Personalized Social Image Organization, Visualization, and Querying Tool using Low- and High-Level Features

Issa Ayoub
E.C.E Dept., School of Engineering
Lebanese American University (LAU)
36 Byblos, Lebanon
issa.ayoub@lau.edu

Karl J. Codoumi
E.C.E Dept., School of Engineering
Lebanese American University (LAU)
36 Byblos, Lebanon
karljoseph.codouni@lau.edu

Joe Tekli
E.C.E Dept., School of Engineering
Lebanese American University (LAU)
36 Byblos, Lebanon
joe.tekli@lau.edu.lb

***Abstract*—The purpose of this study is to create a software system to facilitate the organization of and searching for social images acquired from social sites on the Web (such as Facebook or Flikr), taking into account the images' features as well as user preferences. To achieve our goal, we design a solution based on image clustering, grouping together images sharing similar semantic and visual features, to simplify their organization and querying. This requires low-level and high-level image feature extraction and processing, where: low-level features represent color, texture, and shape image descriptors, whereas high-level features consist of textual descriptors extracted from image annotations and surrounding texts. Our system consists of modular components for: i) feature extraction and representation (low-level and high-level), ii) partitional image clustering (initial clustering phase executed when the user first connects to the system), iii) incremental clustering (updating clusters produces in the previous phase by processing newly published images), iv) fast image querying (using features of cluster representatives), and v) personalized images/search results visualization (using various user-chosen cluster display techniques). Preliminary experiments highlight the efficiency and practicality of our tool.**

***Keywords*—Social Web Images, Image Clustering, Content-based Image Retrieval, Text-based Image Retrieval, Personalized Image Organization.**

I. INTRODUCTION

In the past two decades, the amount of images published on the Web, especially on social sites like Facebook and Flikr, has been increasing exponentially. This was further fueled by the increasing availability of photo taking gadgets such as smart phones, pads, and tables, as well as the increased user connectivity to the Web using wireless network and mobile Internet connectivity. Yet, with the increased availability of social Web images comes the challenge of managing these images in a personalized manner, so that a user can efficiently organize and search for images based on her needs (e.g., grouping together and/or searching for similar images taken at a certain place and/or time, tagged with a certain friend, etc.).

To address this problem, we have designed and implemented a solution called SICOS, for Social Image Cluster-based Organization and Search, allowing to group together images sharing similar semantic and visual features, to simplify their organization and querying. This requires low-level and high-level image feature extraction and processing, where: low-level features represent color, texture, and shape image descriptors, whereas high-level features consist of textual descriptors extracted from image annotations and surrounding text.

The overall architecture of SICOS is shown in Fig. 1, and consists of 7 main components for: i) image retrieval from a social site (e.g., Facebook, Flikr, etc.), ii) feature extraction and representation (low-level and high-level), iii) image similarity computation, needed to perform: iv) partitional image clustering (initial clustering phased executed when the user first connects to the system), v) incremental clustering (updating clusters produces in the initial phase by processing newly published images), vi) image visualization (following different user chosen views, e.g., grid, fish-eye, etc.) and vii) fast image querying (using features of cluster representatives). It accepts as input: social Web images (downloaded from a social site like Facebook), user image organization parameters (highlighting the kinds of image features the user is interested in, as well as image organization parameters), and user search parameters (text-based and/or content based user queries). The system then performs image storage and organization following user organization preferences, and returns search results to answer user queries.

Different from existing image search and result organization solutions which are either: i) generic, addressing Web-based image processing (and not specifically geared toward social image processing), e.g., [1, 2], ii) computationally expensive, performing automatic face or object recognition, e.g., [3, 4], and/or event detection and identification, e.g., [4, 5], in indexing and searching for social images, and iii) requiring specific conditions or contextual data to work properly (cf. Section II); we provide here a computationally efficient solution integrating legacy techniques from Web-based and social image processing, requiring minimal contextual/input data, to provide the following functionality:

- Efficient indexing and storage of images with the corresponding feature information,
- Comparing images based on low-level visual features including: color, texture, and shape descriptors
- Comparing images based on high-level textual features, including: tags, captions, comments, and geographic location,
- Clustering images based on low-level and high-level feature similarities,
- Simple access to images through cluster representatives,
- Allowing different user-friendly cluster visualizations,
- Searching images based on low-level visual features,
- Searching images based on high-level textual features.

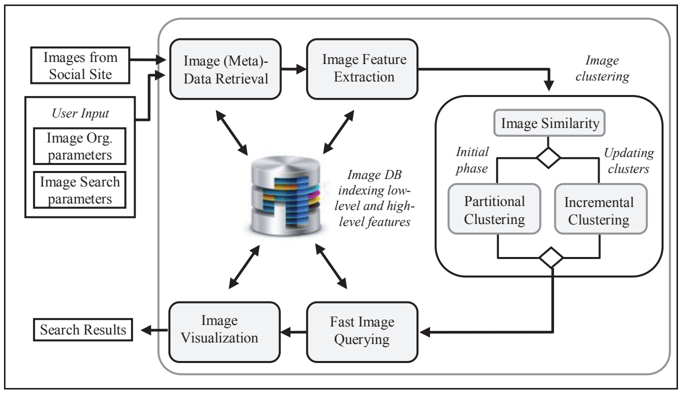


Fig. 1. Overall SICOS architecture.

The remainder of the paper is organized as follows. Section II briefly describes related works. Section III presents background information concerning the different techniques used in developing our solution. Section IV describes our SICOS system, its components, and functionality. Section V presents and discusses experimental results, before concluding with current directions in Section VI.

II. Related Works

Organizing and searching among personal images published on social sites can be extremely tedious and time consuming, since these can be published with or without proper user (publisher) generated descriptions. Hence, a user query (e.g., searching for pictures related to keyword query "*Byblos site"*) would return image results containing multiple topics mixed together, where the user (publisher) cannot be expected to have the time or patience to scroll through the huge result list. Things become even worse when one topic is overwhelming but is not what the user desires [6]. In addition, the user could entirely miss her search goal due to information and/or cognitive overload.

A possible solution is to help the user reformulate the query by suggesting alternative or more precise search criteria. For instance, query disambiguation and relaxation techniques [7-9] could be used to help narrow down the search result diversity (allowing more specific queries such as "*Byblos archeological site*" or "*ancient site in Byblos*"). Yet, such techniques do not entirely solve the problem since even the results of refined queries could be difficult to grasp (e.g., refined query "*ancient site in Byblos*" might return different images describing different sites in Byblos, such as the *Fortress*, the *old market*, the *old house by the shore*, the *roman ruins*, etc.).

Another possibility is to better organize the output information (prior and/or after query refinement), providing a means to facilitate the assimilation of the search results by the user. Hence, image search result organization [10] has been recently investigated as a simple and efficient solution to improving image retrieval quality on the Web [11]. Most methods in this context exploit image clustering (i.e., identifying groups of mutually similar images) as a methodology capable of topic extraction and returning semantically more meaningful image search results to the user [1, 2] (than merely a list of images sorted by similarity to the query). The general assumptions are that (i) mutually similar documents (e.g., images) tend to be similar to the same query [12], and (ii) semantically similar images (i.e., images with similar underlying topics) tend to be grouped together in some feature space [1]. In other words, identifying the different topics (clusters) of image search results would aid the user in understanding and navigating the result set, in order to efficiently identify her search request (cf. sample search result clusters in Figures 4-6).

While most existing image search result organization techniques have been developed for general purpose Web-based image retrieval systems, e.g., [1, 2, 10-12], few approaches have been developed to handle social Web images, e.g., [3, 13, 14]. One group of methods relies on automatic pattern recognition techniques [4], such as face, object, or clothes recognition [3] in order to identify individuals and then tag and organize images accordingly. Methods in this category are computationally expensive and require specific conditions to work properly [4], including: visible faces or distinctive clothing in images, and good lighting [14]. Another group of methods relies on user generated meta-data, namely tag names, allowing to organize images in tree-like structures such as Galois sub-hierarchies [13-15], aiming to incrementally capitalize on existing information by allowing images to inherit descriptions of other existing images. They make use of event extraction and identification techniques which are computationally expensive and require contextual meta-data (e.g., predefined event categories) [16, 17] which might not be always available.

In this study, we integrate different techniques from existing Web-based and social image processing, aiming to design i) a computationally efficient solution, ii) flexible and adaptable following the user's needs (the user is involved in every phase of the processing), iii) requiring minimal (contextual) input data, and iv) providing various functionality toward personalized social image management and search (cf. Section I). Our solution can be viewed as a flexible and open platform on top of which different application specific functionality can be implemented (e.g., pattern recognition, event extraction, image classification, automatic image annotation, semantic image feature learning, among others).

III. Preliminaries

A. Image Representation

1) Low Level Features: A digital image is represented by a number of colored pixels. Different color spaces exist in the literature such as CIE XYZ which attempts to produce a color space based on human eye color perception. Other such color spaces include CIE RGB, CIELAB, etc., [18, 19].

Low level features are image characteristics/descriptors that are related to the color distribution and their combination in an image. Those descriptors can be divided into 3 groups: color, texture, and shape. Color descriptors are used to represent the colors present in the image such as color histogram; texture features are used to describe color patterns and geometric color distributions in an image; while shape descriptors allow detecting different shapes and objects in an image. A prominent image color descriptor is the *color histogram* which represents the color distribution in the image. That is, each cell in the histogram contains the number of pixels in the image having a specific RGB value. For example, one can know the number of pixels that have R = 30, G = 30 and B = 30, by checking a specific index in the histogram. The values for each of the three colors varies from 0 to 255 with the white color being represented by 255 for all three values and the black color being represented by 0 for all three values [19]. A prominent image texture descriptor is the *texture color histogram* which describes certain patterns and shapes in an image based on their color distributions. The combination of the colors forms common shapes that are used for scene understanding. Extracting the texture color histogram is done through image segmentation [20], dividing the image into several parts/regions according to some criteria. A prominent shape descriptor is the *edge histogram* which describes edges in the images. Therefore, images containing cities and buildings might appear similar because of the common edges in the images. Edge (or contour) detection can be performed using different linear (convolution) and/or non-linear (median and mean) local filters [21, 22]. One effective and computationally efficient (linear) filter technique is the *Haar Transform* [23] which we adopt in our study[1].

Note that image features can be represented as vectors, which can be efficiently processed using typical similarity measures such as *cosine similarity* and/or the *Euclidean distance*, to compute the similarity between two images. Similarity evaluation between images is essential in a battery of applications, including image indexing and retrieval [24] as well as image classification and clustering [25, 26]. Note that color descriptors are the most commonly used since they are relatively easy to process (in comparison with texture and/or shape features) while producing good enough results [18].

In developing SICOS, we utilized the open source *Java Lire* library to extract low level image features [27]. The extracted features

[1] Other edge detection filters can be utilized (individually or combined with *Haar Transform*) such as Sobel, Roberts, Laplacien, and/or Canny [21, 22].

are defined using the MPEG 7 standard [28] as descriptors and supporting tools, including: *dominant color* (DC), *scalable color* (SC), and *color layout* (CL) as the main color features, *color and edge directivity descriptor* (CEDD) and *fuzzy color and texture histogram* (FCTH) as the main texture descriptors, and *Gabor filter* (GF) and *edge histogram* (EH) as the main edge descriptors.

2) High Level Features: Another set of descriptors that can be used to describe social Web images are the so-called high-level features, which designate the textual content of the image including textual descriptors such as tags, captions, comments surrounding the image, places, and more recently, hash-tags in social media. In contrast with low-level features which describe the visual content of the image, high-level features attempt to describe the semantics of the image (e.g., who, where, what, etc.) [19], given that the surrounding text of Web and social images can be helpful in portraying the meaning of an image.

Moreover, when performing similarity evaluation between images, using high-level features to perform the comparison process can prove to be faster than comparing bulkier low-level features. Comparison using low-level features requires evaluating the similarity between two large vectors of bytes corresponding to the low-level description of each image, while comparison using high-level features may entail the comparison of merely two different strings of text. With the ever increasing demand for faster results, this may prove to be a very important factor.

In developing our approach, we have explored and utilized the prominent *Apache Lucene* library for extracting high-level features [29], including: i) *Tags:* which easily describe who and how many people are found in a given picture, ii) *Place:* label (name) of place where an image was taken, which can be utilized to also allow address comparison (using a geo-referenced ontology assigning geo-coordinates with place names [30]), iii) *Caption:* i.e., title of the image which is usually the most descriptive user-provided textual feature, providing a direct clue to the meaning and context of the image, and iv) *Comments:* allowing a much larger variety of textual descriptions then the previous features, and which become especially useful when captions have not been provided by the user (publisher).

B. Image Similarity Computation

After extracting features, similarity needs to be computed, to perform more sophisticated tasks such as clustering and similarity-based querying. Image feature similarity computation is usually done separately for high-level and low-level features.

1) Comparing Low-Level Features Comparing low level features can be done by applying the *cosine similarity* measure (and/or any other similarity measure such as the *Euclidian distance* or *Dice*) between the vectors associated with each descriptor. Given two vectors Q and D, *cosine similarity* is evaluated as:

$$\mathrm{Cos}(\vec{Q},\vec{D}) = \frac{\sum_{r=1}^{M} w_Q(n_r)\ .\ w_D(n_r)}{\sqrt{\sum_{r=1}^{M} w_Q(n_r)^2 .\ \sum_{r=1}^{n} w_D(n_r)^2}} \in [0,1] \quad \textbf{(1)}$$

2) Comparing High-Level Features: High-level feature comparison can be undertaken by comparing high-level feature vectors associated with each descriptor, where weights have been computed using legacy TF-IDF scoring developed in information retrieval [31]. TF-IDF (which stands for *Term Frequency – Inverse Document Frequency*) represents the number of times a term appears in a certain entry of a high-level feature (TF) compared to the number of times it appears in all entries of that high-level feature (IDF). It can be computed as:

$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d)\cdot \mathrm{idf}(t,D) \quad \textbf{(2)}$$

$$\mathrm{tf}(t,d) = 0.5 + 0.5\cdot \frac{f_{t,d}}{\max\{f_{t',d} : t' \in d\}} \quad \textbf{(3)}$$

$$\mathrm{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|} \quad \textbf{(4)}$$

where t designates a term, d a document (e.g., image textual feature), D the document (image feature) collection containing d, $f_{t,d}$ the raw term frequency (number of times the term t appears in d), and N designates the total number of documents d in D.

C. Image Clustering

Clustering is a technique used to collect similar objects (in our cases: images) together in groups called clusters. Each image will belong to one and only one cluster, and is more similar to the images in the same cluster then to the images in other clusters [32]. Also, each cluster can have a *representative image*, used to represent (provide a sample of) the other images in the cluster. The choice of the representative image depends on the clustering algorithm adopted.

One of the advantages of using clustering is the speed with which querying can be done subsequently. Having all the images in different clusters, the query only needs to be compared with the representative of each cluster to find the cluster that matches the query. Also, if the user is looking for a certain image, the query can be compared with the images in that cluster, thus greatly decreasing the number of comparisons that need to occur when querying.

In developing SICOS, we made use of two different clustering algorithms: *max-min* clustering [33] and *incremental* clustering [34], which we briefly describe below.

1) Max-Min Clustering: The main clustering algorithm used in our approach is max-min clustering, originally designed for visual diversification of image search results [33]. It is a partitional clustering algorithm grouping separate images together to produce clusters, in contrast with hierarchical (agglomerative/divisive) clustering which takes a combined set of images and aggregates/divides them in a bottom-up/top-down approach.

The *max-min* algorithm's pseudo code is provided in Fig. 2, and proceeds in the following manner. First of all, the first representative is selected at random from the set of images. Second, the average similarity of all pairs images is computed. Third, the second representative is found by finding the image with the farthest distance from the first representative (smallest similarity with the first image). Fourth, the same process is repeated to find the other representatives. That is, to find the next representative, the algorithm finds an image that has the maximum distance from all other representatives, as long as that maximum distance is greater than the average distance (or the minimum similarity is less than the average similarity). Fifth and finally, once all representative images have been selected, a nearest neighbor approach is used to divide the rest of the images into their corresponding clusters. The nearest neighbor approach takes each image and places it in the cluster having the minimum distance (maximum similarity) between the image and the cluster's representative image. In this way, all images are placed in their corresponding clusters and the algorithm terminates.

2) Incremental Clustering: This is an agglomerative clustering algorithm that considers the images one by one in an incremental manner and directly decides what to do with (where to put) each image [34] (cf. pseudo-code in Fig. 3). This means that the algorithm takes the first image and places it in a cluster. Then, for the next image, the algorithm decides based on a (user chosen or average) similarity threshold if the new image should be placed in the same cluster or if a new cluster should be created around that image, by comparing the image with cluster representatives. The algorithm continues in the same manner until all images have been clustered.

In our solution, we utilize *incremental clustering* to add new images to existing clusters which are originally produced by the *max-min* algorithm, rather than re-clustering all images every time a new image is added by the user (described in Section IV).

IV. Solution Design and Functionality

The overall architecture of our solution is shown in Fig. 1. It allows to retrieve images from a social site (we utilize Facebook in our study, even though any other site could have been used), extracts their features and stores them into a database, computes their high- and low-level feature similarities, and then clusters the images based on their similarities w.r.t. (with respect to) user chosen parameters. Subsequently, the system can be used to search for different images based on high-level and/or low-level features. Finally, the results can be displayed through multiple personalized visualization techniques (such as the list view, cluster view, fish-eye view, etc.) which we further describe in the following subsections.

A. Retrieving Images from the Social Site

To retrieve information from a social site such as Facebook, the user first needs to be authenticated to get the data. For this purpose, and given that most social sites (namely Facebook) do not support SDKs for desktop applications, we developed a dedicated Web application (including an imbedded Web browser) for the user to access her social (Facebook) account. Whenever the user first launches the SICOS software, she is required to sign in: granting the application the necessary permission to retrieve images on her behalf.

B. Extracting and Processing Image Features

As mentioned previously, we have utilized the *Java Lire* library to extract the low-level features of images, including various kinds of color, texture, and shape histograms (cf. Section II). Similarly, high-level image feature extraction is undertaken using the *Apache Lucene* library by first retrieving the high-level features corresponding to the image, namely *place*, *caption*, and *comments*, then creating *vector text fields* (i.e., vector dimensions) for each one of the features: one vector text field (dimension) containing a string description for the *place* (location) information, another containing the string description for *caption*, and one containing a string composed of all the image's *comments* concatenated. Consequently, the similarity between two images Img_1 and Img_2 is computed as the weighted sum of the similarities between their corresponding image features:

$$\begin{aligned} Sim_{\text{High-Level}}(Img_1, Img_2) = \; & w_{Place} \times Sim_{Place}(Img_1, Img_2) \\ & w_{Caption} \times Sim_{Caption}(Img_1, Img_2) \\ & w_{Comment} \times Sim_{Comment}(Img_1, Img_2) \end{aligned} \quad \textbf{(5)}$$

where feature weights $w_{Place}, w_{Place}, w_{Place} \geq 0$ and $w_{Place} + w_{Place} + w_{Place} = 1$; and feature similSim$_{Place}$, Sim$_{Caption}$, Sim$_{Comment} \in [0, 1]$.

A similar formula is utilized to evaluate low-level image similarity where the user chooses the features and their corresponding weights. *Cosine* similarity (cf. Formula 1) is used to compute the similarity between individual image feature vectors.

Note that low-level and high-level feature extraction and processing is undertaken for every social image in the user repository, whereas similarity computation is undertaken for every pair of images. This means that for n images, the system needs to compute $(n \times (n-1))/2$ similarity scores, which is done offline (as pre-processing), storing all similarity scores in the database (indexed using image IDs), to be quickly retrieved later on to perform clustering and search while minimizing *on-the-fly* computation time.

C. Choice of Clustering Algorithms

Among the battery of clustering algorithms available in the literature [34], we chose *max-min* to perform the initial clustering phase: executed when the user first connects to the system. The advantages of using *max-min* clustering is that it is relatively fast when compared with legacy hierarchical clustering algorithms, and does not require the preliminary input that is required by other partitional algorithms such as *k-means* clustering. In *k-means* clustering, the user needs to provide (based on previous knowledge) the number of clusters that are going to be produced. However, with the *max-min* algorithm, the number of clusters is automatically produced by the algorithm based on the average image similarity score utilized as clustering threshold (cf. Fig. 2). This makes it easier to cluster the results dynamically, and allows for an increased cluster variety depending on images in the initial set [33]. Also, *max-min* is of average linear $O(n \times k)$ complexity, since n images are compared with k cluster representatives (where k is significantly smaller – almost negligible – w.r.t. n), and thus is generally more efficient than alternative clustering algorithms which loop through all n images n times, and thus typically require $O(n^2)$ (or at best $O(n \times \log(n))$ time [34].

```
Input:  Set of Images to be clustered: I
        Set of features chosen by user: F
Output: Set of image clusters: C

1    Initialize set or representatives R = ∅
2    Compute average similarity between images in I
3    Select random image i ∈ I as first representative, R = R ∪ {i}
4    Remove selected image representative from I
5    for each remaining image i ∈ I
6        for each representative image r ∈ R
7            if Sim_F(i,r) < threshold
8                Add i to R
9                Remove i from I
10               Create new cluster c
11               Add i to c
12               Add c to C
13           endif
14       endfor
15   endfor
16   for each image i ∈ I
17       Find max(Sim_F(i,r, F)) where r ∈ R
18       Add i to cluster c having r
19   endfor
20   return C
```

Fig. 2. Pseudo-code of max-min clustering algorithm.

One of the main limitations of *max-min* clustering is that it might require more processing time than *k-means* clustering, since it first needs to produce the representatives and the number of clusters, whereas *k-means* execution speed can be greatly reduced based on the number of clusters and convergence threshold used. Another limitation of *max-min* is that it might produce less precise and accurate clusters in comparison with hierarchical clustering algorithms [33]. Also, cluster representatives might not be the most accurate representatives because they are first chosen randomly. This could be fixed by later creating a new representative within each cluster which would be the average image between all images in the cluster; however, this would require additional execution time.

In addition to *max-min* used in the initial clustering phased, we utilize *incremental clustering* to update clusters produces after the initial phase by classifying newly published images in the already formed clusters. We adopt *incremental clustering* since it is considered as one of the fastest clustering algorithms compared with alternative approaches (requiring average $O(k)$ time where k represents the number of cluster representatives). Yet, it does not produce the best results which depend on the original order following which images were presented to the algorithm [34] (a different original order can produce a different clustering result all together). Similarly to the *max-min* algorithm, *incremental clustering* also presents a tradeoff of efficiency versus quality that should be taken into account.

```
Input:  Set of initial image clusters: C
        Set of initial cluster representatives: R
        Stream of newly added Images: S
        Set of features chosen by user: F
Output: Set of updated image clusters: C

1     if user wishes to set threshold
2             Set threshold following user input
3     else
4             Set threshold as average similarity of all images in C
5     endif
6     for each incoming image i ∈ S
7        find max($Sim_F$(i,r)) where r ∈ R
7        if max($Sim_F$(i,r)) <threshold
8             create new cluster c
9             add i to c
10            add c to C
11       endif
12       else
13            add i to cluster c having r as representative
14       endif
15    return C
```

Fig. 3. Pseudo-code for incremental clustering algorithm.

Note that in our current study, clusters are labeled following the captions of their cluster representatives. Yet, more sophisticated cluster labeling schemes could be devised later on.

D. Image Querying

Querying the results will return the pictures users are looking for in a simplified manner. Here, SICOS allows three different querying methods described below.

1) Tag-based Image Search: searching for images based on the people tagged in them. This is done by storing tag names in the image database along with the images corresponding to each tag. When the query tag is submitted, the database selects the images having the tag and returns them to the user. Furthermore, our solution allows the user to enter multiple tag names and search for images which contain any or all of them, following user preferences. For example, query "*John Smith, Jane Smith*" will return all images in which *John Smith* and *Jane Smith* appear together, or separately (based on the query formulation).

2) High Level-based Image Search: searching for images using high-level features such as place, caption, and/or comments. The user selects the features against which to perform the query, as well as corresponding feature weights (similar to feature extraction in Section II.B). Then, the query is run against the image clusters, such that the query feature vector is compared with the feature vectors of the representatives of the clusters. The system then returns the images that fit the query in order of decreasing TF-IDF scores. This means that feature vectors which have the same word repeated several times will get a higher score due to the higher term frequency factor. Also, feature vectors which contain more term occurrences yet without having any matches to the query terms will get a lower score. The clusters corresponding to the returned documents will be displayed to the user using special cluster display techniques (covered in Section IV.E).

The number of results to display is based on user preferences. The user has the choice to decide between a k nearest neighbor approach or a range approach. The k nearest neighbor approach displays the top k results, where k is an integer given by the user. The range approach displays all clusters in which the distance (similarity) between the query feature vector and the cluster representative's feature vector is lesser (higher) than a user-given threshold.

3) Low Level-based Image Search: using a sample image as query, such that the image query can be either selected from the available image repository or uploaded by the user. SICOS then extracts the low-level features of the image, and uses them to compare it with the cluster representatives' low-level feature vectors. The results are ranked based on the similarities returned by the comparison. Similarly to high-level querying, the user can select her preferred low-level features and features weights to be utilized in the comparison (retrieval) process. Finally, clusters corresponding to the most similar representatives are returned, displayed to the user using different possible cluster visualization techniques (cf. Section IV.E). Also, the user can select the number of clusters to display, using either the k neighrest neighbor approach or the range selection approach.

Moreover, when a new image is uploaded as a query, the user is asked if she would like the image to be added to the user's image repository (after the query has been processed). If yes, the user is then asked to provide the image's high-level features (if available), which are then processed along with low-level features, and run through the *incremental clustering* component.

E. Result Presentation and Organization

Having images organized into clusters becomes even more effective and practical in retrieval if these clusters can be visualized properly. As a result, our solution allows different visualization techniques, which can be used to display both: i) the cluster organization of images in the repository, as well as ii) image query search results.

Our solution includes five different visualization techniques, introduced to answer different user preferences, and which we describe in the following sub-sections.

1) Representatives Display: Following this layout, the representative image of each cluster is displayed only at first, and then when the user clicks on one of the images, a new window opens containing the images of the corresponding cluster. This is done by looping through the first image in the array list for each cluster and displaying the images in one window. Then, each image is assigned a mouse click listener, which retrieves the corresponding cluster and displays the images in that cluster.

The main advantage of using this view over others is that it is quick in displaying the images since it only requires initially displaying the representative images without having to display the rest of the images in each cluster. In other words, there is no need to load all cluster images unless the user chooses to do so explicitly for a given (number of) clusters. A disadvantage of this view is that it might not be very intuitive in displaying clustering, since the user cannot easily visualize how the clusters are organized w.r.t. image similarities/dissimilarities, or how close/far away clusters are from each other (cf. Fig. 4).

2) Cluster List View Display: In this display, the main view is a list in which each item in the list represents a cluster. For each cluster, the representative image is displayed in large on the left, and then the rest of the images are displayed in smaller size to the right. To implement this view, a separate class was created to represent each list item. The constructor of the class would take the cluster as input and would produce a list item having the view described above. Moreover, each image can be enlarged (upon user selection) to see the image in its actual size, while allowing the user to change the cluster representative (choosing another image from the cluster as its representative). Changing cluster representatives not only affects the visualization of clusters, but also allows replacing the old cluster representatives in the database.

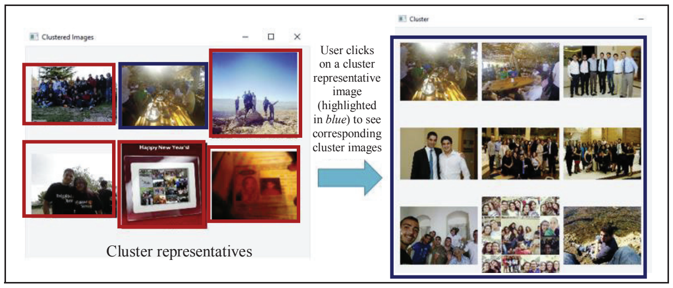

Clustered Images
Cluster
User clicks on a cluster representative image (highlighted in blue) to see corresponding cluster images
Cluster representatives

Fig. 4. Sample *cluster representatives display*.

The main advantage of this view is that it allows the user to view all clusters at once in an organized manner and allows the user to change representative images. A disadvantage is that it is time-consuming and memory-consuming due to the fact that a new instance of a class should be created for every cluster, and all images in the system should be loaded into memory (cf. Fig. 5).

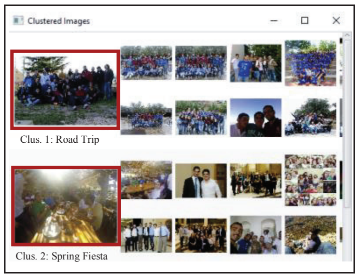


Fig. 5. Sample *cluster list view display*.

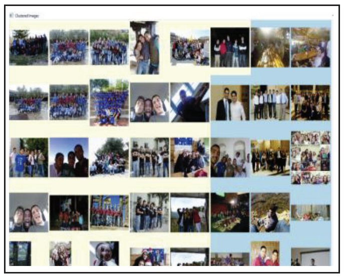


Fig. 6. *Grid view display* (with 2 clusters: white & blue).

3) 2D Display: it presents images in a 2 dimensional plane, where each cluster of images is separated from the rest (using different color indicators), and within each cluster of images: the representative image is placed in the middle, and the rest of the images are placed around it according to the similarity between the (feature vectors of) images and (those of) the representative image. That is, in a given cluster, images that are most similar to the cluster's representative image are displayed closest to the center, while those that are least similar are displayed the farthest from the center. This is done by creating a specific display pane for each cluster in which the images are laid out according to the above description. Then, each display pane is added to the main grid which separates the clusters from each other.

The main advantage of this is approach that it helps visualize clusters while highlighting their intra-cluster image similarities, and inter-cluster similarity-based spatial organization. A disadvantage of this display is that it is more computationally expensive and time-consuming, compared with the previous two displays (cf. Fig. 7).

4) Grid View Display: It places all the images in a 2 dimensional grid in such a way that images in the same cluster are placed as close as possible to each other. The difference between the *grid view* and the *2D display* is that the *grid view* places images in an ordered manner as tiles next to the cluster representative, whereas *2D display* places images in a spiral shape around the representative. Different clusters are distinguished using special background colors for each cluster. To do this, the first image is placed in the grid, and the next image in the cluster with the highest similarity is placed as close to the representative as possible. For each new image, the system identifies the next tile in the grid which can be filled taking into account the distance/similarity w.r.t. the representative (i.e., trying to keep the minimum average distance/max similarity).

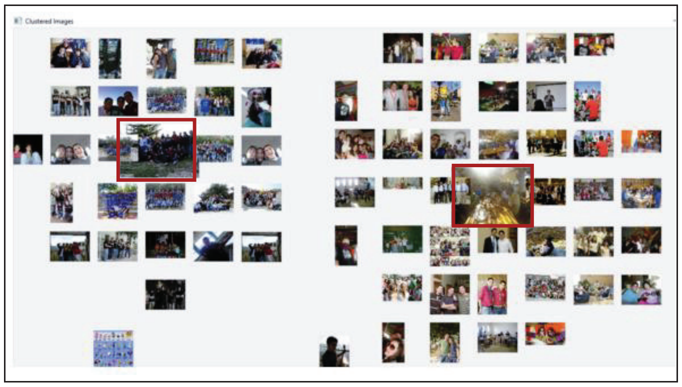


Fig. 7. Sample *2D display*.

The main advantage of *grid view display* is that images can be displayed in 2D manner while requiring less computation and time compared with the *2D display* approach. A disadvantage of this display is that it seems less expressive of the intra- and inter-organization of clusters in comparison with *2D display* (e.g., the distances among images and among clusters cannot be easily spotted with the *grid view display*, compared with *2D display* where these are clearly highlighted, cf. Fig. 6).

5) Fish-Eye View Display: This is similar to *2D display* with one major difference: the sizes of images surrounding the cluster representative decrease as their similarities w.r.t. the representative decrease, causing the images that are farther away from the representative to appear smaller. It carries the already mentioned advantages and limitations of *2D display*.

V. Performance Evaluation

To test the performance of our tool, we evaluated execution time for each of its constituent components while varying user parameters. Experiments were performed on an Intel core i3-2328 2.20 GHz CPU with 4 GB RAM. Each experiment was executed 5 times, retaining average time values. The SICOS prototype system, along with all experimental evaluation results, is available online[2].

A. Feature Extraction Time

To start off, we first evaluated the time needed to extract the different low-level and high-level features from the images. Tests were undertaken on an increasing number of images: 50, 100, 130, 170, and 195. For each of these sets, the time needed to extract each feature was retrieved and can be seen in Fig. 8. We omit the time results for high-level feature extraction here since they require significantly less time (extraction is done almost instantaneously) in comparison with their low-level counterparts.

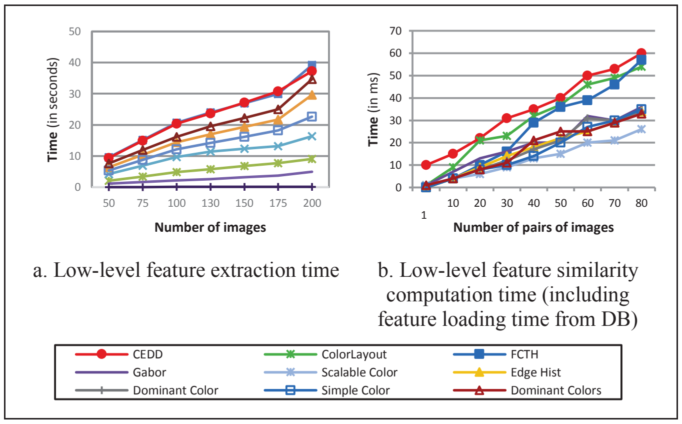
Results show that FCTH and CEDD features are the most expensive to extract (which was expected since they are considerably

[2] http://services.soe.lau.edu.lb/SICOS/

sophisticated and detailed in their descriptions, producing relatively large feature vectors, cf. Section II), such that all features (including the latter) show a linear increase in time w.r.t. number of images. Note the *Gabor filter* feature shows the least increase, and thus seems to be the least variant in extraction time w.r.t. the number of images.

B. *Similarity Computation Time*

We also evaluated the time needed to compute the similarity between image feature vectors, for both high-level and low-level features once those have been extracted. Results obtained with an increasing number of pairs of images (from 2 to 80 pairs, designating the number of similarity computation tasks) are shown in Fig. 8.



a. Low-level feature extraction time

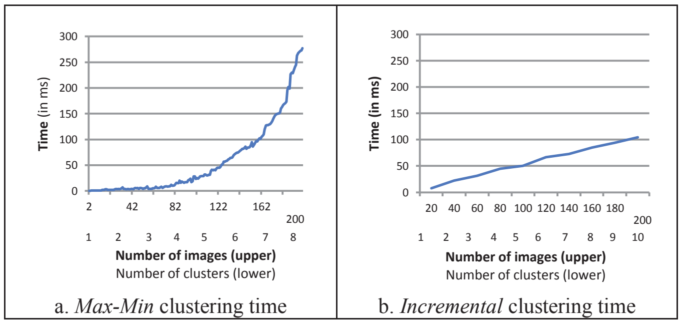
b. Low-level feature similarity computation time (including feature loading time from DB)

Fig. 8. Processing low-level image features.

Results show that similarity computation time increases linearly w.r.t. the number of pairs of images being compared, which reflects the linear complexity of computing the *cosine similarity* measure between two image vectors. Also, one can see that the time needed to compute similarity for CEDD, FCTH, and *color layout* features is higher than the time needed for other features, which is in direct relation with the increased sizes of the corresponding feature vectors. Note that similar results were obtained with high-level features (omitted to simplify presentation), since the same *cosine similarity* measure is utilized to compare corresponding feature vectors.

C. *Max-Min Clustering Time*

As for *max-min* clustering, we conducted performance tests on a random set of 200 images downloaded from the author's Facebook accounts. Results in Fig. 9.a show that *max-min* clustering time varies in an almost linear/slightly polynomial fashion (highlighting *max-min*'s average linear complexity levels (cf. Sections III.C.1 and IV.C). This shows that *max-min* can be very efficient when processing small/moderate sized image datasets, but may require polynomial time when processing a very large number of images.



a. *Max-Min* clustering time

b. *Incremental* clustering time

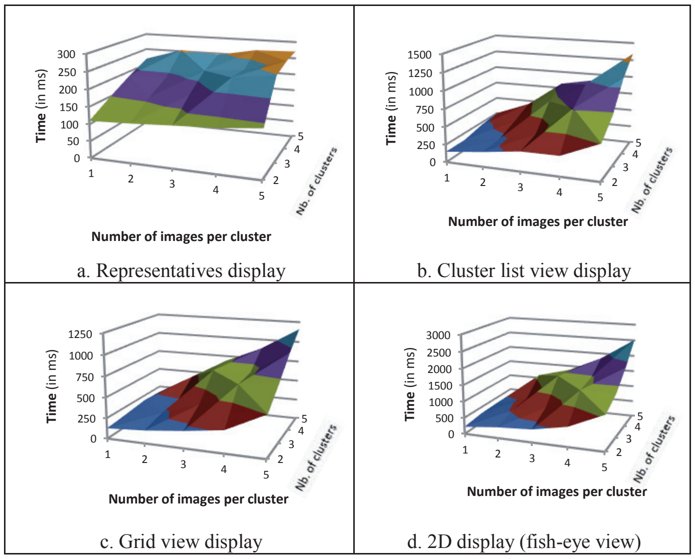
Fig. 9. Clustering time.

D. *Incremental Clustering Time*

As for *incremental clustering* time, results in Fig. 9.b show that execution time is almost perfectly linear in the number of images (which is proportional to the number clusters/cluster representatives), thus rendering the algorithm extremely faster than *max-min* (almost 3 times faster in our implementation) considering the same number of images (clusters) processed by both algorithms.

E. *Cluster Visualization Time*

We have also evaluated the time performance for each of the five cluster visualization techniques provided in SICOS. We ran multiple tests with different parameter variations: in number of clusters (between 2 and 5) and number of images in each cluster (between 1 and 5). Results in Fig. 10 show that all five displays requires average linear time w.r.t. the number of clusters and the number of images per cluster, such that the *representatives display* view is the fastest (since it only displays cluster representatives, and involves less image loading time), followed by the *cluster list view* and *grid view* displays (which display: cluster representatives and image constituents, and thus need more loading time), followed by *2D display* (and *fish-eye view*) which is (are) the most expensive (since they perform additional processing to display images around the representatives, in a spiral shape, following their similarities, cf. Section IV.E).



a. Representatives display

b. Cluster list view display

c. Grid view display

d. 2D display (fish-eye view)

Fig. 10. Time needed to produce different cluster visualizations.

F. *Querying Performance*

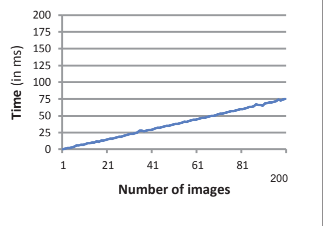
We have also evaluated the time performance of our system in running queries, considering both low- and high-level features.

1) High Level-based Image Search Time: As for high level-based image searching, we have studied the time it takes to return query results when the query is run on a varying number of cluster representatives, considering all high-level features aggregated (i.e., *tag*, *place*, *caption*, and *comments*)^3. Recall that our queries are run against cluster representatives (instead of running them against each and every individual image) in order to reduce processing time (cf. Section IV.D). Results in Fig. 11.a show that query execution time is clearly linear in the number of representative images.

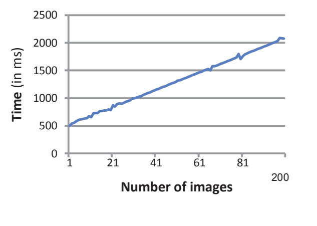
2) Low Level-based Image Search Time: We conducted similar tests considering (all) low-level image features combined^3. Similarly

^{3} Individual feature evaluations were discussed in Section V.B.

to high-level features, results in Fig. 11.b show linear dependency on the number of representative images being processed. Low-level feature querying is naturaly more expensive than high-level querying due to the sheer size of low-level feature vectors (e.g., histogram vectors) in comparsison with high-level feature vectors (e.g., term/frequency vectors, cf. Sections III.A and IV.B).



a. High-level feature querying



b. Low-level feature querying

Fig. 11. Query execution time[4].

To sum up, both query performance experiments highlight the efficiency of our approach in handling large image repositories, where time is mainly dependent the number of clusters/cluster representatives rather than the actual size of the repository (given that image feature extraction and pair-wise similarity computations are executed offline). Cluster visualization techniques then kick in to display images within each and every returned result cluster.

We are currently conducting a battery of (quality) tests to study the effectiveness of our approach in: i) clustering images (fine-tuning the impact of low-level/high-level features to produce more coherent clusters), ii) retrieving "relevant" results (while minimizing false negatives), and iii) visualizing clusters/query results, compared with alternative tree-like representations in existing social image management solutions, e.g., [13-15].

VI. Conclusion

In this paper, we describe our solution called SICOS as a personalized social Web image organization, clustering, and search tool, allowing a battery of low-level (visual) and high-level (textual) image features, run through efficient clustering algorithms, as well as different result visualization displays all of which can be fine-tuned following user preferences. We briefly described the background and related works. From there, we described the functional requirements of our system, leading to its design and implementation. Finally, we conducted various tests to evaluate the time performance of the different components and functionality of our solution.

For the future, we plan to research and add new features, such as producing multiple representative images for each cluster (e.g., allowing *k* representatives instead of only one, following user querying and visualization preferences), and performing automated image annotation through image processing (e.g., automatically annotating new images, based on high-level features of similar images which were already annotated) [14]. Furthermore, we plan to extend our solution, making use of advanced semantic database indexing techniques [35] to allow semantic-aware image querying: considering not only term occurrences, but also their semantic meanings (e.g., term "*orange*" could mean the *color* or the *fruit*, which could produce totally different results). Extending our solution to describe semantic relations between images based on their low- and high-level features [36] and/or using vector graphics annotations [37], in order to facilitate knowledge-based event detection, identification, and description [38] is also an upcoming challenge.

[4] It includes: i) time to load extracted features from the database, ii) feature vector similarity computation time, and iii) result presentation time following the *cluster list view* display.

Acknowledgments

The study is partly funded by the National Council for Scientific Research (CNRS-L) Lebanon: NCSR_00695_01/09/15, LAU research fund: SOERC1516R003, and LAU travel grant: SOERC1516T015

References

[1] Chen Y. *et al.*, *Content-based Image Retrieval by Clustering.* MIR 2003, 193-200

[2] Van Leuken R. H. *et al.*, *Visual Diversification of Image Search Restuls.* Proc. of Inter. World Wide Web Conference, 2009. pp. 341-350

[3] Suh B. and Bederson B., *Semi-Automatic Photo Annotation Strategies using Event-based Clustering and Clothing based Person Recognition.* Interacting with Computers, 2007. 19(4): 524-544

[4] Phillips P. *et al.*, *Preliminary Face Recognition Grand Challenge Results.* Inter. Conf. on Automatic Face and Gesture Recognition, 2006. pp. 15-24

[5] Kang H., Bederson B. and Suh B., *Capture, Annotate, Browse, Find, Share: Novel Interfaces for Personal Photo Management.* IJHCI, 2007. 23(3): 315-337

[6] Cai D. *et al.*, *Hierarchical Clustering of WWW Image Search Results using Visual, Textual and Link Information.* Inter. ACM Multimedia Conf., 2004. pp. 952-959

[7] Weinberger K. *et al.*, *Resolving Tag Ambiguity.* Proc. of the 16th International ACM Multimedia Conference (MM'08), 2008. pp. 111-120, Vancouver, Canada

[8] Cronen-Townsend S.; Zhou Y. and Croft W.B., *Predicting Query Performance.* ACM SIGIR Conf. on Research and Development in IR, 2002. pp. 299-306

[9] Carpineto C. *et al.*, *An Information-Theoretic Approach to Automatic Query Expansion.* ACM Transactions on Information Systems (TOIS), 2001. 19(1):1-27.

[10] Rodden K. *et al.*, *Does Organization by Similarity Assist Image Browsing?* SIGCHI Conf. on Human Factors in Computing Systems, 2001. pp. 190-197

[11] Wang S. *et al.*, *IGroup: Presenting Web Image Search Results in Semantic Clusters.* Computer-Human Interaction Conference, 2007. pp. 587-596

[12] Cutting D.R. *et al.*, *Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections.* ACM SIGIR Inter. Conf. on R&D in IR, 1992, 318-329

[13] Eklund P. *et al.*, *An Intelligent User Interface for Browsing and Search MPEG-7 Images using Concept Lattices.* Inter. LNAI'06 Conf., Springer, 2006. pp. 1-21

[14] Crampes M. *et al.*, *Visualizing Social Photos on a Hasse Diagram for Eliciting Relations and Indexing New Photos.* IEEE TVCG, 2009. 15(6): 985-992

[15] Ferré S., *CAMELIS: Organizing and Browsing a Personal Photo Collection with a Logical Information System.* In proc. of Inter. CLA Conf., 2007. pp. 112-123

[16] Ma Z. et al., *Knowledge Adaptation for Ad Hoc Multimedia Event Detection with Few Exemplars.* In proc. of ACM Multimedia Conf., 2012. pp. 469-478

[17] Liu X., Troncy R. and Huet B., *Using social media to identify events.* In proc. of the 3rd ACM SIGMM International Workshop on Social Media, 2011. pp. 3-8

[18] Datta R.; Joshi D.; Li J. and Wang J.Z., *Image Retrieval: Ideas, Influences and Trends of the New Age.* ACM Computer Surveys, 2008. 40(2):1-60

[19] Liu Y.; Zhang D.; Lu G. and Ma W.-Y., *A Survey of Content-Based Image Retrieval with High-Level Semantics* PAttern Recognition, 2006. 40(1):262-282.

[20] Felzenszwalb P. and Huttenlocher D., *Efficient Graph-Based Image Segmentation.* International Journal of Computer Vision, 2004. 59(2):167-181

[21] Kaur K. and Malhotra S., *A Survey on Edge Detection Using Different Techniques.* Inter. IJAIEM journal, 2013. 2:496-500

[22] Sharifi M. *et al.*, *A Classified and Comparative Study of Edge Detection Algorithms.* Inter. Conf. on Info. Tech.: Coding & Computing (ITCC), 2002. p. 4

[23] Chatzichristofis S. and Boutalis Y., *CEDD: Color and Edge Directivity Descriptor: A Compact Descriptor for Image Indexing and Retrieval.* Computer Vision Systems 2008. LNCS Vol. 5008, pp. 312-322

[24] Wang Y.H., *Image Indexing and Similarity Retrieval based on Spatial Relationship Model.* Informatics and Computer Science J., 2003. 154(1-2):39-58

[25] Rodden K. *et al.*, *Evaluating a Visualization of Image Similarity as a Tool for Image Browsing.* IEEE Symposium on Information Visualization, 1999. pp. 36-43

[26] Hirota M. *et al.*, *A Robust Clustering Method for Missing Metadata in Image Search Results.* Journal of Information Processing, 2012. 20(3):537-547

[27] Lux M. and Chatzichristofis S., *Lire: lucene image retrieval: an extensible java CBIR library.* 16th ACM Multimedia Conf. (MM'08), 2008. pp. 1085-1088

[28] International Organization for Standardization (ISO), *MPEG-7 Overview.* ISO/IEC JTC1/SC29/WG11, Coding for Moving Pictures and Audio, 2004. Martinez J.M.

[29] Lucene, A., *Apache Lucene Core.* https://lucene.apache.org/core/ [April 2016]

[30] Tekli J. *et al.*, *Toward Approximate GML Retrieval Based on Structural and Semantic Characteristics.* Inter. Conf. on Web Eng. (ICWE'09), 2009. pp. 16-34.

[31] McGill M., *Intro. to Modern Information Retrieval.* 1983. McGraw-Hill, NY

[32] Moellic P.A. *et al.*, *Image Clustering based on a Shared Nearest Neighbors Approach for Tagged Collections.* Proc. of the Inter. CIVR Conf., 2008, 269-278

[33] Van Leuken R. H. *et al.*, *Visual Diversification of Image Search Restuls.* Proc. of the International World Wide Web Conference, 2009. pp. 341-350

[34] Jain A.K. *et al.*, *Data Clustering: A Review.* ACM Computing Surveys, 1999. 31(3):264-323

[35] Chbeir R. *et al.*, *SemIndex: Semantic-Aware Inverted Index.* East-European Conf. on Advanced Databases and Information Systems (ADBIS'14), 2014. pp. 290-307.

[36] Jisha K.P., *An image retrieval technique based on texture features using semantic properties.* Inter. ICSIPR Conf., 2013. pp. 248-252

[37] Salameh K. *et al.*., *SVG-to-RDF Image Semantization.* SISAP 2014. pp. 214-228.

[38] Ashagrie M. *et al.*, *A General Multimedia Representation Space Model toward Event-based Collective Knowledge Management.* Inter. CSE Conf. 2016, Paris