

Revised version, submitted to Pattern Recognition, Jan. 2009.

Interactive unsupervised classification and visualization for browsing an image collection

Pierrick Bruneau^{1,2}, Fabien Picarougne¹ and Marc Gelgon^{1,2}

(1) Nantes university, LINA (UMR CNRS 6241), Polytech'Nantes
rue C.Pauc, La Chantrerie, 44306 Nantes cedex 3, France

(2) INRIA Atlas project-team
firstname.surname@univ-nantes.fr

1 Introduction

Content-based image retrieval has been a matter of much attention and literature in the last decade [13]. Distinct goals may be distinguished. On the one hand, one may aim at identifying various occurrences of a single physical scene or object, making for variability of appearance [24]. On the other hand, there exists numerous needs where the target images is not initially accurately expressed, either because they are difficult to express, or because the user rather wants to explore the content of an unknown image collection. Our paper fits this latter setting, in related to a surveillance need. More precisely, our goal¹ is provide a scheme that assists a human operator to monitor the flow of images travelling through network routers from/to a set of users that have previously identified as suspect. The elementary solution, involving display of all images, proves too tedious for the operator, thus an organized and more concise view is sought. We hence put forward a solution based on the following principles :

- the image set is organized into groups of visually similar groups, supplying a graph where image groups are nodes and vertices reflect inter-group similarity. Fig. 1 summarizes this phase;
- the collection is displayed in a way that reveals its visual structure, by means of the abovementioned graph. Fig. 7 supplies an example screenshot.

A major point of the contribution is that the algorithmic solutions proposed for clustering and visualization are can efficiently accommodate change in structure due to time-varying image collections or user feedback via the interface, i.e. low computational cost and temporal consistency of the structure is ensured.

The novelty of the system is not in the image features that are extracted, hence their description are reported with the experimental results. Rather, we select the setting of mixture models in a multivariate feature space for describing images, and the contribution operates in this context, which can in practice capture the probability distribution of feature in many applicative situations.

For instance in [9], the authors modeled color and texture segments with a Gaussian mixture. Users can build a query by choosing significant segments from an image, obtaining sorted relevant results from the database. The framework presented in [11] is closer to our work in that it builds a local cluster structure in the neighborhood of a query image. In [27], authors

¹This work is funded under the ANR Safeimage/Systems and Tools for Global Security programme, in partnership with Alcatel-Lucent.

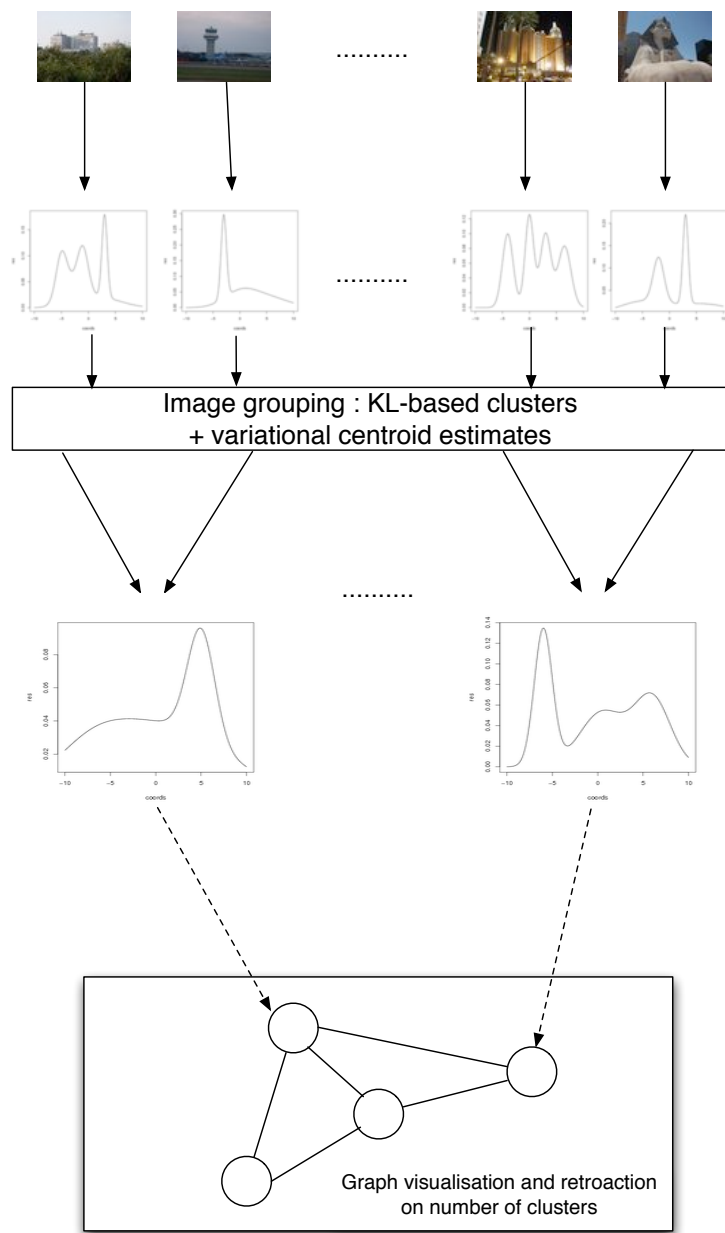


Figure 1: Summary of the proposed approach : for each image, a mixture model is estimated. Then, images are grouped via variational-Bayes parameter-based clustering among mixture models, in an unsupervised fashion. A graph of visually similar images groups is finally built, based on distance between centroid densities of these groups. As new images flow in, the cluster structure may be updated at low cost, as updates only rely on a concise set of model parameters rather than low-level image features. Layout and interaction with the graph are discussed further down in the paper.

rely on appropriate choice of salient features and vector quantization learning to assess hierarchical classification of images in a number of predefined semantic classes (indoor/outdoor, landscape/city, etc...). Our contribution rather proposes a method to cluster a set of images, which is about discovering some information and structure in an unstructured and unlabeled data set. Our clustering algorithm operates on Gaussian mixtures (i.e. a Gaussian mixture being an item to cluster), and to our knowledge, clustering of continuous densities has been little addressed in the literature.

In the present work, user feedback does not appear in terms of improving a query, but rather capture user's intentions regarding the classifier itself. This pertains to semi-supervised learning techniques. Only few examples will be presented here, for a more comprehensive survey see [10]. Nigam et al. [25] initialize their cluster structure with a set of user-labeled examples, and estimate a mixture model on whole data using EM algorithm. Basu et al. [4] model constraints with hidden Markov random fields, and describe a modified objective function for the k-means clustering algorithm [23] that takes these constraints into account. Their scheme also includes a learning step for a flexible distortion measure.

To reduce the cognitive load of the user, we only show, at the beginning of the visualization process, the evolution of the distance between each cluster. We construct a complete graph where nodes represent clusters. To give users an explicit view to the content of each cluster, and meanwhile limit the amount of information on the screen, each nodes displays few thumbnails of its most representative images. The weights of edges relate directly to the distance between clusters. During the clustering process, as the number of clusters and the distance between clusters evolve, the graph is updated accordingly by a spring based algorithm [19, 15].

Visualizing a complete graph is a difficult task. However, since the number of cluster produced by our algorithm is relatively small (maximum 20 clusters), we allow the user to set a distance threshold for edge filtering, as long as the graph remains connected. We also let the user interact with the visualization by viewing all the images of a cluster, rotating the visualization in order to better understand the relations between clusters, and provide feedback to the clustering algorithm by removing or adding nodes.

The remainder of this paper is organised as follows. In section 2, we describe mechanisms and baseline algorithms used to build clusters from an image set. In section 3, we expose a dynamic visualization system that provides user-appealing interaction with the cluster structure. Section 4 contains some experimental evaluations on a real image set. Finally, in section 5 we draw some conclusions and perspectives.

2 Clustering a set of images

This task is composed of two successive phases : choosing appropriate image features, and designing an image grouping technique that runs on these representations. We first introduce a joint spatial and color feature space on image pixels. Pixels from a single image will then be modeled by a Gaussian mixture fitted with the Variational Bayesian EM algorithm [1, 6]. This procedure addresses the choice of a relevant number of components. Then we propose an iterative image grouping algorithm. Akin to k-means, it requires computation of cluster centers. As the items we want to cluster are represented by Gaussian mixtures, the most immediate way to do so would be a direct average of all the Gaussian mixtures forming a cluster. As this will quickly lead to a great number of components, and consequently a high computational cost for the distances of images w.r.t. their current center, we disclose a variational Bayesian procedure for Gaussian mixture reduction that will build an approximation of the true mean. This method is a novel derivation of the above mentioned Variational Bayesian EM algorithm. The intrinsic properties of the baseline method permit us to reduce a Gaussian mixtures input set to the most sensible low complexity model. This reduction is computationally efficient, and obtained centroids allow faster computations for the distances.

2.1 Image representation

Rather than the classic (R,G,B) color space, we will use (L,a,b), as the latter is known for good perceptual properties, leading to meaningful distances [16, 30]. To preserve spatial organization, we add pixel coordinates, thus obtaining the 5-dimensional feature space (L,a,b,x,y).

Clustering a set of images using pixel wise dissimilarities is intractable. Instead we adopt a more parsimonious approach, and represent an image by fitting a Gaussian mixture in the feature space defined above. While reducing data set size, doing so is equivalent to building groups of homogeneous pixels, and building a similarity measure on such abstractions is sensible [16, 28]. Fitting a GMM on a data set is classically achieved with EM algorithm [14] for the estimation of a maximum likelihood solution. Since this method alone cannot determine the number of components in the mixture, we usually apply complexity-related penalty terms. Variational estimation is a more efficient procedure, enabling the Bayesian estimation of a model and a relevant number of components in the same process [12]. This technique has been widely applied to point-wise data clustering [1, 6]. We actually use this framework twice in this paper : in the classical, point-wise case, for modeling a single image; then in an extension, which we propose as a contribution, involved in grouping images. we recall here its main principles.

We consider a set of d-dimensional data vectors $X = (x_1, \dots, x_n)^T$, to which we attempt

to fit a probabilistic model parameterized by θ . For a Gaussian mixture, this parameter set defines the following distribution :

$$p(x_n | \theta) = \sum_{k=1}^K \omega_k \mathcal{N}(x_n | \mu_k, \Lambda_k^{-1}) \quad (1)$$

where ω_k , μ_k and Λ_k are respectively the weight, mean vector and precision matrix for the component θ_k , and the full parameter set is denoted by $\theta = \{\theta_k\}$. We also define the following lightweight notations: $\Omega = \{\omega_k\}$, $\mu = \{\mu_k\}$ and $\Lambda = \{\Lambda_k\}$. The ω_k are under the constraint $\sum_k \omega_k = 1$.

Under i.i.d. assumption for X , we can conveniently decompose the global distribution:

$$p(Z | \Omega) = \prod_{n=1}^N \prod_{k=1}^K \omega_k^{z_{nk}} \quad (2)$$

$$p(X | Z, \mu, \Lambda) = \prod_{n=1}^N \prod_{k=1}^K \mathcal{N}(x_n | \mu_k, \Lambda_k^{-1})^{z_{nk}} \quad (3)$$

where Z is a set of binary variables denoting the component from which each element of X originates, i.e. $z_{nk} = 1 \equiv x_n$ i.i.d. from θ_k . The variational approach applied on the context of a Gaussian mixture consists in defining a variational distribution :

$$q(Z, \Omega, \mu, \Lambda) = q(Z)q(\Omega) \prod_k q(\mu_k, \Lambda_k) \quad (4)$$

The purpose of the method is to maximize a lower bound to the constant marginal likelihood $\ln p(X)$, which is equivalent to minimize $\text{KL}(q(Z, \Omega, \mu, \Lambda) \| p(Z, \Omega, \mu, \Lambda | X))$, where $p(Z, \Omega, \mu, \Lambda | X)$ is the true posterior distribution. This formulation leads to the following general result for the optimal form q^* :

$$\ln q_j^* = \mathbb{E}_{i \neq j} [\ln p(X, Z, \Omega, \mu, \Lambda)] + \text{const} \quad (5)$$

where $i, j \in \{Z, \Omega, \mu, \Lambda\}$, and $\mathbb{E}_{i \neq j}$ denotes an expectation w.r.t. the $i \neq j$ terms.

For a fully Bayesian treatment, we need to define prior distributions over the parameter set. The associated functional forms are chosen so that $p(\Omega)$ and $p(\mu, \Lambda)$ are conjugates of $p(Z|\Omega)$ and $p(X|Z, \mu, \Lambda)$.

$$p(\Omega) = \text{Dir}(\Omega | \alpha_0) = C(\alpha_0) \prod_k \omega_k^{\alpha_0 - 1} \quad (6)$$

$$p(\mu, \Lambda) = p(\mu | \Lambda) p(\Lambda) = \prod_k \mathcal{N}(\mu_k | m_0, (\beta_0 \Lambda_k)^{-1}) \mathcal{W}(\Lambda_k | W_0, \nu_0) \quad (7)$$

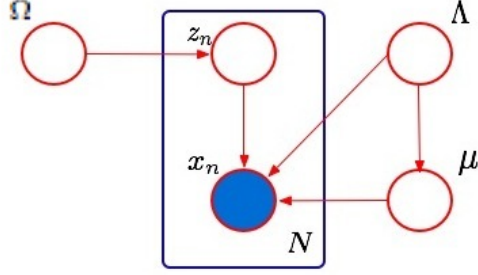


Figure 2: Graphical model associated with the variational Bayesian framework

where $C(\cdot)$ is a normalization constant, $\mathcal{W}(\cdot|\cdot, \cdot)$ is the Wishart distribution, and m_0 , β_0 , W_0 and ν_0 are hyper parameters.

As suggested by the graphical model (see figure 2), we have all the elements needed to define the joint distribution :

$$p(X, Z, \Omega, \mu, \Lambda) = p(\Omega)p(Z|\Omega)p(\mu, \Lambda)p(X|Z, \mu, \Lambda) \quad (8)$$

This joint distribution is used to apply formula (5) for each parameter and latent variable. We obtain inter dependent estimates. Alternative updates to these estimates implements a pseudo EM algorithm [1, 6]. We note that an appropriate choice for the Dirichlet parameters ($\alpha_0 < 1$) influences some components to play no role in the model. Such a component (let's call it k) is detected by controlling that α_k is significantly different from α_0 , and it can be eventually deleted from the final model. Doing so we keep only relevant components, automatically addressing the choice of a good number of components.

2.2 Iterative grouping of Gaussian mixtures

2.2.1 Building image set representatives

In the previous section, we described the feature set and procedure used to obtain a mixture model as a representation for each image. We now introduce a derivation of the classic variational procedure that takes a Gaussian component set as input.

Let us consider an arbitrary mixture defining L components, with parameters $\theta' = \{\Omega', \mu', \Lambda'\}$. We then assume that X and Z were i.i.d sampled from this distribution. It is therefore possible to regroup X according to the component from which its data was drawn. It leads us to the following formalism : $X = \{\hat{x}_1, \dots, \hat{x}_L\}$ with $\text{card}(X) = N$, $\hat{x}_l = \{x_i \mid z_{il} = 1\}$ and $\text{card}(\hat{x}_l) = \omega'_l N$. Let us express (2) and (3) w.r.t this formalism. To achieve tractability, we make the

following assumption : $\forall x_i \in \hat{x}_l, z_{ik} = \text{const} = z_{lk}$. Thus we can rewrite expression (3) as follows :

$$p(X | Z, \mu, \Lambda) = \prod_{k=1}^K \prod_{l=1}^L p(\hat{x}_l | Z, \mu_k, \Lambda_k)^{z_{lk}} \quad (9)$$

$$p(X | Z, \mu, \Lambda) = \prod_{k=1}^K \prod_{l=1}^L \left[\prod_{i=1}^{\omega'_l N} \mathcal{N}(x_{li} | \mu_k, \Lambda_k^{-1}) \right]^{z_{lk}} \quad (10)$$

$$\ln p(X | Z, \mu, \Lambda) = \sum_{k=1}^K \sum_{l=1}^L z_{lk} \left[\sum_{i=1}^{\omega'_l N} \ln \mathcal{N}(x_{li} | \mu_k, \Lambda_k^{-1}) \right] \quad (11)$$

For N sufficiently large, we can make the following approximation :

$$\sum_{i=1}^{\omega'_l N} \ln \mathcal{N}(x_{li} | \mu_k, \Lambda_k^{-1}) \simeq \omega'_l N \mathbb{E}_{\mu'_l, \Lambda'_l} [\ln \mathcal{N}(x | \mu_k, \Lambda_k^{-1})] \quad (12)$$

This statement is known as *virtual sampling*, and was introduced in [29, 28].

The expectation may be explicitated :

$$E_{\mu'_l, \Lambda'_l} [\ln \mathcal{N}(x | \mu_k, \Lambda_k^{-1})] = \int \mathcal{N}(x | \mu'_l, \Lambda'_l{}^{-1}) \ln \mathcal{N}(x | \mu_k, \Lambda_k^{-1}) dx \quad (13)$$

$$E_{\mu'_l, \Lambda'_l} [\ln \mathcal{N}(x | \mu_k, \Lambda_k^{-1})] = -KL \left(\mathcal{N}(x | \mu'_l, \Lambda'_l{}^{-1}) \parallel \mathcal{N}(x | \mu_k, \Lambda_k^{-1}) \right) - H(\mathcal{N}(x | \mu'_l, \Lambda'_l{}^{-1})) \quad (14)$$

with $KL(q_0 \parallel q_1)$ the KL divergence of q_1 from q_0 and $H(q_0)$ the entropy of q_0 . These two terms benefit from closed-form expressions [7]. Thus by reinjecting (14) into (12), and then (12) into (11), we obtain the convenient following expression for $p(X | Z, \mu, \Lambda)$:

$$\ln p(X | Z, \mu, \Lambda) = N \sum_{k=1}^K \sum_{l=1}^L z_{lk} \omega'_l \left[-KL \left(\mathcal{N}(x | \mu'_l, \Lambda'_l{}^{-1}) \parallel \mathcal{N}(x | \mu_k, \Lambda_k^{-1}) \right) - H(\mathcal{N}(x | \mu'_l, \Lambda'_l{}^{-1})) \right] \quad (15)$$

$$\ln p(X | Z, \mu, \Lambda) = N \sum_{k=1}^K \sum_{l=1}^L z_{lk} \omega'_l \left[\frac{1}{2} \ln \det \Lambda_k - \frac{1}{2} \text{Tr}(\Lambda_k \Lambda'_l{}^{-1}) - \frac{1}{2} (\mu'_l - \mu_k)^T \Lambda_k (\mu'_l - \mu_k) - \frac{d}{2} \ln(2\pi) \right] \quad (16)$$

Thus with our adaptation the likelihood term depends solely on θ' . The formalism change also has consequences on (2) : as we previously stated that $z_{lk} = z_{nk} \forall x_n \in \hat{x}_l$, we can write :

$$p(Z | \Omega) = \prod_{n=1}^N \prod_{k=1}^K \omega_k^{z_{nk}} = \prod_{l=1}^L \prod_{k=1}^K \omega_k^{N\omega'_l z_{lk}} \quad (17)$$

Variational update equations are partially based on moments evaluated w.r.t $p(Z)$ and $p(X)$ [6]. Therefore cascading consequences occur relatively to the classical VBEM algorithm.

Normalized estimates $\{r_{nk}\}$ for $q^*(Z)$ are obtained through the calculation of the unnormalized log estimates $\ln \rho_{nk}$. According to equation (17), $q^*(Z)$ will now be factorized over L . Combining the classic variational Bayesian formulation with (16) and (17) leads to this estimate :

$$\begin{aligned} \ln(\rho_{lk}) &= \frac{N\omega'_l}{2} (2\mathbb{E}[\ln \omega_k] + \mathbb{E}[\ln \det \Lambda_k] - d \ln(2\pi)) \\ &\quad - \frac{N\omega'_l}{2} \left(\mathbb{E}_{\mu_k, \Lambda_k} \left[\text{Tr}(\Lambda_k \Lambda'_l{}^{-1}) + (\mu'_l - \mu_k)^T \Lambda_k (\mu'_l - \mu_k) \right] \right) \end{aligned} \quad (18)$$

The moment w.r.t μ_k and Λ_k is easily evaluated to give $\frac{d}{\beta_k} + \nu_k \left[\text{Tr}(W_k \Lambda'_l{}^{-1}) + (\mu'_l - m_k)^T W_k (\mu'_l - m_k) \right]$. Applying formula (5) jointly with our adaptation also leads to modified update estimates for θ . These are given hereunder :

$$\alpha_k = \alpha_0 + \sum_l N\omega'_l r_{lk} \quad (19)$$

$$\beta_k = \beta_0 + \sum_l N\beta'_l r_{lk} \quad (20)$$

$$m_k = \frac{1}{\beta_k} \left(\beta_0 m_0 + \sum_l N\omega'_l r_{lk} \mu'_l \right) \quad (21)$$

$$W_k^{-1} = W_0^{-1} + \beta_0 m_0 m_0^T - \beta_k m_k m_k^T + \sum_l N\omega'_l r_{lk} (\mu'_l \mu'_l{}^T + \Lambda'_l{}^{-1}) \quad (22)$$

$$\nu_k = \nu_0 + \sum_l N\omega'_l r_{lk} \quad (23)$$

The classical variational algorithm is known to monotonically decrease the KL distance between the variational pdf and the true posterior [6]. This is equivalent to maximizing the lower bound of the complete likelihood. As we can compute this lower bound, and as this bound should never decrease, we can test for convergence by comparing two successive values of the bound. Only terms of the bound that depend on Z or X are impacted, resulting in the following changes :

$$\mathbb{E}[\ln p(X | Z, \mu, \Lambda)] = \frac{1}{2} \sum_k \sum_l N \omega'_l r_{lk} \quad (24)$$

$$\left[\ln \tilde{\Lambda}_k - \frac{d}{\beta_k} - \nu_k [\text{Tr}(W_k \Lambda'_l)^{-1}) + (\mu'_l - m_k)^T W_k (\mu'_l - m_k)] \right]$$

$$\mathbb{E}[\ln p(Z | \Omega)] = \sum_l \sum_k N \omega'_l r_{lk} \ln \tilde{\omega}_k \quad (25)$$

Now consider θ' was built by grouping several images, i.e. several Gaussian mixtures. We previously saw that the variational procedure minimizes the KL divergence loss of the posterior distribution w.r.t. the variational distribution. KL divergence is a commonly used dissimilarity measure for Gaussian mixtures, specifically for content-based image retrieval systems using mixtures as image representations [16, 18]. As the estimation takes model complexity into account, we obtain the closest posterior low-complexity model w.r.t. the overall input model. Therefore, the obtained model is roughly a centroid for the input images.

2.2.2 Iterative local minima search

Let the criterion to optimize be the sum of KL divergences of individuals w.r.t. centroids (further denoted as the distortion). We disclose the following iterative algorithm for local minima search relatively to this criterion :

-
- (1) Input : N individuals, number of centroids k
 - (2) Initialisation : choose k centroids from individuals
 - (3) **while** change in assignments **do**
 - (4) **Assign images to classes** :
 - (5) assignment(f_n) = $\arg \min_i \text{KL}_{\text{UT}}(f_n || g_i)$
 - (6) **Update class representative density** :
 - (7) compute "average" density with method introduced in section 2.2.1
 - (8) **endwhile**
-

Figure 3: Iterative algorithm for local minima search

We saw in the previous section how to represent images by Gaussian mixtures, and how to obtain representatives from a group of Gaussian mixtures. In other words we have an implementation for the update step. Image-to-class assignments rely on minimizing the density "distance" between the image and the current class representative, as evaluated by the

Kullback-Leibler divergence between these distributions. Since the densities to be compared both take the form of a Gaussian mixture model, let us discuss its practical computation. As no exact closed-form exists, we seek a trade-off between computation cost and accuracy. A general solution is to resort to Monte-Carlo sampling, but its high computation cost, especially in high-dimension spaces precludes its usage when efficiency is sought. A crude approximation proposed in [17] consists in matching the closest Gaussian components between the two mixtures and approximating the complete divergence as the sum of divergence within these pairs of Gaussians, as this may be an existing closed-form. A more accurate option, which we retain for our present scheme, is based on the unscented transform [16], denoted by KL_{UT} . We recall here the principles of this approximation. Usually, with random sampling, we approximate KL divergence as following :

$$\text{KL}(q||p) = \int q(x) \ln \frac{q(x)}{p(x)} dx \approx \sum_X \ln \frac{q(x_i)}{p(x_i)} \quad (26)$$

where $X = \{x_1, \dots, x_n\}$ is a sample generated by $q(x)$. The idea behind unscented transform is to choose a reduced set of informative points instead of a fully random sample.

Let us consider $q(x) = \mathcal{N}(x|\mu, \Sigma)$. We perform the eigen value decomposition of Σ , giving $\Sigma = UDU^T$, with the eigen vectors set $U = (U_1, \dots, U_d)$, and the eigen values diagonal matrix $D = \text{diag}(\lambda_1, \dots, \lambda_d)$.

A set of $2d + 1$ points is then chosen as following :

$$x_i = \mu + \sqrt{\lambda_i} U_i \quad i = 1, \dots, d \quad (27)$$

$$x_{d+i} = \mu - \sqrt{\lambda_i} U_i \quad i = 1, \dots, d \quad (28)$$

$$x_{2d+1} = \mu \quad (29)$$

This set of points can be used in formula 26. Now if $q(x)$ is a Gaussian mixture, we obtain an approximation by involving the mixture weights in the expression :

$$\text{KL}_{\text{UT}}(q||p) \approx \frac{1}{2d+1} \sum_{k=1}^K \omega_k \sum_{i=1}^{2d+1} \ln \frac{q(x_{i,k})}{p(x_{i,k})} \quad (30)$$

where $\{x_{i,k}\}$ is the set of points obtained from the k -th Gaussian component of q .

Evaluations showed it performs much better than the matching based approximation, at a reasonable computational cost. The convergence aspect must be handled carefully : in our scheme, the true distortion, that is defined over the true KL divergences, is guaranteed to decrease at each step. However, as we resort to an approximation, this might not be true in

exceptional cases. In other words, the true distortion might decrease while the approximate might not, hence we chose here assignments changes as convergence criterion.

In general, KL divergence is not symmetric, i.e. $KL(q||p) \neq KL(p||q)$. In information theory, $KL(q||p)$ measures the expected loss of coding with p a sample generated by q . In the variational Bayesian scheme, we seek a distribution q that is likely to have generated an observed sample. Practically, by minimizing $KL(q||p)$ we obtain a distribution able to generate samples that are very close to the observed sample. In our clustering algorithm, as suggested by the original k-means algorithm, we minimize a distortion measure. In other words, each individual f_n is assigned to a representative g_i so that the loss of coding f_n with g_i is as low as possible, i.e. we minimize $KL(f_n||g_i)$.

2.3 On-line data processing

Now let us consider the case of on-line data, i.e. a stream of incoming data. As, in this context, it is a priori impossible to store and process an entire data set, we need a strategy to discard stale data. The sliding window model is a classic solution for this purpose [3]. The basic idea here is to define a window size N , which will contain the N most recently received elements. Elements that are no longer included in this interval are discarded.

Since the computation of the pseudo-centroids is not based on averaging in a Euclidean space (see section 2.2.1), we can not use update formula like suggested in [8]. Instead, at each time a new data item arrives, we should recompute all centroids and assignments until convergence. To limit this cost, and still aim at finding a local minimum, we monitor the total distortion. When the variation of this distortion is decreasing (i.e. sign of second derivative is changing), we start processing the next item.

This gives rise to an incremental algorithm (figure 4).

3 An interactive visualization of clusters

Our clustering method can be used in an incremental way in order to cluster large image databases. But in this context, it can be quite difficult to interpret the stream of results given by the algorithm and to interact with it. We propose to provide to the user a visualization method that summed up what happened in the clustering process and let him interact with the clustering algorithm.

In this context we will only visualize clusters, and its most representative individuals (images in this case), with the relative distance between them in order to simplify the visualization.

-
- (1) Input : N individuals, number of centroids k
 - (2) Initialisation : k centroids = k first individuals
 - (3) **while** incoming data **do**
 - (4) Add item at first position in the window
 - (5) **if** window full **then**
 - (6) remove oldest item
 - (7) **endif**
 - (8) Iterate through loop of algorithm 3 until $\frac{d(\Delta\text{distortion})}{dt} > 0$
 - (9) **endwhile**
-

Figure 4: Online adaptation of algorithm 3

We then build a complete graph where nodes represent clusters and the weight of each edge is inversely proportional to the similarity between clusters linked by the edge.

The clustering process evolves over time, so we need a graphical representation that can show to the user what has changed at each iteration of the clustering algorithm. Several methods had been proposed in the literature to visualize time series. Many of them [5, 21, 20] provide a new visualization at each change and make a morphing of the new representation with the previous one. But in this case, there are several steps in the visualization process: 1) providing a graphical representation and 2) morphing this representation with the previous one. This leads to a discontinuity in the graphical representation of classes (we see a discrete representation of the time series).

With respect to these considerations, we have chosen to use a spring based algorithm [19, 15] to display our clusters. Our clustering algorithm produces a small set of clusters and a spring-based algorithm displays the graph with good quality results for graphs of medium size (up to 50-100 vertices). This algorithm family (based on force-directed placement) have become one of the most effective techniques for drawing undirected graphs. They provide a physical approximation that can be easily interpreted by a human. They produce a simple visualization where nodes are generally well distributed. They preserve graph symmetries.

We can find two families of spring based algorithm:

- The first model considers a spring-like force for every pair of nodes (i, j) where the ideal length d_{ij} of each spring is proportional to the theoretic distance between nodes i and j .

Then we solve the optimization problem by minimizing the difference between euclidean and ideal distances between nodes.

- The second model considers each edge as a spring and each node as an electrically charged particle. The entire graph is then simulated as if it were a physical system. At each iteration we compute the sum of forces in our system by using the Hooke's law for the edges and the Coulomb's law for the nodes. The first force tries to maintain the required distance for each edge while the Coulomb force tries to move nodes as far as possible from all other nodes.

We will use this second model for our display algorithm because it provides a continuous display of the evolution of a population of nodes in a graph depending on simulated physical principles. These principles are natural for human perception and relatively easy to predict and understand.

3.1 Spring-based algorithm and visualization improvement

In order to let the user interpret how clusters are built, we have chosen to use an incremental 3D version of the spring-based algorithm. This kind of algorithm has already been used to display interactively graphs [22]. It lets the user see naturally the evolution of the drawing of the graph. By drawing the intermediate stages of the graph step by step, the user can follow how the graph evolves and interact with it. The interaction permits to avoid local minima by moving nodes to any position in the space and so initializing the algorithm to another state.

At each iteration of the visualization algorithm, the Coulomb repulsion (see eq. 31 where q_1 and q_2 represents the electric charge of two nodes (n_1 and n_2) and \vec{r} the vector between n_1 and n_2) and the Hooke attraction (see eq. 32 where \vec{L} is a vector between the two nodes linked by an edge and R the equilibrium length of the spring) of the net are computed and the position of each node is updated (see figure 5). Those formulas allow us to update in parallel the characteristics of the graph and therefore, to take into account the changes in the clustering produced by our algorithm. Hence a smooth animation can be presented to the user. Finally we display a subset of images that are close to the cluster center for each node.

$$\begin{aligned}
 E_0 &= 8.85 \cdot 10^{-12} * C^2 / (N * m^2) \\
 \vec{F} &= \frac{q_1 \cdot q_2 \cdot \vec{r}}{4\pi E_0 |\vec{r}|^3}
 \end{aligned}
 \tag{31}$$

$$\vec{F} = -kHook \left(|\vec{L}| - R \right) \frac{\vec{L}}{|\vec{L}|} \quad (32)$$

```

(1) foreach Node  $n_i \in NodeList$  do
(2)    $netForce \leftarrow 0$ 
(3)   foreach Node  $n_j \in NodeList \setminus n_i$  do
(4)      $netForce \leftarrow netForce + CoulombRepulsion(n_i, n_j)$ 
(5)   endfor
(6)   foreach Node  $n_j \in NodeLinkTo(n_i)$  do
(7)      $netForce \leftarrow netForce + HookeAttraction(n_i, n_j)$ 
(8)   endfor
(9)    $V_{n_i}^t \leftarrow (V_{n_i}^{t-1} + TimeStep \times netForce) \times Damping$ 
(10)   $P_{n_i}^t \leftarrow P_{n_i}^{t-1} + V_{n_i}^t \times TimeStep$ 
(11) endfor

```

Figure 5: Incremental spring-based algorithm. $V_{n_i}^t$ represents the velocity of the n_i node at time t ; $P_{n_i}^t$ represents the position of the n_i node at time t ; $TimeStep$ represents the time elapsed between 2 iterations of the algorithm ; and $Damping$ represents a constant set to 0.2 experimentally.

The figure 6 shows a static representation of the visualization part of our application. We can notice that the images that represent clusters are always displayed face to the user camera (3D billboard).

Several enhancement have been done on the visualization process. In order to visualize all the graph representation of our clustering, we center at each visualization iteration the gravity center of the graph into the user screen. But when the gravity center of the graph suddenly changes its position, for example when the number of nodes changes, refocusing the graph induces a jump in the display. To avoid this, we progressively move the gravity center of the graph to the center of the screen. We limit the acceleration that occurs on the graph to a maximum value and gradually reduce the speed according to a constant deceleration when the graph arrives near the center.

Displaying a complete graph can be difficult and reduce the legibility of the visualization. And displaying a link between two dissimilar clusters may not be useful. The threshold limit

to display an edge between two clusters can be very difficult to determine. Then, we let the user set this threshold by using a slider bar (see in figure 6 in the bottom of the window). The system restricts the lower threshold in order to keep a connected graph (compulsory condition of the spring-based algorithm).

Finally, overlap can occur in 3D and it may be difficult to perceive the depth in a 3D visualization. To avoid this problem we have implemented in our application a stereoscopic visualization that works in different modes (anaglyph (see figure 6 *b*)) and stereoscopy with polarized glasses).

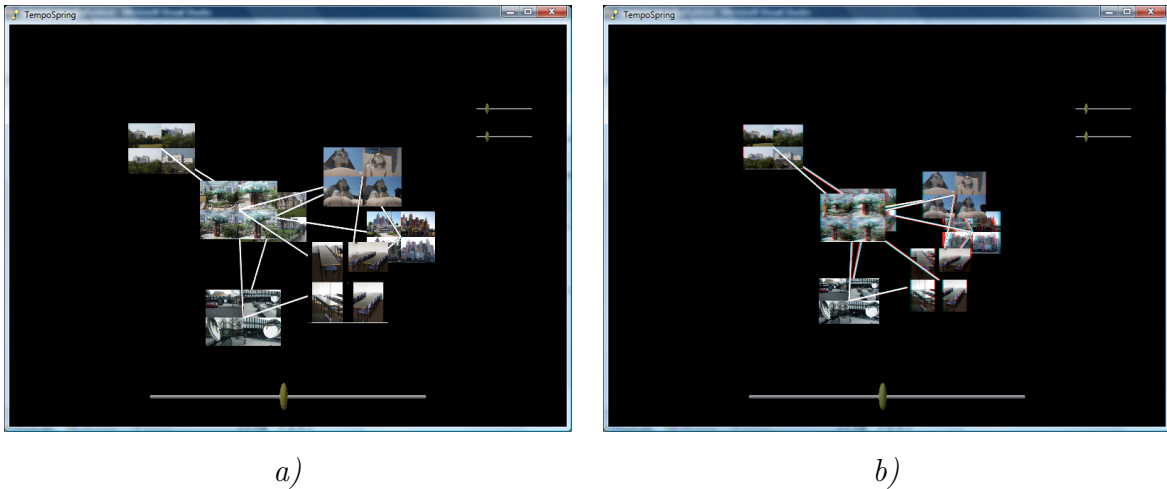


Figure 6: Screen capture of the application in *a*) normal view and *b*) anaglyph view. The two sliders bar on the right of the screen permit respectively to manage the power of the Coulomb force and of the Hooke force. The slider bar at the bottom of the screen permits the user to reduce the number of edges of the graph.

3.2 Interactivity and feedback on the clustering algorithm

The final result of the spring based algorithm can be strongly influenced by the initial layout. In some cases (in particular with a lot of edges) it can lead to low-quality drawing. But an advantage of this class of algorithm is the interactive aspect. In our application, the user can interact in the visualization by rotating the graph representation or selecting any cluster in order to show some details like for example the number of item in the cluster. Then, the selected item is highlighted and we allow the user to move it in 3D space to show how it interacts with others clusters or to found a new visualization. To simplify the movements in a 3D space with a simple and classical mouse, we limit them on plans $\{x, y\}$, $\{x, z\}$ or $\{y, z\}$ (see figure 7).

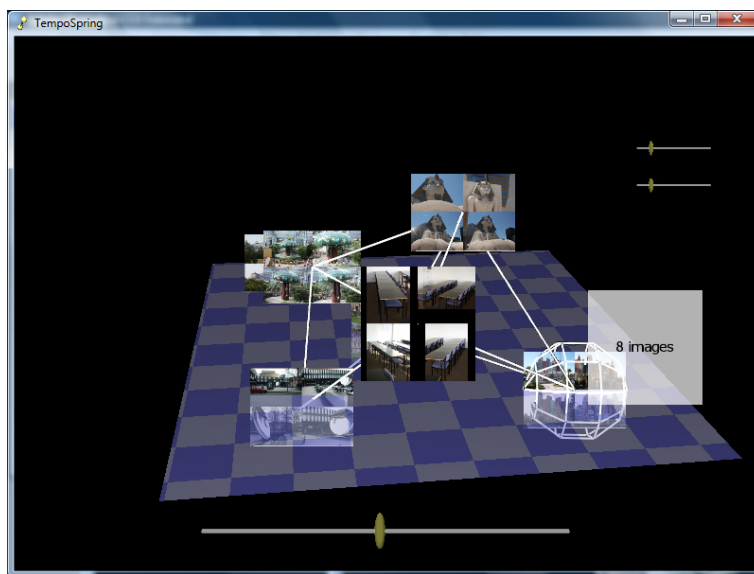


Figure 7: Selection and interactivity in the visualization application.

In order to limit the number of images in the graph, we display on each cluster node only thumbnails of the 4 most representative individuals. But by selecting a particular cluster, the user can display all individuals in the group (see figure 8).

If a clustering error has been made by the algorithm, the user can then send a feedback to the system in three ways:

- by dragging an image outside of the group g_i . At the next iteration, the clustering algorithm will compute a new centroid for g_i without this element. A new cluster is created with the dragged element as its center.
- by dragging an image into an other cluster g_j . At the next iteration, the clustering algorithm will compute a new centroid for g_j with this element. The centroid for g_i without the dragged element is also computed.
- by destroying a cluster g_i . At the next iteration, members of the former cluster become assigned to the closest cluster center.

4 Experiments

In this section we will focus on evaluating the batch version of our algorithm (see paragraph 2.2.2). The purpose of this restriction is to observe the behavior of our method on a real image data set, independently from effects that might be induced by on-line and interactive

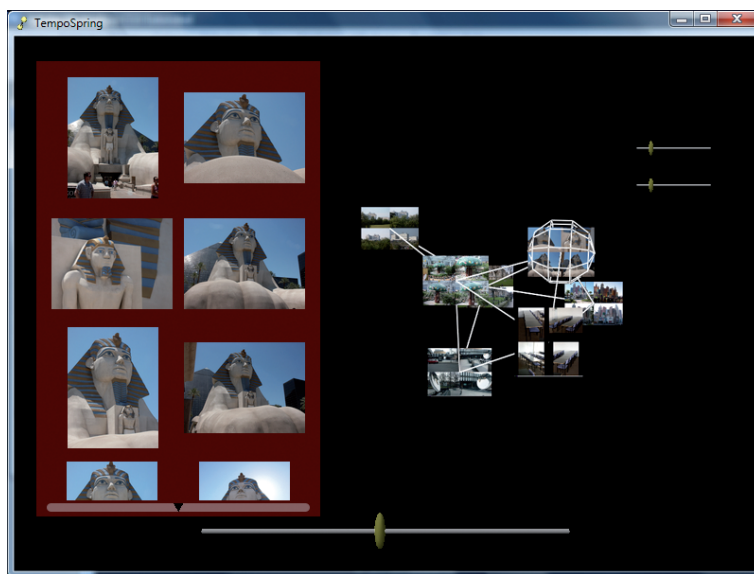


Figure 8: Interactive feedback on the clustering algorithm.

aspects. For our experiments, we used a subset of the UCID image data set [26]. This data set consists of real photos (buildings, people, objects). All images have the same size. Each photo is associated with a *ground truth*, i.e. a specific object or person being photographed. For testing purposes, we selected 134 images from this data set, associated with 11 different ground truths. To keep a relative computational simplicity, we used downsampled versions of these images (128x96 pixels).

We will evaluate the ability of the algorithm to recover the 11 true groups or, at least, some other pertinent groups. We initialize the algorithm with 11 randomly chosen individuals as centroids. We present results averaged over 20 runs. In figure 9, we monitored the evolution of the total distortion (sum of KL divergences of individuals w.r.t centroids). As a reference, we included a dashed line indicating the observed distortion in the true groups. In figure 10, we indicated the evolution of the observed number of groups. Indeed, even if we initialize the algorithm with 11 groups, in the current (non-Euclidian) setting, it is possible for a centroid to become unaffected by any individual. In such a case the corresponding group disappears. For figure 11, we monitored the dispersion of centroids, using an average pairwise metric from the set of centroids. As the role played by each centroid in a couple from this set is symmetric, we used a symmetrized KL divergence $\frac{1}{2}(\text{KL}(p \parallel q) + \text{KL}(q \parallel p))$. Finally, we also measured the couple error [2] of our induced classification w.r.t. the true groups. This error increases when two items belonging to the same true groups are clustered in two different groups, and conversely.

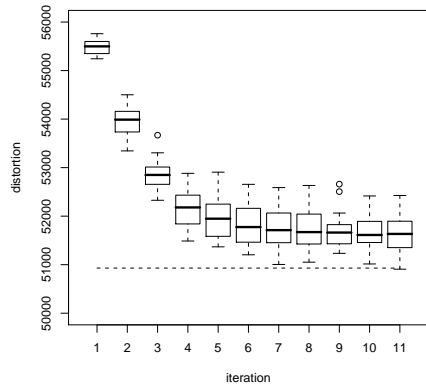


Figure 9: Evolution of the total distortion during first 10 iterations (20 runs). The dashed line indicates the level of distortion in the true groups. This is a Box-and-Whisker plot representation. The center bold lines is the observed median value, the other box lines represent quantiles. Outliers are notified with circles.

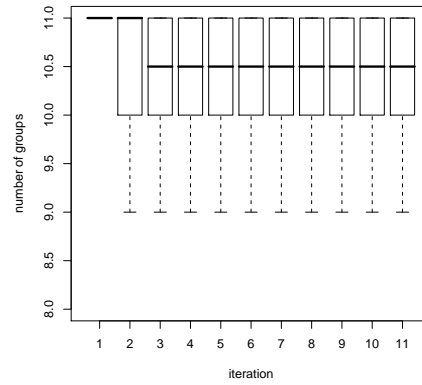


Figure 10: Evolution of the number of groups during first 10 iterations (20 runs).

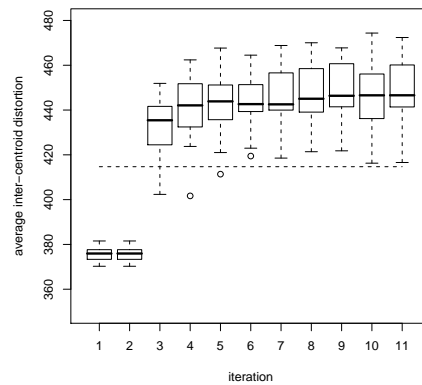


Figure 11: Evolution of the average centroids pairwise distances during first 10 iterations (20 runs). The dashed line indicates this average for the suggested natural groups centroids. Zeroes indicate cases with only 1 cluster remaining.

We can draw the following conclusions :

- Over 20 runs, we obtain an average of 11% for the couple error w.r.t. the true groups.
- As expected, figure 9 suggests that the distortion quickly decreases, and after 5 iterations converges slowly toward a limit value. In the current experimental setting (batch, random initializations), the distortion level observed in the true groups could seldom be obtained. Therefore, with additional information provided by the user interaction, the estimation process might almost recover the true groups.
- Marginally, we observe groups that disappear during process (figure 10). But we remain very close from the suggested initialization.
- The average inter-centroid distortion increases very quickly during estimation (figure 11). This produces well-separated classes. We notice that centroids suggested by natural groups are less separated than estimated ones. This might be compensated by our interactive scheme : allowing users to manually recombine groups during incremental process would help avoiding such local minima (as a better one is suggested on figure 9).

5 Conclusion and perspectives

This paper discloses a scheme for organizing and visualizing a set of images, with a view to image browsing. The first step of the proposal is formulated as statistical clustering of Gaussian mixture models, that can evolve in time as images flow in. The classes and their similarities obtained thereby form a time-varying graph, enabling the use of an efficient graph display technique. While the visualization technique is interactive, it also allows feedback on the classification process. The unsupervised mixture grouping technique can accommodate this feedback in a natural way, as it formulates clustering as an iterative algorithm performing local optimization, operating on mixtures rather than vectors. Besides user feedback, changes in the graph may also result from change in time of the clustering.

The work may be extended as follows. Besides the existing "create/destroy class", the adaptation of semi-supervised classification to grouping mixtures deserves thorough analysis, to integrate constraints such as "(don't) group together" to mixtures. In particular, the way time-evolving clustering should handle user-defined associations or dissociations is complex matter, as these user preferences may be outdated. Second, formalizing the mixture grouping in a Bayesian framework would allow better handling of the number of groups. Finally, in

its present state, the paper does not describe user satisfaction in using the proposed system, beyond the objective classification measurements that may be conducted for image grouping.

References

- [1] H. Attias. A variational Bayesian framework for graphical models. *Advances in Neural Information Processing Systems - MIT Press*, 12, 2000.
- [2] H. Azzag. *Classification hiérarchique par des fourmis artificielles : application à la fouille de données et de textes pour le Web*. PhD thesis, Ecole Doctorale Santé Sciences et Technologies, Université François Rabelais Tours, 2005.
- [3] B. Babcock, M. Datar, R. Motwani, and L. O’Callaghan. Maintaining variance and k-medians over data stream windows. *Proceedings of the 22nd ACM SIGMOD-SIGACT symposium on Principles of database systems*, pages 234–243, 2003.
- [4] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 59–68, 2004.
- [5] Skye Bender-deMoll and Daniel A. McFarland. The art and science of dynamic network visualization. *Journal of Social Structure*, 7(2), 2006.
- [6] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.
- [7] R. E. Blahut. *Principles and Practice of Information Theory*. Addison-Wesley, 1987.
- [8] L. Bottou and Y. Bengio. Convergence properties of the k-means algorithm. *Advances in Neural Information Processing Systems - MIT Press*, 7, 1995.
- [9] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Learning*, 24(8):1026–1038, 2002.
- [10] O. Chapelle, B. Scholkopf, and A. Zien. *Semi-Supervised Learning*. MIT Press, 2006.
- [11] Y. Chen, J. Z. Wang, and R. Krovetz. Content-based image retrieval by clustering. *Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval*, POSTER SESSION:193–200, 2003.

- [12] C. Constantinopoulos and M. K. Titsias. Bayesian feature and model selection for gaussian mixture models. *IEEE Trans. on Pattern Analysis Machine Intelligence*, 28(6):1013–1018, 2006.
- [13] R. Datta, J. Li, and J. Z. Wang. Content-based image retrieval: approaches and trends of the new age. *Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*, pages 253–262, 2005.
- [14] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society - B Series*, B(39):1–38, 1977.
- [15] Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Softw. Pract. Exper.*, 21(11):1129–1164, 1991.
- [16] J. Goldberger, S. Gordon, and H. Greenspan. An efficient image similarity measure based on approximations of KL-divergence between two Gaussian mixtures. *Proceedings of the 9th IEEE International Conference on Computer Vision*, 1:487–493, 2003.
- [17] J. Goldberger and S. Roweis. Hierarchical clustering of a mixture model. *Advances in Neural Information Processing Systems - MIT Press*, 17, 2004.
- [18] H. Greenspan, J. Goldberger, and L. Ridel. A continuous probabilistic framework for image matching. *Computer Vision and Image Understanding*, 84(3):384–406, 2001.
- [19] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inf. Process. Lett.*, 31(1):7–15, 1989.
- [20] Hyunmo Kang, Lise Getoor, and Lisa Singh. Visual analysis of dynamic group membership in temporal social networks. *SIGKDD Explor. Newsl.*, 9(2):13–21, 2007.
- [21] Eloïse Loubier, Wahiba Bahsoun, and Bernard Dousset. Visualization and analysis of large graphs. In *PIKM '07: Proceedings of the ACM first Ph.D. workshop in CIKM*, pages 41–48, New York, NY, USA, 2007. ACM.
- [22] D. Scott McCrickard and Colleen M. Kehoe. Visualizing search results using sqwid. pages 51–60. ACM Press, 1997.
- [23] J. McQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

- [24] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. *Proceedings of Eighth International Conference on Computer Vision*, 1:525, 2001.
- [25] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39:103–134, 2000.
- [26] G. Schaefer and M. Stich. UCID - an Uncompressed Colour Image Database. *Proceedings SPIE, Storage and Retrieval Methods and Applications to Multimedia*, pages 472–480, 2004.
- [27] A. Vailaya, M. A. T. Figueiredo, A. K. Jain, and H. J. Zhang. Image classification for content-based indexing. *IEEE transactions on image processing*, 10(1):117–130, 2001.
- [28] N. Vasconcelos. Image indexing with mixture hierarchies. *Proceedings of IEEE Conference in Computer Vision and Pattern Recognition*, 1:3–10, 2001.
- [29] N. Vasconcelos and A. Lippman. Learning mixture hierarchies. *Advances in Neural Information Processing Systems - MIT Press Neural Information Processing Systems*, II:606–612, 1998.
- [30] G. Wyszecki and W. Stiles. *Color Science : Concepts and methods, Quantitative Data and Formulae*. John Wiley and Sons, 1982.