Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 96 (2016) 1479 – 1488

Procedia Computer Science

20th International Conference on Knowledge Based and Intelligent Information and Engineering Systems, KES2016, 5-7 September 2016, York, United Kingdom

Approximate matching in ACSM dissimilarity measure

Alessia Amelio[a,*]

[a]*DIMES University of Calabria, Via Pietro Bucci Cube 44, 87036 Rende (CS), Italy*

Abstract

The paper introduces a new patch-based dissimilarity measure for image comparison employing an approximation strategy. It extends the Average Common Sub-matrix measure computing the exact dissimilarity among images. In the exact method, dissimilarity between two images is obtained by considering the average area of the biggest square sub-matrices in common between the images, by exact match of the extracted sub-matrices pixel by pixel. As an extension, the proposed dissimilarity measure computes an approximate match between the sub-matrices, which is obtained by omitting a controlled number of pixels at a given column offset inside the sub-matrices. The proposed dissimilarity measure is extensively compared with other well-known approximate methods for image comparison in the state-of-the-art. Experiments demonstrate the superiority of the proposed approximate measure in terms of execution time with respect to the exact method, and in terms of retrieval precision with respect to the other state-of-the-art methods.

Peer-review under responsibility of KES International

Keywords: similarity measure; approximate matching; ACSM; pattern matching; image analysis

2000 MSC: 94A08, 68T10, 68P20

1. Introduction

In recent years, different methods have been proposed in Content-Based Image Retrieval (CBIR), adopting an approximate strategy for feature extraction or similarity computation.

SIMPLIcity is a retrieval system[1] where an image is divided into regions represented by feature vectors, defining the semantic type of the image. Similarity score, based on a greedy method for finding the region matching, is computed only between the query image and images belonging to the same semantic class. Carson et al.[2] introduce the *Blobworld* system, where each image is represented as a set of blobs corresponding to regions modeled as feature vectors. Query image is submitted by its blob representation, from which only a few relevant blobs are selected for matching. Also, an approximate indexing method is introduced to detect images relevant to a given query. In VisualSEEk[3] each image is segmented into regions characterized by color and spatial features. Color is represented by color sets, based on the assumption that regions are characterized by a few equally salient colors. An indexing method based on color properties, region centroids and minimum bounding rectangles is employed for retrieval speed-

* Corresponding author. Tel.: +39-0984-494783.
E-mail address: aamelio@dimes.unical.it

Peer-review under responsibility of KES International
doi:10.1016/j.procs.2016.08.194

up. Furthermore, NeTra system[4] is an image retrieval framework, where texture, color and shape information are extracted from the image and used to segment the image into homogeneous regions. Each image region is represented with a small quantity of colors, associated to a subset of colors from a color codebook, shape and texture content. Indexing strategies in terms of color, texture and shape are adopted for fast image retrieval. Also, Lv et al.[5] propose a new image retrieval system, dividing the image into homogeneous regions, represented as bit vectors and enveloped into a single image feature vector, such that the L_1 norm between two vectors is an approximation of the Earth Mover's Distance (EMD) between the corresponding images. Querying is performed by bit vector transformation for the query image. Again, SIFT feature strategy[6] efficiently identifies local image features through a filtering procedure finding key points in scale space. They are computed by modeling blurred image gradients in different orientations and scales by adopting Difference of Gaussians (DoG) as approximation of Laplacian of Gaussian (LoG). A nearest-neighbor indexing method is employed for determining the matches of key points between images. SURF descriptor[7] approximates LoG with box filter, computed by using the integral images. Non-maximum suppression is applied for detection of the key points, and wavelet responses employed for feature description. An indexing phase considering the sign of the Laplacian is also proposed, improving the speed of matching the key points between images. Finally, Wan et al.[8] propose a deep learning framework for CBIR consisting in learning image features by training large-scale deep Convolutional Neural Networks (CNNs). The activations of the last three fully connected layers are considered for image feature representation directly using one of the activation features, or adopting similarity learning algorithms to refine the previously obtained features, or re-training the CNNs on the new image dataset by initializing the network with the previously trained parameters.

In this paper, a new patch-based method for approximate dissimilarity computation among images is proposed. The method extends the state-of-the-art Average Common Sub-matrix algorithm[9,10], introduced for exact computation of the image dissimilarity. It is pixel-based and does not need to extract any feature representation, considering the image as a pixel matrix. Accordingly, given two images, the area of their largest common sub-matrices is computed to quantify the dissimilarity between them. The largest common sub-matrices are detected by exact matching of the sub-matrices inside the two images. Hence, matching is performed by comparing the sub-matrices in the two images pixel by pixel. As an extension, the proposed approach introduces a new approximation strategy in dissimilarity computation, which is pursued by omitting some pixels at regular intervals inside the sub-matrices. For this reason, matching of the sub-matrices is performed by considering a subset of pixels. It implies that two sub-matrices are considered to perfectly match only if a subset of their pixels perfectly matches. It determines a meaningful improvement in execution time of the dissimilarity computation, at the expense of a low accuracy decrease. The concept is similar to that of *"don't cares"*[11] and pattern matching with *k-mismatches*[12] in 2D arrays. The main difference is the location of the omitted pixels, which is constrained, established a priori in this case and regularly determined in the sub-matrices for the dissimilarity computation. To the best of our knowledge, it is the first time that image dissimilarity is computed by employing a similar approach beforehand.

The paper is organized as follows. Section 2 recalls the main concepts underlying the Average Common Sub-matrix algorithm. Section 3 introduces the approximation strategy, describes the approximation method and presents the algorithm for approximate matching. Section 4 describes the experiment. Section 5 presents the results of the experiment and discusses them. Finally, Section 6 makes the conclusions.

2. ACSM dissimilarity measure

Average Common Sub-matrix (ACSM) is an exact method for computing the dissimilarity among images[9,10]. It is based on the concept that two images are dissimilar to each other if they have a few number of small regions in common. On the contrary, if the two images have a relevant number of large regions in common, they are considered similar to each other. Next, we briefly recall the main concepts underlying ACSM dissimilarity measure.

Let I_A and I_B be two images, with corresponding pixel matrices $\mathbf{A}$ and $\mathbf{B}$, of size respectively $N \times N$ and $M \times M$, and let Σ be a finite alphabet on which $\mathbf{A}$ and $\mathbf{B}$ are defined. ACSM evaluates the average area of the sub-matrices in $\mathbf{A}$ exactly matching inside $\mathbf{B}$ to quantify the similarity between I_A and I_B. If another image I_D with corresponding matrix $\mathbf{D}$ defined on Σ is considered, it is noted that I_A is more similar to I_B than to I_D if the average area of the sub-matrices in common between $\mathbf{A}$ and $\mathbf{B}$ is bigger than the same average area between $\mathbf{A}$ and $\mathbf{D}$. The dissimilarity between I_A and

I_B is computed starting from the obtained similarity value, considering also the size of the matrices **A** and **B** and the similarity value of I_A with itself.

Given a generic matrix **Z** and a position (i, j) in **Z**, it is denoted as $Z^n_{i,j}$ the square sub-matrix starting at that position, of size n and area equal to $(n \times n)$. Accordingly, let $A^t_{i,j}$, $n = t$, be the biggest square sub-matrix of **A** starting at that position whose area is equal to $(t \times t)$ exactly matching a sub-matrix $B^t_{h,k}$ starting from a certain position (h, k) in **B**. The area $(t \times t)$ of $A^t_{i,j}$ is denoted as $W(i, j)$. In the case of $n = min\{i, j\}$, $A^n_{i,j}$ is maximal at that position, and its area $min\{i, j\}^2$ is denoted as $W_D(i, j)$. The similarity measure between the images I_A and I_B is realized by considering the area of the biggest common sub-matrices for all the positions (i, j), $i, j = 1...N$ inside **A** matrix and computing the average value. Consequently, ACSM is defined between I_A and I_B as follows:

$$S_\alpha(I_A, I_B) = \sum_{i=1}^{N} \sum_{j=1}^{N} W(i, j)/N^2 \ s.t.\ W(i, j) \geq \alpha, \tag{1}$$

where α is a parameter fixing the minimum area of the considered sub-matrices. Given Eq.(1), the symmetric dissimilarity measure is computed as:

$$d_s(I_A, I_B) = d_s(I_B, I_A) = \frac{d_\alpha(I_A, I_B) + d_\alpha(I_B, I_A)}{2}, \tag{2}$$

where $d_\alpha(I_A, I_B)$ is defined as:

$$d_\alpha(I_A, I_B) = \frac{log(M^2)}{S_\alpha(I_A, I_B)} - \frac{log(N^2)}{S_\alpha(I_A, I_A)}, \tag{3}$$

and $S_\alpha(I_A, I_A)$ is the similarity measure of **A** with itself.

The straightforward algorithm to compute the ACSM dissimilarity measure between I_A and I_B can be summarized in the following main points:

1. For each position (i, j) in **A**:
   (a) find the area $W(i, j)$ of the biggest square sub-matrix $A^t_{i,j}$ greater than or equal to α starting at that position, **exactly matching** a sub-matrix $B^t_{h,k}$ at some position (h, k) in **B**,
   (b) find the area $W_D(i, j)$ of the square sub-matrix $A^{min\{i,j\}}_{i,j}$ of maximal size greater than or equal to α and starting at that position,
   (c) sum in $W_\alpha(A, B)$ and in $W_\alpha(A, A)$ respectively the obtained $W(i, j)$ and $W_D(i, j)$,
2. divide $W_\alpha(A, B)$ and $W_\alpha(A, A)$ by N^2 for obtaining $S_\alpha(I_A, I_B)$ and $S_\alpha(I_A, I_A)$,
3. calculate $d_\alpha(I_A, I_B)$ as in Eq.(3) by using $S_\alpha(I_A, I_B)$ and $S_\alpha(I_A, I_A)$,
4. repeat steps 1-3 for **B** and **A** and obtain $d_\alpha(I_B, I_A)$,
5. compute $d_s(I_A, I_B)$ as in Eq.(2) by using $d_\alpha(I_A, I_B)$ and $d_\alpha(I_B, I_A)$.

If there is no sub-matrix for some position (i, j) in **A** of area greater than or equal to α exactly matching in **B**, that position won't give any contribution in the dissimilarity measure.

However, if it is supposed that α parameter is equal to 1, in point 1 (a) of the algorithm, the biggest square sub-matrix $A^t_{i,j}$ at position (i, j) in **A** is searched by considering all the square sub-matrices $A^n_{i,j}$, $n = 1...min\{i, j\}$, having the main diagonal up to that position. This means that all these sub-matrices are examined for possible matching with some sub-matrix $B^n_{h,k}$ inside **B**. Procedure is started from the sub-matrix of maximal size $n = min\{i, j\}$ at position (i, j) of **A**. If matching is detected with a sub-matrix in B, the procedure is stopped, otherwise it is continued by examining always smaller sub-matrices of main diagonal up to that position and verifying the matching inside **B**, until the size of the sub-matrix reaches the value of $n = 1$, which corresponds to only one pixel. Point 1 (b) is useful to find the similarity of I_A with itself. It is easily pursued by considering that at position (i, j) of **A** the biggest square sub-matrix $A^t_{i,j}$ exactly matching a some sub-matrix in **A** itself is that of maximal size $t = min\{i, j\}$. It is worth to note that all the positions (i, j) inside the **A** matrix are examined. Furthermore, the size of the sub-matrices is not fixed a priori and the

contribution of all the pixels of the sub-matrices is considered for determining if a positive match occurs. Accordingly, the match between two sub-matrices will have a positive result if all the corresponding pixels in the two sub-matrices perfectly match. For this reason, ACSM algorithm realizes an exact procedure for computing the dissimilarity among images.

Finding exact match of a sub-matrix $A^n_{i,j}$ inside the **B** matrix (see point 1 (a) in the algorithm) takes $O(M^2 n \times n)$ time by adopting a brute force approach, because the exact match of $A^n_{i,j}$ inside **B** has to be verified at each position (h, k) of **B** [13]. It can be reduced to $O(M^2)$ which is independent from the size of $A^n_{i,j}$, determining a worst case complexity of $O(M^2N^3)$ time for ACSM computation [9]. Furthermore, it has been proved that this complexity can be further reduced to $O(M^2N^2log(N))$ by adopting a binary strategy [9]. Finally, a more efficient version of the algorithm has been introduced, employing a generalized suffix tree in two dimensions, for reducing the complexity to $O(M^2 + N^2)$. However, the generalized suffix tree construction takes $O(M^2(logM + log|\Sigma|) + N^2(logN + log|\Sigma|))$ time [14], and it can be particularly demanding if the size of the images is large and the variability of the pixel values is high. Hence, the temporal cost of ACSM algorithm needs further investigation and reduction of its execution time still remains an open problem.

Next, we investigate the effect in terms of execution time and performances of omitting a portion of pixels for ACSM dissimilarity computation. In particular, we start our analysis from the aforementioned straightforward ACSM algorithm where exact match is performed by the brute force approach. In such a context, an approximation is introduced in matching procedure, pursued by omitting the match of some pixels at regular intervals. Accordingly, a match between the two sub-matrices $A^n_{i,j}$ and $B^n_{h,k}$ is considered positive even if only a subset of their pixels successfully matches. Consequently, we relax the constraint and impose that not all the pixels of the two sub-matrices have to be equal for obtaining a positive match.

3. Approximate ACSM

3.1. The method

The introduced concept of **approximation** determines a variation in the ACSM dissimilarity. In particular, I_A and I_B are considered as dissimilar to each other if a few image regions with small area are in common between **A** and **B** with a certain degree of approximation. On the contrary, if **A** and **B** *approximately* share a sufficient number of large regions, their corresponding images I_A and I_B will be considered as similar to each other. Accordingly, given the **A** matrix, the average area of its sub-matrices *approximately* matching inside **B** is computed for similarity evaluation between I_A and I_B. Also, I_A will be more similar to I_B than to I_D if the average area of the sub-matrices in **A** *approximately* matching with sub-matrices in **B** is greater than the same average area in **D**.

Let (i, j) be a given position in **A**, the method finds the biggest square sub-matrix $A^{\hat{t}}_{i,j}$, starting at that position in **A** whose area is equal to $(\hat{t} \times \hat{t})$, *approximately* matching a sub-matrix $B^{\hat{t}}_{h,k}$ inside **B** starting at some position (h, k) in **B**. Area $(\hat{t} \times \hat{t})$ of sub-matrix $A^{\hat{t}}_{i,j}$ is denoted as $\hat{W}(i, j)$. The method computes the area of such sub-matrices for all the positions (i, j), $i, j = 1...N$ inside **A** and determines the average value. Consequently, ACSM similarity measure with approximation, computed between I_A and I_B, is now defined as:

$$\hat{S}_\alpha(I_A, I_B) = \sum_{i=1}^{N} \sum_{j=1}^{N} \hat{W}(i, j)/N^2 \ s.t. \ \hat{W}(i, j) \geq \alpha, \tag{4}$$

Similarly to Eq.(3), ACSM dissimilarity measure between I_A and I_B becomes:

$$\hat{d}_\alpha(I_A, I_B) = \frac{log(M^2)}{\hat{S}_\alpha(I_A, I_B)} - \frac{log(N^2)}{S_\alpha(I_A, I_A)}, \tag{5}$$

Finally, ACSM symmetric dissimilarity measure is the following:

$$\hat{d}_s(I_A, I_B) = \hat{d}_s(I_B, I_A) = \frac{\hat{d}_\alpha(I_A, I_B) + \hat{d}_\alpha(I_B, I_A)}{2}. \tag{6}$$

It is understandable that t' can be greater than t, because approximation relaxes the match constraint. On the contrary, there is no variation in computing $S_\alpha(I_A, I_A)$, because the area of the sub-matrix of maximal size at a given position does not depend on approximation.

3.2. Approximation strategy

Approximate match between $A^n_{i,j}$ and $B^n_{h,k}$ is realized by omitting a controlled number of pixels inside the two sub-matrices at a fixed column interval. Consequently, $A^n_{i,j}$ and $B^n_{h,k}$ will have a positive match only if the remaining pixels at the corresponding positions match. The procedure of the approximate match in pseudo code, called *approximateMatch*, is depicted in Figure 1.

```
approximateMatch(A^n_{i,j}, B^n_{h,k}, Δc){
  1. r = 1
  2. while (r <= n)
  3.     if (r%2)==0
  4.         c := 2
  5.     else
  6.         c := 1
  7.     while c <= n
  8.         if A^n_{i,j}(r, c) ≠ B^n_{h,k}(r, c)
  9.             return false
  10.        end
  11.        c = c + Δc
  12.    end
  13. r = r + 1
  14. end
  15. return true
}
```

Fig. 1. *ApproximateMatch* procedure

It has three input parameters: the first sub-matrix $A^n_{i,j}$ of $\mathbf{A}$, the second sub-matrix $B^n_{h,k}$ of $\mathbf{B}$ and a column offset Δc. The output of the procedure is a boolean value indicating if $A^n_{i,j}$ and $B^n_{h,k}$ have a positive match (true) or a negative match (false). Differently from the exact match procedure, *approximateMatch* introduces the parameter Δc which is a column offset for pixel matching. The procedure iterates step by step along the rows of $A^n_{i,j}$ and $B^n_{h,k}$ and by steps of size Δc along the columns of the two sub-matrices. In order to optimize the coverage of the area of $A^n_{i,j}$ and $B^n_{h,k}$, the column index is unaligned between the odd rows and the even rows.

The procedure runs along the rows of the sub-matrices $A^n_{i,j}$ and $B^n_{h,k}$ at constant step of 1 (see lines 1 and 13). Meanwhile, it verifies if the row index r is an even number (see line 3). If it is so, the column index c is initialized to 2, otherwise it is initialized to 1, for realizing the column disalignment (see lines 4-6). In lines 7-12, the procedure iterates along the columns of $A^n_{i,j}$ and $B^n_{h,k}$, fixed the row index r. Pixels corresponding at position (r, c) inside $A^n_{i,j}$ and $B^n_{h,k}$ are matched (see line 8) and the procedure is stopped as soon as a mismatch is detected (see line 9). Column index c is updated by steps of Δc (see line 11). If no mismatches are detected between pixels in $A^n_{i,j}$ and $B^n_{h,k}$, the procedure returns a positive match between the two sub-matrices (see line 15). Figure 2 shows an example of *approximateMatch* execution.

$$A = \begin{vmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & \mathbf{1} \end{vmatrix} \quad A^4_{5,5} = \begin{vmatrix} \mathbf{0} & 0 & \mathbf{1} & 1 \\ 0 & \mathbf{0} & 0 & \mathbf{1} \\ \mathbf{0} & 1 & \mathbf{0} & 1 \\ 1 & \mathbf{1} & 0 & \mathbf{1} \end{vmatrix} \qquad B = \begin{vmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & \mathbf{1} \\ 0 & 0 & 1 & 0 & 1 \end{vmatrix} \quad B^4_{4,5} = \begin{vmatrix} \mathbf{0} & 1 & \mathbf{1} & 0 \\ 1 & \mathbf{0} & 0 & \mathbf{1} \\ \mathbf{0} & 1 & \mathbf{0} & 0 \\ 0 & \mathbf{1} & 1 & \mathbf{1} \end{vmatrix}$$

Fig. 2. Execution of *approximateMatch* procedure on the sub-matrices $A^4_{5,5}$ and $B^4_{4,5}$ with column offset $\Delta c = 2$. Matched pixels are in bold

Procedure runs step by step along the rows and by steps of $\Delta c = 2$ along the columns of $A^4_{5,5}$ and $B^4_{4,5}$. Pixels selected for matching from the procedure are marked in bold. It is observable that pixels are unaligned in correspondence of consecutive rows (i.e. row 1 and row 2) and aligned only in correspondence of alternating rows (i.e. row 1 and row 3). Only pixels at positions $(r, c + \Delta c)$ are considered for matching inside $A^4_{5,5}$ and $B^4_{4,5}$ (in bold). For example, in row 1, only pixels at positions $(1, 1)$ and $(1, 3)$ are matched inside the two sub-matrices. All the other pixels are

omitted and not considered for matching. Pixels in $A^4_{5,5}$ and $B^4_{4,5}$ are matched, starting from the first selected position $(1, 1)$ and continuing from left to right and from top to bottom. In this case, all the pixels at the selected positions perfectly match in $A^4_{5,5}$ and $B^4_{4,5}$. It implies that the two sub-matrices will have a positive match and that the procedure will return true.

3.3. The algorithm

ACSM algorithm with approximation is composed of the same points of the exact ACSM algorithm described in Section 2. However, the brute force procedure finding exact matches of the sub-matrices of **A** inside **B** is modified by introducing the approximate match procedure. Hence, the approximate algorithm becomes the following:

1. For each position (i, j) in **A**:
   (a) find the area $\hat{W}(i, j)$ of the biggest square sub-matrix $A^{\hat{t}}_{i,j}$ greater than or equal to α starting at that position, **approximately matching** a sub-matrix $B^{\hat{t}}_{h,k}$ at some position (h, k) in **B**,
   (b) find the area $W_D(i, j)$ of the square sub-matrix $A^{min\{i,j\}}_{i,j}$ greater than or equal to α starting at that position,
   (c) sum in $\hat{W}_\alpha(A, B)$ and in $W_\alpha(A, A)$ respectively the obtained $\hat{W}(i, j)$ and $W_D(i, j)$,
2. divide $\hat{W}_\alpha(A, B)$ and $W_\alpha(A, A)$ by N^2 for obtaining $\hat{S}_\alpha(I_A, I_B)$ and $S_\alpha(I_A, I_A)$,
3. calculate $\hat{d}_\alpha(I_A, I_B)$ as in Eq.(5) by using $\hat{S}_\alpha(I_A, I_B)$ and $S_\alpha(I_A, I_A)$,
4. repeat steps 1-3 for **B** and **A** and obtain $\hat{d}_\alpha(I_B, I_A)$,
5. compute $\hat{d}_s(I_A, I_B)$ as in Eq.(6) by using $\hat{d}_\alpha(I_A, I_B)$ and $\hat{d}_\alpha(I_B, I_A)$.

Figure 3 shows the biggest common square sub-matrices, extracted from ACSM algorithm with approximation, between two example binary images I_A and I_B with Δc fixed to 2 and α equal to 4.

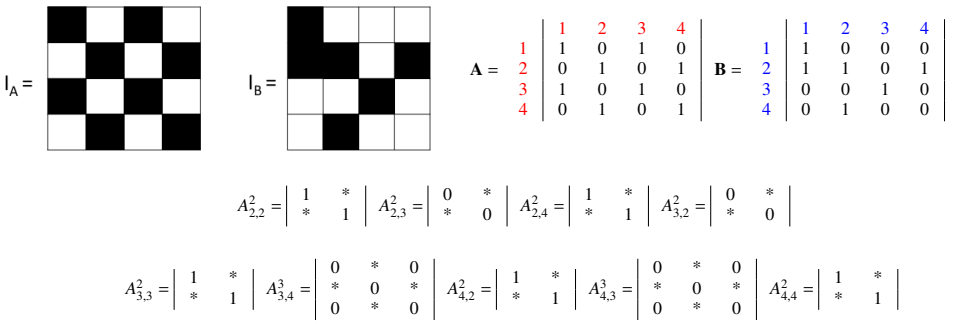

Fig. 3. Biggest common square sub-matrices, extracted from ACSM algorithm with approximation, between two example binary images I_A and I_B whose pixel matrices are **A** and **B**. Black pixels are identified as '1' and white pixels are identified as '0'. Symbol '*' indicates that pixel at that position is not considered for matching from the *approximateMatch* procedure

At position $(1, 1)$ of **A**, the biggest square sub-matrix approximately matching inside **B** is $A^1_{1,1} = \left| \; 1 \; \right|$. However, it is not taken into account in the dissimilarity computation, because its area is less than $\alpha = 4$. The same is for positions $(1, 2)$, $(1, 3)$, $(1, 4)$, $(2, 1)$, $(3, 1)$, and $(4, 1)$.

An example is at position $(3, 4)$ of **A**, where the biggest square sub-matrix approximately matching a sub-matrix in **B** at position $(3, 4)$ is $A^3_{3,4}$. It is extracted from **A** by considering its bottom-right corner at $(3, 4)$ and diagonal up to that position, and checking the approximate match (with $\Delta c = 2$) inside **B**. The set of all the candidate square sub-matrices at position $(3, 4)$ and of area $\geq \alpha = 4$, are the following:

$$A^3_{3,4} = \begin{vmatrix} 0 & * & 0 \\ * & 0 & * \\ 0 & * & 0 \end{vmatrix}, \quad A^2_{3,4} = \begin{vmatrix} 0 & * \\ * & 0 \end{vmatrix}.$$

Another example is at position $(4, 4)$ of $\mathbf{A}$, where the candidate square sub-matrices with bottom-right corner at that position and diagonal up to that position, of area $\geq \alpha = 4$, approximated by $\Delta c = 2$, are the following:

$$A^4_{4,4} = \begin{vmatrix} 1 & * & 1 & * \\ * & 1 & * & 1 \\ 1 & * & 1 & * \\ * & 1 & * & 1 \end{vmatrix}, \quad A^3_{4,4} = \begin{vmatrix} 1 & * & 1 \\ * & 1 & * \\ 1 & * & 1 \end{vmatrix}, \quad A^2_{4,4} = \begin{vmatrix} 1 & * \\ * & 1 \end{vmatrix}.$$

In this case, the maximal square sub-matrix of $\mathbf{A}$ at position $(4, 4)$ is $A^4_{4,4}$. However, it does not match approximately any sub-matrix inside $\mathbf{B}$. The same is for $A^3_{4,4}$. Consequently, $A^2_{4,4}$ is selected as the biggest square sub-matrix at position $(4, 4)$ of $\mathbf{A}$. In fact, it approximately matches at positions $(2, 2)$ and $(3, 3)$ inside $\mathbf{B}$.

4. Experimentation

Next, we evaluate the ability of ACSM dissimilarity measure with approximation in capturing the differences among the images. It is performed by embedding the dissimilarity measure inside the image retrieval context. In particular, let Q be a query image and D an image dataset. The aim is to retrieve the top k images in D which are the most (less) (dis)similar to Q. Success in retrieval mostly depends on the (dis)similarity measure adopted for image comparison.

For retrieval evaluation, the well-known *retrieval precision* criterion has been employed[9,15] . It is defined as the ratio between the T number of relevant images which are retrieved and the total k number of images which are retrieved[9]. Images are considered as relevant if they belong to the same class of Q. Because retrieval result strongly depends on the selected query, procedure is repeated multiple times for different queries and average precision results are computed. In particular, given N queries, retrieval is performed N times on D. For each query Q, the top k most (less) (dis)similar images to the query are retrieved. It is realized by computing the (dis)similarity between Q and each image $I_D \in D$, by sorting the images in D based on the (dis)similarity score, and by selecting the k images with the highest (lowest) score. Then, the T relevant images are selected from the top k retrieved images and retrieval precision for Q is computed. Finally, retrieval precision is averaged on all the queries.

Experimentation has been performed on a subset of 50 grayscale images employed in[9] and on 4 randomly selected queries from Columbia Object Image Library (COIL-20) dataset, and on a subset of 200 RGB color images from Caltech-256 Object Category Dataset, freely available online at `http://www.vision.caltech.edu/Image_Datasets/Caltech256/`. To create the dataset of 200 images, a subset of 10 object classes has been randomly selected. Then, a subset of 20 images has been randomly chosen for each class. Finally, a set of 5 queries has been randomly selected from the 10 objects. Images have been resized to 128×128.

ACSM dissimilarity measure with approximation has been compared to exact ACSM dissimilarity measure and to other two methods based on well-known approximate descriptors for image representation[16,17,18]: (dis)similarity strategy adopting SIFT features[6] and (dis)similarity method employing SURF features[7]. The approximation strategy can be considered as a sampling method, selecting only a subset of pixels for matching. For this reason, ACSM dissimilarity measure with approximation has been compared to exact ACSM dissimilarity measure employed on the original images and on the resized images to a smaller size (i.e. 64×64). Resizing method was based on *nearest neighbor*, *bilinear* and *bicubic* interpolation.

Image comparison based on SIFT and SURF local features has been performed by employing two different methods. In particular, let d_x and d_y be two images to compare and p_x and p_y their corresponding local descriptor sets (SIFT descriptors or SURF descriptors). Dissimilarity between d_x and d_y has been computed as *1-NN Similarity Average*[19]. The only difference is that euclidean distance has been adopted instead of similarity between the local descriptors[20]. Hence, dissimilarity between d_x and d_y has been evaluated as the average euclidean distance between the local descriptors in d_x and their nearest neighbor descriptors in d_y:

$$d^1(d_x, d_y) = \frac{1}{|d_x|} \sum_{p_x \in d_x} min_{p_y \in d_y}(d(p_x, p_y)) \tag{7}$$

where $|d_x|$ is the number of local descriptors of d_x. Similarity between d_x and d_y has been computed as *Percentage of Matches*[19], defined as the percentage of local descriptors in d_x having a correspondence in d_y:

$$s^m(d_x, d_y) = \frac{1}{|d_x|} \sum_{p_x \in d_x} m(p_x, d_y) \quad (8)$$

where $m(p_x, d_y)$ assumes value of 1 if p_x has a correspondence in d_y, 0 otherwise.

Experimentation has been performed in MATLAB R2012a and in Eclipse 3.8 with Java 1.7, on a laptop computer Intel Core i7 2.3 GHz, 8 GB RAM and UNIX platform. A trial and error procedure has been employed on benchmark images different from the datasets for tuning the α parameter. Hence, it has been fixed to 3 in all the experiments.

5. Results and Discussion

Two experiments have been performed for analysis of the ACSM dissimilarity measure with approximation. The first one evaluates if an improvement of execution time is obtained in dissimilarity computation when ACSM measure with approximation is employed, with respect to the exact ACSM measure on the original images and on the resized images, and the change in retrieval precision when approximation progressively grows. The second one evaluates the retrieval precision of ACSM dissimilarity measure with approximation compared to the other approximate (dis)similarity methods. In the following, exact ACSM dissimilarity measure on the original images will be referred as ACSM, exact ACSM dissimilarity measure on the resized images as $\text{ACSM}_{nearest(64\times64)}$ (nearest neighbor interpolation), $\text{ACSM}_{bilinear(64\times64)}$ (bilinear interpolation), $\text{ACSM}_{bicubic(64\times64)}$ (bicubic interpolation), and ACSM dissimilarity measure with approximation as A-ACSM.

Table 1 shows the results of the first experiment, for each value of Δc varying from 3 to 9. In fact, it is proved that very small improvements in execution time occur when Δc is equal to 1 and 2, and that a value of Δc higher than 9 determines an abrupt lowering of retrieval precision for some types of images. Exact ACSM and approximate ACSM dissimilarity measures are run on the subset of 50 images of COIL-20 dataset. Retrieval precision is averaged on the 4 queries for the top k less dissimilar images, where k varies from 1 to 5. Furthermore, for each query, the time (in seconds) spent to compute the dissimilarity between the given query and the complete dataset, is considered. The average time on all the 4 queries together with the average retrieval precision are reported inside the table. It is observable that the maximum retrieval precision is obtained from ACSM. In fact, it reaches the highest set of values of 1.00, 1.00, 0.92, 0.94 and 0.75 for k varying from 1 to 5. However, average execution time per query is pretty high, reaching a value of 10258 s. As soon as the value of Δc increases, the execution time starts decreasing. In particular, when the value of Δc varies from 0 (in the case of ACSM) to 3, execution time lowers from 10258 s to 3360 s, which is really noticeable. On the contrary, the corresponding retrieval precision slightly decreases. In fact, it remains equal to 1.00 for $k = 1$ and decreases at most of 0.10 for k varying from 2 to 5. For values of Δc higher than 3, the execution time exhibits a slower but meaningful decrease, until it reaches the value of 2512 s for Δc equal to 9. Accordingly, the retrieval precision remains satisfactory for all the k values from 1 to 5. In particular, for $k = 1$, A-ACSM reaches in all the cases the same maximum value of ACSM which is 1.00. When $k = 2$, A-ACSM obtains the same maximum retrieval precision of ACSM equal to 1.00 for $\Delta c = 4, 6, 7$ and 8 (4 cases out of 7). For $k = 3$, the retrieval precision of ACSM is 0.92, which is equal to the retrieval precision of A-ACSM when $\Delta c = 4, 5$. However, when $\Delta c = 8$, the retrieval precision of A-ACSM has the maximum value of 1.00, outperforming the value of 0.92 obtained from ACSM. Again, for $k = 4$, the retrieval precision of ACSM, which is 0.94, overcomes the retrieval precision of A-ACSM for all the values of Δc. In fact, A-ACSM reaches 0.81 when $\Delta c = 3$ and 4, 0.75 when $\Delta c = 5, 6, 8$, and 0.62 when $\Delta c = 7, 9$. Finally, when $k = 5$, the retrieval precision of ACSM, equal to 0.75, is slightly higher than the retrieval precision of A-ACSM. In fact, it takes a value of 0.65 for $\Delta c = 3, 4, 5, 8, 9$, a value of 0.70 for $\Delta c = 6$ and a value of 0.60 for $\Delta c = 7$. In conclusion, A-ACSM has the advantage to strongly lower the execution time, while retrieval precision slowly decreases as the value of Δc becomes higher. It indicates that the loss generated from the approximation method does not strongly affect the efficacy of the base algorithm, while it strongly improves its efficiency. Furthermore, in some cases, an increase of the value of Δc determines an improvement in retrieval precision (i.e. Δc varying from 7 to 8). It demonstrates that some pixels can be misleading in image comparison and should be omitted for a better dissimilarity evaluation. Hence, introducing approximation sometimes becomes a valid help for retrieval improvement. On the other hand, it is observable that execution time of ACSM employed on the resized images is reasonably smaller. However, retrieval precision obtained from A-ACSM outperforms in many cases that obtained from ACSM employed on the resized images. Specifically, A-ACSM overcomes $\text{ACSM}_{nearest(64\times64)}$ and

$ACSM_{bilinear(64\times64)}$ in retrieval precision for $k = 1, 2, 3$, and $ACSM_{bicubic(64\times64)}$ for $k = 2$. For the other k values, ACSM employed on the resized images reaches the low retrieval precision values which are reached from A-ACSM for some Δc.

Table 1. Average execution time (in seconds) and retrieval precision of exact ACSM and ACSM with approximation, with Δc parameter varying from 3 to 9, for the top k less dissimilar images, with k varying from 1 to 5, on a subset of COIL-20 dataset

Method	k=1	k=2	k=3	k=4	k=5	Avg. Time Q (s)
ACSM	1.00	1.00	0.92	0.94	0.75	10258
A-ACSM ($\Delta c = 3$)	1.00	0.87	0.75	0.81	0.65	3360
A-ACSM ($\Delta c = 4$)	1.00	1.00	0.92	0.81	0.65	3059
A-ACSM ($\Delta c = 5$)	1.00	0.87	0.92	0.75	0.65	2874
A-ACSM ($\Delta c = 6$)	1.00	1.00	0.83	0.75	0.70	2743
A-ACSM ($\Delta c = 7$)	1.00	1.00	0.83	0.62	0.60	2660
A-ACSM ($\Delta c = 8$)	1.00	1.00	1.00	0.75	0.65	2579
A-ACSM ($\Delta c = 9$)	1.00	0.87	0.83	0.62	0.65	2512
$ACSM_{bicubic(64\times64)}$	1.00	0.75	0.75	0.69	0.60	56.75
$ACSM_{bilinear(64\times64)}$	0.75	0.62	0.67	0.62	0.65	56.00
$ACSM_{nearest(64\times64)}$	0.75	0.62	0.67	0.62	0.65	56.00

Table 2 shows the results of the second experiment, in terms of average retrieval precision obtained from A-ACSM on the subset of 200 images and 5 queries of Caltech-256 dataset, for the top k less dissimilar images, where k varies from 1 to 10. A-ACSM results are compared on the same dataset to *1-NN Similarity Average* approach using SIFT descriptors ($SIFT_{avg}$), *Percentage of Matches* approach using SIFT descriptors ($SIFT_{match_{\%}}$), *1-NN Similarity Average* approach employing SURF descriptors ($SURF_{avg}$), *Percentage of Matches* approach adopting SURF descriptors ($SURF_{match_{\%}}$). Because *1-NN Similarity Average* using euclidean distance is a dissimilarity measure, the top k less dissimilar images are selected for retrieval precision. On the contrary, *Percentage of Matches* determines a similarity measure, for which the top k most similar images are selected. Although the dataset is particularly complex, it is worth to note that A-ACSM outperforms all the competitor approaches in 7 cases out of 10. In fact, the retrieval precision obtained from A-ACSM outperforms the retrieval precision obtained from the other approaches for k=1 and for k varying from 5 to 10. Only in 3 cases corresponding to $k = 2, 3, 4$, A-ACSM obtains the same results of $SURF_{avg}$. In fact, it reaches 0.50 for $k = 2$, 0.47 for $k = 3$, and 0.40 for $k = 4$. Obtained results demonstrate the superiority of the new proposed approximate method in capturing the visual similarity with respect to the well-known approximate methods in the state-of-the-art.

Table 2. Average retrieval precision of A-ACSM, *1-NN Similarity Average* approach using SIFT descriptors ($SIFT_{avg}$), *Percentage of Matches* approach using SIFT descriptors ($SIFT_{match_{\%}}$), *1-NN Similarity Average* approach employing SURF descriptors ($SURF_{avg}$), *Percentage of Matches* approach adopting SURF descriptors ($SURF_{match_{\%}}$), on a subset of Caltech-256 dataset for the top k most similar (less dissimilar) images, where k varies from 1 to 10. In bold are the cases when A-ACSM outperforms all the other approaches.

k	$SIFT_{avg}$	$SIFT_{match_{\%}}$	$SURF_{avg}$	$SURF_{match_{\%}}$	A-ACSM
1	0.20	0.20	0.40	0.40	**0.60**
2	0.40	0.30	*0.50*	0.30	*0.50*
3	0.33	0.33	*0.47*	0.27	*0.47*
4	0.30	0.30	*0.40*	0.30	*0.40*
5	0.32	0.28	0.32	0.24	**0.44**
6	0.30	0.27	0.33	0.23	**0.43**
7	0.26	0.26	0.29	0.26	**0.43**
8	0.25	0.25	0.27	0.27	**0.40**
9	0.22	0.22	0.29	0.29	**0.40**
10	0.22	0.22	0.30	0.28	**0.38**

6. Conclusions

The paper presented a new approximate dissimilarity measure for image comparison in CBIR systems, extending the ACSM dissimilarity measure, based on the average area of the biggest square sub-matrices in common in the im-

ages. ACSM was pursued by exactly matching the sub-matrices of the different images at the corresponding positions pixel by pixel. Accordingly, a positive match between two sub-matrices was detected if a perfect correspondence of the pixels occurred. The proposed ACSM extension introduced an approximation method in matching of the sub-matrices. It was realized by omitting a portion of the pixels, at a given column offset, during the match. Accordingly, two sub-matrices perfectly matched only if a subset of their pixels was equal at the corresponding positions. Hence, an algorithm for computing the approximate measure was proposed and an example provided. Experimentation performed on two benchmark image datasets demonstrated the validity of the proposed approximate approach in both execution time and retrieval precision, when the method was compared with the exact ACSM measure and with other well-known approximate measures in the state-of-the-art. In particular, it achieved a huge improvement in execution time with respect to exact ACSM measure, while maintaining a satisfactory value of retrieval precision.

Future work will extend the experimentation to large databases and will test the approach in the field of biometrics.

References

1. Wang, J., Li, J., Wiederhold, G.. Simplicity: semantics-sensitive integrated matching for picture libraries. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 2001;**23**(9):947–963. doi:10.1109/34.955109.
2. Carson, C., Thomas, M., Belongie, S., Hellerstein, J., Malik, J.. Blobworld: A system for region-based image indexing and retrieval. In: Huijsmans, D., Smeulders, A., editors. *Visual Information and Information Systems*; vol. 1614 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. ISBN 978-3-540-66079-8; 1999, p. 509–517. URL: http://dx.doi.org/10.1007/3-540-48762-X_63. doi:10.1007/3-540-48762-X_63.
3. Smith, J.R., Chang, S.F.. Visualseek: A fully automated content-based image query system. In: *Proceedings of the Fourth ACM International Conference on Multimedia*; MULTIMEDIA '96. New York, NY, USA: ACM. ISBN 0-89791-871-1; 1996, p. 87–98. URL: http://doi.acm.org/10.1145/244130.244151. doi:10.1145/244130.244151.
4. Ma, W., Manjunath, B.. Netra: a toolbox for navigating large image databases. In: *Image Processing, 1997. Proceedings., International Conference on*; vol. 1. 1997, p. 568–571 vol.1. doi:10.1109/ICIP.1997.647976.
5. Lv, Q., Charikar, M., Li, K.. Image similarity search with compact data structures. In: *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*; CIKM '04. New York, NY, USA: ACM. ISBN 1-58113-874-1; 2004, p. 208–217. URL: http://doi.acm.org/10.1145/1031171.1031213. doi:10.1145/1031171.1031213.
6. Lowe, D.. Object recognition from local scale-invariant features. In: *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*; vol. 2. 1999, p. 1150–1157 vol.2. doi:10.1109/ICCV.1999.790410.
7. Bay, H., Tuytelaars, T., Van Gool, L.. Surf: Speeded up robust features. In: *Computer Vision ECCV 2006*; vol. 3951 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. ISBN 978-3-540-33832-1; 2006, p. 404–417. URL: http://dx.doi.org/10.1007/11744023_32. doi:10.1007/11744023_32.
8. Wan, J., Wang, D., Hoi, S.C.H., Wu, P., Zhu, J., Zhang, Y., et al. Deep learning for content-based image retrieval: A comprehensive study. In: *Proceedings of the ACM International Conference on Multimedia*; MM '14. New York, NY, USA: ACM. ISBN 978-1-4503-3063-3; 2014, p. 157–166. URL: http://doi.acm.org/10.1145/2647868.2654948. doi:10.1145/2647868.2654948.
9. Amelio, A., Pizzuti, C.. Average common submatrix: a new image distance measure. In: *17th international Conference on Image Analysis and Processing, (ICIAP 2013)*. LNCS 8156, Springer-Verlag; 2013, p. 170–180.
10. Amelio, A., Pizzuti, C.. A patch-based measure for image dissimilarity. *Neurocomputing* 2016;**171**:362–378. URL: http://dx.doi.org/10.1016/j.neucom.2015.06.044. doi:10.1016/j.neucom.2015.06.044.
11. Apostolico, A., Parida, L., Rombo, S.E.. Motif patterns in 2d. *Theoretical Computer Science* 2008;**390**(1):40 – 55. URL: http://www.sciencedirect.com/science/article/pii/S0304397507007645. doi:http://dx.doi.org/10.1016/j.tcs.2007.10.019.
12. Ranka, S., Heywood, T.. Two-dimensional pattern matching with k mismatches. *Pattern Recognition* 1991;**24**(1):31 – 40. URL: http://www.sciencedirect.com/science/article/pii/003132039190114K. doi:http://dx.doi.org/10.1016/0031-3203(91)90114-K.
13. Crochemore, M., Lecroq, T.. Pattern-matching and text-compression algorithms. *ACM Comput Surv* 1996;**28**(1):39–41. URL: http://doi.acm.org/10.1145/234313.234331. doi:10.1145/234313.234331.
14. Giancarlo, R.. A generalization of the suffix tree to square matrices, with applications. *SIAM Journal on Computing* 1995;**24**(3):520–562.
15. Tourassi, G.D., Harrawood, B.. Evaluation of information-theoretic similarity measures for content-based retrieval and detection of masses in mammograms. *Medical Physics* 2007;**34**(1):140–150.
16. Bicego, M., Lagorio, A., Grosso, E., Tistarelli, M.. On the use of sift features for face authentication. In: *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW '06. Conference on*. 2006, p. 35–35. doi:10.1109/CVPRW.2006.149.
17. Chincha, R., Tian, Y.. Finding objects for blind people based on surf features. In: *Bioinformatics and Biomedicine Workshops (BIBMW), 2011 IEEE International Conference on*. 2011, p. 526–527. doi:10.1109/BIBMW.2011.6112423.
18. Zagoruyko, S., Komodakis, N.. Learning to compare image patches via convolutional neural networks. *CoRR* 2015;**abs/1504.03641**. URL: http://arxiv.org/abs/1504.03641.
19. Amato, G., Falchi, F., Rabitti, F.. Landmark recognition in visito tuscany. In: Grana, C., Cucchiara, R., editors. *Multimedia for Cultural Heritage*; vol. 247 of *Communications in Computer and Information Science*. Springer Berlin Heidelberg. ISBN 978-3-642-27977-5; 2012, p. 1–13. URL: http://dx.doi.org/10.1007/978-3-642-27978-2_1. doi:10.1007/978-3-642-27978-2_1.
20. Hua, S., Chen, G., Wei, H., Jiang, Q.. Similarity measure for image resizing using sift feature. *EURASIP Journal on Image and Video Processing* 2012;**2012**(1):1–11. URL: http://dx.doi.org/10.1186/1687-5281-2012-6. doi:10.1186/1687-5281-2012-6.