Web Image Clustering by Consistent Utilization of Visual Features and Surrounding Texts

Bin Gao[1, 2], Tie-Yan Liu[1], Tao Qin[1, 3], Xin Zheng[1, 4], Qian-Sheng Cheng[2], and Wei-Ying Ma[1]

[1]Microsoft Research Asia
5F, Sigma Center, No. 49, Zhichun Road,
Beijing, 100080, P. R. China
{tyliu, wyma}@microsoft.com

[2]LMAM, Dept. of Information Science,
School of Mathematical Sciences, Peking University,
Beijing, 100871, P. R. China
gaobin@math.pku.edu.cn, qcheng@pku.edu.cn

[3]MSP Laboratory, Dept. of Electronic Engineering,
Tsinghua University,
Beijing 100084, P. R. China
qinshitao99@mails.tsinghua.edu.cn

[4]Key Lab of Pervasive Computing,
Dept. of Computer Science and Technology,
Tsinghua University,
Beijing 100084, P. R. China
zhengxin99@mails.tsinghua.edu.cn

ABSTRACT

Image clustering, an important technology for image processing, has been actively researched for a long period of time. Especially in recent years, with the explosive growth of the Web, image clustering has even been a critical technology to help users digest the large amount of online visual information. However, as far as we know, many previous works on image clustering only used either low-level visual features or surrounding texts, but rarely exploited these two kinds of information in the same framework. To tackle this problem, we proposed a novel method named consistent bipartite graph co-partitioning in this paper, which can cluster Web images based on the consistent fusion of the information contained in both low-level features and surrounding texts. In particular, we formulated it as a constrained multi-objective optimization problem, which can be efficiently solved by semi-definite programming (SDP). Experiments on a real-world Web image collection showed that our proposed method outperformed the methods only based on low-level features or surround texts.

Categories and Subject Descriptors

I.5.3 [**Pattern Recognition**]: Clustering – *algorithms*; I.5.4 [**Pattern Recognition**]: Applications – *Computer vision*.

General Terms

Algorithms, Performance, Design, Experimentation, Theory.

Keywords

Co-clustering, Consistency, Spectral Graph, Image Processing.

1. INTRODUCTION

Along with the fast development of Web search engines, Web image search has become a more and more popular application, which can provide users with relevant images to the queries they issued. Considering the numerous online images, the numbers of image search results for many queries are usually very large. In such a scenario, image clustering will be very helpful to users because it can provide a concise summarization and visualization of image search results.

To the best of our knowledge, most of the traditional image clustering algorithms were based on the low-level visual features of the images [7][18][25]. That is, some low-level visual features such as color histogram and wavelet texture were first extracted from the raw images, and then clustering algorithms such as *k*-means [11], maximum likelihood estimation [11] and spectral clustering [1][27] were applied to group similar images together. For example, as an interesting piece of such works, Qiu [24] proposed to use a bipartite graph[1] to model the relations between images and their low-level features, so as to convert the image clustering problem to a graph partitioning problem that could be solved by singular value decomposition [16]. Although low-level feature based image clustering has been used in many applications [7][18], its effectiveness is doubtful due to the problem of semantic gap. That is, many images whose appearances are very similar to each other actually belong to quite different categories. For instance, an image of a hawk flying in the sky and another image of a black duck swimming in a lake are quite similar in their colors and textures, even if their semantics are far from each other. Actually, it is the same reason that prevents content-based image retrieval (CBIR) from being widely used in real-world applications.

In contrast to the embarrassment of CBIR, image search tools in today's Web search engines have partially fitted people's information need. Their successes lie in that they have taken a

[1] If the vertices of a graph can be decomposed into two disjoint subsets such that no two vertices within the same set are adjacent, the graph is named a bipartite graph.

different approach from CBIR: the search indexes were actually built on the surrounding texts of the images[2], but not visual features. In such a way, the image clustering problem is converted to a text clustering problem, where traditional text mining techniques [2][6], such as *tf-idf* weighting, cosine similarity measure and so on can be applied. However, it is clear that an image is not a textual document after all. So simply converting image clustering to text clustering is not a perfect solution. One can expect better clustering results if both textural and visual features are utilized to cluster Web images.

Actually, there has been some works [5][20][21][29] on integrating visual and textual information in the literature, although not many. For instance, Cai *et al* [5] proposed to use three representations of a Web image, i.e. representation based on visual features, representation based on textual features and representation induced from link analysis to construct an image relationship graph. Then they used spectral techniques to cluster the search results into different semantic groups by textual features and link information. After that, low-level visual features were used to further cluster images in each semantic category. Therefore, they used textual and visual features successively but not simultaneously, so errors in the first clustering step might propagate to the next step so that the clustering performance might be depressed in some cases. La Cascia *et al* [20] proposed to combine textual and visual statistics in a single index vector for content based search of a WWW image database. Textual statistics were captured in vector form using latent semantic indexing (LSI) [12] based on text in the containing HTML document. Visual statistics were captured in vector form using color and orientation histograms. In another similar work [29], textual feature vector and visual feature vector were firstly combined directly into a global feature vector, and then the LSI technique was applied in this global feature vector for CBIR. In the above two methods [20][29], textual and visual features were combined in a stiff way. However, in our opinion, textual features reflect the external description, interpretation or comment imposed on an image by people, while visual features reflect the internal attributes held by the image. They come from totally different sources and we consider it improper to combine them in such a stiff way. Li *et al* [21] also took visual features, textual features and link information into account when clustering images. They combined the co-clustering between images and terms in surrounding texts and the one-side image clustering based on low-level features into an iterative process. However, they did not prove the convergence property of this algorithm, and in our opinion, this kind of combination is unsymmetrical according to the status of visual and textual features. In this regard, we had better develop some more advanced technology to fuse the heterogeneous information for a better clustering.

Illumined by the idea of image and low-level feature co-clustering [24], in this paper, we propose to use a tripartite graph[3] to model the relations among low-level features, images and their surrounding texts. Then we partition this tripartite graph using a novel technology named consistent bipartite graph co-partitioning (CBGC), which is based on the consistent fusion of two co-clustering sub-problems: the co-clustering of low-level visual features and the images, and the co-clustering of textual features and the images. In other words, we look for such two clustering schemes for the aforementioned two sub-problems, provided that each of them might not be locally optimal but their clustering results on the images are identical and the overall clustering scheme is globally optimal. Actually, similar ideas have been proposed in our former works. The consistent bipartite spectral graph co-partitioning algorithm, which was based on generalized singular value decomposition (GSVD) [16], was proposed in [14] to solve the above tripartite model. This algorithm has a spectral interpretation but does not have a distinct objective function, and the computation cost of GSVD is rather high. In [15], the concept of consistent bipartite graph co-partitioning was proposed and the above problem was modeled by a single objective optimization problem which could be efficiently solved by semi-definite programming (SDP) [4]. In this paper, we model this problem as a multi-objective optimization problem so that a better interpretation might be given under this model. Then the model is solved by the similar technique as in [15]. Tested on real-world image collections, the proposed algorithm showed its high feasibility and validity in Web image clustering.

The rest of this paper is organized as follows. In Section 2 the background knowledge on spectral clustering is introduced while the novel model for image clustering is proposed in Section 3. Then in Section 4 the method to solve consistent bipartite graph co-partitioning is described in details and the experimental results are discussed in Section 5. Concluding remarks and future work directions are listed in the last section.

2. RELATED WORKS

In this section, we will review some research works on spectral clustering, which is the foundation of our proposed method.

2.1 Spectral Clustering

Spectral clustering [1][27] refers to a category of clustering algorithms based on spectral graph partitioning [22], which was proposed and well studied in the literature. To explain how this method works, we need to introduce some basic knowledge about graph theory first.

A graph G=(V, E) is composed by a set of vertices V={1,2,…,|V|} and a set of edges E={<i, j>| i, j∈ V}, where |V| represents the number of vertices. If using E_{ij} to denote the weight of edge <i,j>, we can further define the adjacency matrix M of the graph as follows

$$M_{ij} = \begin{cases} E_{ij}, & if < i, j > \in E \\ 0, & \text{otherwise} \end{cases} . \quad (1)$$

In the spectral graph partitioning methods for image clustering, the vertices correspond to images, and the edges correspond to the similarities between images. The weights of the edges correspond to the strength of the similarities, which can be calculated by a certain measure in the low-level feature space. Supposing that the vertex set V is partitioned into two subsets V_1 and V_2, the corresponding *cut* can be defined as:

[2] In the scenario of Web image search, such an approach is meaningful because most of the images in the web pages are surrounded with rich textual information.

[3] Generally speaking, a k-partite graph is a graph whose graph vertices can be partitioned into k disjoint sets so that no two vertices within the same set are adjacent.

$$cut(V_1,V_2)=\sum_{i\in V_1,\, j\in V_2} M_{ij}\ . \qquad (2)$$

One can easily extend the above definition to the case of k subsets:

$$cut(V_1,V_2,\ldots,V_k)=\sum_{\eta<\theta} cut(V_\eta,V_\theta)\ . \qquad (3)$$

Image clustering is to find clusters such that images in the same cluster are similar while images in different clusters are dissimilar. Then it is easy to see that the clustering objective is equivalent to minimizing the *cut*. Usually, balanced clusters are more preferred, so some variations of the definition of *cut* were proposed and therefore different kinds of spectral clustering methods [10][19][27] were derived. For example, Ratio Cut [19] is achieved by balancing cluster sizes, while Normalized Cut [27] is attained by balancing cluster weights. Among these variations, Normalized Cut (or *NCut*) is one of the most popularly-used spectral clustering methods. Its objective function is shown in (4), where e is the column vector with all its elements equal to 1:

$$\min \frac{q^T Lq}{q^T Dq},\ \text{subject to } q^T De = 0,\ q \neq 0\ . \qquad (4)$$

Here D is a diagonal matrix with $D_{ii}=\sum_k E_{ik}$, and $L=D$-M is called Laplacian matrix. q is a column vector with $q_i= c_1$ if $i\in V_1$ and $q_i=$ -c_2 if $i\in V_2$, where c_1 and c_2 are constants derived from D. By relaxing q_i from discrete values to continuous values, it can be proved that the solution for (4) is the eigenvector corresponding to the second smallest eigenvalue λ_2 of the following generalized eigenvalue problem [9][16][27] :

$$Lq=\lambda Dq\ . \qquad (5)$$

Then we can obtain the desired image clusters by running some routine clustering algorithms such as k-means [11] on this eigenvector q (called the Fiedler vector). However, the efficiency of this method in image clustering is low in many cases, for the computation cost on generating the similarity matrix M is high especially when the dimensionality of the feature vector is large. Besides, different forms of similarity measures might affect the clustering results more or less.

2.2 Bipartite Spectral Graph Partitioning

To depress the computation cost and avoid the effect by different similarity measures in image clustering, Qiu [24] used the undirected bipartite graph in Figure 1 to represent the relationship between images and their low-level features. In this figure, squares and circles represent low-level features $F = \{f_1, f_2,\ldots, f_m\}$ and images $H = \{h_1, h_2,\ldots, h_n\}$ respectively. Then the bipartite graph can be represented by a triplet $G=(F, H, E)$, where E is a set of edges connecting vertices from different vertex sets, i.e., $E=\{<i, j> \mid i\in F, j\in H\}$. If we further use A to denote the inter-relation matrix in which A_{ij} equals to the weight of edge E_{ij}, i.e., the value of low-level feature i for image j, the adjacency matrix of the bipartite graph will be written as:

$$M = \begin{array}{cc} & \begin{array}{cc} F & H \end{array} \\ \begin{array}{c} F \\ H \end{array} & \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \end{array}, \qquad (6)$$

where the vertices have been ordered such that the first m vertices index low-level features while the last n index images.

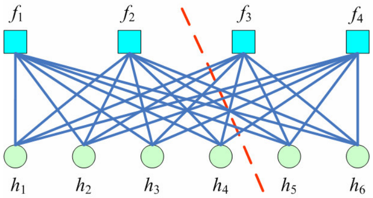

Figure 1. The Bipartite Graph of Low-level Features and Images.

Suppose the dashed line in Figure 1 shows the very partition that minimizes (4), we will obtain two subsets $\{f_1,f_2,h_1,h_2,h_3,h_4\}$ and $\{f_3,f_4,h_5,h_6\}$. Therefore, the low-level features are clustered into two subsets $\{f_1,f_2\}$ and $\{f_3,f_4\}$, while the images are clustered into two subsets $\{h_1,h_2,h_3,h_4\}$ and $\{h_5,h_6\}$ simultaneously. To work out this very partition, we also need to solve a generalized eigenvalue problem like (5). Due to the bipartite property of the graph, after some trivial deduction, this problem can be converted to a singular value decomposition (SVD) [16] problem, which can be computed more efficiently. For the details of this algorithm, please refer to [9][24].

3. LOW-LEVEL FEATURE, IMAGE AND TERM IN SURROUNDING TEXT CO-CLUSTERING

In this section, a tripartite graph model is first proposed to represent the relations among low-level features, images and surrounding texts. And then the concept of *consistency* is presented.

3.1 The Tripartite Graph Model

To make use of both the visual information and the textual information for image clustering, we use the tripartite graph as shown in Figure 2 to model the relations between images and their visual and textual features.

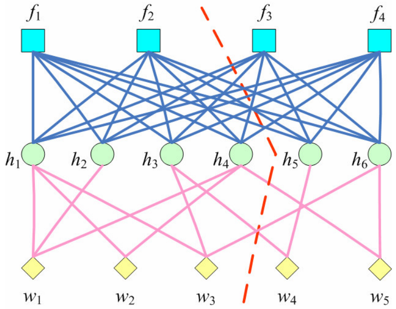

Figure 2. The Tripartite Graph of Low-level Features, Images and Terms in Surrounding Texts.

In this figure, squares, circles and diamonds represent low-level features $F = \{f_1,f_2,\ldots,f_m\}$, images $H = \{h_1, h_2,\ldots, h_n\}$ and terms in surrounding texts $W=\{w_1, w_2,\ldots, w_t\}$ respectively. The weight of an edge between low-level feature i and image j equals the value of low-level feature i in image j, while the weight of an edge between image j and term k equals the frequency of term k in the surrounding text of image j.

If we use A and B to denote the inter-relationship matrices between low-level features and images, and between images and terms respectively, it is easy to derive the adjacency matrix for Figure 2:

$$M = \begin{array}{c|ccc} & F & H & W \\ \hline F & 0 & A & 0 \\ H & A^T & 0 & B \\ W & 0 & B^T & 0 \end{array}, \qquad (7)$$

where the vertices have been ordered such that the first m vertices index low-level features, the next n index images and the last t index terms in surrounding texts.

To co-cluster low-level features, images and surrounding texts simultaneously, it seems natural to partition the graph in Figure 2 by working out the generalized eigenvalue problem corresponding to the adjacency matrix (7). However, we would like to point out that this idea does not always work as it seems. Actually, if we move the vertices of low-level features in Figure 2 to the side of the vertices of terms, it is not difficult to see that the original tripartite graph will turn to be a bipartite graph. Therefore, we are actually working on an {images}-{low-level features & terms in surrounding texts} bipartite graph and the loss of cutting an edge between an image and a low-level feature contributes to the loss function identically to the loss of cutting an edge between an image and a term. However, these two kinds of edges are heterogeneous and might not be comparable. To tackle this problem, in the next subsection, we will present a novel method to avoid this situation.

3.2 Consistent Bipartite Graph Co-Partitioning (CBGC)

To tackle the aforementioned problem, as we have done in [15], we propose to treat the tripartite graph in Figure 2 as two bipartite graphs in Figure 1 and Figure 3 respectively, which share the central part of images in Figure 2. Then we transform the original problem to the fusion of the pair-wise co-clustering problems over these two bipartite graphs.

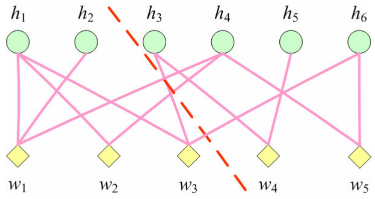

Figure 3. The Bipartite Graph of Images and Terms.

However, if we conduct bipartite spectral graph partitioning [9][24][28] on Figure 1 and 3 independently, it will have a great probability that the partitioning schemes for images are different in the two solutions. In other words, the two locally optimal partitioning schemes in images do not match in most cases. This is not what we want. Actually, we are looking for such two partitions for Figure 1 and 3, provided that each of them is not locally optimal, but their clustering results on images are the same, and the overall partitioning is globally optimal under a certain objective function. We call it by *consistent bipartite graph co-partitioning* (CBGC).

To make the aforementioned concept of CBGC computable, we will give a specific objective function and discuss how to optimize it efficiently. In this paper, we will focus on bi-partitioning, where the three substances will be simultaneously clustered into two groups respectively. For this purpose, we let f, h, w act as the indicating column vectors of m, n, t dimensions for low-level features, images and terms respectively. We denote $q=(f, h)^T$ and $p=(h, w)^T$ as the indicating vectors for the two local bipartite graphs, and denote $D^{(f)}$, $D^{(w)}$, $L^{(f)}$ and $L^{(w)}$ as the diagonal matrices and Laplacian matrices for the adjacent matrices A and B. Then we mathematically model the consistent co-partitioning problem in a manner of multi-objective optimization,

$$\begin{aligned} &\min \frac{q^T L^{(f)} q}{q^T D^{(f)} q} \\ &\min \frac{p^T L^{(w)} p}{p^T D^{(w)} p} \qquad\qquad , \\ &\text{s. t. } (i)\ q^T D^{(f)} e = 0,\ q \neq 0 \\ &\qquad (ii)\ p^T D^{(w)} e = 0,\ p \neq 0 \end{aligned} \qquad (8)$$

4. OPTIMIZING ALGORITHM BASED ON SEMI-DEFINITE PROGRAMMING

In this section we will propose an algorithm to compute the solution of the optimization problem (8) defined in Section 3.2. Actually a very commonly-used approach to solve the multi-objective optimization problem is linearly combining the two objective functions, which is shown as follows,

$$\begin{aligned} &\min \left[\beta \frac{q^T L^{(f)} q}{q^T D^{(f)} q} + (1-\beta) \frac{p^T L^{(w)} p}{p^T D^{(w)} p} \right] \\ &\text{s. t. } (i)\ q^T D^{(f)} e = 0,\ q \neq 0 \\ &\qquad (ii)\ p^T D^{(w)} e = 0,\ p \neq 0 \\ &\qquad (iii)\ 0 < \beta < 1 \end{aligned} \qquad (9)$$

where β is a weighting parameter to balance which local graph we trust more. This form of objective function materializes the concept of consistent bipartite graph co-partitioning. Note that the aforementioned linear combination is only one of the approaches to solve multi-objective programming. One can choose to use other approaches [17] as well.

Then the following derivations are very similar with what we have done in [15]. By setting $\omega=(f, h, w)^T$ to be a combined indicating vector of $s=m+n+t$ dimensions, and extending the matrices $L^{(f)}$,$L^{(w)}$,$D^{(f)}$ and $D^{(w)}$ to adapt the dimension of ω as follows[4]:

$$\Gamma_1 = \begin{bmatrix} L^{(f)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}_{s\times s},\ \Gamma_2 = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & L^{(w)} \end{bmatrix}_{s\times s}, \qquad (10)$$

$$\Pi_1 = \begin{bmatrix} D^{(f)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}_{s\times s},\ \Pi_2 = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & D^{(w)} \end{bmatrix}_{s\times s}, \qquad (11)$$

we have

[4] Here the **0**’s are matrix blocks with all the elements equal to zero.

$$
\begin{aligned}
&\min\left[\beta\frac{\omega^T\Gamma_1\omega}{\omega^T\Pi_1\omega}+(1-\beta)\frac{\omega^T\Gamma_2\omega}{\omega^T\Pi_2\omega}\right].\\
&\text{s.t. } (i)\ \omega^T\Pi_1 e=0\\
&\qquad (ii)\ \omega^T\Pi_2 e=0\\
&\qquad (iii)\ \omega\neq 0,\ 0<\beta<1
\end{aligned}
\tag{12}
$$

Problem (12) is a typical sum-of-ratios quadratic fractional programming problem [13], which is hard and complicated to solve although there has been some branch-and-bound algorithms [3]. To avoid solving this fractional programming problem, we use a familiar skill in spectral clustering to simplify it: by fixing the values of the denominators in (12) to $e^T\Pi_1 e$ and $e^T\Pi_2 e$ respectively, we have:

$$
\begin{aligned}
&\min\ \omega^T\Gamma\omega\\
&\text{s.t. } (i)\ \omega^T\Pi_1\omega=e^T\Pi_1 e\\
&\qquad (ii)\ \omega^T\Pi_2\omega=e^T\Pi_2 e\\
&\qquad (iii)\ \omega^T\Pi_1 e=0\\
&\qquad (iv)\ \omega^T\Pi_2 e=0
\end{aligned}
\ ,
\tag{13}
$$

where

$$
\Gamma=\frac{\beta}{e^T\Pi_1 e}\Gamma_1+\frac{1-\beta}{e^T\Pi_2 e}\Gamma_2,\ 0<\beta<1.
\tag{14}
$$

Optimization problem (13) turns to be a quadratically constrained quadratic programming (QCQP) [4] problem, and it is not difficult to verify that the constraints are all convex because matrices Π_1 and Π_2 are both positive semi-definite. As we know, convex QCQP problem can be cast in the form of a semi-definite programming problem (SDP) [4] for efficient computation.

SDP is an optimization problem with the form as below:

$$
\begin{aligned}
&\min\ C\bullet W\\
&\text{s.t. } (i)\ A_i\bullet W=b_i,\ i=1,\ldots,k\\
&\qquad (ii)\ W\ \textit{is positive semidefinite}
\end{aligned}
\ ,
\tag{15}
$$

where C is a symmetric coefficient matrix and W is a symmetric parameter matrix; A_i (and b_i), i=1,…,k are coefficient matrices (and vectors) for the constraints; the matrix inner-product is defined as:

$$
C\bullet W=\sum_{i,j}C_{ij}W_{ij}\ .
\tag{16}
$$

As done in [15], we further reformulate this QCQP as a SDP by relaxing the product terms $\omega_i\omega_j$ to an element Ω_{ij} of a symmetric matrix Ω.:

$$
\begin{aligned}
&\min_{\omega,\Omega}\ \begin{bmatrix}0 & \mathbf{0}\\ \mathbf{0} & \Gamma\end{bmatrix}\bullet\begin{bmatrix}1 & \omega^T\\ \omega & \Omega\end{bmatrix}\\
&\text{s.t. } (i)\ \begin{bmatrix}-e^T\Pi_1 e & \mathbf{0}\\ \mathbf{0} & \Pi_1\end{bmatrix}\bullet\begin{bmatrix}1 & \omega^T\\ \omega & \Omega\end{bmatrix}=0\\
&\qquad (ii)\ \begin{bmatrix}-e^T\Pi_2 e & \mathbf{0}\\ \mathbf{0} & \Pi_2\end{bmatrix}\bullet\begin{bmatrix}1 & \omega^T\\ \omega & \Omega\end{bmatrix}=0\\
&\qquad (iii)\begin{bmatrix}0 & e^T\Pi_1/2\\ \Pi_1 e/2 & \mathbf{0}\end{bmatrix}\bullet\begin{bmatrix}1 & \omega^T\\ \omega & \Omega\end{bmatrix}=0\\
&\qquad (iv)\begin{bmatrix}0 & e^T\Pi_2/2\\ \Pi_2 e/2 & \mathbf{0}\end{bmatrix}\bullet\begin{bmatrix}1 & \omega^T\\ \omega & \Omega\end{bmatrix}=0\\
&\qquad (v)\ \begin{bmatrix}1 & \omega^T\\ \omega & \Omega\end{bmatrix}\ \textit{is positive semidefinite}
\end{aligned}
\tag{17}
$$

As it has been proved that the SDP relaxation of a QCQP may produce an approximation to the original problem with a good error bound [26], we further ignore the constraints of $\Omega=\omega_i\omega_j$ and get the following relaxation:

$$
\begin{aligned}
&\min_W\ \begin{bmatrix}0 & \mathbf{0}\\ \mathbf{0} & \Gamma\end{bmatrix}\bullet W\\
&\text{s.t. } (i)\ \begin{bmatrix}-e^T\Pi_1 e & \mathbf{0}\\ \mathbf{0} & \Pi_1\end{bmatrix}\bullet W=0\\
&\qquad (ii)\ \begin{bmatrix}-e^T\Pi_2 e & \mathbf{0}\\ \mathbf{0} & \Pi_2\end{bmatrix}\bullet W=0\\
&\qquad (iii)\begin{bmatrix}0 & e^T\Pi_1/2\\ \Pi_1 e/2 & \mathbf{0}\end{bmatrix}\bullet W=0\\
&\qquad (iv)\begin{bmatrix}0 & e^T\Pi_2/2\\ \Pi_2 e/2 & \mathbf{0}\end{bmatrix}\bullet W=0\\
&\qquad (\mathrm{v})\ \begin{bmatrix}1 & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\bullet W=1,\\
&\qquad (\mathrm{vi})\ \begin{bmatrix}0 & e\\ e & \mathbf{0}\end{bmatrix}\bullet W=\theta_1,\\
&\qquad (\mathrm{vii})\ \begin{bmatrix}0 & \mathbf{0}\\ \mathbf{0} & E\end{bmatrix}\bullet W=\theta_2\\
&\qquad (viii)\ W\ \textit{is positive semidefinite}
\end{aligned}
\tag{18}
$$

where E is a matrix block with all the elements equal to one; the constraint $(v)\begin{bmatrix}1 & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\bullet W=1$ guarantees W_{11}=1, and the next two constrains ((vi) and (vii)) are bound controllers with some constants θ_1 and θ_2. We will discuss these parameters in Section 5.

Up to now, we have got a standard form of SDP. The first column of W (except W_{11}) can be regarded as the representation of ω. As SDP is a hot research field [26] in recent years, there are many toolkits available such as SDPA[5], SDPT3[6] and SeDuMi[7], almost

[5] http://grid.r.dendai.ac.jp/sdpa/

[6] http://www.math.nus.edu.sg/~mattohkc/sdpt3.html

all of which are based on fast iterative algorithms. We could use any of these toolkits to compute an efficient solution to the optimization problem (18).

To summarize, our algorithm to solve the co-clustering of low-level features, images and terms can be listed as below. This algorithm was firstly proposed by us in [15] and is modified to adapt the multimedia applications in this paper. For ease of reference, we use F-I-T (low-level ***F***eatures, ***I***mages, ***T***erms in surrounding texts) to abbreviate it in the future discussions.

The F-I-T Algorithm

1. Set the parameters β, θ_1 and θ_2.
2. Given the inter-relation matrices A and B, form the corresponding diagonal matrices and Laplacian matrices $D^{(f)}$, $D^{(w)}$, $L^{(f)}$ and $L^{(w)}$.
3. Extend $D^{(f)}$, $D^{(w)}$, $L^{(f)}$ and $L^{(w)}$ to Π_1, Π_2, Γ_1 and Γ_2, and form Γ, such that the coefficient matrices in SDP (18) can be computed.
4. Solve (18) by a certain iterative algorithm such as SDPA.
5. Extract ω from W and regard it as the embedding vector of low-level features, images and terms.
6. For image clustering, extract the embedding vector h of images from ω and run some traditional clustering algorithms such as the k-means algorithm or threshold split algorithm on h to obtain the desired clusters of images.

5. EXPERIMENTAL EVALUATION

In this section, we present the experiments that we used to evaluate the effectiveness of the proposed consistency concept and the corresponding SDP-based algorithm. For this purpose, we first show the influence of the parameters β, θ_1 and θ_2 on the clustering accuracy, and then compare the proposed algorithm with low-level feature based image clustering and surrounding text based image clustering respectively.

5.1 Data Preparation

All the data used in our experiments were crawled from the Photography Museums and Galleries[8] of the Yahoo! Directory. Images and their surrounding texts were extracted from the crawled Web pages. We filtered out those images whose width-height ratios are larger than 5 or smaller than 1/5, and those images whose width and height are both less than 60 pixels, because such kinds of images are most probably of low quantity. After that, the remaining 17,000 images were assigned to 48 categories manually.

In our experiment, we randomly selected 10 categories of images from the aforementioned dataset, the names and sizes of which are listed in Table 1. To give a more vivid impression, we randomly selected 8 samples from each category and put their thumbnails in Figure 4. We totally extracted 530-dimension color and texture features as the low-level visual representation of the images (See Table 2).

Table 1. The Image Categories Used in the Experiments.

Category Name	Category Size	Category Name	Category Size
Bat	48	*Hill*	82
Bear	57	*Hummingbird*	69
Caterpillar	64	*Map*	31
Coral	87	*Moth*	87
Flying	70	*Owl*	86

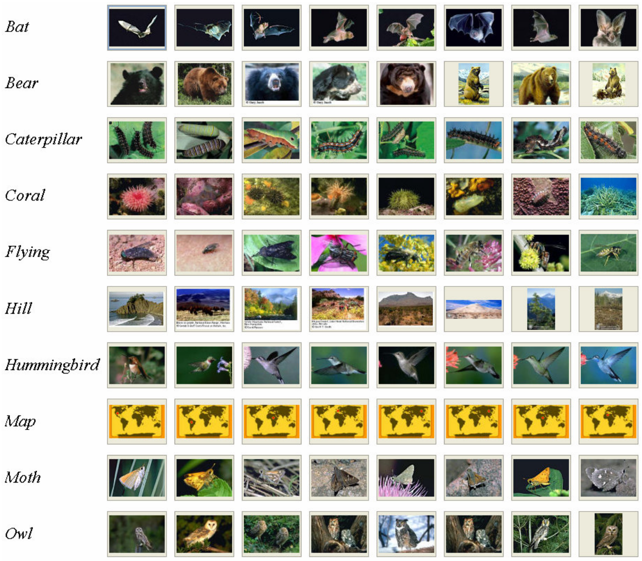

Figure 4. Thumbnails of Samples from the Collection.

Table 2. The Low-Level Features Extracted from Images.

Feature Category	Feature Name	Dimensions
Color	Color Histogram Features	256
	Color Moment Features	9
	Color Coherence Features	128
Texture	Tamura Texture Features	18
	Wavelet Features[7]	104
	MRSAR [22]	15

And for the surrounding texts, we removed the stop words such as prepositions, conjunctions, articles and pronouns and so on. The remaining words were regarded as textual representations of the images in our experiments. The dimensionality of the textual feature ranges from several hundred to more than one thousand, change with different subset of images. Because there are not many textual features for one single image, the term-image adjacency matrix B might be very sparse. This may affect the connectivity of the corresponding image-term bipartite graph and make the corresponding spectral analysis less robust. To tackle this problem, we smoothed the matrix B by adding an additional term that connects to all the images and setting the corresponding edge weights to be the reciprocal of the number of images.

[7] http://fewcal.kub.nl/sturm/software/sedumi.html

[8] http://dir.yahoo.com/Arts/Visual_Arts/Photography/Museums_and_Galleries/

5.2 Experiment Settings

For comparison, we also tested low-level feature based image clustering method and surrounding text based image clustering method on the above data set. Low-level ***F***eature based ***I***mage clustering, abbreviated by us as F-I algorithm, uses bipartite spectral graph partitioning to get image clusters. (For details of this algorithm, please refer to Section 2.2 and [24].) Surrounding text based image clustering, treats terms in the surrounding texts of images as textual features and also uses bipartite spectral graph partitioning to get image clusters [9]. For ease of reference, we abbreviate it as I-T (***I***mages and ***T***erms in their surrounding texts) algorithm.

In our experiments, we simply used 0 as a threshold to partition the embeddings of images to get the bi-clustering results. To evaluate different algorithms, we used cross accuracy as metric. If the concerned subset is mixed with category *I* and category *II* with n_1 and n_2 images respectively, the ground truth can be represented by a Boolean vector *rt*,

$$rt = (1,1,...,1,0,0,...,0)\,, \qquad (19)$$

in which the first n_1 elements are set to 1 and the rest n_2 elements are set to 0.

After image clustering, the results can also be converted to a Boolean vector *rc*, the element arrangement of which is the same with *rt*. Then the definition of cross accuracy is given as follows, where *XOR* means the exclusive-OR operator.

$$accuracy = \max\left\{ \frac{\sum_i (rt_i\ XOR\ rc_i)}{n_1+n_2},\ 1-\frac{\sum_i (rt_i\ XOR\ rc_i)}{n_1+n_2} \right\}. \qquad (20)$$

5.3 Parameter Tuning

As we know, the parameter β in the proposed F-I-T algorithm controls which local bipartite graph we can trust more. To see how this parameter will influence the clustering performance, we mixed the images in the categories of *Coral* and *Bat* (the thumbnails[9] of which are shown in Figure 5 and Figure 6) and tuned β in the interval of [0, 1]. The corresponding results are plotted in Figure 7. From this figure, we can see that the accuracy drops seriously at the extreme points of $\beta = 0$ and $\beta = 1$, while in a wide range within (0.2, 0.8) it is correspondingly stable. In particular, we chose $\beta = 0.6$ as the basic setting of the following experiments.

As for the influence of the other two parameters, θ_1 and θ_2 in the F-I-T algorithm on the clustering performance, we plot the performance surface with respect to different values of θ_1 and θ_2 in Figure 8. From this figure we can see that there is a large high-performance area when $\theta_1 \geq \theta_2$, while the accuracy drops severely when $\theta_1 < \theta_2$. Without loss of generality and for ease, we would set both θ_1 and θ_2 to 1 when comparing the F-I-T algorithm to the two reference algorithms.

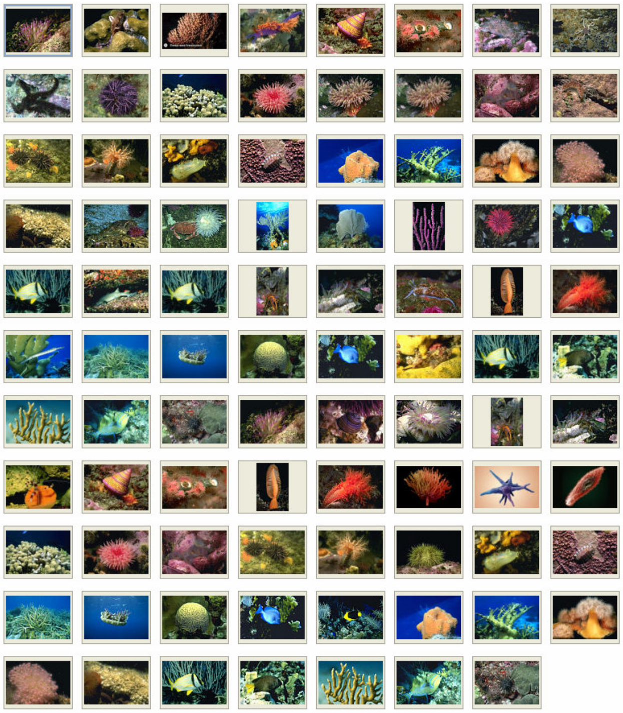

Figure 5. Thumbnails of Images in the Category of *Coral*.

Figure 6. Thumbnails of Images in the Category of *Bat*.

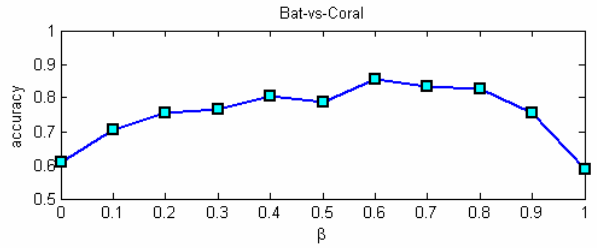

Figure 7. Clustering Performance under Different Values of *β*.

[9] From the thumbnails in Figure 5 and Figure 6, readers might find there seems to lay some repeated images. Actually, they were crawled from different Website and might have different resolutions, color spaces and surrounding texts.

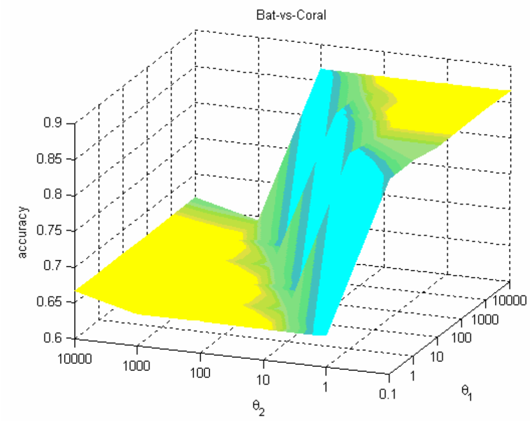

Figure 8. Clustering Performance under Different Values of θ_1 and θ_2.

5.4 A Glance of the Clustering Results

In this subsection, we randomly select two category pairs (*Hill* vs. *Owl* and *Flying* vs. *Map*) to investigate the clustering performance of the F-I-T algorithm as well as the two reference algorithms (F-I and I-T). The corresponding results can be seen in Figure 9 and Figure 10, where "o" and "+" indicate the different clustering results of the images.

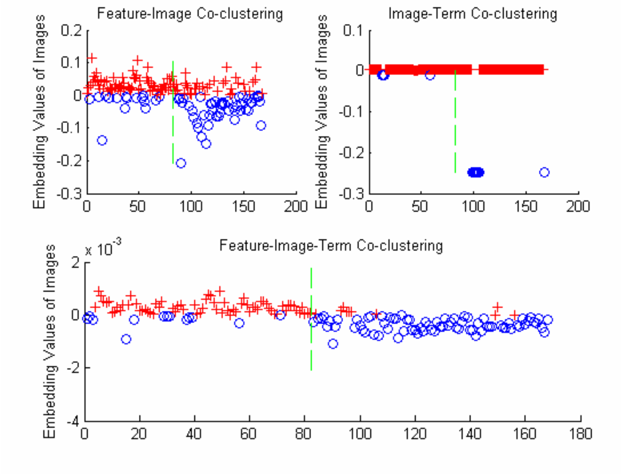

Figure 9. Embedding Values for *Hill* and *Owl*.

In each sub-figure of Figure 9, the vertical axis indicates the embedding values of the images, and the horizontal axis indicates the indices of them, which have been ordered such that the first 82 points index the images in *Hill* while the next 86 points index the images in *Owl*. We can see that the clustering results of F-I are bad. This is because the low-level features of some images from different categories are quite similar. We know the main color of a hill and an owl might both be puce or dark, and we found there are images of a flying owl in the background of hills. The performance of I-T is even worse because the surrounding texts are so infrequent in this subset that many images only have a few words with them. The F-I-T algorithm utilizes the information from both low-level features and surrounding texts and output the best clusters among the three algorithms.

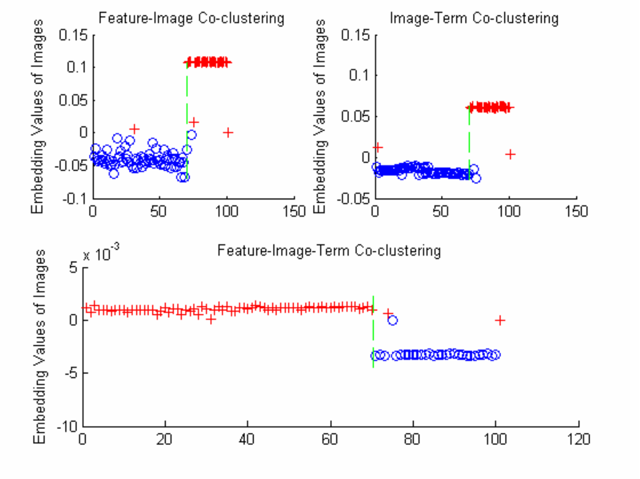

Figure 10. Embedding Values for *Flying* and *Map*.

From Figure 10, we can see that all three methods performed excellently. This is because the low-level features of images in *Flying* and *Map* are quite different, and the surrounding texts in this subset are rich and easily distinguishable.

From the above two figures, we can see that: on one hand, when information from low-level features or surrounding texts are good enough for image clustering, the F-I-T algorithm can also get nice results as F-I or I-T algorithms; on the other hand, when low-level features or surrounding texts are not good enough for distinguishing different categories of images, the F-I-T algorithm can leverage these information to get better output.

5.5 Average Performance

In this section, we would like to report the clustering performance for all possible pairs of categories in our experimental dataset. We plot the F-I-T vs F-I and F-I-T vs I-T figures in Figure 11, each point in which represents a possible category pair. We can see that most of the points fall in the upper side of the diagonal in either sub-figure, indicating that the F-I-T algorithm outperforms the other two methods in most cases. Though there are several cases that our algorithm performs inferior to one of the two reference algorithms or even both of them, they are infrequent and do not affect the superiority of the average performance of F-I-T.

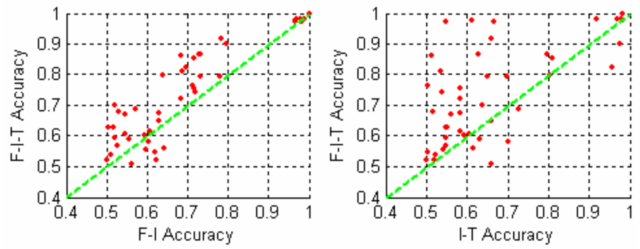

Figure 11. Performance Comparison.

The averaged performance for between each category and all the other categories are listed in Table 3, where the surpassing values are blackened. From Table 3, we can see that, averagely speaking, our algorithm succeeds in 80% categories. We can also see that in a global average view, the F-I-T algorithm outperforms F-I and I-

T methods by 5%, and in an average view of category-respective, F-I-T is at best 10% better than the reference algorithms.

Table 3. Average Performance.

Category Name	F-I	I-T	F-I-T
Bat	0.7307	0.6866	**0.8266**
Bear	0.6303	0.5920	**0.6857**
Caterpillar	0.6297	0.6240	**0.6805**
Coral	0.6494	0.6351	**0.6932**
Flying	0.6554	**0.6917**	0.6892
Hill	0.7369	0.6500	**0.8203**
Hummingbird	**0.6567**	0.6308	0.6518
Map	0.9594	0.8488	**0.9708**
Moth	0.6332	0.7071	**0.7213**
Owl	0.6302	0.5483	**0.6633**
Total Average	0.6912	0.6614	**0.7403**

After a macro view of the results, we would like to investigate several special cases to get more insights. In the case of *Map*, the accuracies of the three algorithms are high because their low-level features are almost the same within this category and quite different with others (please refer to Figure 4), and their surrounding texts are also extremely alike with each other. We tracked down by following clues from the crawling list and found that these images come from a Website illuminating the distributing of dinosaurs. Almost all of their surrounding texts contain terms like *map*, *dinosaur*, *locate*, etc. Similarly, the performance in *Bat* is high since most of the backgrounds of the bats are black, which causes the similarity of their low-level features. In the cases of *Flying* and *Hummingbird*, F-I-T failed to hit the top (but it is only no more than 0.5% lower than the winner). In these cases, either the low-level features are bad-regulated or the surrounding texts are confused and inaccurate, and thus they would have negative effect on the proposed algorithm.

To sum up, the F-I-T algorithm would achieve better cross accuracy than the two reference algorithms in the majority cases.

5.6 An Image Search System

At the end of the section, we would like to show an application of our method. We organized the corpora (17,000 images in total) described in Section 5.1 in a database and built an image search system based on the proposed algorithm. When a user submits a query, the system will search in the table of the surrounding texts of all images and retrieve the images whose surrounding texts contains this query. Then the F-I-T algorithm is implemented on the retrieved images to get clusters, which are organized in a friendly interface to the user.[10] For example, when the query "*bird*" was submitted by a user, the system retrieved 832 images. After clustering by F-I-T, they were re-organized as 3 clusters shown in the left part of Figure 12. We can see that the three clusters are *birds in forests*, *birds on water* and *birds in the sky*. If the user clicked one of the clusters, all images grouped in this cluster would be shown in the right part.

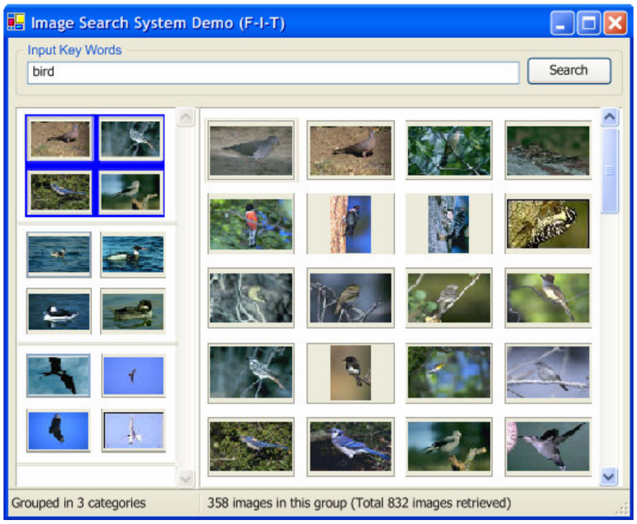

Image Search System Demo (F-I-T)

Input Key Words: bird

Grouped in 3 categories

358 images in this group (Total 832 images retrieved)

Figure 12. A View of Our Image Search System.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we used a tripartite graph to model the co-clustering problem of low-level features, images and terms in surrounding texts, and proposed the concept of consistent bipartite graph co-partitioning to get the co-clustering of the three substances simultaneously. Then we proved our desired consistent co-clustering can be achieved by optimizing a certain objective function based on semi-definite programming. Experiments on a collection of digital photographs showed the effectiveness and validity of our approach. For the future work, we will further explore whether there are any more reasonable objective functions, and whether it is possible to get a close-form solution for them.

7. ACKNOWLEDGMENTS

This work was done when the first, the third and the fourth authors were interns at Microsoft Research Asia. We would like to thank Professor Kurt M. Anstreicher for his great generosity and enthusiasm in helping us nail down some facts on semi-definite programming. We should also thank Hang-Hang Tong, Xin-Jing WANG and Deng CAI for their enthusiasm in helping us prepare the collection of images and surrounding texts.

8. REFERENCES

[1] Bach, F.R., and Jordan, M.I. Learning spectral clustering. Neural Info. Processing Systems 16 (NIPS 2003), 2003.

[2] Baeza-Yates, R., and Ribeiro-Neto, B. Modern information retrieval. ACM Press, a Division of the Association for Computing Machinery, Inc. (ACM). 1999.

[3] Benson, H.P. Global Optimization Algorithm for the Nonlinear Sum of Ratios Problem. Journal of Optimization Theory and Applications: Vol. 112, No. 1, pp. 1–29, January 2002.

[4] Boyd, S., and Vandenberghe, L. Convex Optimization. Cambridge University Press, 2004.

[10] As we focus on bi-clustering problem in this paper, for *k*-clustering cases, we simply ran the *k*-means algorithm on the extracted embeddings of images to get the desired clusters. Note that there must be some better ways to generalize our method to the case of k-clustering, but it has been beyond the scope of this paper.

[5] Cai, D., He, X., Li, Z., Ma, W., and Wen, J. Hierarchical Clustering of WWW Image Search Results Using Visual, Textual and Link Information. In ACM Multimedia 2004, 2004.

[6] Cai, D., He, X., Ma, W., Wen, J., and Zhang, H. Organizing WWW Images Based on The Analysis of Page Layout and Web Link Structure. In the 2004 IEEE International Conference on Multimedia and EXPO, 2004.

[7] Chang, T., and Kuo, C. -CJ. Texture analysis and classification with tree-structured wavelet transform. IEEE Transactions on Image Processing, 2, 4(Oct. 1993), 429-441.

[8] Chen, Y., Wang, J. Z., and Krovetz, R. Content-based image retrieval by clustering. In Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval, pages 193–200. ACM Press, 2003.

[9] Dhillon, I.S. Co-clustering documents and words using bipartite spectral graph partitioning. In KDD'01, 2001.

[10] Ding, C., He, X., Zha, H., Gu, M., and Simon, H. A min-max cut algorithm for graph partitioning and data clustering. Proc. IEEE Int'l Conf. Data Mining, 2001.

[11] Duda, R.O., Hart, P.E., and Stork, D.G. Pattern classification, Second Edition. John Wiley & Sons Inc. 2001.

[12] Dumais, S.T. Latent semantic analysis. Annual Review of Information Science and Technology (ARIST), Volume 38, Chapter 4, 189-230, 2004.

[13] Frenk, J.B.G., and Schaible, S. Fractional Programming. ERIM Report Series Reference No. ERS-2004-074-LIS. http://ssrn.com/abstract=595012.

[14] Gao, B., Liu, T., Cheng, Q., Feng, G., Qin, T., and Ma, W. Hierarchical Taxonomy Preparation for Text Categorization Using Consistent Bipartite Spectral Graph Co-partitioning. IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 9, pp. 1263-1273, September 2005.

[15] Gao, B., Liu, T., Zheng, X., Cheng, Q., and Ma, W. Consistent Bipartite Graph Co-Partitioning for Star-Structured High-Order Heterogeneous Data Co-Clustering. In Proceedings of ACM SIGKDD 2005.

[16] Golub, G.H., and Loan, C.F.V. Matrix computations. Johns Hopkins University Press, 3rd edition, 1996.

[17] Freitas, Alex A. A Critical Review of Multi-Objective Optimization in Data Mining, SIGKDD Explorations, vol.6, Issue.2: 77-86, 2004.

[18] Gordon, S., Greenspan, H., and Goldberger, J. Applying the information bottleneck principle to unsupervised clustering of discrete and continuous image representations. In ICCV, 2003.

[19] Hagen, L., and Kahng, A.B. New spectral methods for ratio cut partitioning and clustering. IEEE. Trans. on Computed Aided Desgin, 11:1074-1085, 1992.

[20] La Cascia, M., Sethi, S., and Sclaroff, S. Combining Textual and Visual Cues for Content-based Image Retrieval on the World Wide Web. IEEE Workshop on Content-based Access of Image and Video Libraries, June 1998.

[21] Li, Z., Xu, G., Li, M., Ma, W., and Zhang, H. Group WWW image search results by novel inhomogeneous clustering method. In proceedings of MMM'04, 2004.

[22] Mao, J. C., and Jain, A. K. Texture classification and segmentation using multiresolution simultaneous autoregressive models. Pattern Recognition, 25, 2(1992), 173-188.

[23] Pothen, A., Simon, H.D., and Liou, K.P. Partitioning sparse matrices with eigenvectors of graph. SIAM Journal of Matrix Anal. Appl., 11:430-452, 1990.

[24] Qiu, G. Image and Feature Co-clustering. ICPR (4) 2004: 991-994.

[25] Rodden, K., Basalaj, W., Sinclair, D., and Wood, K. R. Does organisation by similarity assist image browsing? In Proceedings of Human Factors in Computing Systems, 2001.

[26] Semidefinite Programming. http://www-user.tu-chemnitz.de/~helmberg/semidef.html.

[27] Shi, J., and Malik, J. Normalized cuts and image segmentation. IEEE. Transactions on Pattern Analysis and Machine Intelligence, 22:888--905, 2000.

[28] Zha, H., Ding, C., and Gu, M. Bipartite graph partitioning and data clustering. In proceedings of CIKM'01, 2001.

[29] Zhao, R. and Grosky, W.I. Narrowing the Semantic Gap - Improved Text-Based Web Document Retrieval Using Visual Features. IEEE Transactions on Multimedia, Vol. 4, No. 2, June 2002.