Correct and Efficient Evaluation of Region-Based Image Search *

Ilaria Bartolini and Marco Patella

DEIS - CSITE-CNR, University of Bologna, Italy
{ibartolini, mpatella}@deis.unibo.it

Abstract. Content-based image retrieval systems allow the user to interactively search image databases looking for those images which are similar to a specified query image. To this end, region-based systems decompose each database image into a set of "homogeneous" regions. Similarity between images is then assessed by computing similarity between regions and combining the results at image level. In this paper we propose a correct approach to region matching maximizing the overall similarity score between images. The presented approach only relies on the assumption that regions' scores are to be combined using a monotonic function. Experimental results obtained using an existing system show the effectiveness of our approach with respect to existing region matching heuristics. Then, to reduce query costs, we present an index-based algorithm that can make use of any distance-based access structure, and demonstrate its efficiency on a medium-size image data-set.

1 Introduction

Many real world applications, in the field of medicine, weather prediction, criminal investigations, and communications, to name a few, require efficient access to image databases based on content. To this end, the goal of content-based image retrieval (CBIR) systems is to define a set of properties (*features*) able to effectively characterize the content of images and then to use such features during retrieval. In order to allow automatic storing and retrieval of images, the features should be simple enough to allow their automatic extraction but meaningful enough to capture the image content.

The vast majority of CBIR systems uses *global features* to represent image semantics, by extracting information on color and texture on the whole image [FEF+94]. Other approaches divide each image into a set of pre-defined regions in order to extract localized features [SO95]. The usual approach of such systems is to represent each image as a feature vector and to assess similarity between images by way of a distance function. Of course, each method is characterized by the number and the type of such features and by the distance function used.

It is common, for a user accessing a CBIR system, to request for images containing particular "objects", possibly arranged in a specific spatial organization. The use of global features, therefore, cannot effectively fulfill users needs, since no information about objects contained in the image can be extracted from global descriptors. Only in recent times, a number of *region-based* image retrieval systems have been presented [CTB+99,NRS99,ABP99], that fragment each image into regions, i.e. sets of pixels sharing common visual characteristics, like color and texture. Similarity assessment between images is then performed by associating regions in the query image with those contained in database images and by taking into account similarity between associated regions. To

* This work has been partially supported by InterData and ex-60% grants from MURST

this end, features are extracted for each region and a metric function is used to compare region descriptors. Existing systems, however, either consider a scenario, which is beyond the scope of our work, where spatial constraints are taken into account [BDV99], or use naïve heuristic matching algorithms when associating regions of the images being compared, thus obtaining incorrect results. As an example, suppose that a user asks for an image containing two regions each representing a tiger: If a database image contains a single "tiger" region, it is *not* correct to associate both query regions to the single "tiger" region of the database image, since, in this case, information on the number of query regions is lost.

In our discussion, we will focus on k nearest neighbor queries, where the user asks for the k images in the database which are most similar, according to the similarity measurement implemented by the CBIR system, to a query image.

In this paper we present a *correct* approach to region matching and apply it to an existing region-based image retrieval prototype, the WINDSURF system [ABP99]. In [ABP99], the region matching technique uses an heuristic algorithm (see Section 3.2) that is not sound. It has to be noted that our approach is independent of the underlying CBIR system, and only requires that similarity between images is computed by way of similarity scores between image regions. Moreover, since the complexity of sequential evaluation of queries is linear in the database size, we propose an index-based solution, showing its efficiency on a medium-size image database. We begin our discussion by presenting existing region-based image retrieval systems, and outlining their limits in query processing (Section 2). Section 3 introduces the WINDSURF system, the region-based image retrieval prototype on which we will concentrate throughout the rest of the paper. In Section 4 we precisely formalize the region matching problem, and present a sequential correct algorithm (`ERASE`) for its resolution. We then present an index-based solution ($\mathcal{A}_0^{WS}$) for the problem, in order to speed up query resolution when the image database is large (Section 5). Analysis of the proposed solutions, with respect to existing heuristics used in the WINDSURF system, follows in Section 6, by looking at both efficiency and effectiveness issues. Finally, Section 7 concludes the paper, drafting possible directions for future work.

2 Background

Region-based image retrieval systems divide images into a set of "homogeneous" regions, i.e. sets of pixels having common visual characteristics. This is usually performed by applying a clustering algorithm on pixels constituting the image. In order to obtain homogeneity within regions, it is not correct to separately consider pixel features such as color, texture, and position, since image semantics would be lost: If we say, for example, that a red and striped object is only red, or only striped, we give a right information about the object, but not an effective description of it. Toward this goal, we need to fragment an image into regions by defining a feature vector on the combined space (e.g. on a *color-texture* space) for each of them. In this way, querying is based on a set of regions of interest, rather than a description of the whole image, and this allows to support more specific queries like: "find all those images containing a small red and striped region under a big blue region". It is obviously a need for CBIR systems to pass from an image-based approach to a region-based one.

As an example of a region-based system, in [SC96] the VisualSEEk system is proposed, considering information on both the spatial and the frequency domain in order to decompose each

image into regions. The similarity between two images is computed by taking into account color, location, dimension, and relative positioning of regions. Query processing, however, is carried out using a simple heuristics: First, for each region of the query image, a range query on color, location, and dimension is issued with similarity thresholds provided by the user; then, the *candidate set* of images is built, by taking into account only those images that present regions in all the result regions sets; finally, the optimum match is computed on the set of candidate images. Thus, if a user would request for, say, the 10 images most similar to a given one, he/she is also asked for the specification of similarity thresholds that have no physical counterparts in user's mind.

WALRUS (WAveLet-based Retrieval of User-specified Scenes) [NRS99] is a region-based similarity retrieval system which fragments images using wavelets [Dau92]. The matching phase of WALRUS consists in retrieving all the DB regions which are similar to at least one query region with a score $\geq \epsilon$. To this end, descriptors of DB regions are indexed using an R*-tree [BKSS90] and a range query, with a radius determined by ϵ, is issued for each query region. Then, in the combining phase, which is applied only to those images containing regions obtained in the matching phase, the relative sizes of matching regions are added up to obtain the overall similarity score between images. Images for which the similarity with the query image is higher than a user-specified ξ threshold are returned as the query result.

The main limitation of both VisualSEEk and WALRUS resides in the fact that they require the specification of similarity thresholds. Indeed, range queries are not well suited for the scenario we envision: Since the user has no a priori knowledge on the distribution of similarities between images, he/she has no way to guess the "right" value for a similarity threshold; a high value for it could lead to an empty result, and slightly lowering this value could result in an overwhelming number of returned images.

Blobworld [CTB+99] is a CBIR system which fragments an image into regions (*blobs*), homogeneous with respect to color and texture, by using an Expectation-Maximization clustering algorithm. The Blobworld index-based query resolution algorithm uses an R-tree-like structure to index color descriptors of blobs. The matching phase is performed by requesting, for each blob in the query image, a predetermined number (in the order of the hundreds) of most similar DB regions by issuing a nearest neighbors query on the index. The combining phase only considers regions obtained in the matching phase and computes the overall image similarity using (weighted) fuzzy-logic operators to combine regions' scores. This approach has two major limitations. First, since best matches for query blobs are computed by ignoring matches for other blobs, a single blob in the database image can be associated to two distinct query blobs (see the "two tigers" example in Section 1). Second, the number of regions that are returned by the matching phase is a priori determined, thus it is unrelated to the number k of images requested by the user and to the specific query image. As we will show in Section 4, this can lead to miss the correct best images.

3 The Windsurf system

The WINDSURF system [ABP99] is a region-based image retrieval system that uses the Discrete Wavelet Transform (DWT, [Dau92]) to divide each image into regions. The global architecture of the system is sketched in Figure 1. Each image is processed through a number of steps described as follows:

DWT The image is analyzed in the time-frequency domain using a 2-D DWT. In detail, the Haar wavelet is used as a special case of the bihorthogonal wavelet of Cohen-Daubechies-Feauveau (CDF), as provided by the WAILI software library [UVJ+97]. To this end, the image is divided into the correspondent color channels and the DWT is applied to each channel. Since the RGB color space is not suitable to reflect human perception of color [Smi97], we consider the HSV color space, because, in this space, each color component is perceptually independent and uniform. We refer to the j-th wavelet coefficient as $w_j^{l;B} = (w_{0_j}^{l;B}, w_{1_j}^{l;B}, w_{2_j}^{l;B})$, where B is a sub-band of frequency ($B \in \mathcal{B} = \{LL, LH, HL, HH\}$), l is the DWT level and $c \in \{0, 1, 2\}$ denotes a color channel.

Clustering The image is fragmented into a set of regions using the wavelet coefficients (*clustering features*). In particular, we use a simple *k-means* algorithm whose goal is to minimize a function depending on the distance between each pixel coefficient and the centroid of each cluster. Obviously, the value of such function depends both on the number of clusters and on the choice of the distance function. As for the distance between pixels descriptors, we adopted the *Mahalanobis* distance applied on the 3-D wavelet coefficients of the LL sub-band of the 3-rd level. This because, intuitively, regions should be built by taking into account low frequency descriptors. To compute the "optimal" value for the number of clusters (m), we iterate the computation of the above function between a minimum ($m_{\min} = 2$) and a maximum ($m_{\max} = 10$) value, choosing the best solution which minimizes the adopted validity function. Thus, the k-means algorithm provides, as the output, the total set of image pixels divided in m clusters. Details on the region fragmentation algorithm of the WINDSURF system can be found in [ABP99].

Feature Indexing Regions so obtained are described using a set of similarity features. When comparing regions, we consider information on size and color-texture as provided by all the frequency sub-bands. To this end, the similarity features for a region R_{s_i} of image I_s are defined as a 37-D vector:

Size The number of pixels in the region, $size(R_{s_i})$.

Centroid The centroid of R_{s_i} is defined through a 12-D vector $V_{R_{s_i}} = (\mu_{R_{s_i}}^{LL}, \mu_{R_{s_i}}^{LH}, \mu_{R_{s_i}}^{HL}, \mu_{R_{s_i}}^{HH})$, where, for each sub-band B, $\mu_{R_{s_i}}^{B}$ is a 3-D point representing the average value for each color channel. This represents the color information for the region.

Features These correspond to the coefficients of the 3×3 covariance matrices, $\mathcal{C}_{R_{s_i}}^{3;B}$, of the points contained in R_{s_i}. Since the covariance matrices are symmetrical, we only store 6 values for each matrix $\mathcal{C}_{R_{s_i}}^{3;B}$, obtaining a 24-$D$ vector $\mathcal{C}_{R_{s_i}}$. Since coefficients of the covariance matrix take into account variations in color for pixels in R_{s_i}, these represent texture information of the region.

3.1 Region Similarity

In the WINDSURF system, the similarity between two regions, R_{q_i} of a query image I_q and R_{s_j} of a database image I_s, is computed as follows:

$$r_{sim}(R_{q_i}, R_{s_j}) = h(d(R_{q_i}, R_{s_j})) \tag{1}$$

where $d()$ is a distance function, and $h()$ is a so-called *correspondence function* [CPZ98] mapping distance values to similarity scores. The function $h : \Re_0^+ \rightarrow [0, 1]$ has to satisfy the following

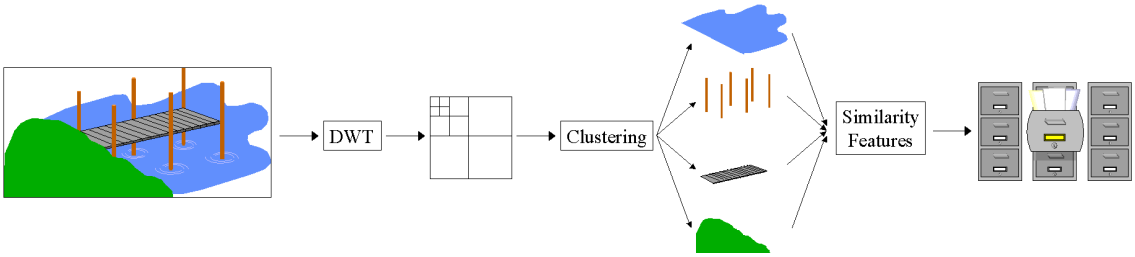


Fig. 1. Steps for image features indexing in the WINDSURF system.

properties: $h(0) = 1$ and $d_1 \leq d_2 \Rightarrow h(d_1) \geq h(d_2), \forall d_1, d_2 \in \Re_0^+$. In all the experiments, we used $h(d) = e^{-d/\sigma_d}$, where σ_d^2 is the variance of the distances computed over a sample of database regions. The overall distance between regions R_{q_i} and R_{s_j} takes into account both differences in the color and textures descriptors and in their relative size (see [ABP99] for details).

3.2 Image Similarity

In the first version of our prototype, we used simple heuristics to compute the overall similarity between two images. This will be used in Section 6 as a yardstick to compare effectiveness and efficiency of our proposed approach.

When matching regions in I_q with regions in I_s, we have to satisfy two basic constraints:

1. A region of I_q cannot match with two different regions in I_s.
2. Two different regions of I_q cannot match with the same region of I_s.

To this end, each region R_{q_i} of I_q is associated to its "best match" region R_{s_j} in I_s by considering $r_{sim}(R_{q_i}, R_{s_j})$. If, however, two regions R_{q_i} and $R_{q_{i'}}$ of I_q are associated to the same region R_{s_j} of I_s, only the best match is kept, e.g. if $r_{sim}(R_{q_i}, R_{s_j}) > r_{sim}(R_{q_{i'}}, R_{s_j})$, then the match between $R_{q_{i'}}$ and R_{s_j} is removed and, following the WINDSURF heuristics, $R_{q_{i'}}$ is associated with no region of I_s, therefore contributing with a score $r_{sim} = 0$ to the overall similarity score between images I_q and I_s. The region in I_s associated with region R_{q_i} is indicated as $\Gamma_s(R_{q_i})$.

We are now ready to compute the overall similarity between two images I_q and I_s as the average similarity between matched regions:

$$I_{sim}(I_q, I_s) = \frac{1}{n} \sum_{i=1}^{n} r_{sim}(R_{q_i}, \Gamma_s(R_{q_i})) = \frac{1}{n} \sum_{i=1}^{n} h(d(R_{q_i}, \Gamma_s(R_{q_i}))) \quad (2)$$

Note that the best match for a certain region R_{q_i} can be undefined. Of course, $r_{sim}(R_{q_i}, \Gamma_s(R_{q_i})) = 0$ if $\Gamma_s(R_{q_i})$ is undefined.

4 Optimal Region Matching

Given a reference (query) image I_q, divided into a set of regions $\{R_{q_1}, \ldots, R_{q_n}\}$, and an archive (data-set) image I_s, also divided into a set of regions $\{R_{s_1}, \ldots, R_{s_m}\}$, the problem of *optimal region matching* consists in associating (matching) each region R_{q_i} of I_q to a region $R_{s_j} = \Gamma_s(R_{q_i})$

of I_s (possibly, no region is associated to R_{q_i}, i.e. $\Gamma_s(R_{q_i}) = \emptyset$) such that the overall similarity score between images I_q and I_s, $I_{sim}(I_q, I_s)$, is maximized. Similarity between regions is assessed by way of the $r_{sim}(R_{q_i}, R_{s_j})$ function. Every Γ_s matching of regions has to satisfy the following constraint: two regions of I_q cannot be associated to the same region of I_s, therefore if $R_{q_i} \neq R_{q_j}$ and $\Gamma(R_{q_i}) = \Gamma(R_{q_j})$, it is $\Gamma(R_{q_i}) = \Gamma(R_{q_j}) = \emptyset$. Similarity between images is computed by taking into account similarity between associated regions, i.e. $I_{sim}(I_q, I_s) = RM_{sim}(r_{sim}(R_{q_1}, \Gamma_s(R_{q_1})), \ldots, r_{sim}(R_{q_n}, \Gamma_s(R_{q_n})))$. The only requirement for the function RM_{sim} is that it has to be a monotonic increasing function, that is if $s_i \leq s'_i, i \in \{1, n\}$, then it is $RM_{sim}(s_1, \ldots, s_i, \ldots, s_n) \leq RM_{sim}(s_1, \ldots, s'_i, \ldots, s_n)$. This is intuitive, since better matches between regions can only increase the overall similarity score between corresponding images. Moreover, for the sake of simplicity, in the following we will assume that RM_{sim} is a commutative function. The optimal matching between regions, i.e. that for which $I_{sim}(I_q, I_s)$ is maximum, will be denoted as Γ_s^{opt}.

The above problem can be expressed as a generalized assignment problem: Let $s_{ij} = r_{sim}(R_{q_i}, R_{s_j})$ be the similarity score between region R_{q_i} of I_q and region R_{s_j} of I_s, denote with $\mathcal{H}$ the index set of matched regions $\mathcal{H} = \{(i,j)|R_{s_j} = \Gamma_s(R_{q_i})\}$; of course, it is $|\mathcal{H}| \leq \min\{m, n\}$. The goal is to maximize the function $RM_{sim}(s_{i_1 j_1}, \ldots, s_{i_{|\mathcal{H}|} j_{|\mathcal{H}|}})$, with $(i_h j_h), (i_l j_l) \in \mathcal{H}, (i_h j_h) \neq (i_l j_l)$, i.e. maximize $I_{sim}(I_q, I_s)$. To this end, we introduce the variables $x_{ij} = 1$ iff $R_{s_j} = \Gamma_s(R_{q_i})$, $x_{ij} = 0$ otherwise.

$$z = \max RM_{sim}(s_{i_1 j_1}, \ldots, s_{i_{|\mathcal{H}|} j_{|\mathcal{H}|}}), \quad (i_h j_h), (i_l j_l) \in \mathcal{H}, (i_h j_h) \neq (i_l j_l) \tag{3}$$

$$\mathcal{H} = \{(i,j)|x_{ij} = 1\} \tag{4}$$

$$\sum_{j=1}^{m} x_{ij} \leq 1 \quad (i = 1, \ldots, n), \tag{5}$$

$$\sum_{i=1}^{n} x_{ij} \leq 1 \quad (j = 1, \ldots, m), \tag{6}$$

$$x_{ij} \in \{0, 1\} \quad (i = 1, \ldots, n)(j = 1, \ldots, m) \tag{7}$$

Equation 3 means that to determine the overall score $I_{sim}(I_q, I_s)$ we have to consider only the matches $\Gamma_s()$ in $\mathcal{H}$ (Equation 4). Equation 5 (Equation 6) expresses the constraint that at most one region R_{s_j} of I_s (resp. R_{q_i} of I_q) can be assigned to a region R_{q_i} of I_q (resp. R_{s_j} of I_s).

Definition 1 (Correct matching). *A set of x_{ij} values that satisfies the constraints expressed by Equations 5, 6, and 7 is called a* correct *matching.*

Definition 2 (Complete matching). *A correct matching for which it is $\sum_{j=1}^{m} x_{ij} = 1, (i = 1, \ldots, n)$ (i.e. each query region is associated to a region of the database image) is called a* complete *matching. It should be noted that* any *correct matching for a database image having a number of regions lower than that of the query regions is obviously not complete.*

Definition 3 (Optimal matching). *The correct matching that maximizes the function expressed by Equation 3 is called the* optimal *(or* exact*) matching.*

A typical form of the scoring function RM_{sim} is that of a sum (this is indeed the case, save for a constant scale factor, for two of the image retrieval systems introduced in previous Sections, i.e.

WALRUS and WINDSURF, whereas Blobworld uses fuzzy functions for computing image similarity), leading to a re-formulation of Equation 3 as follows:

$$z = \max \sum_{i=1}^{n} \sum_{j=1}^{m} s_{ij} \cdot x_{ij} \tag{8}$$

The generalized assignment problem, in this case, takes the form of the well known Assignment Problem (AP), one of the most popular topics in combinatorial optimization. To resolve it, we apply the Hungarian Algorithm [Kuh55] `HUNG` to the matrix $\{s_{ij}\}$ of similarity scores between regions. More precisely, sequential evaluation of a k nearest neighbor query is performed by way of the `ERASE` (Exact Region Assignment SEquential) algorithm shown in Figure 2.

```
ERASE(I_q: query image, k: integer, C: data-set)
{ ∀ image I_s in the data-set C
  { ∀ region R_{s_j} of I_s
      ∀ region R_{q_i} of I_q compute s_ij = s(R_{q_i}, R_{s_j});
    invoke HUNG({s_ij}) obtaining, as the result, the value I_sim(I_q, I_s); }
  return the k images having the highest overall similarity scores I_sim(I_q, I_s); }
```

Fig. 2. The Exact Region Assignment SEquential algorithm.

Correct resolution of k nearest neighbor queries by way of the `ERASE` algorithm, therefore, requires the computation of similarity scores between regions in the query image and *all* the regions contained in the database images. Algorithm complexity is, hence, linear in the database size.

5 Index Evaluation

In order to obtain a complexity sub-linear in the data-set size, in this Section we present an index-based algorithm for the resolution of the optimal region matching problem.

Since similarity between images is computed by combining distances between regions' features, we need to use a distance-based access method (DBAM), like the R*-tree [BKSS90] or the M-tree [CPZ97], to index regions contained in database images. Such index structures are able to efficiently answer to range- and k nearest neighbor queries, as well as to perform a *sorted access* to the data, i.e. to output regions one by one in increasing order of distance with respect to a query region [HS95]. In order to deal with "compound" queries where multiple query regions are specified, we need to extend query capabilities of DBAMs. In our experimentation, we used the M-tree index [CPZ97].

A first naïve approach to resolve compound queries with DBAMs is the following: For each region R_{q_i} of the query image I_q, we execute a k nearest neighbor query, thus returning the k regions in the data-set most similar to R_q. Then, we compute the exact region assignment for the images that contained the regions obtained in the previous step (this is, indeed, the query processing strategy used in the Blobworld system). This set of images is called the *candidate set*.

This algorithm guarantees that the number of candidate images is not higher than $n \cdot k$. Such a solution is indeed very efficient, but does not guarantee that the correct result is returned. As an example, consider the case where $n = 2$, $k = 1$, and consider the similarity scores obtained by a sorted access to a DBAM and given in Table 1. It is plain to see that the image most similar to the query image I_q is the image I_2 (the overall similarity score, computed as the average sum of regions similarities, for image I_1 is $\frac{0.9+0.7}{2} = 0.80$, for I_2 is $\frac{0.85+0.79}{2} = 0.82$, for I_3 is $\frac{0.71+0.87}{2} = 0.79$, and other images lead to lower scores), whereas the *candidate* set is only composed by images I_1 and I_3.

R_{q_1}			R_{q_2}		
region	image	similarity	region	image	similarity
R_{1_1}	I_1	0.90	R_{3_2}	I_3	0.87
R_{2_2}	I_2	0.85	R_{2_1}	I_2	0.79
R_{4_1}	I_4	0.83	R_{3_3}	I_3	0.75
R_{3_3}	I_3	0.71	R_{1_1}	I_1	0.72
R_{2_1}	I_2	0.69	R_{1_2}	I_1	0.70
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 1. A sorted access example for a query image with two regions: R_{q_1} and R_{q_2}.

In order to guarantee that the correct result for the query is included in the candidate set, i.e. in the images retrieved by the sorted access phase, we have to find a suitable condition to stop the sorted accesses. The reference solution is the Fagin's $\mathcal{A}_0$ algorithm [Fag96]. The $\mathcal{A}_0$ algorithm stops the sorted accesses when at least k objects are included in all the index scans results (this is called the *stop condition* for the algorithm). The only requirement for the $\mathcal{A}_0$ algorithm is that the function applied to combine objects' scores (in our case, the RM_{sim} function) has to be monotonic. Applying the $\mathcal{A}_0$ algorithm to the optimal region matching problem would be as in Figure 3.

$\mathcal{A}_0$(`query image` I_q, `integer` k, `DBAM` $\mathcal{T}$)
{ $\forall$ `region` R_{q_i} `of` I_q, `open a` *`sorted access`* `index scan on` $\mathcal{T}$ `and insert images containing result regions in the set` X^i;
`stop the sorted accesses when there are at least` k `images in the intersection` $L = \cap_i X^i$;
`for each image` I_s `in the` *`candidate set`* $\cup_i X^i$, `compute the optimal assignment;`
(*`random access`*)
`return the` k `images having the highest overall similarity scores` $I_{sim}(I_s, I_q)$; }

Fig. 3. The $\mathcal{A}_0$ algorithm for the optimal region matching problem.

This algorithm, however, does not guarantee yet that the best k images are included in the candidate set, since its stopping condition does not take into account *correct* assignment of regions. Just consider, as an example, the case depicted in Table 2, where $n = 2$ and $k = 1$. Here, as opposed to the case of Table 1, it is not correct to stop the sorted access phase at the second step, since

image I_2 has been found for both query regions with the same region R_{2_1}; therefore, we cannot find a correct assignment for image I_2 using only regions that have been seen during the sorted access phase.

R_{q_1}			R_{q_2}		
region	image	similarity	region	image	similarity
R_{1_1}	I_1	0.90	R_{3_2}	I_3	0.87
R_{2_1}	I_2	0.85	R_{2_1}	I_2	0.79
R_{4_1}	I_4	0.83	R_{3_3}	I_3	0.75
R_{3_3}	I_3	0.71	R_{1_1}	I_1	0.72
R_{2_3}	I_2	0.69	R_{1_2}	I_1	0.70
⋮	⋮	⋮	⋮	⋮	⋮

Table 2. Another sorted access example for a query image with two regions: R_{q_1} and R_{q_2}.

To ensure that the best k results are included into the set of candidate images, the stopping condition of $\mathcal{A}_0$ algorithm has to be modified to test correctness of regions' assignments (see Definition 1). The sorted access phase can be stopped as soon as a complete assignment (Definition 2) is found, taking into account only regions returned by index scans.[1] In the example of Table 2, hence, we stop the sorted accesses after the fourth step, since image I_3 has a complete assignment ($\Gamma_3(R_{q_1}) = R_{3_3}$ and $\Gamma_3(R_{q_2}) = R_{3_2}$). It should be noted, however, that this is *not* the best result for I_q (image I_1 leads to the best overall score of 0.8). This leads to the modified $\mathcal{A}_0$ algorithm shown in Figure 4.

```
A_0^WS(I_q: query image, k: integer, T: DBAM)
{ ∀ region R_qi of I_q, open a sorted access index scan on T and insert result regions
    in the set X^i;
  stop the sorted accesses when there are at least k images for which
    a complete assignment exists, considering only regions in ∪_i X^i;
  for each image I_s having regions in ∪_i X^i, compute the optimal assignment;
    (random access)
  return the k images having the highest overall similarity scores I_sim(I_s, I_q); }
```

Fig. 4. The $\mathcal{A}_0^{WS}$ algorithm.

The random access phase consists in computing the optimal assignment $\Gamma_s^{opt}(R_{q_i})$ for each image I_s in the candidate set. To this end, a number of similarity scores have to be computed, since the sorted access phase only produces similarity scores between each query region R_{q_i} and database regions included in X^i, i.e. the result set of i-th scan.

[1] By the way, this is the reason why Blobworld algorithm is not correct, since its stopping condition cannot guarantee the existence of a complete assignment.

As for algorithm correctness, we give here only an intuitive explanation. Exact proof follows the same steps of that of $\mathcal{A}_0$ algorithm [Fag96] and is omitted for the sake of brevity. Suppose $k = 1$, we have to prove that the best image is included in the candidate set. Consider, again, the example of Table 2: We stop the index scans after the fourth step, thus the candidate set consists of images I_1, I_2, I_3, and I_4. For each i, by definition of sorted access, images not included in the candidate set can only have regions with similarity scores lower than those included in X^i. Since the RM_{sim} function is monotonic increasing, such images will have an overall score lower than that of candidate images having a complete assignment. Therefore, the correct result cannot include images outside of the candidate set.

The index evaluation of compound queries, thus, will have a twofold impact on query evaluation: First, the use of an index can reduce the number of distance computations needed for assessing image similarity; second, the number of images on which the Hungarian algorithm has to be run is reduced by considering only the candidate images.

6 Experimental results

Preliminary experimentation on proposed techniques has been performed on a sample medium-size data-set consisting of ca. 2000 real-life images, producing over 8000 regions, extracted from a CD-ROM of *IMSI-PHOTOS*.[2] The query workload consists in about a hundred randomly chosen images not included in the data-set. All experiments were performed on a Pentium II 450 MHz workstation equipped with 64MB of main memory running Windows NT 4.0.

6.1 Efficiency

The first set of experiments we present concerns the efficiency of the proposed approach. In order to test the performance of the $\mathcal{A}_0^{WS}$ index-based algorithm, in Figure 5 we compare the number of candidate images, i.e. the images on which the Hungarian algorithm has to be applied, as a function of the number of query regions.[3] Of course, for the `ERASE` algorithm the number of candidate images equals the number of images in the data-set, whereas for the index version this number depends both on values of k and of the number of query regions. As the graph shows, the $\mathcal{A}_0^{WS}$ algorithm is indeed very efficient in reducing the number of candidate images, even if its performance degrades as the number of query regions increases. This is intuitive, since the complexity of finding k objects in the intersection of n sets augments with n.

Another element affecting algorithm performance is the number of computed distances between regions. In Figure 6 (a) we show the number of computed distances for the `ERASE` and the $\mathcal{A}_0^{WS}$ algorithms, as a function of k, with a number of query regions $n = 3$ (this is the average number of regions for the data-set images). In order to reduce the number of distances to be computed for the index-based algorithm, we also considered an approximate version of the $\mathcal{A}_0^{WS}$ algorithm: $\mathcal{A}_0^{WS_{app}}$. In this case, the random access phase computes the optimal assignment for each candidate image by taking into account only regions returned by the sorted access phase, i.e. no new distance is computed. Average number of distance computations for the $\mathcal{A}_0^{WS_{app}}$ algorithm is also shown

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

[3] Unless otherwise specified, all the graphs presented here show numbers averaged over all the images contained in the query workload.

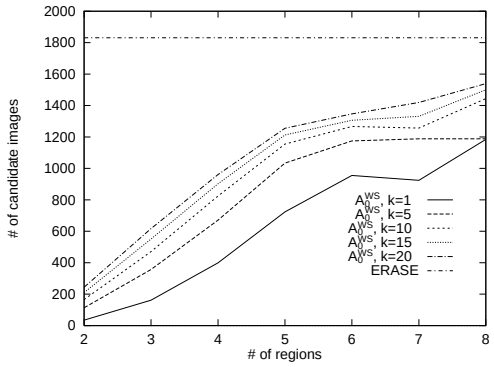
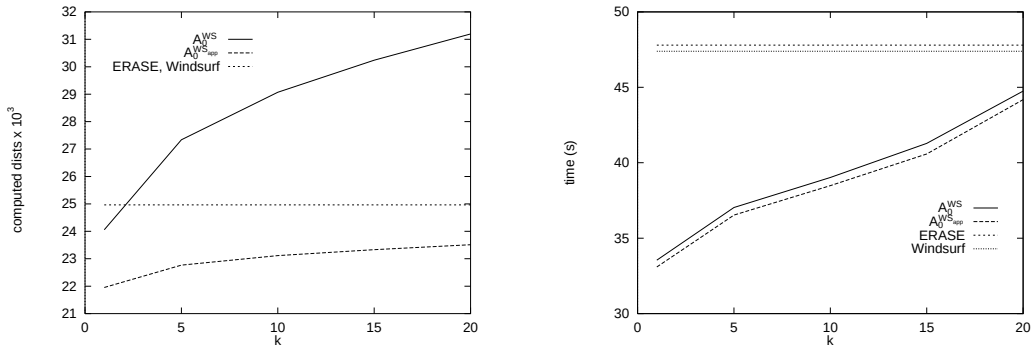


Fig. 5. Average number of candidate images as a function of the number of query regions.

in Figure 6 (a). The graph shows that the index-based approach is not very efficient in reducing the number of computed distances. We believe that this is due to the reduced cardinality of the considered data-set: Increasing the number of images in the data-set would have a beneficial effect on performance of index-based algorithms (whose search costs grow logarithmically with the number of indexed objects) with respect to that of sequential ones.

Finally, in Figure 6 (b) we compare query response times as a function of k (with a constant value of $n = 3$). The graph shows average query evaluation times (in seconds) for the `ERASE` algorithm, the original WINDSURF algorithm and the two index-based algorithms, $\mathcal{A}_0^{WS}$ and $\mathcal{A}_0^{WS_{app}}$, respectively. From the graph it can be deduced that: (i) The lower complexity of region matching for the original WINDSURF algorithm with respect to the `ERASE` algorithm does not pay off in reducing query evaluation times; this is due to the fact that, if n is low (as it is in our case), finding the optimal matching is very easy. (ii) The index-based algorithms really succeed in cutting down query resolution times, even if difference in performance reduces with increasing values of k. (iii) The approximate $\mathcal{A}_0^{WS_{app}}$ algorithm has performance similar to that of the exact $\mathcal{A}_0^{WS}$ algorithm; this demonstrates that the increase in performance with respect to sequential query evaluation is due to the reduced number of candidate images, and that the number of computed distances has a minor impact on performance.



(a) (b)

Fig. 6. Average number of computed distances (a) and average query resolution time (b) as a function of k ($n = 3$).

6.2 Effectiveness

In order to compare the "goodness" of results obtained by approximated algorithms (i.e. the original WINDSURF image matching algorithm and $\mathcal{A}_0^{WS_{app}}$) with respect to those obtained by exact ones (`ERASE` and $\mathcal{A}_0^{WS}$), we need a performance measure able to compare results of k nearest neighbor queries. Such measure should compare two sorted lists of results: This is obtained by contrasting *ranks* (positions) of objects in exact and approximate results. Given the i-th image in the approximate result, its rank, rank(i), is given by the position of that image in the exact result. As an example, consider the case when $k = 1$: The "goodness" of an approximate result with respect to the exact one can be obtained by only taking into account rank(1). The measure can be easily extended to the case where $k > 1$ by considering all the rank for the k images that form the approximate result, i.e. rank(1), ... , rank(k).

In [WB00], the *normalized rank sum* is used to quantify the loss of result quality when k nearest neighbor queries are approximately evaluated.

$$nrs = \frac{k(k+1)}{2 \cdot \sum_{i=1}^{k} \text{rank}(i)} \tag{9}$$

The overall measure, hence, is computed by the inverse sum of all the exact ranking for the objects in the approximate result. This measure, however, is not able to capture inversions in the result (e.g. when image I_s is ranked higher than image $I_{s'}$ in the approximate result and lower in the exact result), since no difference between ranking of objects in the approximate and in the exact result is taken into account.

In [ZSAR98], the *precision of approximation* measure P is introduced, which is defined as:

$$P = \frac{\sum_{i=1}^{k} \frac{i}{\text{rank}(i)}}{k} \tag{10}$$

P, therefore, measures the relative error in ranking for all the objects in the approximate result. This measure, however, relies on the assumption that $i \leq \text{rank}(i)$, thus no inversions on results are allowed.

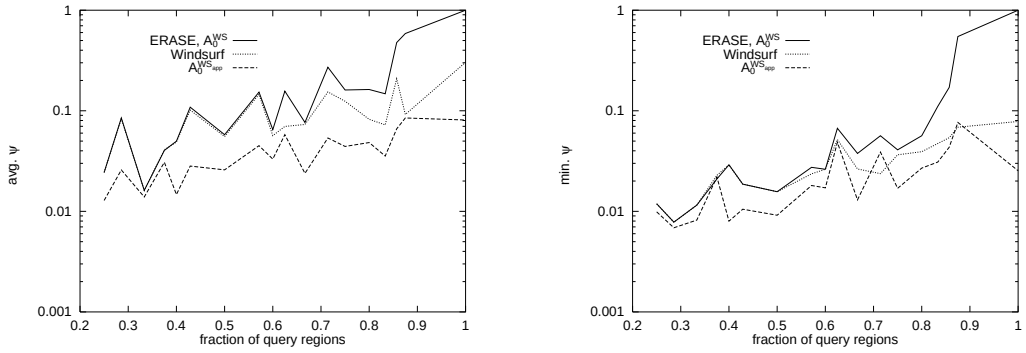
To overcome above limitations in quality measures, we introduce a new measure: the normalized rank difference sum ψ. To compute ψ, we sum differences in ranking for objects in approximate result and divide by k. Normalization of the measure in the interval $[0, 1]$ leads to the formulation of ψ as follows:

$$\psi = \frac{1}{1 + \frac{1}{k}\left(\sum_{i=1}^{k} \mathbf{1}(\text{rank}(i) - i)^p\right)^{1/p}} \tag{11}$$

where $\mathbf{1}()$ is the ramp function ($\mathbf{1}(x) = 0$ if $x < 0$, $\mathbf{1}(x) = x$ if $x \geq 0$), and p is an integer parameter (in our experiments, we always used $p = 2$). The use of the ramp function $\mathbf{1}()$ is necessary for correct accounting of inversions in ranking. Consider, as an example, the case where $k = 3$ and the exact result is I_1, I_2, and I_3. If the approximated result is I_1, I_3, I_2, it is *not* correct to add to the error the rank difference for both I_2 and I_3, since the error for I_2 is generated from the shift of I_3 (or vice versa). Hence, by using the ramp function, we only consider downward shifts in ranking. Values of ψ close to 1 indicate high quality of the approximate result.

Figure 7 shows average (a) and minimum (b) values of ψ for exact and approximated algorithms as a function of the fraction of query regions used to query the database (the value of k is kept

fixed at 20, other values lead to similar results and are omitted here for brevity). This is to show the effectiveness of different approaches when only some regions of the query image are used for the query (this can be done in order to reduce the query response time or just because we are interested only in some objects included in the query image). Both graphs exhibit similar trends: The effectiveness of the $\mathcal{A}_0^{WS_{app}}$ algorithm is almost always the lowest, and, for all curves, ψ only reaches high values when the fraction of query regions is close to 1. Figure 7 (b) shows that, in order to find a "good" result, we have to use *all* the regions in the query image. From Figure 7 (a), on the other hand, we see that approximate algorithms lead to a very low effectiveness, even if, as we have seen before, they attain slightly better performance with respect to their exact counterparts.



(a) (b)

Fig. 7. Average (a) and minimum (b) ψ as a function of the fraction of query regions ($k = 20$).

7 Conclusions

In this work we introduced an original approach to correct resolution of similarity queries on region-based image retrieval systems. In particular, a sequential algorithm (`ERASE`) was presented computing the optimal matching between regions of the query image and regions of a database image, in order to maximize the overall similarity score between images. We then proposed an index-based approach ($\mathcal{A}_0^{WS}$) to reduce query resolution times for large image data-sets. Preliminary experiments conducted over an existing CBIR system (WINDSURF) showed that the presented approach is indeed very effective with respect to existing region matching heuristics. However, from the efficiency point of view, we still have room for improvement. In particular, we observed that the index-based algorithm, even if it reduces query resolution times with respect to the sequential one, is not successful in reducing the number of distance computations during the sorted access phase. In order to overcome this limitation, we plan to employ approximate techniques for index access [CP00]. Another issue that needs to be investigated regards the possible parallelization of sorted access to the index [BEKS00]; in fact, in the worst case the same index node is retrieved once for each query region, whereas a "shared" access for all query regions could substantially reduce query costs.

References

[ABP99] S. Ardizzoni, I. Bartolini, and M. Patella. Windsurf: Region-based image retrieval using wavelets. In *IWOSS'99*, pp. 167–173, Florence, Italy, September 1999.

[BDV99] S. Berretti, A. Del Bimbo, and E. Vicario. Managing the complexity of match in retrieval by spatial arrangement. In *ICIAP'99*, Venezia, Italy, September 1999.

[BEKS00] B. Braunmüller, M. Ester, H.-P. Kriegel, and J. Sander. Efficiently supporting multiple similarity queries for mining in metric databases. In *ICDE 2000*, pp. 256–267, San Diego, CA, February 2000.

[BKSS90] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *SIGMOD'90*, pp. 322–331, Atlantic City, NJ, May 1990.

[CP00] P. Ciaccia and M. Patella. PAC nearest neighbor queries: Approximate and controlled search in high-dimensional and metric spaces. In *ICDE 2000*, pp. 244-255, San Diego, CA, February 2000.

[CPZ97] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *VLDB'97*, pp. 426–435, Athens, Greece, August 1997.

[CPZ98] P. Ciaccia, M. Patella, and P. Zezula. Processing complex similarity queries with distance-based access methods. In *EDBT'98*, pp. 9–23, Valencia, Spain, March 1998.

[CTB+99] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik. Blobworld: A system for region-based image indexing and retrieval. In *VISUAL'99*, pp. 509–516, Amsterdam, The Netherlands, June 1999.

[Dau92] I. Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1992.

[Fag96] R. Fagin. Combining fuzzy information from multiple systems. In *PODS'96*, pp. 216–226, Montreal, Canada, June 1996.

[FEF+94] C. Faloutsos, W. Equitz, M. Flickner, W. Niblack, D. Petkovic, and R. Barber. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3(3/4):231–262, July 1994.

[HS95] G. R. Hjaltason and H. Samet. Ranking in spatial databases. In *SSD'95*, pp. 83–95, Portland, ME, August 1995.

[Kuh55] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2:83–97, 1955.

[NRS99] A. Natsev, R. Rastogi, and K. Shim. WALRUS: A similarity retrieval algorithm for image databases. In *SIGMOD'99*, pp. 395–406, Philadelphia, PA, June 1999.

[SC96] J. R. Smith and S.-F. Chang. VisualSEEk: A fully automated content-based image query system. In *ACM Multimedia '96*, pp. 87–98, Boston, MA, November 1996. `http://www.ctr.columbia.edu/visualseek/`.

[Smi97] J. R. Smith. *Integrated Spatial and Feature Image Systems: Retrieval, Analysis and Compression*. PhD thesis, Columbia University, 1997.

[SO95] M. Stricker and M. Orengo. Similarity of color images. In *Storage and Retrieval for Image and Video Databases SPIE*, volume 2420, pp. 381–392, San Jose, CA, February 1995.

[UVJ+97] G. Uytterhoeven, F. Van Wulpen, M. Jansen, D. Roose, and A. Bultheel. WAILI: Wavelets with integer lifting. Technical Report 262, Department of Computer Science, Katholieke Universiteit Leuven, Heverlee, Belgium, July 1997.

[WB00] R. Weber and K. Böhm. Trading quality for time with nearest-neighbor search. In *EDBT2000*, Konstanz, Germany, March 2000.

[ZSAR98] P. Zezula, P. Savino, G. Amato, and F. Rabitti. Approximate similarity retrieval with M-trees. *The VLDB Journal*, 7(4):275–293, 1998.