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

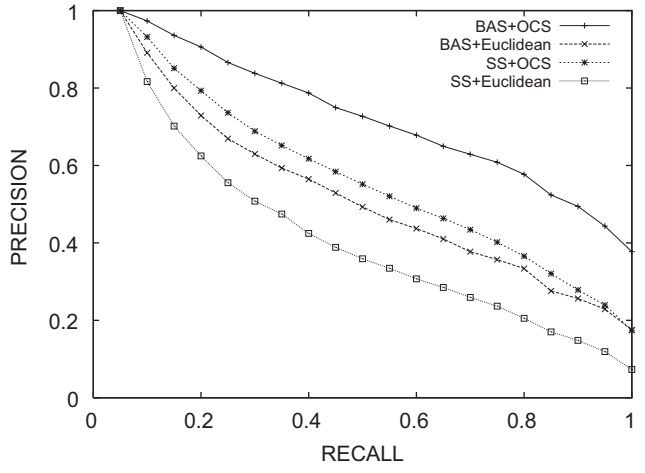

Fig. 1. Precision–recall curves for BAS and SS descriptors in MPEG-7 database using two different similarity functions.

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).

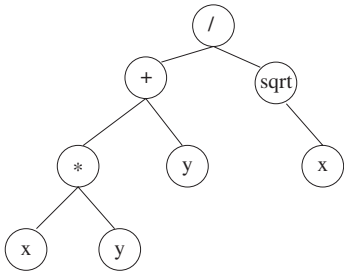

Fig. 2. A sample tree representation.