 test if $\Lambda$ generates a better combination function and update the parameter vector. The algorithm stops when no combination better than $\theta$ is found, returning in Line 19 the last parameter vector $\theta^*$, which is then used in Eq. (7) to define the final composite descriptor for feedback-based learning using the OPF classifier.

4.2. Composite descriptors using genetic programming

In the genetic programming (GP) approach [64], the combination function $\delta D$ is a tree of mathematical operations applied to the distance values that result from the simple descriptors (Fig. 5). Leaf nodes are represented by these distance values, while the mathematical operations are defined in the internal nodes. Fig. 5 exemplifies an individual with three distinct descriptors using the set $\{+, -, /, sqrt\}$ of operators as internal nodes. The best
combination function $d^*(s,t)$ is $d^*(s,t) = \sqrt{d_1(s,t) + d_2(s,t)}$ in this example.

The GP approach [6] searches for the best combination function by evolving a population $\mathcal{P}$ of $n_p$ candidate individuals per iteration during $n_g$ generations. At each iteration, all individuals $d_\mathcal{D}_i, i = 1, 2, \ldots, n_p$, are evaluated by function $F(T, D, \delta d_i)$ (Eq. (6)). The best individuals are identified and modified by applying genetic transformations, such as reproduction, mutation, and crossover. The reproduction operator copies them to the next generation. Mutation randomly selects a point in the GP tree of an individual and replaces the subtree of that point with a new randomly generated subtree. The crossover operator swaps a subtree of one parent with a subtree of another [64].

Algorithm 4 describes the search for the best individual $d^*$ during $n_g$ generations, starting from an initial population $\mathcal{P}$ with $n_p$ individuals. The population starts with individuals created randomly (Line 1). This population evolves generation by generation through genetic operations (Lines 2–9). The criterion function $F(T, D, \delta d_i)$ (Eq. (6)) is used to assign the fitness value for each individual (Line 5). This value is used to select the best individuals (Line 7). Next, genetic operators are applied to this population in order to create more diverse and better performing individuals (Line 8). The last step consists of determining the best individual (Line 10). The commonest choice is the individual with the best performance in the last generation of $\mathcal{P}$.

**Algorithm 4. GP Algorithm**

```plaintext
Input: A training set $T$, a set $\mathcal{D}$ of descriptors, the size $n_p$ of the population, the number $n_g$ of generations, and percentage of reproduction, mutation and crossover.

Output: The best individual $d^*$ (a tree structure).

Auxiliary: Set $\mathcal{P}$ (population) of $n_p$ individuals $\delta d_i$, a set $\mathcal{A}$ of pairs $(\delta d_i, F(T, D, \delta d_i))$, and variables $i$ and $g$.

1. $\mathcal{A} \leftarrow \{ \delta d_i \} $; initial random population of $n_p$ individuals;
2. $A \leftarrow \emptyset$;
3. for each generation $g = 1, 2, \ldots, n_g$ do
4. for each individual $\delta d_i \in \mathcal{P}$, $i = 1, 2, \ldots, n_p$ do
5. insert $A \leftarrow A \cup (\delta d_i, F(T, D, \delta d_i))$;
6. end
7. sort $A$ in the decreasing order of $F(T, D, \delta d_i)$;
8. create a new population $\mathcal{P}$ of size $n_p$ by reproduction, crossover and mutation among the best individuals in $A$;
9. end
10. return the best individual $d^*$ in $\mathcal{A}$ with the highest value of $F(T, D, \delta d^*)$.
```

5. Experiments and results

We call OPFGP and OPF$_{\text{MSPS}}$ the feedback-based learning process using OPF classification and the optimization techniques MSPS and GP, respectively.

Table 1 shows the values used for GP parameters [64] used in the OPF$_{\text{CP}}$ feedback-based learning method. Table 2 presents the population size and the number of generations of GP in the OPF$_{\text{CP}}$ method for each dataset.

For each image database, we simulate the user behavior by using each image as initial query point and marking the relevant points (images from the same class of the query) among 30 returned images per iteration.

We compare the effectiveness of OPF$_{\text{MSPS}}$ and OPF$_{\text{CP}}$ against OPF$_{\text{PRF}}$, as proposed in [17] and described in Section 3 for simple descriptors (the best one for each dataset), and another approach, named GP$^*$. GP$^*$ is a feedback-based learning method which uses genetic programming to compute the best composite descriptor given a training set composed only by relevant (positive)
Fig. 6. Mean $P \times R$ curves in (a) Coil-100 and (b) Corel databases after 3rd iteration.

Fig. 7. Mean $P \times R$ curves in (a) Coil-100 and (b) Corel databases 5th iteration.

Fig. 8. Mean $P \times R$ curves in (a) Coil-100 and (b) Corel databases after 8th iteration.
Fig. 9. Mean P × R curves in (a) ETH-80 and (b) MPEG7 databases after 3rd iteration.

Fig. 10. Mean P × R curves in (a) ETH-80 and (b) MPEG7 databases after 5th iteration.

Fig. 11. Mean P × R curves in (a) ETH-80 and (b) MPEG7 databases after 8th iteration.
GP+ ranks the database images by a voting scheme among the best individuals of the last population. In this voting scheme, selected individuals vote for $N$ candidate images. The most voted images are showed to the user.

GP+ outperforms several other feedback-based learning techniques [38,11,13,65,66] as well as the ranking process based on the best individual $d$ obtained by Algorithm 4 with no OPF classification. In our experiments, the values for the GP parameters are the same as those used for the Corel Collection in [10].

We use the precision-recall ($P \times R$) curve for 3, 5, and 8 iterations to measure effectiveness. We use the $P \times R$ curve considering the results obtained at the last RF iteration. The higher the curve, the better the method. The first comparison evaluates the importance of having a composite descriptor while the second comparison evaluates the importance of having a pattern classifier.

We also present the percentage of relevant images retrieved to the user given a number of relevance feedback iterations of OPFMSPS, OPFGP, and GP+. This curve allows us to evaluate how well the number of retrieved relevant images grows over iterations. Both curves show the average result considering all database images computed using the leave-one-out cross-validation. Since the curves use the entire image database $\mathcal{Z}$, Line 10 of Algorithm 2 is replaced by: Return $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{Y}$ in their increasing order of $d(t, x, s_i)$.

The experiments used six heterogeneous image databases, representing different challenges for CBIR.

- Coil-100 [67]: It is an image database of 100 objects. Pictures of each object were taken in 72 different poses (total of 7,200 images).
- Corel [68]: This database is a collection with 200,000 images from the Corel Gallery Magic–Stock Photo Library 2. We use a subset of 3,906 samples, pre-classified into 85 classes. These classes have different number of images varying from 7 to 98 images each.
- ETH-80 [69]: This database is available in the project COGVIS. The project includes images of objects from 8 basic-level categories performing a total of 2,384 images, distributed uniformly among the classes.
- MPEG7 (MPEG7 CE Shape-1 Part B) [70]: It is a database of 1,400 binary images of 70 shape categories, being 20 objects per category.
- MSRCORID [71]: It contains a set of 4,320 images grouped into 20 categories – about 36–652 per category. Most categories have about 200 images.
Fig. 14. Mean $P \times R$ curves in (a) MSRCORID and (b) PASCAL databases after 8th iteration.

Fig. 15. Rel $x$ It to (a) Coil-100 and (b) Corel databases from 1st to 8th iteration.

Fig. 16. Rel $x$ It to (a) ETH-80 and (b) MPEG7 databases from 1st to 8th iteration.
PASCAL [72]: This database consists of images from Flickr. We use a subset of 3448 grouped into 23 classes with different number of images, varying from 72 to 446 subimages each.

We use several image descriptors to test the methods. The results presented in this paper make use of the following descriptors: ACC [43], BIC [1], Color Bitmap [73], Fourier [74], JAC [3], LAS [44], LBP [75], MSF (Multi-Scale Fractal) [47], SASI [2], SID [4] and TSDIZ [5]. ACC, BIC, Color Bitmap and JAC are color descriptors, while LAS, LBP, SASI and SID are texture descriptors. Fourier, Multi-Scale Fractal and TSDIZ are used for databases with shape information.

For each image database, we chose the best simple descriptors for combination, as shown in Table 3 (except for the case of Coil-100, we chose the worst ones, because the problem was too easy with the best one – BIC [1]). For Corel, MSRCORID and PASCAL databases we selected three color descriptors and two texture descriptors. In ETH-80 we used shape, color, and texture descriptors. Fourier, Multi-Scale Fractal and TSDIZ are used for databases with shape information.

Figs. 6–14 show the mean precision-recall curves of each method in all image databases for 3, 5 and 8 iterations of relevance feedback. These images show the evolution in the performance of each method. One may observe that both OPFMSPS and OPFRF outperformed GP$^+$ in all tested databases. OPFMSPS outperformed OPFRF in coil–100 and PASCAL image databases while OPFRF was more effective than OPFMSPS in ETH-80 and MSRCORID databases. In Corel database, OPFMSPS had better result up to the fifth iteration and OPFRF beat OPFMSPS after eight iterations. In MPEG7 shape database, OPFMSPS had a good result for three iterations while OPFRF outperforms OPFMSPS from five iterations. In addition to that, OPFRF using the best descriptor (TSDIZ for the MPEG7 database and BIC for the others) was more effective than GP$^+$ in most results, proving that the classification step is very important in the feedback-based learning process. Figs. 15–17 present the mean curves of relevant returned images per iteration ($Rel \times It$) for all databases from the first to the eighth iteration, confirming the previous results.

6. Conclusion

We extended a recent feedback-based learning approach for CBIR using OPF classification (OPF+$G^+$) to handle composite descriptors. The new methods, OPFMSPS and OPFRF, use optimization techniques, such as multi-scale parameter search (MSPS) and genetic programming (GP), to find the best combination function for a given set of simple descriptors at each iteration of relevance feedback.

Recent works [10,16] compare the baseline methods with the most effective approaches according to the state-of-the-art. Experiments with several datasets and descriptors have demonstrated that both OPFMSPS and OPFRF are very effective methods, with better performance than other state-of-the-art approaches, GP$^+$ and OPFRF. In [16], however, we have demonstrated that our planned approach outperforms our greedy approach used in this work for simple descriptors. The extension of our planned approach for composite descriptors is not direct, given that it returns the most informative images (being many of them irrelevant images) per iteration, while the composite descriptors are being found based on the rank of the relevant images in the training set. This would require to investigate an effective optimization criterion to find composite descriptors based on sets with more irrelevant images than relevant ones. We understand that this is a subject for a future work.

The experiments show the importance of using multiple descriptors and pattern classifiers for CBIR. The best choice between OPFMSPS and OPFRF will depend on the dataset (application). However, it is important to note that MSPS assumes a fixed equation for the combination function and searches for its best parameters, while GP is more flexible in the choice of the best combination function. On the other hand, MSPS is faster than GP, and this favors the choice of OPFMSPS. The execution time of MSPS varies according to the number of descriptors to be combined. The number of scales has some influence, but tests have indicated that lower number of scales usually imply a higher number of iterations to achieve the best combination function.

A drawback in GP is its sensitivity to the initial parametrization (values of $n_p$ and $n_g$ for instance). This requires a previous study of the impact of each parameter for each dataset. Its efficiency depends on the number of generations, population size, training set size, and the size of the individuals [63].

The results also indicate that OPFRF and OPFMSPS require a few iterations of relevance feedback to retrieve the most relevant images.

Our future work includes a study of indexing schemes to provide scalability in large image databases with fast image access, the use of irrelevant images in the training set used by GP in the OPFRF method, and the extension of our planned method [16] to
use composite descriptors. The OPF classifier is a multiple-class approach, as mentioned in Section 2, so the extension of our methods for multiple levels of relevance is straightforward and the efficacy gain may be higher. We also intend to investigate that.

Acknowledgments

Authors thank CAPES, CNPq (grants 140968/2007-5, 481556/ 2009-5, 303673/2010-9, and 306587/2009-2), and FAPESP (grants 52015-0, 2008/57248-2, 2008/58528-2, and 2009/18438-7) for financial support.

References


[7] R. Li, J. Lu, Y. Zhang, T. Zhao, Dynamic adaboost learning with feature selection gain may be higher. We also intend to investigate that.


