Pattern Anal Applic (2012) 15:345–366
DOI 10.1007/s10044-011-0261-7

SURVEY

An experimental comparison of clustering methods for content-based indexing of large image databases

Hien Phuong Lai · Muriel Visani · Alain Boucher · Jean-Marc Ogier

Received: 4 January 2011 / Accepted: 27 December 2011 / Published online: 13 January 2012

Abstract In recent years, the expansion of acquisition devices such as digital cameras, the development of storage and transmission techniques of multimedia documents and the development of tablet computers facilitate the development of many large image databases as well as the interactions with the users. This increases the need for efficient and robust methods for finding information in these huge masses of data, including feature extraction methods and feature space structuring methods. The feature extraction methods aim to extract, for each image, one or more visual signatures representing the content of this image. The feature space structuring methods organize indexed images in order to facilitate, accelerate and improve the results of further retrieval. Clustering is one kind of feature space structuring methods. There are different types of clustering such as hierarchical clustering, density-based clustering, grid-based clustering, etc. In an interactive context where the user may modify the automatic clustering results, incrementality and hierarchical structuring are properties growing in interest for the clustering algorithms. In this article, we propose an experimental comparison of different clustering methods for structuring large image databases, using a rigorous experimental protocol. We use different image databases of increasing sizes (Wang, PascalVoc2006, Caltech101, Corel30k) to study the scalability of the different approaches.

Keywords Image indexing · Feature space structuring · Clustering · Large image database · Content-based image retrieval · Unsupervised classification

H. P. Lai (✉) · M. Visani · J.-M. Ogier
L3I, Université de La Rochelle,
17042 La Rochelle cedex 1, France
e-mail: lhienphuong@gmail.com; hien_phuong.lai@univ-lr.fr

M. Visani
e-mail: muriel.visani@univ-lr.fr

J.-M. Ogier
e-mail: jean-marc.ogier@univ-lr.fr

H. P. Lai · A. Boucher
IFI, MSI team, IRD, UMI 209 UMMISCO,
Vietnam National University, 42 Ta Quang Buu,
Hanoi, Vietnam
e-mail: alain.boucher@auf.org

1 Originality and contribution

In this paper, we present an overview of different clustering methods. Good surveys and comparisons of clustering techniques have been proposed in the literature a few years ago [3–12]. However, some aspects have not been studied yet, as detailed in the next section. The first contribution of this paper lies in analyzing the respective advantages and drawbacks of different clustering algorithms in a context of huge masses of data where incrementality and hierarchical structuring are needed. The second contribution is an experimental comparison of some clustering methods (global k-means, AHC, R-tree, SR-tree and BIRCH) with different real image databases of increasing sizes (Wang, PascalVoc2006, Caltech101, Corel30k) to study the scalability of these approaches when the size of the database is increasing. Different feature descriptors of different sizes are used in order to evaluate these approaches in the context of high-dimensional data. The clustering results are evaluated by both internal (unsupervised) measures and external (supervised) measures, the latter being closer to the users' semantic.

2 Introduction

With the development of many large image databases, the traditional content-based image retrieval in which the feature vector of the query image is exhaustively compared to that of all other images in the database for finding the nearest images is not compatible. Feature space structuring methods (clustering, classification) are necessary for organizing indexed images to facilitate and accelerate further retrieval.

Clustering, or unsupervised classification, is one of the most important unsupervised learning problems. It aims to split a collection of unlabelled data into groups (clusters) so that similar objects belong to the same group and dissimilar objects are in different groups. In general, clustering is applied on a set of feature vectors (signatures) extracted from the images in the database. Because these feature vectors only capture low level information such as color, shape or texture of an image or of a part of an image (see Sect. 3), there is a semantic gap between high-level semantic concepts expressed by the user and these low-level features. The clustering results are therefore generally different from the intent of the user. Our work in the future aims to involve the user into the clustering phase so that the user could interact with the system in order to improve the clustering results (the user may split or group some clusters, add new images, etc.). With this aim, we are looking for clustering methods which can be incrementally built in order to facilitate the insertion, the deletion of images. The clustering methods should also produce hierarchical cluster structure where the initial clusters may be easily merged or split. It can be noted that the incrementality is also very important in the context of very large image databases, when the whole data set cannot be stored in the main memory. Another very important point is the computational complexity of the clustering algorithm, especially in an interactive context where the user is involved.

Clustering methods may be divided into two types: hard clustering and fuzzy clustering methods. With hard clustering methods, each object is assigned to only one cluster while with fuzzy methods, an object can belong to one or more clusters. Different types of hard clustering methods have been proposed in the literature such as hierarchical clustering (AGNES [37], DIANA [37], BIRCH [45], AHC [42], etc.), partition-based clustering (k-means [33], *k*-medoids [36], PAM [37], etc.), density-based clustering (DBSCAN [57], DENCLUE [58], OPTICS [59], etc.), grid-based clustering (STING [53], WaveCluster [54], CLICK [55], etc.) and neural network based clustering (SOM [60]). Other kinds of clustering approaches have been presented in the literature such as the genetic algorithm [1] or the affinity propagation [2] which exchange real-valued messages between data points until having a high-quality set of exemplars and corresponding clusters. More details on the basic approaches will be given in Sect. 4. Fuzzy clustering methods will be studied in further works.

A few comparisons of clustering methods [3–10] have been proposed so far with different kinds of databases. Steinbach et al. [3] compared agglomerative hierarchical clustering and k-means for document clustering. In [4], Thalamuthu et al. analyzed some clustering methods with simulated and real gene expression data. Some clustering methods for word images are compared in [5]. In [7], Wang and Garibaldi compared hard (k-means) and fuzzy (fuzzy C-means) clustering methods. Some model-based clustering methods are analyzed in [9]. These papers compared different clustering methods using different kinds of data sets (simulated or real), most of these data sets have a low number of attributes or a low number of samples. More general surveys of clustering techniques have been proposed in the literature [11, 12]. Jain et al. [11] presented an overview of different clustering methods and give some important applications of clustering algorithms such as image segmentation, object recognition, but they did not present any experimental comparison of these methods. A well-researched survey of clustering methods is presented in [12], including analysis of different clustering methods and some experimental results not specific to image analysis. In this paper, we present a more complete overview of different clustering methods and analyze their respective advantages and drawbacks in a context of huge masses of data where incrementality and hierarchical structuring are needed. After presenting different clustering methods, we experimentally compare five of these methods (global k-means, AHC, R-tree, SR-tree and BIRCH) with different real image databases of increasing sizes (Wang, Pascal-Voc2006, Caltech101, Corel30k) (the number of images is from 1,000 to 30,000) to study the scalability of different approaches when the size of the database is increasing. Moreover, we test different feature vectors which size (per image) varies from 50 to 500 in order to evaluate these approaches in the context of high-dimensional data. The clustering results are evaluated by both internal (unsupervised) measures and external (supervised and therefore semantic) measures.

The most commonly used Euclidean distance is referred by default in this paper for evaluating the distance or the dissimilarity between two points in the feature space (unless another dissimilarity measure is specified).

This paper is structured as follows. Section 3 presents an overview of feature extraction approaches. Different clustering methods are described in Sect. 4. Results of different clustering methods on different image databases of increasing sizes are analyzed in Sect. 5. Section 6 presents some conclusions and further work.

3 A short review of feature extraction approaches

There are three main types of feature extraction approaches: global approach, local approach and spatial approach.

- With regards to the global approaches, each image is characterized by a signature calculated on the entire image. The construction of the signature is generally based on color, texture and/or shape. We can describe the color of an image, among other descriptors [13], by a color histogram [14] or by different color moments [15]. The texture can be characterized by different types of descriptors such as co-occurrence matrix [16], Gabor filters [17, 18], etc. There are various descriptors representing the shape of an image such as Hu's moments [19], Zernike's moments [20, 21], Fourier descriptors [22], etc. These three kinds of features can be either calculated separately or combined for having a more complete signature.
- Instead of calculating a signature on the entire image, local approaches detect interest points in an image and analyze the local properties of the image region around these points. Thus, each image is characterized by a set of local signatures (one signature for each interest point). There are some different detectors for identifying the interest points of an image such as the Harris detector [23], the difference of Gaussian [24], the Laplacian of Gaussian [25], the Harris–Laplace detector [26], etc. For representing the local characteristics of the image around these interest points, there are various descriptors such as the local color histogram [14], Scale-Invariant Feature Transform (SIFT) [24], Speeded Up Robust Features (SURF) [27], color SIFT descriptors [14, 28–30], etc. Among these descriptors, SIFT descriptors are very popular because of their very good performance.
- Regarding to the spatial approach, each image is considered as a set of visual objects. Spatial relationships between these objects will be captured and characterized by a graph of spatial relations, in which nodes often represent regions and edges represent spatial relations. The signature of an image contains descriptions of visual objects and spatial relationships between them. This kind of approach relies on a preliminary stage of objects recognition which is not straightforward, specially in the context of huge image databases where the contents may be very heterogeneous. Furthermore, the sensitivity of regions segmentation methods generally leads to use inexact graph matching techniques, which correspond to a N–P complete problem.

In content-based image retrieval, it is necessary to measure the dissimilarity between images. With regards to the global approaches, the dissimilarity can be easily calculated because each image is represented by a n-dimensional feature vector (where the dimensionality n is fixed). In the case of the local approaches, each image is represented by a set of local descriptors. And, as the number of interest points may vary from one image to another, the sizes of the feature vectors of different images may differ and some adapted strategies are generally used to tackle the variability of the feature vectors. In that case, among all other methods, we present hereafter two among the most widely used and very different methods for calculating the distance between two images:

- In the first method, the distance between two images is calculated based on the number of *matches* between them [31]. For each interest point P of the query image, we consider, among all the interest points of the image database, the two points $P1$ and $P2$ which are the closest to P ($P1$ being closer than $P2$). A *match* between P and $P1$ is accepted if $D(P, P1) \leq distRatio * D(P, P2)$, where D is the distance between two points (computed using their n-dimensional feature vectors) and *distRatio* is a fixed threshold, $distRatio \in (0,1)$. Note that for two images A_i and A_j, the matching of A_i against A_j (further denoted as (A_i, A_j)) does not produce the same *matches* as the matching of A_j against A_i (denoted as (A_j, A_i).) The distance between two images A_i and A_j is computed using the following formula:

$$D_{i,j} = D_{j,i} = 100 * \left(1 - \frac{M_{ij}}{\min\{K_{A_i}, K_{A_j}\}}\right) \quad (1)$$

  where K_{A_i}, K_{A_j} are respectively the numbers of interest points found in A_i, A_j and M_{ij} is the maximum number of matches found between the pairs (A_i, A_j) and (A_j, A_i).
- The second approach is based on the use of the "bags of words" method [32] which calculates, from a set of local descriptors, a global feature vector for each image. Firstly, we extract local descriptors of all the images in the database and perform clustering on these local descriptors to create a dictionary in which each cluster center is considered as a visual word. Then, each local descriptor of every image in the database is encoded by its closest visual word in the dictionary. Finally, each image in the database is characterized by a histogram vector representing the frequency of occurrence of all the words of the dictionary, or alternatively by a vector calculated by the tf-idf weighting method. Thus, each image is characterized by a feature vector of size n (where n is the number of words in the dictionary, i.e. the number of clusters of local descriptors) and the distance between any two images can be easily calculated using the Euclidean distance or the χ^2 distance.

In summary, the global approaches represent the whole image by a feature descriptor, these methods are limited by the loss of topological information. The spatial approaches represent the spatial relationships between visual objects in the image, they are limited by the stability of the region segmentation algorithms. The local approaches represent each image by a set of local feature descriptor, they are also limited by the the loss of spatial information, but they give a good trade-off.

4 Clustering methods

There are currently many clustering methods that allow us to aggregate data into groups based on the proximity between points (vectors) in the feature space. This section presents an overview of hard clustering methods where each point belongs to one cluster. Fuzzy clustering methods will be studied in further work. Because of our applicative context which involves interactivity with the user (see Sect. 2), we analyze the application capability of these methods in the incremental context. In this section, we use the following notations:

- $X = x_i | i = 1, \ldots, N$: the set of vectors for clustering
- N: the number of vectors
- $K = K_j | j = 1, \ldots, k$: the set of clusters

Clustering methods are divided into several types:

- Partitioning methods partition the data set based on the proximities of the images in the feature space. The points which are close are clustered in the same group.
- Hierarchical methods organize the points in a hierarchical structure of clusters.
- Density-based methods aim to partition a set of points based on their local densities.
- Grid-based methods partition a priori the space into cells without considering the distribution of the data and then group neighboring cells to create clusters.
- Neural network based methods aim to group similar objects by the network and represent them by a single unit (neuron).

4.1 Partitioning methods

Methods based on data partitioning are intended to partition the data set into k clusters, where k is usually predefined. These methods give in general a "flat" organization of clusters (no hierarchical structure). Some methods of this type are: k-means [33], k-medoids [36], PAM [37], CLARA [37], CLARANS [38], ISODATA [40], etc.

K-means [33] K-means is an iterative method that partitions the data set into k clusters so that each point belongs to the cluster with the nearest mean. The idea is to minimize the within-cluster sum of squares:

$$I = \sum_{j=1}^{k} \sum_{x_i \in K_j} D(x_i, \mu_j) \tag{2}$$

where μ_j is the gravity center of the cluster K_j.

The k-means algorithm has the following steps:

1. Select k initial clusters.
2. Calculate the means of these clusters.
3. Assign each vector to the cluster with the nearest mean.
4. Return to step 2 if the new partition is different from the previous one, otherwise, stop.

K-means is very simple to implement. It works well for compact and hyperspherical clusters and it does not depend on the processing order of the data. Moreover, it has relatively low time complexity of $O(Nkl)$ (note that it does not include the complexity of the distance) and space complexity of $O(N + k)$, where l is the number of iterations and N is the number of feature vectors used for clustering. In fact, l and k are usually much small compared to N, so k-means can be considered as linear to the number of elements. K-means is therefore effective for the large databases. On the other side, k-means is very sensitive to the initial partition, it can converge to a local minimum, it is very sensitive to the outliers and it requires to predefine the number of clusters k. K-means is not suitable to the incremental context.

There are several variants of k-means such as k-harmonic means [34], global k-means [35], etc. Global k-means is an iterative approach where a new cluster is added at each iteration. In other words, to partition the data into k clusters, we realize the k-means successively with $k = 1, 2, \ldots, k-1$. In step k, we set the k initial means of clusters as follows:

- $k-1$ means returned by the k-means algorithm in step $k-1$ are considered as the first $k-1$ initial means in step k.
- The point x_n of the database is chosen as the last initial mean if it maximizes b_n:

$$b_n = \sum_{j=1}^{N} (d_{k-1}^{j} - ||x_n - x_j||^2, 0) \tag{3}$$

where d_{k-1}^{j} is the squared distance between x_j and the nearest mean among the $k-1$ means found in the previous iteration. Thus, b_n measures the possible reduction of the error obtained by inserting a new mean at the position x_n.

The global k-means is not sensitive to initial conditions, it is more efficient than k-means, but its computational complexity is higher. The number of clusters k may not be

determined a priori by the user, it could be selected automatically by stopping the algorithm at the value of k having acceptable results following some internal measures (see Sect. 5.1.)

k-medoids [36] The k-medoids method is similar to the k-means method, but instead of using means as representatives of clusters, the k-medoids uses well-chosen data points (usually referred as to medoids[1] or exemplars) to avoid excessive sensitivity towards noise. This method and other methods using medoids are expensive because the calculation phase of medoids has a quadratic complexity. Thus, it is not compatible to the context of large image databases. The current variants of the k-medoids method are not suitable to the incremental context because when new points are added to the system, we have to compute all of the k medoids again.

Partitioning Around Medoids (PAM) [37] is the most common realisation of k-medoids clustering. Starting with an initial set of medoids, we iteratively replace one medoid by a non-medoid point if that operation decreases the *overall distance* (the sum of distances between each point in the database and the medoid of the cluster it belongs to). PAM therefore contains the following steps:

1. Randomly select k points as k initial medoids.
2. Associate each vector to its nearest medoid.
3. For each pair $\{m, o\}$ (m is a medoid, o is a point that is not a medoid):
   - Exchange the role of m and o and calculate the new overall distance when m is a non-medoid and o is a medoid.
   - If the new overall distance is smaller than the overall distance before changing the role of m and o, we keep the new configuration.
4. Repeat step 3 until there is no more change in the medoids.

Because of its high complexity $O(k(n-k)^2)$, PAM is not suitable to the context of large image databases. Like every variant of the k-medoids algorithm, PAM is not compatible with the incremental context either.

CLARA [37] The idea of Clustering LARge Applications (CLARA) is to apply PAM with only a portion of the data set ($40 + 2k$ objects) which is chosen randomly to avoid the high complexity of PAM, the other points which are not in this portion will be assigned to the cluster with the closest medoid. The idea is that, when the portion of the data set is chosen randomly, the medoids of this portion would approximate the medoids of the entire data set. PAM is applied several times (usually five times), each time with a different part of the data set, to avoid the dependence of the algorithm on the selected part. The partition with the lowest average distance (between the points in the database and the corresponding medoids) is chosen.

Due to its lower complexity of $O(k(40+k)^2 + k(N-k))$, CLARA is more suitable than PAM in the context of large image databases, but its result is dependent on the selected partition and it may converge to a local minimum. It is more suitable to the incremental context because when there are new points added to the system, we could directly assign them to the cluster with the closest medoid.

CLARANS [38] Clustering Large Application based upon RANdomize Search (CLARANS) is based on the use of a graph $G_{N,k}$ in which each node represents a set of k candidate medoids $(O_{M1}, \ldots, O_{Mk})$. All nodes of the graph represent the set of all possible choices of k points in the database as k medoids. Each node is associated with a cost representing the *average distance* (the average distance between between all the points in the database and their closest medoids) corresponding to these k medoids. Two nodes are neighbors if they differ by only one medoid. CLARANS will search, in the graph $G_{N,k}$, the node with the minimum cost to get the result. Similar to CLARA, CLARANS does not search on the entire graph, but in the neighborhood of a chosen node. CLARANS has been shown to be more effective than both PAM and CLARA [39], it is also able to detect the outliers. However, its time complexity is $O(N^2)$, therefore, it is not quite effective in very large data set. It is sensitive to the processing order of the data. CLARANS is not suitable to the incremental context because the graph changes when new elements are added.

ISODATA [40] Iterative Self-Organizing Data Analysis Techniques (ISODATA) is an iterative method. At first, it randomly selects k cluster centers (where k is the number of desired clusters). After assigning all the points in the database to the nearest center using the k-means method, we will:

- Eliminate clusters containing very few items (i.e. where the number of points is lower than a given threshold)
- Split clusters if we have too few clusters. A cluster is split if it has enough objects (i.e. the number of objects is greater than a given threshold) or if the average distance between its center and its objects is greater than the overall average distance between all objects in the database and their nearest cluster center.
- Merge the closest clusters if we have too many clusters.

The advantage of ISODATA is that it is not necessary to permanently set the number of classes. Similar to k-means, ISODATA has a low storage complexity (space) of $O(N+k)$ and a low computational complexity (time) of $O(Nkl)$, where N is the number of objects and l is the

[1] The medoid is defined as the cluster object which has the minimal average distance between it and the other objects in the cluster.

number of iterations. It is therefore compatible with large databases. But its drawback is that it relies on thresholds which are highly dependent on the size of the database and therefore difficult to settle.

The partitioning clustering methods described above are not incremental, they do not produce hierarchical structure. Almost of them are independent to the processing order of the data (except CLARANS) and do not depend on any parameters (except ISODATA). K-means, CLARA and CLARANS are adapted to the large databases, while CLARANS and ISODATA are able to detect the outliers. Among these methods, k-means is the best known and the most used because of its simplicity and its effectiveness for the large databases.

4.2 Hierarchical methods

Hierarchical methods decompose hierarchically the database vectors. They provide a hierarchical decomposition of the clusters into sub-clusters while the partitioning methods lead to a "flat" organization of clusters. Some methods of this kind are: AGNES [37], DIANA [37], AHC [42], BIRCH [45], ROCK [46], CURE [47], R-tree family [48–50], SS-tree [51], SR-tree [52], etc.

DIANA [37] DIvisitive ANAlysis (DIANA) is a top-down clustering method that divides successively clusters into smaller clusters. It starts with an initial cluster containing all the vectors in the database, then at each step the cluster with the maximum diameter is divided into two smaller clusters until all clusters contain only one singleton. A cluster K is split into two as follows:

1. Identify x^* in cluster K with the largest average dissimilarity with other objects of cluster K, then x^* initializes a new cluster K^*
2. For each object $x_i \notin K^*$, compute:
$$d_i = [\text{average}[d(x_i, x_j)]|x_j \in K \setminus K^*] - [\text{average}[d(x_i, x_j)]|x_j \in K^*] \quad (4)$$
where $d(x_i, x_j)$ is the dissimilarity between x_i and x_j
3. Choose x_k for which d_k is the largest. If $d_k > 0$ then add x_k into K^*
4. Repeat steps 2 and 3 until $d_k < 0$

The dissimilarity between objects can be measured by different measures (Euclidean, Minkowski, etc.). DIANA is not compatible with an incremental context. Indeed, if we want to insert a new element x into a cluster K that is divided into two clusters K_1 and K_2, the distribution of the elements of the cluster K into two new clusters K'_1 and K'_2 after inserting the element x may be very different to K_1 and K_2. In that case, it is difficult to reorganize the hierarchical structure. Moreover, the execution time to split a cluster into two new clusters is also high (at least quadratic to the number of elements in the cluster to be split), the overall computational complexity is thus at least $O(N^2)$. DIANA is therefore not suitable for a large database.

Simple Divisitive Algorithm (*Minimum Spanning Tree* (*MST*)) [11] This clustering method starts by constructing a Minimum Spanning Tree (MST) [41] and then, at each iteration, removes the longest edge of the MST to obtain the clusters. The process continues until there is no more edge to eliminate. When new elements are added to the database, the minimum spanning tree of the database changes, therefore it may be difficult to use this method in an incremental context. This method has a relatively high computational complexity of $O(N^2)$, it is therefore not compatible for clustering large databases.

Agglomerative Hierarchical Clustering (*AHC*) [42] AHC is a bottom-up clustering method which consists of the following steps:

1. Assign each object to a cluster. We obtain thus N clusters.
2. Merge the two closest clusters.
3. Compute the distances between the new cluster and other clusters.
4. Repeat steps 2 and 3 until it remains only one cluster

There are different approaches to compute the distance between any two clusters:

- In single-linkage, the distance between two clusters K_i and K_j is the minimum distance between an object in cluster K_i and an object in cluster K_j.
- In complete-linkage, the distance between two clusters K_i and K_j is the maximum distance between an object in cluster K_i and an object in cluster K_j.
- In average-linkage, the distance between two clusters K_i and K_j is the average distance between an object in cluster K_i and an object in cluster K_j.
- In centroid-linkage, the distance between two clusters K_i and K_j is the distance between the centroids of these two clusters.
- In Ward's method [43], the distance between two clusters K_i and K_j measures how much the total sum of squares would increase if we merged these two clusters:
$$\begin{aligned} D(K_i, K_j) &= \sum_{x_i \in K_i \cup K_j} (x_i - \mu_{K_i \cup K_j})^2 \\ &\quad - \sum_{x_i \in K_i} (x_i - \mu_{K_i})^2 - \sum_{x_i \in K_j} (x_i - \mu_{K_j})^2 \\ &= \frac{N_{K_i} N_{K_j}}{N_{K_i} + N_{K_j}} (\mu_{K_i} - \mu_{K_j})^2 \end{aligned} \quad (5)$$
where $\mu_{K_i}, \mu_{K_j}, \mu_{K_i \cup K_j}$ are respectively the center of clusters K_i, K_j, $K_i \cup K_j$, and N_{K_i}, N_{K_j} are respectively the numbers of points in clusters K_i and K_j.

Using AHC clustering, the tree constructed is deterministic since it involves no initialization step. But it is not capable to correct possible previous misclassification. The other disadvantages of this method is that it has a high computational complexity of $O(N^2 \log N)$ and a storage complexity of $O(N^2)$, and therefore is not really adapted to large databases. Moreover, it has a tendency to divide, sometimes wrongly, clusters including a large number of examples. It is also sensitive to noise and outliers.

There is an incremental variant [44] of this method. When there is a new item x, we determine its location in the tree by going down from the root. At each node R which has two children $G1$ and $G2$, the new element x will be merged with R if $D(G1, G2) < D(R, X)$; otherwise, we have to go down to $G1$ or $G2$. The new element x belongs to the influence region of $G1$ if $D(X, G1) \leq D(G1, G2)$.

BIRCH [45] Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) is developed to partition very large databases that can not be stored in main memory. The idea is to build a Clustering Feature Tree (CF-tree).

We define a CF-Vector summarizing information of a cluster including M vectors $(\mathbf{X}_1, \ldots, \mathbf{X_M})$, as a triplet $CF = (M, \mathbf{LS}, SS)$ where $\mathbf{LS}$ and SS are respectively the linear sum and the square sum of vectors $(\mathbf{LS} = \sum_{i=1}^{M} \mathbf{X_i}; SS = \sum_{i=1}^{M} \mathbf{X_i}^2)$. From the CF-vector of a cluster, we can simply compute the mean, the average radius and the average diameter (average distance between two vectors of the cluster) of a cluster and also the distance between two clusters (e.g. the Euclidean distance between their means).

A CF-Tree is a balanced tree having three parameters B, L and T:

- Each internal node contains at most B elements of the form $[CF_i, child_i]$ where $child_i$ is a pointer to its ith child node and CF_i is the CF-vector of this child.
- Each leaf node contains at most L elements of the form $[CF_i]$, it also contains two pointer *prev* and *next* to link leaf nodes.
- Each element CF_i of a leaf must have a diameter lower than a threshold T.

The CF-tree is created by inserting successive points into the tree. At first, we create the tree with a small value of T, then if it exceeds the maximum allowed size, T is increased and the tree is reconstructed. During reconstruction, vectors that are already inserted will not be reinserted because they are already represented by the CF-vectors. These CF-vectors will be reinserted. We must increment T so that two closest micro-clusters could be merged. After creating the CF-tree, we can use any clustering method (AHC, k-means, etc.) for clustering CF-vectors of the leaf nodes.

The CF-tree captures the important information of the data while reducing the required storage. And by increasing T, we can reduce the size of the CF-tree. Moreover, it has a low time complexity of $O(N)$, so BIRCH can be applied to a large database. The outliers may be eliminated by identifying the objects that are sparsely distributed. But it is sensitive to the data processing order and it depends on the choice of its three parameters. BIRCH may be used in the incremental context because the CF-tree can be updated easily when new points are added into the system.

CURE [47] In Clustering Using REpresentative (CURE), we use a set of objects of a cluster for representing the information of this cluster. A cluster K_i is represented by the following characteristics:

- $K_i.mean$: the mean of all objects in cluster K_i.
- $K_i.rep$: a set of objects representing cluster K_i. To choose the representative points of K_i, we select firstly the farthest point (the point with the greatest average distance with the other points in its cluster) as the first representative point, and then we choose the new representative point as the farthest point from the representative points.

CURE is identical to the agglomerative hierarchical clustering (AHC), but the distance between two clusters is computed based on the representative objects, which leads to a lower computational complexity. For a large database, CURE is performed as follows:

- Randomly select a subset containing N_{sample} points of the database.
- Partition this subset into p sub-partitions of size N_{sample}/p and realize clustering for each partition. Finally, clustering is performed with all found clusters after eliminating outliers.
- Each point which is not in the subset is associated with the cluster having the closest representative points.

CURE is insensitive to outliers and to the subset chosen. Any new point can be directly associated with the cluster having closest representative points. The execution time of CURE is relatively low of $O(N_{\text{sample}}^2 \log N_{\text{sample}})$, where N_{sample} is the number of elements in the subset chosen, so it can be applied on a large image database. However, CURE relies on a tradeoff between the effectiveness and the complexity of the overall method. Two few samples selected may reduce the effectiveness, while the complexity increases with the number of samples. This tradeoff may be difficult to find when considering huge databases. Moreover, the number of clusters k has to be fixed in order to associate points which are not selected as samples with the cluster having the closest representative points. If the number of clusters is changed, the points have to be reassigned. CURE is thus not suitable to the context that users are involved.

R-tree family [48–50] R-tree [48] is a method that aims to group the vectors using multidimensional bounding rectangles. These rectangles are organized in a balanced tree corresponding to the data distribution. Each node contains at least $N_{\min}$ and at most $N_{\max}$ child nodes. The records are stored in the leaves. The bounding rectangle of a leaf covers the objects belonging to it. The bounding rectangle of an internal node covers the bounding rectangles of its children. And the rectangle of the root node therefore covers all objects in the database. The R-tree thus provides "hierarchical" clusters, where the clusters may be divided into sub-clusters or clusters may be grouped into super-clusters. The tree is incrementally constructed by inserting iteratively the objects into the corresponding leaves. A new element will be inserted into the leaf that requires the least enlargement of its bounding rectangle. When a full node is chosen to insert a new element, it must be divided into two new nodes by minimizing the total volume of the two new bounding boxes.

R-tree is sensitive to the insertion order of the records. The overlap between nodes is generally important. The R+-tree [49] and R*-tree [50] structures have been developed with the aim of minimizing the overlap of bounding rectangles in order to optimize the search in the tree. The computational complexity of this family is about $O(N\log N)$, it is thus suitable to the large databases.

SS-tree [51] The Similarity Search Tree (SS-tree) is a similarity indexing structure which groups the feature vectors based on their dissimilarity measured using the Euclidean distance. The SS-tree structure is similar to that of the R-tree but the objects of each node are grouped in a bounding sphere, which permits to offer an isotropic analysis of the feature space. In comparison to the R-tree family, SS-tree has been shown to have better performance with high dimensional data [51] but the overlap between nodes is also high. As for the R-tree, this structure is incrementally constructed and compatible to the large databases due to its relatively low computational complexity of $O(N\log N)$. But it is sensitive to the insertion order of the records.

SR-tree [52] SR-tree combines two structures of R*-tree and SS-tree by identifying the region of each node as the intersection of the bounding rectangle and the bounding sphere. By combining the bounding rectangle and the bounding sphere, SR-tree allows to create regions with small volumes and small diameters. That reduces the overlap between nodes and thus enhances the performance of nearest neighbor search with high-dimensional data. SR-tree also supports incrementality and compatibility to deal with the large databases because of its low computational complexity of $O(N\log N)$. SR-tree is still sensitive to the processing order of the data.

The advantage of hierarchical methods is that they organize data in hierarchical structure. Therefore, by considering the structure at different levels, we can obtain different number of clusters. DIANA, MST and AHC are not adapted to large databases, while the others are suitable. BIRCH, R-tree, SS-tree and SR-tree structures are built incrementally by adding the records, they are by nature incremental. But because of this incremental construction, they depend on the processing order of the input data. CURE is enable to add new points but the records have to be reassigned whenever the number of clusters k is changed. CURE is thus not suitable to the context where users are involved.

4.3 Grid-based methods

These methods are based on partitioning the space into cells and then grouping neighboring cells to create clusters. The cells may be organized in a hierarchical structure or not. The methods of this type are: STING [53], Wave-Cluster [54], CLICK [55], etc.

STING [53] STatistical INformation Grid (STING) is used for spatial data clustering. It divides the feature space into rectangular cells and organizes them according to a hierarchical structure, where each node (except the leaves) is divided into a fixed number of cells. For instance, each cell at a higher level is partitioned into 4 smaller cells at the lower level.

Each cell is described by the following parameters:

- An attribute-independent parameter:
  - n: number of objects in this cell
- For each attribute, we have five attribute-dependent parameters:
  - μ: mean value of the attribute in this cell.
  - σ: standard deviation of all values of the attribute in this cell.
  - *max*: maximum value of the attribute in the cell.
  - *min*: minimum value of the attribute in the cell.
  - *distribution*: the type of distribution of the attribute value in this cell. The potential distributions can be either normal, uniform, exponential, etc. It could be "None" if the distribution is unknown.

The hierarchy of cells is built upon entrance of data. For cells at the lowest level (leaves), we calculate the parameters n, μ, σ, *max*, *min* directly from the data; the *distribution* can be determined using a statistical hypothesis test, for example the χ^2-test. Parameters of the cells at higher level can be calculated from parameters of lower lever cell as in [53].

Since STING goes through the data set once to compute the statistical parameters of the cells, the time complexity of STING for generating clusters is $O(N)$, STING is thus suitable for large databases. Wang et al. [53] demonstrated

that STING outperforms the partitioning method CLARANS as well as the density-based method DBSCAN when the number of points is large. As STING is used for spatial data and the attribute-dependent parameters have to be calculated for each attribute, it is not adapted to high-dimensional data such as image feature vectors. We could insert or delete some points in the database by updating the parameters of the corresponding cells in the tree. It is able to detect outliers based on the number of objects in each cell.

CLIQUE [55] CLustering In QUEst (CLIQUE) is dedicated to high dimensional databases. In this algorithm, we divide the feature space into cells of the same size and then keep only the dense cells (whose density is greater than a threshold σ given by user). The principle of this algorithm is as follows: a cell that is dense in a k-dimensional space should also be dense in any subspace of $k-1$ dimensions. Therefore, to determine dense cells in the original space, we first determine all 1-dimensional dense cells. Having obtained $k-1$ dimensional dense cells, recursively the k-dimensional dense cells candidates can be determined by the candidate generation procedure in [55]. Moreover, by parsing all the candidates, the candidates that are really dense are determined. This method is not sensitive to the order of the input data. When new points are added, we only have to verify if the cells containing these points are dense or not. Its computational complexity is linear to the number of records and quadratic to the number of dimensions. It is thus suitable to large databases. The outliers may be detected by determining the cells which are not dense.

The grid-based methods are in general adapted to large databases. They are able to be used in an incremental context and to detect outliers. But STING is not suitable to high dimensional data. Moreover, in high dimensional context, data is generally extremely sparse. When the space is almost empty, the hierarchical methods (Sect. 4.2) are better than grid-based methods.

4.4 Density-based methods

These methods aim to partition a set of vectors based on the local density of these vectors. Each vector group which is locally dense is considered as a cluster. There are two kinds of density-based methods:

- Parametric approaches, which assume that data is distributed following a known model: EM [56], etc.
- Non-parametric approaches: DBSCAN [57], DENCLUE [58], OPTICS [59], etc.

EM [56] For the Expectation Maximization (EM) algorithm, we assume that the vectors of a cluster are independent and identically distributed according to a Gaussian mixture model. EM algorithm allows to estimate the optimal parameters of the mixture of Gaussians (means and covariance matrices of clusters).

The EM algorithm consists of four steps:

1. Initialize the parameters of the model and the k clusters.
2. E-step: calculate the probability that an object x_i belongs to any cluster K_j.
3. M-step: Update the parameters of the mixture of Gaussians so that it maximize the probabilities.
4. Repeat steps 2 and 3 until the parameters are stable.

After setting all parameters, we calculate, for each object x_i, the probability that it belongs to each cluster K_j and we will assign it to the cluster associated with the maximum probability.

EM is simple to apply. It allows to identify outliers (e.g. objects for which all the membership probabilities are below a given threshold). The computational complexity of EM is about $O(Nk^2l)$, where l is the number of iterations. EM is thus suitable to large databases when k is small enough. However, if the data is not distributed according to a Gaussian mixture model, the results are often poor, while it is very difficult to determine the distribution of high dimensional data. Moreover, EM may converge to a local optimum, and it is sensitive to the initial parameters. Additionally, it is difficult to use EM in an incremental context.

DBSCAN [57] Density Based Spatial Clustering of Applications with Noise (DBSCAN) is based on the local density of vectors to identify subsets of dense vectors that will be considered as clusters. For describing the algorithm, we use the following terms:

- *ϵ-neighborhood* of a point p contains all the points q, whose distance $D(q,p)<\epsilon$.
- *MinPts* is a constant value used for determining the *core points* in a cluster. A point is considered as a *core point* if there are at least *MinPts* points in its *ϵ-neighborhood*.
- *directly density-reachable*: a point p is *directly density-reachable* from a point q if q is a core point and p is in the ϵ *-neighborhood* of q.
- *density-reachable*: a point p is *density-reachable* from a core point q if there is a chain of points $p_1,\ldots,p_n$ such that $p_1=q$, $p_n=p$ and p_{i+1} is *directly density-reachable* from p_i.
- *density-connected*: a point p is *density-connected* to a point q if there is a point o such that p and q are both *density-reachable* from o.

Intuitively, a cluster is defined to be a set of *density-connected* points. The DBSCAN algorithm is as follows:

1. For each vector x_i which is not associated with any cluster:

- If x_i is a core point, we try to find all vectors x_j which are *density-reachable* from x_i. All these vectors x_j are then classified in the same cluster of x_i.
- Else label x_i as noise.

2. For each noise vector, if it is *density-connected* to a core point, it is then assigned to the same cluster of the core point.

This method allows to find clusters with complex shapes. The number of clusters does not have to be fixed a priori and no assumption is made on the distribution of the features. It is robust to outliers. But on the other hand, the parameters ϵ and *MinPts* are difficult to adjust and this method does not generate clusters with different levels of scatter because of the ϵ parameter being fixed. The DBSCAN fails to identify clusters if the density varies and if the data set is too sparse. This method is therefore not adapted to high dimensional data. The computational complexity of this method being low $O(N\log N)$, DBSCAN is suitable to large data sets. This method is difficult to use in an incremental context because when we insert or delete some points in the database, the local density of vectors is changed and some non-core points could become core points and vice versa.

OPTICS [59] OPTICS (Ordering Points To Identify the Clustering Structure) is based on DBSCAN but instead of a single neighborhood parameter ϵ, we work with a range of values $[\epsilon_1, \epsilon_2]$ which allows to obtain clusters with different scatters. The idea is to sort the objects according to the minimum distance between object and a core object before using DBSCAN; the objective is to identify in advance the very dense clusters. As DBSCAN, it may not be applied to high-dimensional data or in an incremental context. The time complexity of this method is about $O(N\log N)$. Like DBSCAN, it is robust to outliers but it is very dependent on its parameters and is not suitable to an incremental context.

The density-based clustering methods are in general suitable to large databases and are able to detect outliers. But these methods are very dependent on their parameters. Moreover, they does not produce hierarchical structure and are not adapted to an incremental context.

4.5 Neural network based methods

For this kind of approaches, similar records are grouped by the network and represented by a single unit (neuron). Some methods of this kind are Learning Vector Quantization (LVQ) [60], Self-Organizing Map (SOM) [60], Adaptive Resonance Theory (ART) models [61], etc. In which, SOM is the best known and the most used method.

Self-Organizing Map (*SOM*) [60] SOM or Kohonen map is a mono-layer neural network which output layer contains neurons representing the clusters. Each output neuron contains a weight vector describing a cluster. First, we have to initialize the values of all the output neurons.

The algorithm is as follows:

- For each input vector, we search the best matching unit (BMU) in the output layer (output neuron which is associated with the nearest weight vector).
- And then, the weight vectors of the BMU and neurons in its neighborhood are updated towards the input vector.

SOM is incremental, the weight vectors can be updated when new data arrive. But for this method, we have to fix a priori the number of neurons, and the rules of influence of a neuron on its neighbors. The result depends on the initialization values and also the rules of evolution concerning the size of the neighborhood of the BMU. It is suitable only for detecting hyperspherical clusters. Moreover, SOM is sensitive to outliers and to the processing order of the data. The time complexity of SOM is $O(k'Nm)$, where k' is the number of neurons in the output layer, m is the number of training iterations and N is the number of objects. As m and k' are usually much smaller than the number of objects, SOM is adapted to large databases.

4.6 Discussion

Table 1 compares formally the different clustering methods (partitioning methods, hierarchical methods, grid-based methods, density-based methods and neural network based methods) based on different criteria (complexity, adapted to large databases, adapted to incremental context, hierarchical structure, data order dependence, sensitivity to outliers and parameters dependence). Where:

- N: the number of objects in the data set.
- k: the number of clusters.
- l: the number of iterations.
- N_{sample}: the number of samples chosen by the clustering methods (in the case of CURE).
- m: the training times (in the case of SOM).
- k' : the number of neurons in the output layer (in the case of SOM).

The partitioning methods (k-means, k-medoids (PAM), CLARA, CLARANS, ISODATA) are not incremental; they do not produce hierarchical structure. Most of them are independent of the processing order of the data and do not depend on any parameters. K-means, CLARA and ISODATA are suitable to large databases. K-means is the baseline method because of its simplicity and its effectiveness for large database. The hierarchical methods (DIANA, MST, AHC, BIRCH, CURE, R-tree, SS-tree, SR-tree) organize data in hierarchical structure. Therefore,

by considering the structure at different levels, we can obtain different numbers of clusters that are useful in the context where users are involved. DIANA, MST and AHC are not suitable to the incremental context. BIRCH, R-tree, SS-tree and SR-tree are by nature incremental because they are built incrementally by adding the records. They are also adapted to large databases. CURE is also adapted to large databases and it is able to add new points but the results depend much on the samples chosen and the records have to be reassigned whenever the number of clusters k is changed. CURE is thus not suitable to the context where users are involved. The grid-based methods (STING, CLIQUE) are in general adapted to large databases. They are able to be used in incremental context and to detect outliers. STING produce hierarchical structure but it is not suitable to high dimensional data such as features image space. Moreover, when the space is almost empty, the hierarchical methods are better than grid-methods. The density-based methods (EM, DBSCAN, OPTICS) are in general suitable to large databases and are able to detect outliers. But they are very dependent on their parameters, they do not produce hierarchical structure and are not adapted to incremental context. Neural network based methods (SOM) depend on initialization values and on the rules of influence of a neuron on its neighbors. SOM is also sensitive to outliers and to the processing order of the data. SOM does not produce hierarchical structure. Based on the advantages and the disadvantages of different clustering methods, we can see that the hierarchical methods (BIRCH, R-tree, SS-tree and SR-tree) are most suitable to our context.

We choose to present, in Sect. 5, an experimental comparison of five different clustering methods: global k-means [35], AHC [42], R-tree [48], SR-tree [52] and BIRCH [45]. Global k-means is a variant of the well known and the most used clustering method (k-means). The advantage of the global k-means is that we can automatically select the number of clusters k by stopping the algorithm at the value of k providing acceptable results. The other methods provide hierarchical clusters. AHC is chosen because it is the most popular method in the hierarchical family and there exists an incremental version of this method. R-tree, SR-tree and BIRCH are dedicated to large databases and they are by nature incremental.

5 Experimental comparison and discussion

5.1 The protocol

In order to compare the five selected clustering methods, we use different image databases of increasing size (Wang,[2] PascalVoc2006,[3] Caltech101,[4] Corel30k). Some examples of these databases are shown in Figs. 1, 2, 3 and 4. Small databases are intended to verify the performance of descriptors and also clustering methods. Large databases are used to test clustering methods for structuring large amount of data. Wang is a small and simple database, it contains 1,000 images of 10 different classes (100 images per class). PascalVoc2006 contains 5,304 images of 10 classes, each image containing one or more object of different classes. In this paper, we analyze only hard clustering methods in which an image is assigned to only one cluster. Therefore, in PascalVoc2006, we choose only the images that belong to only one class for the tests (3,885 images in total). Caltech101 contains 9,143 images of 101 classes, with 40 up to 800 images per class. The largest image database used is Corel30k, it contains 31,695 images of 320 classes. In fact, Wang is a subset of Corel30k. Note that we use for the experimental tests the same number of clusters as the number of classes in the ground truth.

Concerning the feature descriptors, we implement one global and different local descriptors. Because our study focuses on the clustering methods and not on the feature descriptors, we choose some feature descriptors that are widely used in literature for our experiment. The global descriptor of size 103 is built as the concatenation of three different global descriptors:

- RGB histobin: 16 bins for each channel. This gives a histobin of size $3 \times 16 = 48$.
- Gabor filters: we used 24 Gabor filters on 4 directions and 6 scales. The statistical measure associated with each output image is the mean and standard deviation. We obtained thus a vector of size $24 \times 2 = 48$ for the texture.
- Hu's moments: 7 invariant moments of Hu are used to describe the shape.

For local descriptors, we implemented the SIFT and color SIFT descriptors. They are widely used nowadays for their high performance. We use the SIFT descriptor code of David Lowe[5] and color SIFT descriptors of Koen van de Sande.[6] The "Bag of words" approach is chosen to group local features into a single vector representing the frequency of occurrence of the visual words in the dictionary (see Sect. 3).

As mentioned in Sect. 4.6, we implemented five different clustering methods: global k-means [35], AHC [42], R-tree [48], SR-tree [52] and BIRCH [45]. For the agglomerative

[2] http://wang.ist.psu.edu/docs/related/.

[3] http://pascallin.ecs.soton.ac.uk/challenges/VOC/.

[4] http://www.vision.caltech.edu/Image_Datasets/Caltech101/.

[5] http://www.cs.ubc.ca/~lowe/keypoints/.

[6] http://staff.science.uva.nl/~ksande/research/colordescriptors/.

Table 1 Formal comparison of different clustering methods (partitioning methods, hierarchical methods, grid-based methods, density-based methods and neural network based methods) based on different criterias (complexity, adapted to large databases, adapted to incremental context, hierarchical structure, data order dependence, sensitivity to outliers, parameters dependence)

Methods	Complexity	Adapted to large database	Adapted to incremental context	Hierarchical structure	Data order dependence	Sensitivity to outliers	Parameters Dependence
K-means (Partitioning)	$O(Nkl)$ (time) $O(N + k)$ (space)	Yes	No	No	No	Sensitive	No
k-medoids (PAM) (Partitioning)	$O(k\ (n - k)^2)$ (time)	No	No	No	No	Less sensitive	No
CLARA (Partitioning)	$O(k\ (40 + k)^2 + k(N - k))$ (time)	Yes	Able to add new points	No	No	Less sensitive	No
CLARANS (Partitioning)	$O(N^2)$ (time)	No	No	No	Yes	Enable outliers detection	No
ISODATA (Partitioning)	$O(Nkl)$ (time) $O(N + k)$ (space)	Yes	No	No	No	Enable outliers detection	Yes
DIANA (Hierarchical)	$O(N^2)$ (time)	No	No	Yes	No	Less sensitive	No
Simple Divisitive Algorithm (MST) (Hierarchical)	$O(N^2)$ (time)	No	No	Yes	No	Sensitive	No
AHC (Hierarchical)	$O(N^2 \log N)$ (time) $O(N^2)$ (space)	No	Have incremental version	Yes	No	Sensitive	No
BIRCH (Hierarchical)	$O(N)$ (time)	Yes	Yes	Yes	Yes	Enable outliers detection	Yes
CURE (Hierarchical)	$O(N_{\text{sample}}^2 \log N_{\text{sample}})$ (time)	Yes	Able to add new points	Yes	No	Less sensitive	No
R-tree (Hierarchical)	$O(N \log N)$(time)	Yes	Yes	Yes	Yes	Sensitive	Yes
SS-tree (Hierarchical)	$O(N \log N)$ (time)	Yes	Yes	Yes	Yes	Sensitive	Yes
SR-tree (Hierarchical)	$O(N \log N)$ (time)	Yes	Yes	Yes	Yes	Sensitive	Yes
STING (Grid-based)	$O(N)$ (time)	Yes	Yes	Yes	No	Enable outliers detection	No
CLIQUE (Grid-based)	Linear to the number of element, quadratic to the number of dimensions (time)	Yes	Able to add new points	No	No	Enable outliers detection	Yes
EM (Density-based)	$O(Nk^2l)$ (time)	Yes	No	No	No	Enable outliers detection	Yes
DBSCAN (Density-based)	$O(N \log N)$ (time)	Yes	No	No	No	Enable outliers detection	Yes
OPTICS (Density-based)	$O(N \log N)$ (time)	Yes	No	No	No	Enable outliers detection	Yes
SOM (Neural network based)	$O(k'Nm)$ (time)	Yes	Yes	No	Yes	Sensitive	Yes

Methods in bold are chosen for experimental comparison

Fig. 1 Examples of Wang

Fig. 2 Examples of Pascal

Fig. 3 Examples of Caltech101

Fig. 4 Examples of Corel30k

hierarchical clustering (AHC), in our tests with five different kinds of distance described in Sect. 4.2 (single-linkage, complete-linkage, average-linkage, centroid-linkage and Ward). The distance of the Ward's method [43] gives the best results. It is used therefore in the experiment of the AHC method.

In order to analyze our clustering results, we use two kinds of measures:

– *Internal measures* are low-level measures which are essentially numerical. The quality of the clustering is evaluated based on intrinsic information of the data set.

They consider mainly the distribution of the points into clusters and the balance of these clusters. For these measures, we ignore if the clusters are semantically meaningful or not ("meaning" of each cluster and validity of a point belonging to a cluster). Therefore, internal measures may be considered as unsupervised. Some measures of this kind are: homogeneity or compactness [62], separation [62], Silhouette Width [63], Huberts Γ statistic [64], etc.

– *External measures* evaluate the clustering by comparing it to the distribution of data in the ground truth, which is often created by humans or by a source of knowledge "validated" by humans. The ground truth provides the semantic meaning and therefore, external measures are high-level measures that evaluate the clustering results compared to the wishes of the user. Thus, we can consider external measures as supervised. Some measures of this kind are: Purity [65], Entropy [65], F-measure [66], V-measure [67], Rand Index [68], Jaccard Index [69], Fowlkes–Mallows Index [70], Mirkin metric [71], etc.

The first type of measures (internal) does not consider the semantic point of view, it can therefore be applied automatically by the machine, but it is a numerical evaluation. The second type (external) forces the human to provide a ground truth. It is therefore more difficult to be done automatically, but the evaluation is closer to the wishes of the user. It is thus a semantic evaluation. As clustering is an unsupervised learning problem, we should know nothing about the ground truth. But our work in the future is to embed human for improving the result of the clustering. Thus, we supposed, in this paper, that the class of each image is known, this will help us to evaluate the results of the clustering compared to what the human wants. Among all different measures, we use here five different measures that seem representative: Silhouette Width (SW) [63] as internal measure and V-measure [67], Rand Index [68], Jaccard Index [69] and Fowlkes–Mallows Index [70] as external measures. The Silhouette Width measures the compactness of each cluster and the separation between clusters based on the distance between points in a same cluster and between points from different clusters. V-measure evaluates both the homogeneity and the completeness of the clustering solution. A clustering solution satisfies the homogeneity criterion if all clusters contain only members of one class and satisfies the completeness if all points belonging to a given class are also the elements of a same cluster. Rand Index estimates the probability that a pair of points are similarly classified (together or separately) in the clustering solution and in the ground truth. Jaccard and Fowlkes–Mallows indexes measure the probability that a pair of points are classified together in both the clustering solution and the ground truth, provided they are classified together in at least one of these two partitions. The greater values of these measures means better results. Almost all the evaluation measures used are external measures because they compare the results of clustering to what the human wants. The internal measure is used for evaluating the differences between numerical evaluation and semantic evaluation.

5.2 Experimental results

5.2.1 Clustering methods and feature descriptors analysis

The first set of experiments aims at evaluating the performances of different clustering methods and different feature descriptors. Another objective is evaluate the stability of each method depending on different parameters such as the threshold T in the case of the BIRCH method or the number of children in the cases of R-tree and SR-tree. The Wang image database was chosen for these tests because of its simplicity and its popularity in the field of image analysis. We fix the number of clusters $k = 10$ for all the following tests with the Wang image database (because its ground truth contains 10 classes).

Methods analysis Figure 5 shows the result of five different clustering methods (global k-means, AHC, R-tree, SR-tree and BIRCH) using the global feature descriptor on the Wang image database. The global feature descriptor is used because of its simplicity. We can see that, for this image database, the global k-means and BIRCH methods give in general the best results, while R-tree, SR-tree and AHC give similar results.

Now, we will analyze the stability of these methods towards their parameters (when needed). AHC and global k-means are parameter-free. In the case of BIRCH, the threshold T is an important parameter. As stated in the previous section, BIRCH includes two main steps, the first step is to organize all points in a CF-tree so that each leaf entry contains all points within a radius smaller than T; the second step considers each leaf entry as a point and realizes the clustering for all the leaf entries. The value of the threshold T, has an influence on the number of points in each entry leaf, and thus on the results of the clustering. Figure 6 shows the influence of T on the results of BIRCH. Note that for the second stage of BIRCH, we use k-means for clustering the leaf entries because of its simplicity; the k first leaf entries are used as k initial means so that the result is not influenced by the initial condition. We can see that for the Wang image database and for the value of T that we tested, the results are very stable when the values of T are varied, even though $T = 1$ gives slightly better result. For the R-tree and the SR-tree clustering methods,

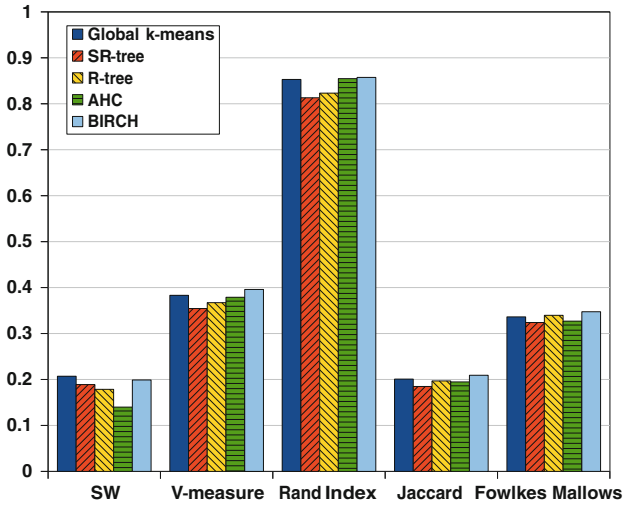

Fig. 5 Clustering methods (Global k-means, SR-tree, R-tree, AHC, BIRCH) comparison using global feature descriptor on the Wang image database. Five measures (Silhouette Width, V-measure, Rand Index, Jaccard Index, Fowlkes–Mallows Index) are used. The higher are these measures, the best are the results

the minimum and maximum number of children of each node are the parameters that we have to fix. We can add more points in a node if the number of children is high enough. Figure 7 shows the influence of these parameters to the result of these two methods. We can see that these parameters may influence so much on both the R-tree and the SR-tree clustering results. For instance, the result of the R-tree clustering when a node has at least 5 children and at most 15 children is much worse than when the minimum and maximum numbers of children are respectively 4 and 10. Selecting the best values for these parameters is crucial, especially for the tree based methods which are not stable. However, it may be difficult to choose a convenient value for these parameters, as it depends on the feature descriptor and on the image database used. In the following tests, we try different values of these parameters and choose the best compromise on all the measures.

Feature descriptors analysis Figure 8 shows the results of the global k-means and BIRCH, which gave previously the best results, on the Wang image database using different feature descriptors (global feature descriptor, SIFT, rgSIFT, CSIFT, local RGB histogram, RGBSIFT and OpponentSIFT). Note that the dictionary size of the "Bag of Words" approach used jointly with the local descriptors is 500. We can see that the global feature descriptor provides good results in general. The SIFT, RGBSIFT and OpponentSIFT are worse than the other local descriptors (CSIFT, rgSIFT, local RGB Histogram). This may be explained by the fact that in the Wang database, color is very important and that the Wang database contains classes (e.g. dinosaurs) which are very easy to differentiate from the others. Thus, in the remaining tests, we will use three feature descriptors (global feature descriptor, CSIFT and rgSIFT).

Methods and feature descriptors comparisons As stated in Sect. 3, the dictionary size determines the size of the feature vector of each image. Given the problems and difficulties related to large image databases, we prefer to choose a relatively small size of the dictionary in order to reduce the number of dimensions of the feature vectors. Working with a small dictionary can also reduce the execution time, which is an important criterion in our case where we aim at involving the user. But, a too small dictionary may not be sufficient to accurately describe the database. Therefore, we must find the best trade-off between the performance and the size of the dictionary. Figures 9, 10, 11 and 12 analyze the results of global k-means, R-tree, SR-tree, BIRCH and AHC on the Wang image database with different feature descriptors (global descriptor, CSIFT and rgSIFT when varying the dictionary size from 50 to 500) using four measures (V-measure, Rand Index, Fowlkes–Mallows Index and SW). Jaccard Index is not used because it is very similar with the Fowlkes–Mallows Index (we can see that they give similar evaluations in the previous results). Note that *LocalDes* means the local descriptor (CSIFT on the left-hand side figure or rgSIFT on the right-hand side figure). *GlobalDes* means global descriptor, which is not influenced by any size of dictionary, but we choose to represent its results on the same graphics using a straight line for comparison and presentation matters. We can see that different feature descriptors give different results, and the size of the dictionary has also an important influence on the clustering results, especially for R-tree and SR-tree. When using external measures (V-measure, Rand Index and Fowlkes–Mallows Index), BIRCH using rgSIFT with a dictionary of 400 to 500 visual words always give the best results. On the contrary, when using the internal measure (silhouette width), the global descriptor is always the best descriptor. We have to remind that the internal and external measures do not evaluate the same aspects. The internal measure used here evaluates without any supervisor the compactness and the separation of clusters based on the distance between elements in a same cluster and in different clusters, while the external measures compare the distributions of points in the clustering result and in the ground truth. We choose the value 200 for the dictionary size in the case of CSIFT and rgSIFT for the following tests and for our future works because it is a good tradeoff between the size of the feature vector and the performance. Concerning the different methods, we can see that global k-means, BIRCH and AHC are in general more effective and stable than R-tree and SR-tree. BIRCH is the best method when we use external measures but the global k-means is better following the internal measure.

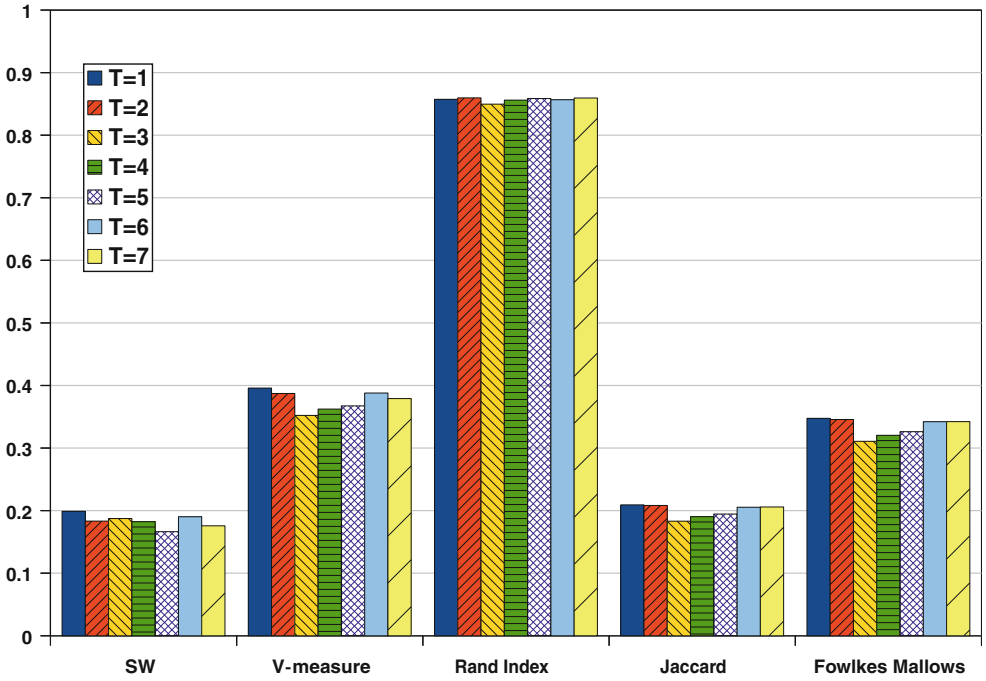

Fig. 6 Influence of the threshold $T(T = 1, \ldots, 7)$ on the BIRCH clustering results using the Wang image database. Five measures (Silhouette Width, V-measure, Rand Index, Jaccard Index, Fowlkes–Mallows Index) are used

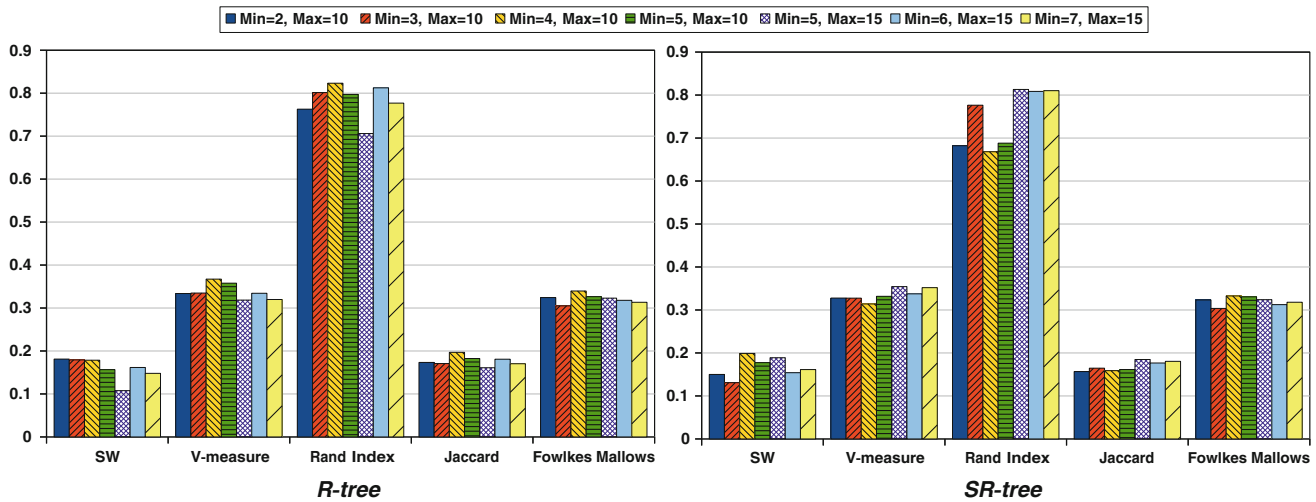

Fig. 7 Influence of minimum and maximum numbers of children on R-tree and SR-tree results, using the Wang image database. Five measures (Silhouette Width, V-measure, Rand Index, Jaccard Index, Fowlkes–Mallows Index) are used

5.2.2 Verification on different image databases

Because the Wang image database is very simple and small, the results of clustering obtained on it may not be very representative of what happens with huge masses of data. Thus, in these following tests, we will analyze the clustering results with larger image databases (PascalVoc2006, Caltech101 and Corel30k). Global k-means, BIRCH and AHC are used because of their high performances and stability. In the case of the Corel30k image database, the AHC method is not used because of the lack of RAM memory. In fact, the AHC clustering requires a large amount of memory when treating more than 10,000 elements, while the Corel30k contains more than 30,000 images. This problem could be solved using the incremental version [44] of AHC which allows to process databases containing about 7 times more data than the classical AHC. The global feature descriptor, CSIFT and rgSIFT are also used for these databases. Note

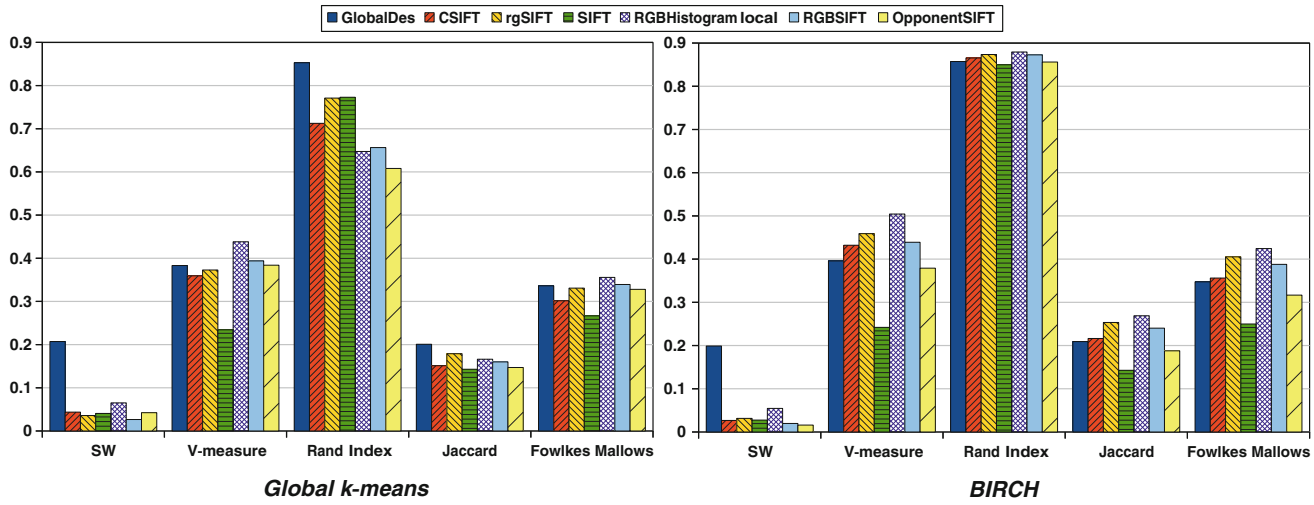

Fig. 8 Feature descriptors (global feature descriptor, local feature descriptors (CSIFT, rgSIFT, SIFT, RGBHistogram, RGBSIFT, OpponentSIFT)) comparison using the global k-means and BIRCH clustering methods on the Wang image database

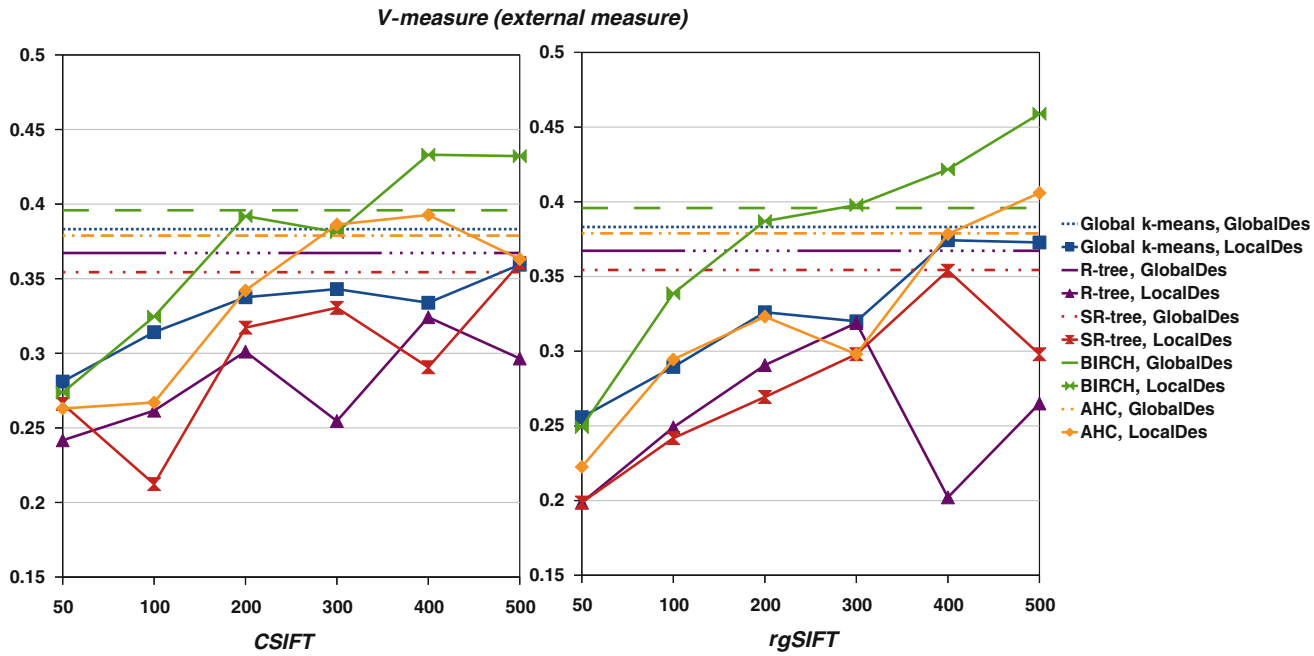

Fig. 9 Clustering methods (global k-means, R-tree, SR-tree, BIRCH and AHC) and feature descriptors (global k-means, CSIFT and rgSIFT) comparisons using V-measure on the Wang image database. The dictionary size of the "Bag of Words" approach used jointly with the local descriptors is from 50 to 500

that the size of the dictionary used for both local descriptors here is 200.

Figures 13, 14 and 15 show the results of the global k-means, the BIRCH and the AHC clustering methods on the PascalVoc2006, Caltech101 and Corel30k. We can see that the numerical evaluation (internal measure) always appreciates the global descriptor but the semantic evaluations (external measures) appreciates the local descriptors in almost all cases, except in the case of global k-means on the Corel30k image database. And in general, the global k-means and the BIRCH clustering methods give similar results, with rgSIFT providing slightly better results than CSIFT. AHC gives better results using CSIFT than rgSIFT, but it gives worse results than BIRCH+rgSIFT in general. Concerning the execution time, we notice that the clustering using the global descriptor is faster than that using the local descriptor (because the global feature descriptor has a dimension of 103 while the dimension of the local

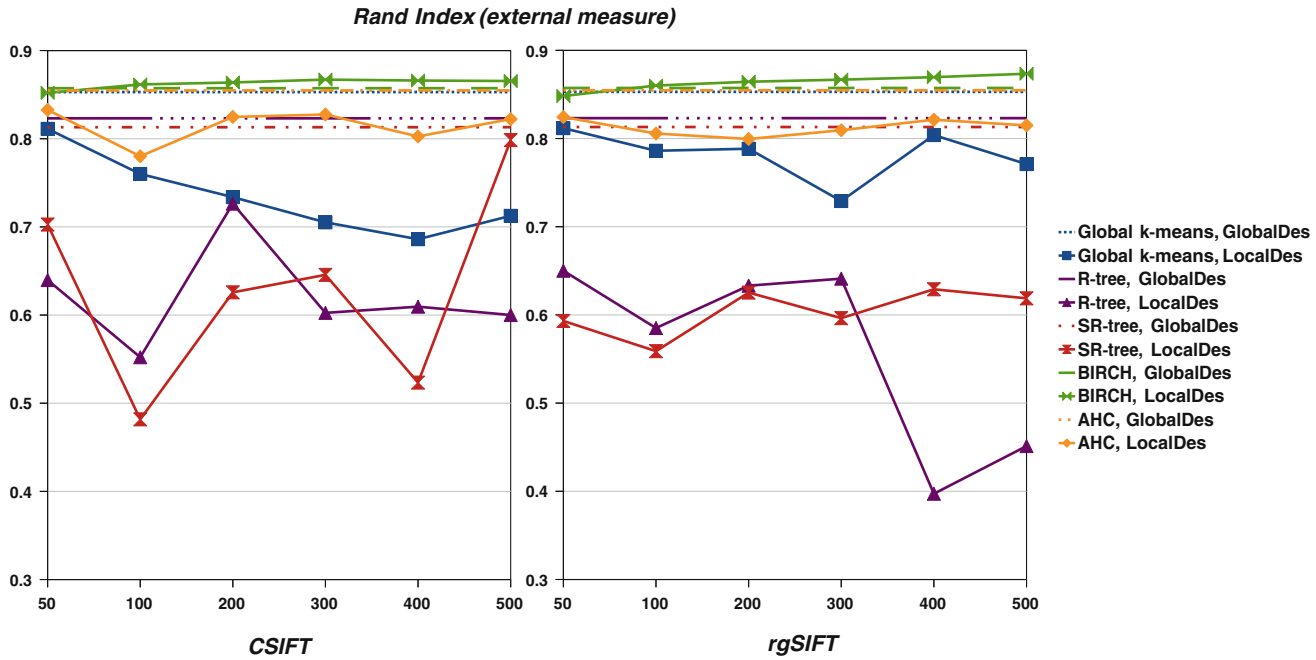

Fig. 10 Clustering methods (global k-means, R-tree, SR-tree, BIRCH and AHC) and feature descriptors (global k-means, CSIFT and rgSIFT) comparisons using Rand Index on the Wang image database. The dictionary size of the "Bag of Words" approach used jointly with the local descriptors is from 50 to 500

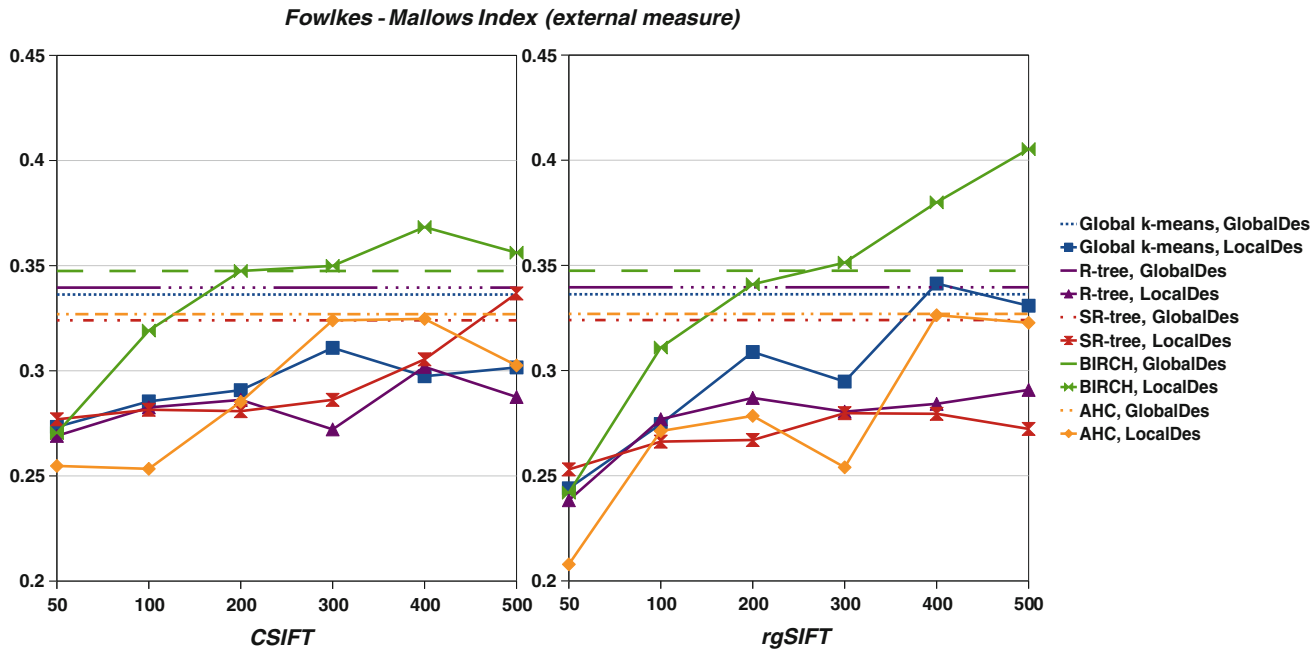

Fig. 11 Clustering methods (global k-means, R-tree, SR-tree, BIRCH and AHC) and feature descriptors (global k-means, CSIFT and rgSIFT) comparisons using Fowlkes–Mallows Index on the Wang image database. The dictionary size of the "Bag of Words" approach used jointly with the local descriptors is from 50 to 500

descriptors is 200) and among the local descriptor, the clustering using rgSIFT is faster. And in comparison to global k-means and AHC, BIRCH is much faster, especially in the case of the Caltech101 and the Corel30k image databases (e.g. the execution time of BIRCH in the case of the Corel30k is about 400 times faster than that of the global k-means, using rgSIFT where the dictionary size is 200).

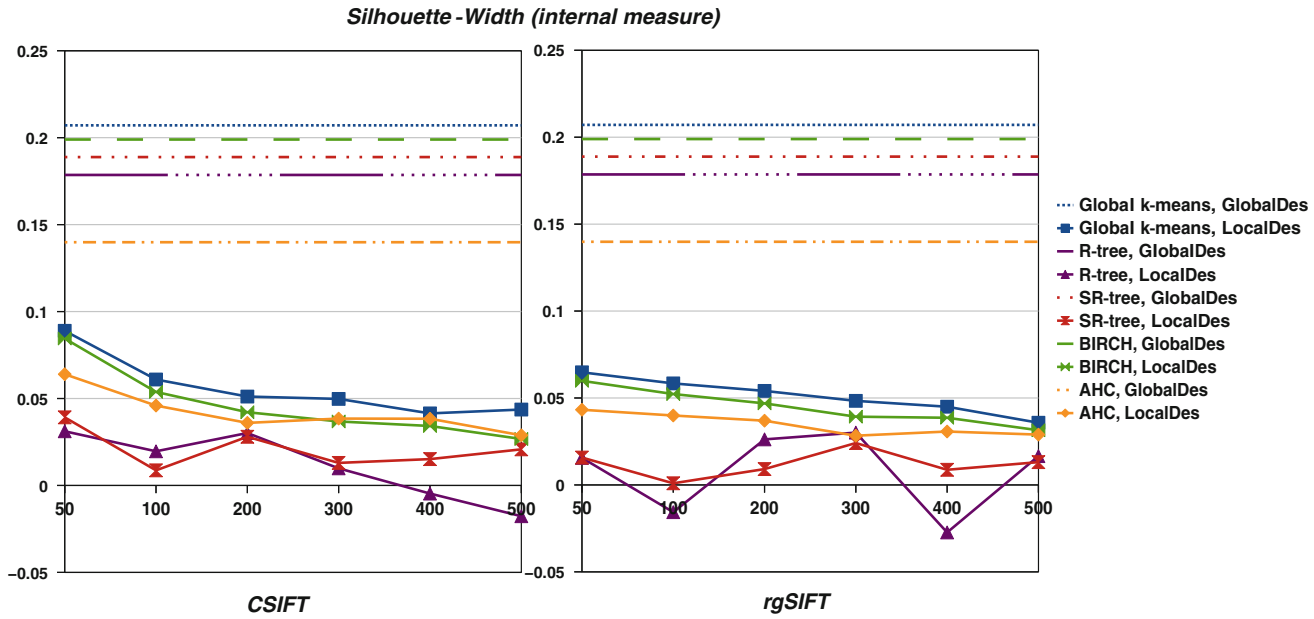

Fig. 12 Clustering methods (global k-means, R-tree, SR-tree, BIRCH and AHC) and feature descriptors (global k-means, CSIFT and rgSIFT) comparisons using Silhouette Width on the Wang image database. The dictionary size of the "Bag of Words" approach used jointly with the local descriptors is from 50 to 500

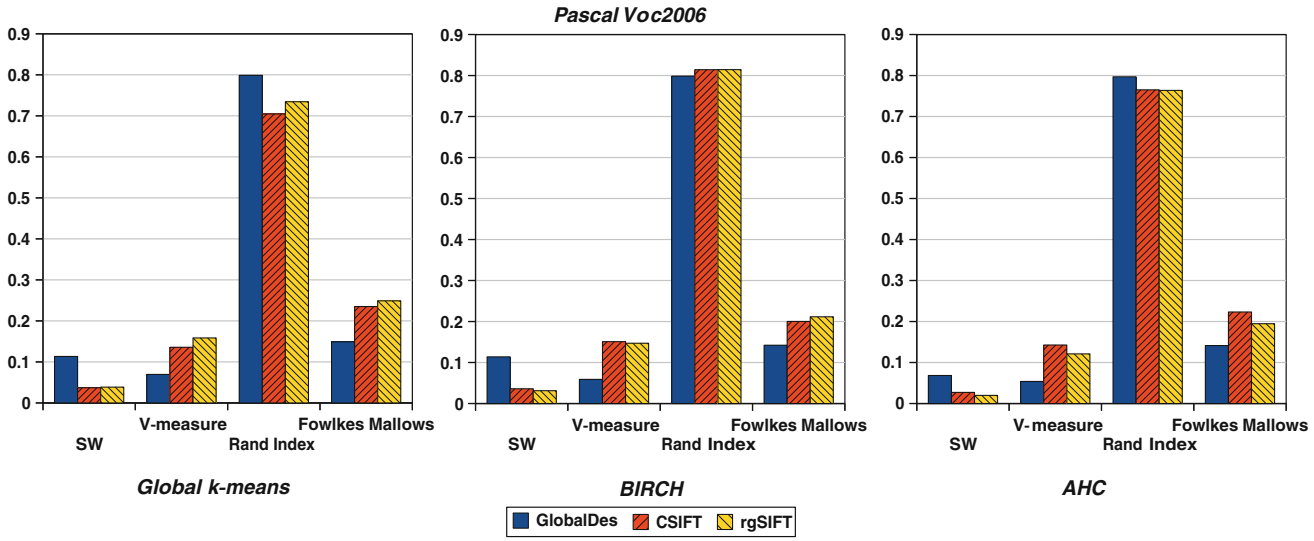

Fig. 13 Global k-means, BIRCH and AHC clustering methods and features descriptors (global descriptor, CSIFT, rgSIFT) comparison using Silhouette Width (SW), V-measure, Rand Index and Fowlkes–Mallows measures using the PascalVoc2006 image database

5.3 Discussion

Concerning the different feature descriptors, the global descriptor is appreciated by the internal measure (numerical evaluation) but the external measures or semantic evaluations appreciate more the local descriptors (CSIFT and rgSIFT). Thus, we can say that CSIFT and rgSIFT descriptors are more compatible with the semantic point of view than the global descriptor at least in our tests.

Concerning the stability of the different clustering methods, k-means and AHC are parameter-free (provided the number k of desired clusters). On the other hand, the results of R-tree, SR-tree and BIRCH vary depending on the value of their input parameters (the maximum and minimum child numbers of each node or the threshold T determining the density of each leaf entry). Therefore, if we want to embed human in the clustering to put semantics on, it is more difficult in the case of k-means and AHC

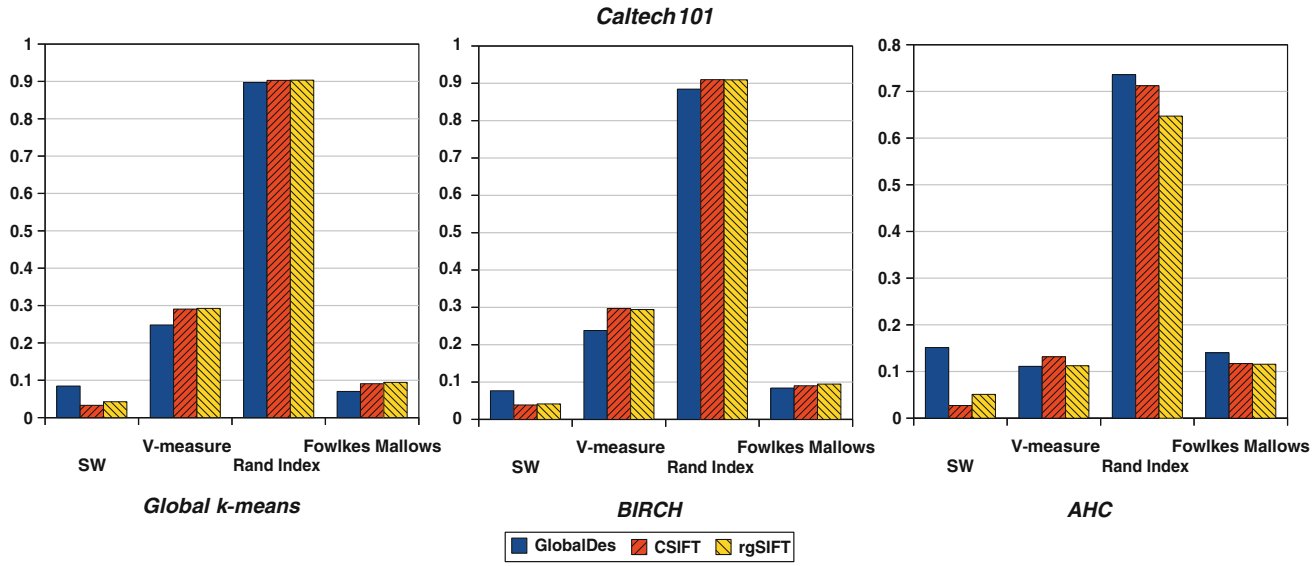

Fig. 14 Global k-means, BIRCH and AHC clustering methods and features descriptors (global descriptor, CSIFT, rgSIFT) comparison using Silhouette Width (SW), V-measure, Rand Index and Fowlkes–Mallows measures using the Caltech101 image database

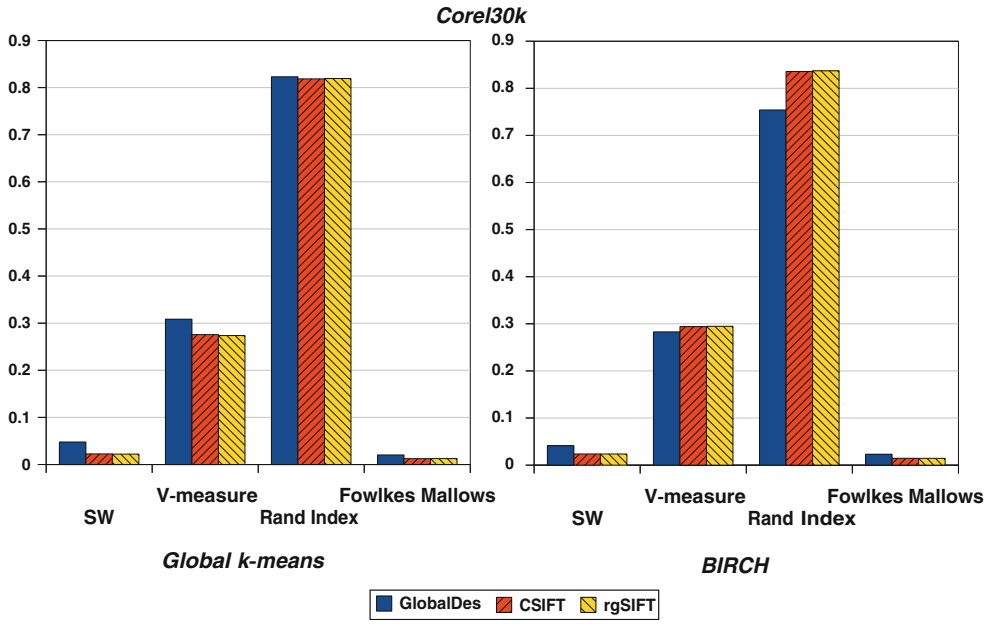

Fig. 15 Global k-means and BIRCH clustering methods and features descriptors (global descriptor, CSIFT, rgSIFT) comparison using Silhouette Width (SW), V-measure, Rand Index and Fowlkes–Mallows measures using the Corel30k image database

because there is no parameter to fix, while in the case of R-tree, SR-tree and BIRCH, if the results are not good from the point of view of the user, we can try to modify the value of the input parameters in order to improve the final results. R-tree and SR-tree have a very unstable behavior when varying their parameters or the number of visual words of the local descriptor. AHC is also much more sensitive to the number of visual words of the local descriptor. Therefore, we consider here that BIRCH is the most interesting method from the stability point of view.

The previous results show a high performance of global k-means, AHC and BIRCH compared to the other methods. BIRCH is slightly better than global k-means and AHC. Moreover, with BIRCH, we have to pass over the data only one time for creating the CF tree while with global k-means, we have to pass the data many times, one time for each iteration. Global k-means clustering is much more computationally complex than BIRCH, especially when the number of clusters k is high because we have to compute the k-means clustering k times (with the number of clusters varying from 1 to k). And the AHC clustering costs much time and memory when the number of elements is high, because of its time complexity $O(N^2 \log N)$ and its space complexity $O(N^2)$. The incremental version [44] of the

AHC could save memory but its computational requirements are still very important. Therefore, in the context of large image databases, BIRCH is more efficient than global k-means and AHC. And because the CF-tree is incrementally built by adding one image at each time, BIRCH may be more promising to be used in an incremental context than global k-means. R-tree and SR-tree give worse results than global k-means, BIRCH and AHC. For all these reasons, we can consider that BIRCH+rgSIFT is the best method in our context.

6 Conclusions and further work

This paper compares both formally and experimentally different clustering methods in the context of multi-dimensional data with image databases of large size. As the final objective of this work is to allow the user to interact with the system in order to improve the results of clustering, it is therefore important that a clustering method is incremental and that the clusters are hierarchically structured. In particular, we compare some tree-based clustering methods (AHC, R-tree, SR-tree and BIRCH) with the global k-means clustering method, which does not decompose clusters into sub-clusters as in the case of tree-based methods. Our results indicate that BIRCH is less sensitive to variations in its parameters or in the features parameters than AHC, R-tree and SR-tree. Moreover, BIRCH may be more promising to be used in the incremental context than the global k-means. BIRCH is also more efficient than global k-means and AHC in the context of large image database.

In this paper, we compare only different hard clustering methods in which each image belongs to only one cluster. But in fact, there is not always a sharp boundary between clusters and an image could be on the boundaries of two or more clusters. Fuzzy clustering methods are necessary in this case. They will be studied in the near future. We are currently working on how to involve the user in the clustering process.

Acknowledgment Grateful acknowledgment is made for financial support by the Poitou-Charentes Region (France).

References

1. Goldberg DE (1989) Genetic algorithms in search optimization and machine learning. Addison-Wesley, Redwood City
2. Frey BJ, Dueck D (2007) Clustering by passing messages between data points. Science 315:972–976
3. Steinbach M, Karypis G, Kumar V (2000) A comparison of document clustering techniques. In: Proceedings of the workshop on text mining, 6th ACM SIGKDD international conference on knowledge discovery and data mining (KDD-2000)
4. Thalamuthu A, Mukhopadhyay I, Zheng X, Tseng GC (2006) Evaluation and comparison of gene clustering methods in microarray analysis. Bioinformatics 22:2405–2412
5. Marinai S, Marino E, Soda G (2008) A comparison of clustering methods for word image indexing. In: The 8th IAPR international workshop on document analysis system, pp 671–676
6. Serban G, Moldovan GS (2006) A comparison of clustering techniques in aspect mining. Studia Universitatis Babes Bolyai Informatica LI(1):69–78
7. Wang XY, Garibaldi JM (2005) A comparison of fuzzy and non-fuzzy clustering techniques in cancer diagnosis. In: Proceedings of the 2nd international conference on computational intelligence in medicine and healthcare (CIMED), pp 250–256
8. Hirano S, Tsumoto S (2005) Empirical comparison of clustering methods for long time-series databases. Lecture notes in artificial intelligence (LNAI) (Subseries of Lecture notes in computer science), vol 3430, pp 268–286
9. Meila M, Heckerman D (2001) An experimental comparison of model-based clustering methods. Mach Learn 42:9–29
10. Hirano S, Sun X, Tsumoto S (2004) Comparison of clustering methods for clinical databases. Inf Sci 159(3-4):155–165
11. Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surv 31:264–323
12. Xu R, Wunsch DII (2005) Survey of clustering algorithms. IEEE Trans Neural Netw 16(3):645–678
13. Plataniotis KN, Venetsanopoulos AN (2000) Color image processing and applications. Springer, Berlin, pp 25–32, 260–275
14. van de Sande KEA, Gevers T, Snoek CGM (2008) Evaluation of color descriptors for object and scene recognition. In: IEEE proceedings of the computer society conference on computer vision and pattern recognition (CVPR), Anchorage, Alaska
15. Mindru F, Tuytelaars T, Van Gool L, Moons T (2004) Moment invariants for recognition under changing viewpoint and illumination. Comput Vis Image Underst 94(1–3):3–27
16. Haralick RM (1979) Statistical and structural approaches to texture. IEEE Proc 67(5):786–804
17. Lee TS (1996) Image representation using 2D Gabor wavelets. IEEE Trans Pattern Anal Mach Intell 18(10):959–971
18. Kuizinga P, Petkov N, Grigorescu S (1999) Comparison of texture features based on gabor filters. In: Proceedings of the 10th international conference on image analysis and processing (ICIAP), pp 142–147
19. Hu MK (1962) Visual pattern recognition by moment invariants. IRE Trans Inf Theory 8:179–187
20. Teague MR (1979) Image analysis via the general theory of moments. J Opt Soc Am 70(8):920–930
21. Khotanzad A, Hong YH (1990) Invariant image recognition by zernike moments. IEEE Trans PAMI 12:489–498
22. Fonga H (1996) Pattern recognition in gray-level images by Fourier analysis. Pattern Recogn Lett 17(14):1477–1489
23. Harris C, Stephens MJ (1998) A combined corner and edge detector. In: Proceedings of the 4th Alvey vision conference, Manchester, UK, pp 147–151
24. Lowe D (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 2(60):91–110
25. Lindeberg T (1994) Scale-space theory: a basic tool for analysing structures at different scales. J Appl Stat 21(2):224–270
26. Zhang J, Marszaek M, Lazebnik S, Schmid C (2007) Local features and kernels for classification of texture and object categories: a comprehensive study. IJCV 73(2):213–238
27. Bay H, Ess A, Tuytelaars T, Van Gool L (2008) SURF: speeded up robust features. CVIU 110(3):346–359
28. Bosch A, Zisserman A, Muoz X (2008) Scene classification using a hybrid generative/discriminative approach. IEEE Trans PAMI 30(4):712–727

29. van de Weijer J, Gevers T, Bagdanov A (2006) Boosting color saliency in image feature detection. IEEE Trans PAMI 28(1):150–156
30. Abdel-Hakim AE, Farag AA (2006) CSIFT: a SIFT descriptor with color invariant characteristics. In: IEEE conference on CVPR, New York, pp 1978–1983
31. Antonopoulos P, Nikolaidis N, Pitas I (2007) Hierarchical face clustering using SIFT image features. In: Proceedings of IEEE symposium on computational intelligence in image and signal processing (CIISP), pp 325–329
32. Sivic J, Zisserman A (2003) Video Google: a text retrieval approach to object matching in videos. In: Proceedings of IEEE international conference on computer vision (ICCV), Nice, France, pp 1470–1477
33. McQueen J (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of 5th Berkeley symposium on mathematical statistics and probability, pp 281–297
34. Zhang B, Hsu M, Dayal U (1999) K-harmonic means—a data clustering algorithm. Technical report HPL-1999-124, Hewlett-Packard Labs
35. Likas A, Vlassis N, Verbeek J (2003) The global k-means clustering algorithm. Pattern Recogn 36(2):451–461
36. Berrani SA (2004) Recherche approximative de plus proches voisins avec contrôle probabiliste de la précision; application à la recherche dimages par le contenu, PhD thesis
37. Kaufman L, Rousseeuw PJ (1990) Finding groups in data: an introduction to cluster analysis. Wiley, New York
38. Ng RT, Han J (2002) CLARANS: a method for clustering objects for spatial data mining. IEEE Trans Knowl Data Eng 14(5): 1003–1016
39. Han J, Kamber M (2006) Data mining: concepts and techniques, 2nd edn. The Morgan Kaufmann, San Francisco
40. Ball G, Hall D (1967) A clustering technique for summarizing multivariate data. Behav Sci 12(2):153–155
41. Gallager RG, Humblet PA, Spira PM (1983) A distributed algorithm for minimum-weight spanning trees. ACM Trans Program Lang Syst 5:66–77
42. Lance GN, Williams WT (1967) A general theory of classification sorting strategies. II. Clustering systems. Comput J 10:271–277
43. Ward JH (1963) Hierarchical grouping to optimize an objective function. J ACM 58(301):236–244
44. Ribert A, Ennaji A, Lecourtier Y (1999) An incremental hierarchical clustering. In: Proceedings of the 1999 vision interface (VI) conference, pp 586–591
45. Zhang T, Ramakrishnan R, Livny M (1996) BIRCH: an efficient data clustering method for very large databases. SIGMOD Rec 25(2):103–114
46. Guha S, Rastogi R, Shim K (1999) ROCK: a robust clustering algorithm for categorical attributes. In: Proceedings of the 15th IEEE international conference on data engineering (ICDE), pp 512–521
47. Guha S, Rastogi R, Shim K (1998) CURE: an efficient clustering algorithms for large databases. In: Proceedings of the ACM SIGMOD international conference on management of data, Seattle, WA, pp 73–84
48. Guttman A (1984) R-tree: a dynamic index structure for spatial searching. In: Proceedings of the ACM SIGMOD international conference on management of data, Boston, MA, pp 47–57
49. Sellis T, Roussopoulos N, Faloutsos C (1987) The R+-tree: a dynamic index for multi-dimensional objects. In: Proceedings of the 16th international conference on very large databases (VLDB), pp 507–518
50. Beckmann N, Kriegel HP, Schneider R, Seeger B (1990) The R*-tree: an efficient and robust access method for points and rectangles. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 322–331
51. White DA, Jain R (1996) Similarity indexing with the SS-tree. In: Proceedings of the 12th IEEE ICDE, pp 516–523
52. Katayama N, Satoh S (1997) The SR-tree: an index structure for high-dimensional nearest neighbor queries. In: Proceedings of the ACM SIGMOD international conference on management of data, Tucson, Arizon USA, pp 369–380
53. Wang W, Yang J, Muntz R (1997) STING: a statistical information grid approach to spatial data mining. In: Proceedings of the 23th VLDB, Athens, Greece. Morgan Kaufmann, San Francisco, pp 186–195
54. Sheikholeslami G, Chatterjee S, Zhang A (1998) WaveCluster: a multi-resolution clustering approach for very large spatial databases. In: Proceedings of the 24th VLDB, New York, NY, pp 428–439
55. Agrawal R, Gehrke J, Gunopulos D, Raghavan P (1998) Automatic subspace clustering of high dimensional data for data mining applications. In: Proceedings of the ACM SIGMOD international conference on management of data, New York, NY, USA, pp 94–105
56. Mclachlan G, Krishnan T (1997) The EM algorithm and extensions. Wiley, New York
57. Ester M, Kriegel H-P, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the 2nd international conference on knowledge discovery and data mining, pp 226–231
58. Hinneburg A, Keim DA (2003) A general approach to clustering in large databases with noise. Knowl Inf Syst 5(4):387–415
59. Ankerst M, Breunig MM, Kriegel HP, Sande J (1999) OPTICS: ordering points to identify the clustering structure. In: Proceedings of the 1999 ACM SIGMOD international conference on management of data, pp 49–60
60. Koskela M (2003) Interactive image retrieval using self-organizing maps. PhD thesis, Helsinki University of Technology, Dissertations in Computer and Information Science, Report D1, Espoo, Finland
61. Carpenter G, Grossberg S (1990) ART3: hierarchical search using chemical transmitters in self-organizing pattern recognition architectures. Neural Netw 3:129–152
62. Shamir R, Sharan R (2002) Algorithmic approaches to clustering gene expression data. Current topics in computational biology. MIT Press, Boston, pp 269–300
63. Rousseeuw PJ (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math 20:53–65
64. Halkidi M, Batistakis Y, Vazirgiannis M (2002) Cluster validity methods: part I and II. SIGMOD Record 31(2):40–45
65. Zhao Y, Karypis G (2001) Criterion functions for document clustering: experiments and analysis. Technical report TR 0140, Department of Computer Science, University of Minnesota
66. Fung BCM, Wang K, Ester M (2003) Hierarchical document clustering using frequent itemsets. In: Proceedings of the SIAM international conference on data mining, pp 59–70
67. Rosenberg A, Hirschberg J (2007) V-measure: a conditional entropy-based external cluster evaluation measure. In: Joint conference on empirical methods in natural language processing and computational language learning, Prague, pp 410–420
68. Rand WM (1971) Objective criteria for the evaluation of clustering methods. J Am Stat Assoc 66(336):846–850
69. Milligan GW, Soon SC, Sokol LM (1983) The effect of cluster size, dimensionality and the number of clusters on recovery of true cluster structure. IEEE Trans PAMI 5:40–47
70. Fowlkes EB, Mallows CL (1983) A method for comparing two hierarchical clusterings. J Am Stat Assoc 78:553–569
71. Mirkin BG (1996) Mathematical classification and clustering. Kluwer, Dordrecht, pp 105–108