Demo of the SICOS tool for Social Image Cluster-based Organization and Search*

Issa Ayoub, Karl J. Codouni, and Joe Tekli
E.C.E Department, School of Engineering (SOE)
Lebanese American University (LAU)
36 Byblos, Lebanon
issa.ayoub@lau.edu, karl.codouni@lau.edu, joe.tekli@lau.edu.lb

***Abstract*—This paper briefly describes and evaluates *SICOS*, a tool 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 following user preferences. The 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 and search results visualization (using various user-chosen cluster display techniques). Experiments highlight the efficiency of the tool.**

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

I. INTRODUCTION

In the past few 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 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. 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: i) efficient indexing and storage of images with the corresponding feature information, ii) comparing images based on low-level visual features including: color, texture, and shape descriptors; iii) comparing images based on high-level textual features, including: tags, captions, comments, and geographic location, iv) clustering images based on low- and high-level feature similarities, v) simple access to images through cluster representatives, vi) allowing different user-friendly cluster visualizations, vii) searching images based on low-level visual features, and viii) searching based on high-level textual features.

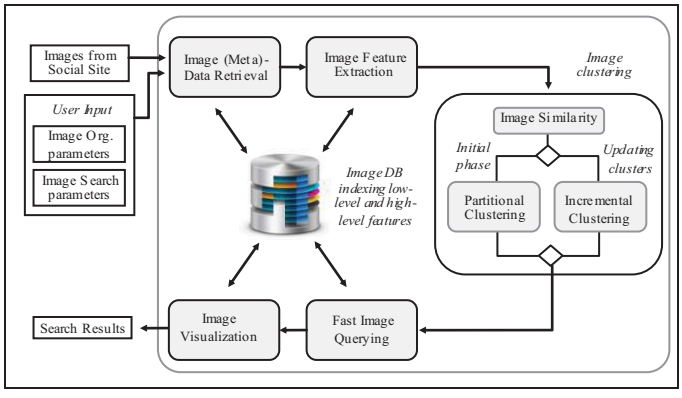


Fig. 1. Overall SICOS architecture.

* This study is partly funded by CNRS Lebanon project: NCSR_00695_01/09/15, and by LAU research fund: SOERC1516R003.

The overall design and groundwork results of SICOS have been described in [6]. In this demonstration, we aim to highlight SICOS's architecture and functionality, and then discuss current and ongoing experimental evaluations. The remainder of the paper is organized as follows. Section II briefly discusses related works. Section III describes SICOS, its components, and functionality. Section VI describes experiments and results, before concluding in Section V.

II. Related Works

Most existing image search result organization techniques have been developed for general purpose Web-based image retrieval systems, e.g., [1, 2, 7-9], few approaches have been developed to handle social Web images, e.g., [3, 10, 11]. 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 [11]. 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 [10-12], 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) [13, 14] 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 in 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. 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 tool the necessary permission to retrieve images on her behalf.

B. Extracting and Processing Image Features

1) Low Level Features: A digital image is represented by a number of colored pixels. 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 to detect different shapes and objects in an image [15]. 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 [16] as well as image classification and clustering [17, 18]. 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 [19].

In developing SICOS, we utilized the open source *Java Lire* library to extract low level image features [20]. The extracted features are defined using the MPEG 7 standard [21] as descriptors and supporting tools, including: *dominant color* (DC), *scalable color* (SC), 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* and *edge histogram* (EH) as the main edge descriptors.

2) High Level Features: Another set of descriptors that can be used to describe images encompasses the so-called high-level features of a Web image, 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.) [15], given that the surrounding text of Web and social images can be very helpful in portraying the meaning of an image.

In developing our approach, we have explored and utilized the prominent *Apache Lucene* library for extracting high-level features [22], 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 [23]), 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).

Note that high-level feature comparison can be undertaken by comparing high-level feature vectors using *cosine* (or any other vector similarity measure, similarly to comparing high-level feature vectors), where vector weights are computed using legacy TF-IDF scoring developed in information retrieval [24]. 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).

C. Image Clustering

Clustering is a technique used to collect similar objects (e.g., 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 [25]. 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 [26] and *incremental* clustering [27], 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 [26]. 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 of 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. 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.

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

1       Compute average similarity
2       Select first image in list as first representative
3       Remove first image from S
4       for each image L ∈ S
5               for each representative r ∈ R
6                       if sim(L,r) < threshold
7                               Add L to R
8                               Remove L from S
9                               Create new cluster
10                              Add L to new cluster
11                              Add new cluster to C
12                      endif
13              endfor
14      endfor
15      for each image L ∈ S
16              Find max(sim(L,r)) where r ∈ R
17              Add L to cluster having r
18      endfor
```

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

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 [27]. Also, it does not require the preliminary input that is required by other partitional algorithms such as *k-means* (e.g., initial number of clusters, and a convergence threshold).

2) Incremental Clustering: 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. 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 [27]. 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 chosen similarity threshold if the new image should be placed in the same cluster or if a new cluster should be created around that image. The algorithm continues in the same manner until all images have been placed in the corresponding clusters (pseudo-code is omitted for space limitations).

While very efficient following our scenario (i.e., of average $O(k)$ time in processing incoming images, where k is the number of cluster representatives), yet *incremental clustering* does not produce the best results: which depend on the original order following which images were presented to the algorithm [27] (a different original order can produce a different clustering result all together).

D. Image Querying

Querying the results will return the pictures users are looking for, within their repository of social photos, 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(s) and returns them to the user.

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). 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.

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 target low-level features and feature 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. subsection 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 (cf. sample display in Fig. 3). 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, and ii) image query search results. Our solution includes five different visualization techniques, introduced to answer different user preferences:

Fig. 3. Sample *2D display* of image clusters.

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.

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.

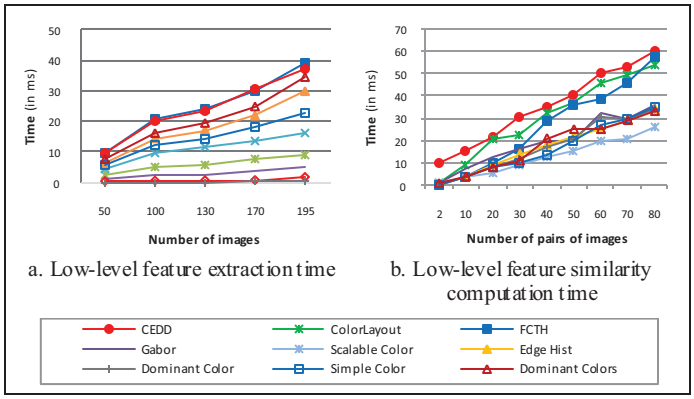
3) 2D Display: it presents images in a 2 dimensional plane, where each cluster of images is separated from the rest (using different color indicators), and within each cluster of images, the representative image is placed in the middle, and the rest of the images are placed around it according to the similarity between the (feature vectors of) images and (those of) the representative image.

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 around the representative in a spiral shape.

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.

IV. Experimental Set-Up and Tests

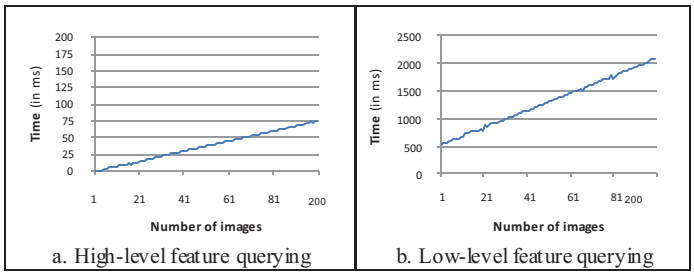
To test the performance of our solution, we evaluated execution time for each of its constituent components while varying user parameters, including: i) image feature extraction, ii) image similarity computation, iii) *max-min* clustering, iv) *incremental* clustering, v) low- and high-level based image search, and vi) image result visualization (considering our different display techniques). 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.



a. Low-level feature extraction time

b. Low-level feature similarity computation time

Fig. 4. Processing low-level image features.



a. High-level feature querying

b. Low-level feature querying

Fig. 5. Query execution time.

In a nutshell, performance experiments (cf. extract of results in Figures 4-5) highlight the efficiency of our approach in handling large image repositories, where time is mainly linearly dependent on the number of clusters/cluster representatives rather than the actual size of the repository. The SICOS prototype, along with all experimental evaluation results, is available online[2].

V. Conclusion

This paper introduces a tool called SICOS for personalized social Web image organization, clustering, and search. It allows to process a battery of low-level (visual) and high-level (textual) image features, through efficient clustering algorithms and different cluster visualization displays, while allowing the user to fine-tune the whole process (from feature extraction to result visualization) following her needs.

In the demonstration of SICOS, we aim to showcase the tool's logical design, implementation, and functionality: fine-tuning the different system parameters, and then highlighting their impact w.r.t. the images being tested. We will also present and discuss our latest experimental evaluation and results, highlighting the tool's effectiveness and efficiency, as well as its strong and weak points in social image querying and organization, emphasizing ongoing design and technical improvements toward semantic-aware indexing [28], vector graphics semantization [29], and knowledge-based social event detection, identification, and description [30].

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

References

[1] Chen Y. *et al.*, *Content-based Image Retrieval by Clustering.* Proc. of the ACM Inter. Conf. on Multimedia IR (MIR'03), 2003. pp. 193-200.

[2] Van Leuken R. H *et al.*, *Visual Diversification of Image Search Restuls.* Proc. of the 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.* Proc. of Inter. IEEE FG'06 Conf., 2006. pp. 15-24.

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

[6] Ayoub I. *et al.*, *Personalized Social Image Organization, Visualization, and Querying Tool using Low- and High-Level Features.* IEEE Inter. Conf. on Computational Science and Engineering (CSE'16), 2016.

[7] Cutting D.R. *et al.*, *Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections.* ACM SIGIR, 1992, 318-329.

[8] Wang S. *et al.*, *IGroup: Presenting Web Image Search Results in Semantic Clusters.* Computer-Human Interaction (CHI), 2007, 587-596.

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

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

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

[12] Ferré S., *CAMELIS: Organizing and Browsing a Personal Photo Collection with a Logical Information System*. Proc. of the 5th Inter. Conf. on. Concept Lattices and Their Applications, 2007. pp. 112-123.

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

[14] Liu X. *et al.*, *Using social media to identify events.* In Proc. of the 3rd ACM SIGMM International Workshop on Social Media, 2011. pp. 3-8.

[15] Liu Y. *et al.*, *A Survey of Content-Based Image Retrieval with High-Level Semantics* Pattern Recognition, 2006. 40(1):262-282.

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

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

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

[19] Datta R. *et al.*, *Image Retrieval: Ideas, Influences and Trends of the New Age.* Proc. of ACM Computer Surveys, 2008. 40(2):1-60.

[20] Lux M. and Chatzichristofis S., *Lire: lucene image retrieval: an extensible java CBIR library.* ACM Multimedia, 2008. pp. 1085-1088.

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

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

[23] Tekli J. *et al.*, *Toward Approximate GML Retrieval Based on Structural and Semantic Characteristics.* Proc. of ICWE Conf., 2009. pp. 16-34.

[24] McGill M., *Introduction to Modern Information Retrieval.* 1983. McGraw-Hill.

[25] Moellic P.A. *et al.*, *Image Clustering based on a Shared Nearest Neighbors Approach for Tagged Collections.* CIVR, 2008. pp. 269-278.

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

[27] Jain A.K.; Murty M.N. and Flynn P.J., *Data Clustering: A Review.* ACM Computing Surveys, 1999. 31(3):264-323.

[28] Chbeir R. *et al.*, *SemIndex: Semantic-Aware Inverted Index.* Proc. of Inter. ADBIS Conf. 2014, pp. 290-307

[29] Salameh K. *et al.*, *SVG-to-RDF Image Semantization.* Proc. of Inter. SISAP Conf., 2014, pp. 214-228.

[30] Ashagrie M. *et al.*, *A General Multimedia Representation Space Model toward Event-based Collective Knowledge Management.* IEEE Inter. Conf. on Computational Science and Engineering (CSE'16), 2016.