Thèse de doctorat

présentée pour obtenir le grade de

Docteur de l'université de La Rochelle

soutenue publiquement le 2 octobre 2013 par

LAI Hien Phuong

Vers un système interactif de structuration des index pour une recherche par le contenu dans des grandes bases d'images

(Towards an interactive index structuring system for content-based image retrieval in large image databases)

sous la direction de
M. Jean-Marc OGIER, Mlle Muriel VISANI et M. Alain BOUCHER

Composition du jury

M. Christophe GARCIA	INSA de Lyon	Rapporteur
Mme. Florence SEDES	Université Paul Sabatier	Rapporteur
M. Matthieu CORD	LIP6/UPMC	Examinateur
M. Bernard MERIALDO	EURECOM	Examinateur
M. Jean-Marc OGIER	Université de La Rochelle	Directeur de thèse
Mlle Muriel VISANI	Université de La Rochelle	Co-encadrant scientifique
M. Alain BOUCHER	IFI-AUF, Hanoi, Vietnam	Co-encadrant scientifique

Ce travail de thèse a été effectué au sein des laboratoires suivants :

- L3i, Université de La Rochelle, France
- IFI, Equipe MSI ; IRD, UMI 209 UMMISCO, Institut de la Francophonie pour l'Informatique, Université Nationale du Vietnam à Hanoi, Vietnam

Contents

List of Figures

List of Tables

Remerciements

Je souhaite adresser ici tous mes remerciements aux personnes qui m'ont apporté leur aide et qui ont ainsi contribué à la réalisation de cette thèse.

Je tiens en premier lieu à remercier Jean-Marc OGIER, Muriel VISANI et Alain BOUCHER, d'avoir accepté de diriger et encadrer cette thèse. Sans eux, je n'aurais pas pu finir ce travail de recherche. Leur confiance, leurs multiples conseils et leur aide précieuse de tous les jours m'ont été indispensables au cours de ces quatre années. J'aimerais également leur adresser mes plus sincères remerciements pour leur grande disponibilité et leur respect sans faille des délais serrés de relecture des documents que je leur ai envoyés.

Je tiens également à remercier les membres du jury d'avoir accepté d'évaluer mon travail de thèse. Je remercie les Professeurs Florence SEDES de l'Université Paul Sabatier et Christophe GARCIA de l'INSA de Lyon, d'avoir accepté d'être les rapporteurs de cette thèse. Leur remarques et suggestions lors de la relecture m'ont permis d'améliorer ce manuscrit ainsi que de préparer au mieux ma soutenance. Je remercie M. Bernard MERIALDO, Professeur de l'Eurecom, d'avoir accepté de présider le jury de cette thèse. Je remercie également M. Matthieu CORD, Professeur de l'Université Pierre et Marie Curie, d'examiner ce travail de thèse et de faire partie du jury.

Ce travail de thèse a été réalisé au sein des laboratoires L3i (Université de La Rochelle, France) et MSI (Institut de la Francophonie pour l'Informatique, Vietnam). Je voudrais remercier les directions de ces laboratoires pour m'avoir accueilli pendant tout ce temps. Mes remerciements sincères vont également à Kathy THEUIL, Dominique LIMOUSIN, Jennifer DE LA CORTE GOMEZ, Isabelle HIRSCH et Brigitte HUDELAINE qui m'ont rendu la vie plus facile dans les différentes démarches administratives du laboratoire L3i et de l'école doctorale S2IM. Je voudrais aussi remercier les autres membres des laboratoires L3i et MSI, qui ont créé un environnement idéal et convivial où j'ai travaillé pendant les quatres années de ma thèse.

Enfin et bien sur, un grand merci à mes parents, à mon frère, à toute ma famille et à mes amis qui me soutiennent toujours et m'encouragent pendant toutes ces années d'études.

Résumé

Cette thèse s'inscrit dans la problématique de l'indexation et la recherche d'images par le contenu dans des bases d'images volumineuses. Les systèmes traditionnels de recherche d'images par le contenu se composent généralement de trois étapes : l'indexation, la structuration et la recherche. Dans le cadre de cette thèse, nous nous intéressons plus particulièrement à l'étape de structuration qui vise à organiser, dans une structure de données, les signatures visuelles des images extraites dans la phase d'indexation afin de faciliter, d'accélérer et d'améliorer les résultats de la recherche ultérieure. A la place des méthodes traditionnelles de structuration, nous étudions les méthodes de regroupement des données (*clustering*) qui ont pour but d'organiser les signatures en groupes d'objets homogènes (*clusters*), sans aucune contrainte sur la taille des clusters, en se basant sur la similarité entre eux.

Afin de combler le fossé sémantique entre les concepts de haut niveau sémantique exprimés par l'utilisateur et les signatures de bas niveau sémantique extraites automatiquement dans la phase d'indexation, nous proposons d'impliquer l'utilisateur dans la phase de clustering pour qu'il puisse interagir avec le système afin d'améliorer les résultats du clustering, et donc améliorer les résultats de la recherche ultérieure.

En vue d'impliquer l'utilisateur dans la phase de clustering, nous proposons un nouveau modèle de clustering semi-supervisé interactif en utilisant les contraintes par paires (*must-link* et *cannot-link*) entre les groupes d'images. Tout d'abord, les images sont regroupées par le clustering non supervisé BIRCH (Zhang et al., 1996). Ensuite, l'utilisateur est impliqué dans la boucle d'interaction afin d'aider le clustering. Pour chaque itération interactive, l'utilisateur visualise les résultats de clustering et fournit des retours au système via notre interface interactive. Par des simples cliques, l'utilisateur peut spécifier les images positives ainsi que les images négatives pour chaque cluster. Il peut aussi glisser les images entre les clusters pour demander de changer l'affectation aux clusters des images. Les contraintes par paires sont ensuite déduites en se basant sur les retours de l'utilisateur ainsi que les informations de voisinage. En tenant compte de ces contraintes, le système réorganise les clusters en utilisant la méthode de clustering semi-supervisé proposée dans cette thèse. La boucle d'interaction peut être répétée jusqu'à ce que le résultat du clustering satisfasse l'utilisateur.

Différentes stratégies pour déduire les contraintes par paires entre les images sont proposées. Ces stratégies sont analysées théoriquement et expérimentalement. Afin d'éviter que les résultats expérimentaux dépendent subjectivement de l'utilisateur humain, un agent logiciel simulant le comportement de l'utilisateur humain pour donner des retours est utilisé pour nos expérimentations. En comparant notre méthode avec la méthode de clustering semi-supervisé la plus populaire HMRF-kmeans (Basu et al., 2004), notre méthode donne de meilleurs résultats.

Abstract

This thesis deals with the problem of Content-Based Image Retrieval (CBIR) on large image databases. Traditional CBIR systems generally rely on three phases: feature extraction, feature space structuring and retrieval. In this thesis, we are particularly interested in the structuring phase, which aims at organizing the visual feature descriptors of all images into an efficient data structure in order to facilitate, accelerate and improve further retrieval. The visual feature descriptor of each image is extracted from the feature extraction phase. Instead of traditional structuring methods, clustering methods which aim at organizing image descriptors into groups of similar objects (*clusters*), without any constraint on the cluster size, are studied.

In order to reduce the "semantic gap" between high-level semantic concepts expressed by the user and the low-level features automatically extracted from the images, we propose to involve the user in the clustering phase so that he/she can interact with the system so as to improve the clustering results, and thus improve the results of further retrieval.

With the aim of involving the user in the clustering phase, we propose a new interactive semi-supervised clustering model based on pairwise constraints between groups of images. Firstly, images are organized into clusters by using the unsupervised clustering method BIRCH (Zhang et al., 1996). Then the user is involved into the interaction loop in order to guide the clustering process. In each interactive iteration, the user visualizes the clustering results and provide feedback to the system via our interactive interface. With some simple clicks, the user can specify the positive and/or negative images for each cluster. The user can also drag and drop images between clusters in order to change the cluster assignment of some images. Pairwise constraints are then deduced based on the user feedback as well as the neighbourhood information. By taking into account these constraints, the system re-organizes the data set, using the semi-supervised clustering proposed in this thesis. The interaction loop can be iterated until the clustering result satisfies the user.

Different strategies for deducing pairwise constraints are proposed. These strategies are theoretically and experimentally analyzed. In order to avoid the subjective dependence of the clustering results on the human user, a software agent simulating the behaviour of the human user for providing feedback to the system is used in our experiments. By comparing our method with the most popular semi-supervised clustering HMRF-kmeans (Basu et al., 2004), our method gives better results.

Chapter 1

Introduction

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.

Content-Based Image Retrieval (CBIR) refers to the techniques for solving the image retrieval problem, by using the visual contents of images rather than the other metadata associated with the images. Figure 1.1 presents the schema of a traditional CBIR system which generally relies on three principal phases. The first phase, feature extraction (generally performed offline), extracts the visual feature descriptors from all the images in the database. Visual features are usually low-level descriptors of the contents of the image, describing its color, shape, texture, etc. The second phase, feature space structuring (generally performed offline), aims at organizing the feature descriptors of all images into an efficient index data structure used for further retrieval. Feature space structuring methods are normally called indexing methods. Therefore, in this thesis, the term indexing and feature space structuring are interchangeable and are used to refer to the second phase. The third phase, retrieval (generally performed online), uses a similarity measure for searching the images which are the most similar to the query image in the index data structure. With the development of many large image database, the second phase of feature space structuring is required for facilitating, accelerating and improving the further retrieval phase.

In this thesis, we are particularly interested in the structuring phase while considering that the feature extraction phase is previously completed and the image feature descriptors are available. Feature space structuring methods which are currently used in traditional CBIR systems can be classified into space partitioning methods and data partitioning methods.

For space partitioning methods (Quadtree [28, 116], K-D-tree [14], K-D-B-tree [112], grid file [98], LSD tree [48], etc.), the feature space is generally partitioned into disjoint cells of fairly similar cardinality (in terms of number of objects per cell) or into regular disjoint cells (in terms of size of cell) by different hyperplanes. As the data classes may be unbalanced or the data distribution may not be homogeneous among the different classes, the resulting index may include dissimilar points in a same cell and similar points in different cells. In the retrieval phase, the user generally wants to retrieve images which are similar to the query image, and the

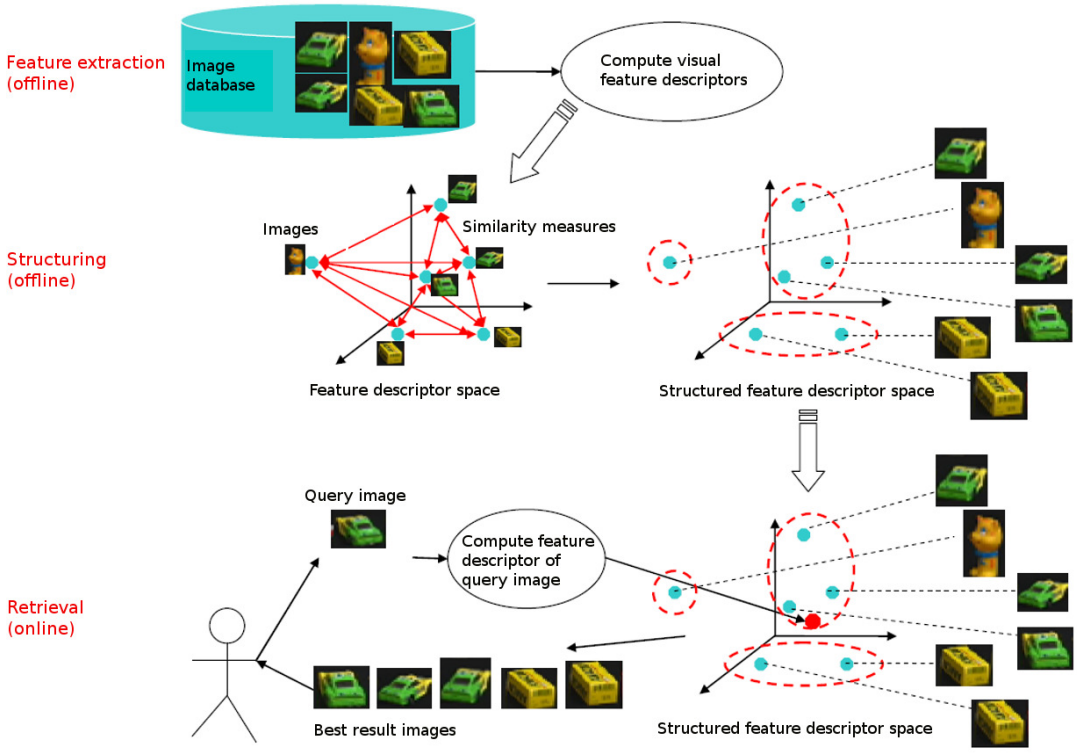


Figure 1.1 – Traditional CBIR system.

indexed structure produced by space partitioning methods may be not optimal for retrieval. Moreover, the partition of the whole feature space into cells may result in many empty or almost empty cells, especially in the case of high dimensionality which is generally the case of image feature descriptors, therefore leading to poor storage utilization of the resulting indexed structure.

For data partitioning methods (R-tree family [13, 42, 117], SS-tree [139], SR-tree [60], X-tree [16], etc.), data objects are partitioned into regions based on their distribution in the feature space. Contrary to space partitioning methods, these regions can be overlapping. No empty cells are produced, and therefore, the storage utilization is improved. However, the limitations on the cardinality of the cells remain. It is difficult to tune the cardinality parameters, especially when groups of similar objects are unbalanced, *i.e.* composed of very different numbers of objects. Therefore, the resulting index is generally non-optimal for retrieval.

Clustering, a part of machine learning techniques, can also be used for the structuring phase of the CBIR systems. In fact, clustering methods aim at splitting an object data set into groups (clusters) of objects which are similar in the feature space, with no constraint on the cluster size. Our claim is that using clustering methods instead of methods which are currently used in the structuring phase may result in indexed structure more adapted to the retrieval of high dimensional and unbalanced data. Indeed, when using clustering in the structuring phase, data objects are organized in a "flat" or a "hierarchical" structure of clusters. Therefore, in the retrieval phase, we only have to navigate to the clusters which are similar to the input feature descriptor and compare the input descriptor with a small number of feature descriptors included in these clusters. Clustering is thus suitable for

indexing large databases. Moreover, as there is no limitation on the cardinality of the clusters, clustering can be used for indexing both balanced or unbalanced data. Furthermore, when using clustering methods, the empty cell problem is avoided. Based on these advantages of clustering methods compared to structuring methods which are currently used in CBIR systems, in our work, the clustering methods are used for structuring objects in the feature space.

As feature descriptors generally capture low-level information of the image such as color, shape or texture, there is a "semantic gap" between high-level semantic concepts expressed by the user and these low-level features. The clustering results and therefore the retrieval results are generally different from the wishes of the user. For example, in Figure 1.1, the user submits a query image of a car toy with the hope of receiving all the images of toys (car and doll). But the system, based only on low-level features, returns a tea box image before the image of the doll. In this thesis, we propose to involve the user in the clustering phase so that the user can interact with the system in order to improve the clustering results, and therefore improve the performance of the retrieval phase. The idea of the proposed approach is presented in Figure 1.2. The user, after observing the clustering results, gives feedback to the system in order to guide the re-clustering phase. The system then re-organizes the data set by using not only the similarity between objects, but also the feedback given by the user in order to reduce the effect of the semantic gap problem. The interaction loop can be iterated until the clustering result satisfies the user.

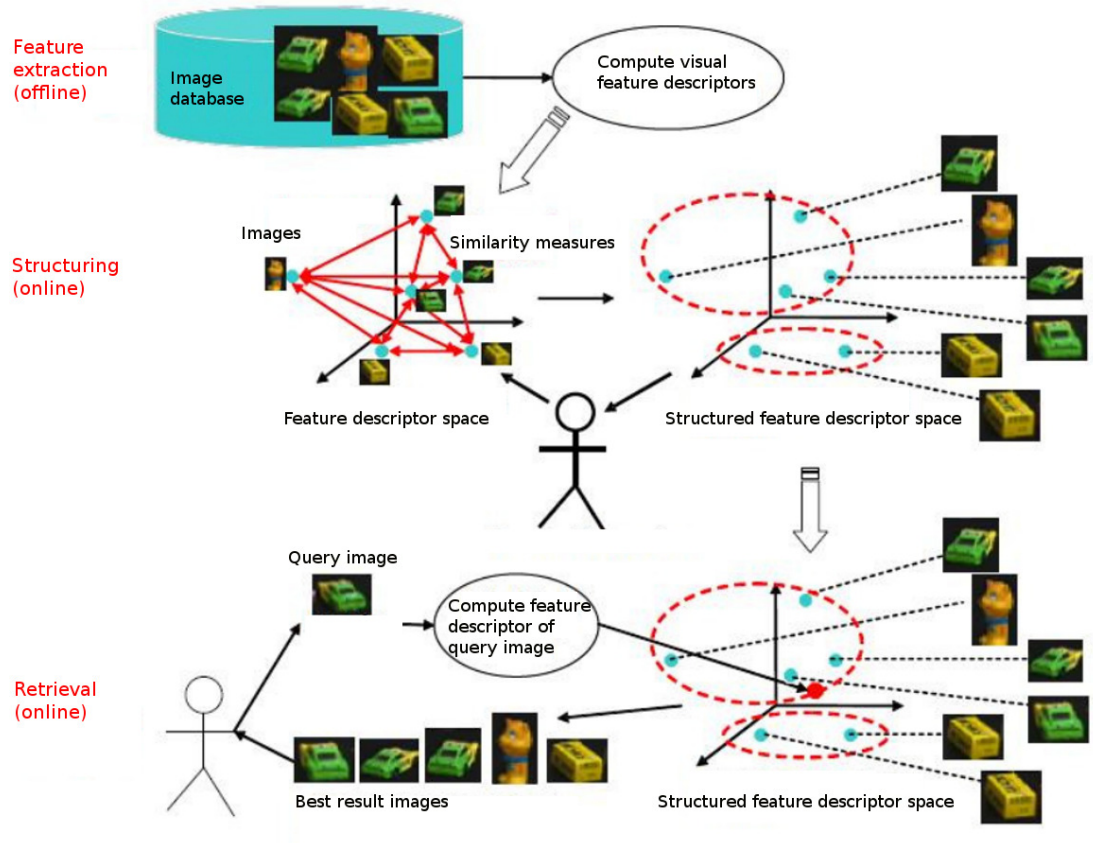


Figure 1.2 – CBIR system involving the user's interaction in the structuring phase.

In the case of large image database indexing, we may be interested in unsupervised clustering or semi-supervised clustering. While no information about the ground truth is provided in the case of unsupervised clustering, a limited amount of supervised information is provided in the case of semi-supervised clustering. The provided supervised information may consist of class labels (for some objects) or pairwise constraints (must-link or cannot-link) between objects.

In this thesis, we assume that the user has no prior knowledge about the image database. Instead, the user observes the clustering results during each interactive iteration and gives feedback in order to guide the re-clustering phase. Therefore, an unsupervised clustering method is suitable to be used for the initial clustering, when no supervised information is available yet. Then, after receiving the user feedback in each interactive iteration, a semi-supervised clustering can be used for the re-clustering process.

Based on a deep study of the state of the art of different unsupervised clustering methods as well as semi-supervised clustering approaches, in this thesis we propose a new interactive semi-supervised clustering model involving the user in the clustering phase in order to improve the clustering results. From the analysis of different unsupervised clustering methods, we chose to experiment some methods which appear to be the most suitable to be used in an incremental context involving the user in the clustering stage. The hierarchical BIRCH unsupervised clustering [145] which gives the best performance from these experiments is then chosen to be used as the initial clustering. Then, an interactive loop in which the user provides the feedback to the system and the system re-organizes the image database using our semi-supervised clustering is iterated until the clustering results satisfies the user. As the user has no prior knowledge about the image database, it is difficult for him/her to label the clusters or the images in the clusters using classes. Therefore, we provide to the user an interactive interface allowing him/her to give feedback to the system. Using this interface, the user can click for identifying relevant or non-relevant images for each cluster, as well as drag and drop images from a cluster to another. Pairwise constraints between groups of similar images are then deduced from the user feedback in order to be used as supervised information for the semi-supervised clustering method proposed in this thesis. Different strategies for deducing pairwise constraints from the user feedback are proposed and experimentally compared in this thesis. In order to avoid the subjective dependence of the experimental results on the human user and in order to automatize the interactive experiments, we propose a software agent simulating the behaviour of the human user when interacting with the system. Experimental results show that our proposed semi-supervised clustering outperforms the semi-supervised HMRF-kmeans clustering, which gives the best performance in our experimental comparison of different semi-supervised clustering methods.

This dissertation is organized in two parts: general state-of-the-art (Chapters 2 and 3) and interaction (Chapters 4 and 5). The contents of these chapters is as follows:

Chapter 2 presents a brief state of the art of different techniques used in a traditional content-based image retrieval system.

Chapter 3 proposes a survey of unsupervised clustering methods and analyzes the advantages and disadvantages of different methods in the incremental context involving the user in the clustering of huge masses of data. An experimental comparison of different unsupervised clustering methods using different image databases of increasing sizes (Wang, PascalVoc2006, Caltech101, Corel30k) is also studied in this chapter.

Chapter 4 presents and analyzes different semi-supervised clustering methods. We propose in this chapter a framework to implement any semi-supervised method in the interactive context. An experimental comparison of some well known semi-supervised methods using a software agent simulating the behaviour of the human user for giving feedback is also presented.

Chapter 5 proposes a new interactive semi-supervised clustering model using pairwise constraints deduced from the user feedback as supervised information. Different strategies for deducing pairwise constraints from the user feedback are proposed and experimentally compared in this chapter. This chapter also presents an experimental comparison of our proposed semi-supervised clustering method with the HMRF-kmeans semi-supervised clustering which gives the best performance from the experiments in Chapter 4.

Chapter 6 gives some conclusions and future directions for this thesis.

Part I

State of the art

Chapter 2

Traditional Content-based image retrieval

2.1 Introduction

The term Content-Based Image Retrieval (CBIR) seems to be used the first time in the literature in 1992 by Kato [61] in order to describe his experiments of an image retrieval system based on color and shape features. CBIR is used to refer to the process which uses visual contents to search images from the database according to the user's queries. Visual contents, normally called features, usually describe color, shape, texture or any other kind of information corresponding to the properties of the images. These properties are usually encoded into feature vectors and can be automatically extracted from the images themselves. In recent years, with the development of many large image databases, the main challenge of a CBIR system is how to locate an image in these huge masses of data. Many CBIR systems have been developed, such as QBIC [30], Photobook [102], Visualseek [122], Netra [83], etc. See [23, 132] for more complete surveys of CBIR systems.

Traditional CBIR systems generally rely on three phases: the first phase consists in extracting the feature vectors from all the images in the database, the second phase consists in structuring them into an efficient index data structure, while the last phase is to efficiently search in the indexed feature space in order to find the most similar images to the query image. Methods which are used to extract the characteristics of each image in the database are called feature extraction methods. The characteristics of the images are in general presented in the form of feature vectors. They are usually low-level features commonly based on shape, texture, color properties of images. Feature space structuring methods (normally called indexing methods) aim at organizing the feature vectors of all images in the database into an efficient index data structure that facilitates further retrieval. In the retrieval phase, the user first submits a query image to the system. The system then computes the feature vector of the query image and uses a similarity measure for finding and returning to the user the images which are most similar to the query image in the structured feature space. These images are usually returned in decreasing order of similarity.

In the present chapter, we present a brief state of the art of different techniques used in content-based image retrieval. In Section 2.2, we analyze current feature extraction approaches. Section 2.3 presents an overview of different indexing meth-

ods. And the retrieval strategy is described in Section 2.4. Finally, a discussion is given in Section 2.5.

2.2 Feature extraction

In content-based image retrieval, feature extraction is intended to transform each input image into a set of features (normally in the form of feature vector) describing the image content. In general, these are low-level features automatically extracted based on different properties (color, texture, shape, etc.) of the image itself. Before calculating features, images could be preprocessed (color space conversion, denoising, quantization, etc.). Good features should carry enough information about the image and should well relate to the user perception. Finding relevant features representing visual content of images in a large database is still a challenging task. It depends on the image database, on the application and also on the user's wishes. Many research focuses on this problem. We can divide feature extraction approaches into three main types: global approaches, local approaches and spatial approaches.

- With regard to 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.

- Instead of calculating a signature on the entire image, local approaches compute color, texture and/or shape features for segmented regions or for region around interest points of each image. Thus, each image is characterized by a set of local signatures (one signature for each interest point, for instance).

- Regarding to the spatial approaches, 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 between regions. Thus, the signature of an image contains descriptions of visual objects and spatial relationships between them.

2.2.1 Global approaches

Global approaches aim at extracting features over the entire image instead of segmented regions. Global features are generally calculated based on color, texture or shape properties of the image. These three kinds of features can be either calculated separately or combined to get a more complete signature. We present in the next three paragraphs some basic global features representing the visual contents of image.

2.2.1.1 Color

Color features are among the most widely used features in content-based image retrieval. The color of an image could be represented by color histograms [130], color moments [92], color correlograms [54], etc.

Color histogram Color histograms are the most widely used representation of color features; they are used in many content-based image retrieval systems such as QBIC [133], MARS [133], PicToSeek [133], etc. A color histogram is in general a n-dimensional vector $[h_1, ..., h_n]$ in which h_j is the number of pixels having the color j in the image and n is the number of color values. A color histogram represents thus the distribution of colors in an image. As an image can be represented in many color spaces (RGB, HSV, L*a*b*, L*u*v*,...) [105] which are related to each other by mathematical formulas, the color histogram can be built for any color space [130]. In the case of gray scale images, the term intensity histogram may be used instead of color histogram for representing the frequencies of occurrence of each gray level. In the case of color images, the histogram can be calculated on a specific channel or on each channel of the selected color space. The histogram of an image in a specific color space could also capture the joint probabilities of the intensities of two or three color channels (R and/or G and/or B of the RGB color space and similarity for other color spaces). A quantization is often used to divide each channel of the color space into a number of ranges of similar intensity values (called bins) for reducing the dimension of the histogram and also the computational cost.

We present here some most popular color histograms based on different color spaces:

- *RGB color space*: The RBG histogram is the most commonly used. In [130], Van de Sande *et al.* define the RBG histogram as a combination of three separate histograms on each color channels R, G and B. In [58], Jeong quantizes each channel R, G and B into 8 ranges and computes the RGB histogram having 8x8x8=512 bins.

- *Normalized RGB color space*: The normalized RGB color space [130] is transformed from the RGB space as in Equation (2.1):

$$\begin{pmatrix} r \\ g \\ b \end{pmatrix} = \begin{pmatrix} \frac{R}{R+G+B} \\ \frac{G}{R+G+B} \\ \frac{B}{R+G+B} \end{pmatrix} \tag{2.1}$$

  As $r + g + b = 1$, only two components (r and g) are used to combine the rghistogram [130] of this color space. By using the normalized RGB space, the feature vectors are less sensible to the luminance changes.

- *Opponent color space*: In [130], the opponent histogram is the combination of three histograms on each channel of the opponent color space:

$$\begin{pmatrix} O_1 \\ O_2 \\ O_3 \end{pmatrix} = \begin{pmatrix} \frac{R-G}{\sqrt{2}} \\ \frac{R+G-2B}{\sqrt{6}} \\ \frac{R+G+B}{\sqrt{3}} \end{pmatrix} \tag{2.2}$$

  The opponent color space has three components in which O_3 represents the luminance information, O_1 is the red-green channel and O_2 is the blue-yellow channel.

- *HSV color space*: The HSV color space [105] consists of three channels: (1) Hue represents the color in its pure form varying from red through yellow, green, cyan, blue, magenta and back to red; (2) Saturation defines if the color is grey or pure; (3) Value represents the brightness of the color.

$$\begin{aligned} H &= cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right\} \\ S &= \frac{max(R,G,B)-min(R,G,B)}{max(R,G,B)} \\ V &= \frac{max(R,G,B)}{255} \end{aligned} \tag{2.3}$$

  In this color space, hue represents most of the information about the color. Therefore, in [58], the H channel is quantized into 18 bins while S and V are quantized into 3 bins to create a HSV histogram having 18x3x3=162 bins. Van de Sande *et al.*, in their article [130], use only the Hue histogram.

Color moments Under the assumption that the color in an image may follow a certain probability distribution, the moments of the color distribution could be used as color features of the image.

Stricker and Orengo [124] define different color moments for each color channel in an image.

- The first color moment (mean) of the i^{th} color channel is defined as follows:

$$M_i^1 = \frac{1}{N}\sum_{j=1}^{N} I_{i,j} \tag{2.4}$$

  where $I_{i,j}$ is the value of the i^{th} color channel of the j^{th} pixel and N is the total number of pixel in the image.

- Then, the h^{th} moment of the i^{th} color channel is defined as:

$$M_i^h = \left(\frac{1}{N}\sum_{j=1}^{N}(I_{i,j}-M_i^1)^h\right)^{\frac{1}{h}} \tag{2.5}$$

Since most of the information is concentrated in the low-order moments, Stricker and Orengo [124] only use the first moment (mean), the second moment (standard deviation) and the third moment (skewness) as features. Since 3 moments are used for each color channel, an image is thus characterized by a feature vector of 9 moments.

Note that color histograms and color moments described above do not contain any information about the spatial layout of the color in an image. Mindru *et al.* [92] define generalized color moments including spatial information as follows:

$$M_{pq}^{abc} = \iint x^p y^q [R(x,y)]^a [G(x,y)]^b [B(x,y)]^c dxdy \tag{2.6}$$

where M_{pq}^{abc} is the generalized color moment of order $p+q$ and degree $a+b+c$. $R(x,y)$, $G(x,y)$ and $B(x,y)$ are respectively the R, G, B values of the pixel at the position (x,y) of the image. Note that in Equation (2.6), the generalized color moments are defined in the RGB color space. Color moments of any other space could be computed in a similar way. We can see that generalized color moments of degree 0 (M_{pq}^{000}) do not contain any photometric information while the generalized color moments of order 0 (M_{00}^{abc}) do not contain any spatial information. The generalized color moments of higher order and higher degree contain simultaneously color and spatial information.

With different values of order and degree, we can create a large number of generalized color moments. In general, generalized color moments up to the first order and the second degree are used. There are nine possible combinations of the degree ($(a,b,c) \in \{(0,0,0), (1,0,0), (0,1,0), (0,0,1), (1,1,0), (0,1,1), (1,0,1), (2,0,0), (0,2,0), (0,0,2)\}$) and three possible combinations of the order ($(p,q) \in \{(0,0), (1,0), (0,1)\}$). The color moment feature vector has thus 27 dimensions.

By properly combining different moments, Mindru *et al.* [92] computed different color moment invariants which are robust towards photometric changes (or illuminance changes) in one, two or three color channels of the color space. A feature vector could be created by using different color moment invariants.

Color correlogram The color correlograms [54] were proposed to include the spatial correlation of pairs of colors in an image. Like color histograms and color moments, the color correlograms could be used for any kind of color space.

Assume that the colors in an image I are quantized into m colors c_1,...,c_m. The distance between two pixels $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ of the image I is defined as:

$$|p_1 - p_2| = max\{|x_1 - x_2|, |y_1 - y_2|\} \tag{2.7}$$

Let $I_{c_i} = \{p \in I \mid I(p) = c_i\}$ represent the set of pixels of the image I having the color c_i. The color correlogram is defined for a pair of colors $c_i, c_j \in \{c_1, ..., c_m\}$ and a distance $k \in \{1, ..., d\}$ (d is fixed a priori) as:

$$\gamma_{c_i,c_j}^{(k)}(I) = \Pr_{p_1 \in I_{c_i}, p_2 \in I} \left[p_2 \in I_{c_j} / |p_1 - p_2| = k\right] \tag{2.8}$$

Given any pixel of color c_i, $\gamma_{c_i,c_j}^{(k)}(I)$ specifies the probability of finding a pixel of color c_j at a distance k from the given pixel. By considering all possible combinations of color pairs in $\{c_1, ..., c_m\}$ and all distances $k \in \{1, ..., d\}$, the color correlogram feature vector has a large size of $O(m^2 d)$. The *autocorrelogram*, a simple version of the correlogram, is instead often used for reducing the dimension of the feature vector. Instead of capturing the spatial correlation between each pair of colors, the color autocorrelogram characterizes only the spatial correlation between identical colors ($\gamma_{c_i,c_i}^{(k)}$, $c_i \in \{c_1, ..., c_m\}$). The dimension of the color autocorrelogram is therefore $O(md)$.

2.2.1.2 Texture

Texture is a concept that is closely related to the human perception. We can define a textured region as a non-uniform intensity region which can be perceived

(a) Regular texture (b) Irregular texture (c) Stochastic texture

Figure 2.1 – Examples of textures (*Source:* `http://www.cgtextures.com`)

as homogeneous at some large spatial scale. Texture features are also important visual features of images. For instance, tigers and cheetahs cannot be distinguished only by color or shape, texture features are needed in this case.

Textures can be arranged from regular to stochastic textures. In the regular case, a texture can be interpreted as a repetition of some basic patterns. In the stochastic case, there is no basic pattern, the pixels are randomly organized. Some examples of regular, irregular and stochastic textures are respectively shown in Figures 2.1a, 2.1b, 2.1c.

Many texture representation methods are presented in the literature [44, 52, 87, 129]. We can divide them into structural and statistical methods. Structural methods characterize texture by identifying primitives or basic patterns (such as circles, hexagons, ...) and their placement rules to form the texture. These methods are more effective for regular textures (e.g. brick wall image). Some methods of this kind are described in [44,88,128,134]. Statistical methods characterize texture by a set of statistical feature vectors, based on properties such as contrast, entropy, etc. Some methods of this kind are described in [44,45]. We describe in this section some of the most frequently used methods.

Gray-Level Co-occurrence matrices [45] One of the most famous and widely used texture features are gray-level co-occurrence matrices. Before computing co-occurrence matrices, a texture image of size nxm is quantized into G gray-levels. A co-occurrence matrix $P_{d,\theta}$ of size GxG is computed for representing the frequency of transition between pairs of gray-level according to a distance d and a direction θ. The entry $P_{d,\theta}(i,j)$ of this matrix represents the number of occurrences of a pixel of gray-level g_j which is at a distance d and a direction θ from a pixel of gray-level g_i. Figure 2.2 represents a transition from the gray-level g_i to the gray-level g_j with a distance $d = 2$ and a direction $\theta = \pi/4$.

Many co-occurrence matrices can be computed for each image according to different distances (d = 1,2,3,...) and different directions (θ = 0, $\pi/4$, $\pi/2$, $3\pi/4$, ...). These matrices represent certain information about the spatial distribution of different gray-levels of the texture. For each co-occurrence matrix, Haralick [45] proposed 14 statistical features characterizing texture properties. Some main features extracted from the $P_{d,\theta}$ co-occurrence matrix are described in Table 2.1.

where:

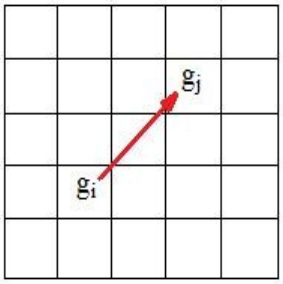


Figure 2.2 – Transition from gray-level g_i to gray-level g_j with a distance $d = 2$ and a direction $\theta = \pi/4$.

Energy	$\sum_i \sum_j p_{d,\theta}^2(i,j)$
Entropy	$-\sum_i \sum_j p_{d,\theta}(i,j) \log(p_{d,\theta}(i,j))$
Contrast (Inertia)	$\sum_i \sum_j (i-j)^2 p_{d,\theta}(i,j)$
Difference moment	$\sum_i \sum_j \frac{1}{1+(i-j)^2} p_{d,\theta}(i,j)$
Variance	$\sum_i \sum_j (i-\mu)^2 p_{d,\theta}(i,j)$
Maximum probability	$\max_{i,j} p_{d,\theta}(i,j)$
Correlation	$\sum_i \sum_j \frac{ijp_{d,\theta}(i,j)-\mu_x\mu_y}{\sigma_x\sigma_y}$

Table 2.1 – Some features extracted from Haralick's gray-level co-occurrence matrices.

- $p_{d,\theta}(i,j)$ is the normalized value corresponding to the $P_{d,\theta}(i,j)$ value of the matrix.
- μ is the mean of the matrix.
- μ_x, μ_y, σ_x, σ_y are the means and standard deviations of the row and column sums of the matrix.

Gabor filters [67, 74] Based on the principle that a texture is a repetition of primitives with a certain frequency, the idea of this method is to use a set of Gabor filters to analyze the structure of the texture at different scales (frequencies) and orientations.

A Gabor filter is the product of a Gaussian kernel function and a complex sinusoid. The Gabor function is defined as follows:

- Complex:

$$g(x,y) = e^{-\frac{(x'^2+\gamma^2 y'^2)}{2\sigma^2}} \times e^{i(\frac{2\pi x'}{\lambda}+\varphi)} \tag{2.9}$$

$$x' = x\cos\theta + y\sin\theta \tag{2.10}$$

$$y' = -x\sin\theta + y\cos\theta \tag{2.11}$$

- Real part:

$$g_r(x,y) = e^{-\frac{(x'^2+\gamma^2 y'^2)}{2\sigma^2}} \times \cos(\frac{2\pi x'}{\lambda}+\varphi) \tag{2.12}$$

- Imaginary part:

$$g_i(x,y) = e^{-\frac{(x'^2+\gamma^2 y'^2)}{2\sigma^2}} \times \sin(\frac{2\pi x'}{\lambda} + \varphi) \tag{2.13}$$

Where:

- σ is the standard deviation of the Gaussian factor and γ is the spatial aspect ratio specifying the ellipticity of the Gabor function.
- λ represents the wavelength and $\frac{1}{\lambda}$ represents the spatial frequency of the sinusoidal factor.
- θ specifies the orientation of the Gabor filter.
- φ is the phase offset.

To analyze the texture features, the input image is convolved with a set of Gabor filters with different frequencies and orientations. An input image $I(x,y)$, where $(x,y) \in \Omega$ (Ω is the set of pixels in the image) is convolved with a Gabor function $g(x,y)$ to obtain two Gabor feature images ($r_r(x,y)$ as real part and $r_i(x,y)$ as imaginary part) as follows:

$$r_r(x,y) = (I * g_r)(x,y) = \iint_{\Omega} I(\xi,\eta) g_r(x-\xi, y-\eta) d\xi d\eta \tag{2.14}$$

$$r_i(x,y) = (I * g_i)(x,y) = \iint_{\Omega} I(\xi,\eta) g_i(x-\xi, y-\eta) d\xi d\eta \tag{2.15}$$

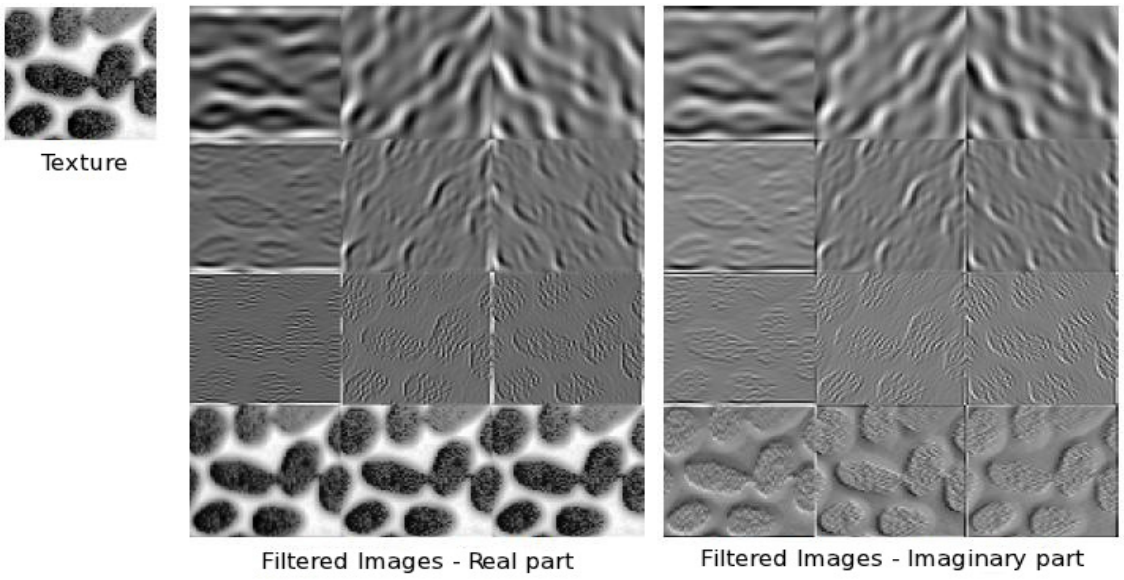


Figure 2.3 – Filtered images using Gabor filters according to 4 frequencies (vertical axis, the frequencies increase from top to bottom) and 3 orientations (horizontal axis, from left to right $\theta = 0$, $\theta = \pi/3$, $\theta = 2\pi/3$).

Thus, for each texture image, we obtain a set of Gabor feature images in real part and imaginary part. Figure 2.3 shows the response images of an input texture according to 3 frequencies and 4 orientations; in total we have 24 response images ($3 \times 4 = 12$ images in real part and $3 \times 4 = 12$ image in imaginary part). The feature vector of the input image is then created by combining the statistics such as mean and standard deviation [144] calculated for each response image. For example, we calculate the mean μ and the standard deviation σ for each response image in Figure 2.3, the feature vector has thus $24 \times 2 = 48$ dimensions $f = (\mu_0, \sigma_0, ..., \mu_{23}, \sigma_{23})$.

2.2.1.3 Shape

(a) A ball (b) A box

Figure 2.4 – Examples of objects with similar color and texture but having different shapes.

Shape features have widely been used in many CBIR systems [133]. This kind of features is useful for object and region description. Figures 2.4a and 2.4b show two images containing two objects having similar colors and textures but their shapes are very different. In this case, the color or texture criteria do not allow us to distinguish these two images. Shape features are useful in this kind of situation.

Shape descriptors are often extracted after segmenting the images into objects and regions. As image segmentation is also a difficult task, shape features are in general less developed than color and texture features. Many shape feature descriptors are presented in the literature [93]. We can divide them into two groups: contour-based descriptors and region based descriptors. While contour-based descriptors (such as Fourier-based shape descriptor [31, 141, 143], ...) represent an object or region by its outer boundary, the region-based descriptors (such as Hu moments [53], Zernike moments [125], medial axis [75], ...) represent the properties of entire region. A good shape feature should be invariant to translation, rotation and scale. In this section, we describe the shape descriptors that are most widely used in content-based image retrieval systems.

Fourier descriptors [31, 141, 143] Fourier descriptor is the most widely used among contour-based descriptors for describing shape information of images. In general, a shape signature is a one-dimensional function representing the boundary coordinates of a shape. By applying a Fourier transform on a shape signature, we obtain the Fourier descriptors.

Let us assume that the shape boundary coordinates are $(x(t), y(t))$, $t = 0, 1, ..., N-1$ where N is the number of sampled points in the boundary of the shape. Boundary coordinates can be represented by different shape signatures:

- Complex coordinates function:

$$s(t) = z(t) = x(t) + iy(t) \tag{2.16}$$

- Shifted complex coordinates function:

$$s(t) = z(t) = [x(t) - x_c] + i[y(t) - y_c] \tag{2.17}$$

where (x_c, y_c) is the centroid of the shape:

$$\begin{aligned} x_c &= \frac{1}{L}\sum_{t=0}^{L-1} x(t) \\ y_c &= \frac{1}{L}\sum_{t=0}^{L-1} y(t) \end{aligned} \tag{2.18}$$

- Centroid distance function:

$$s(t) = r(t) = \sqrt{[x(t) - x_c]^2 + [y(t) - y_c]^2} \tag{2.19}$$

The discrete Fourier transform of a given shape signature $s(t)$ is given by:

$$a_n = \frac{1}{N}\sum_{t=0}^{N-1} s(t) exp(\frac{-i2\pi nt}{N}), n = 0, 1, ..., N-1 \tag{2.20}$$

The coefficients a_n, $n = 0, 1, ..., N-1$ form the Fourier descriptors of the shape. Fourier descriptors are simple to implement but require a pre-processing step to extract the contour from the images. Their main drawback is the lack of description of occlusions in the shape.

Hu moments [53] Hu [53] proposed seven moments which are useful to capture the shape characteristics. These moments are invariant to translation, rotation and scaling. These 7 moment are calculated as follows:

$$\begin{aligned} I_1 &= \eta_{20} + \eta_{02} && (2.21)\\ I_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 && (2.22)\\ I_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 && (2.23)\\ I_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 && (2.24)\\ I_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ &\quad +(3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] && (2.25) \end{aligned}$$

$$
\begin{aligned}
I_6 &= (\eta_{20}-\eta_{02})[(\eta_{30}+\eta_{12})^2-(\eta_{21}+\eta_{03})^2] \\
&\quad +4\eta_{11}(\eta_{30}+\eta_{12})(\eta_{21}+\eta_{03}) \qquad (2.26)\\
I_7 &= (3\eta_{21}-\eta_{03})(\eta_{30}+\eta_{12})[(\eta_{30}+\eta_{12})^2-3(\eta_{21}+\eta_{03})^2] \\
&\quad -(\eta_{30}-3\eta_{12})(\eta_{21}+\eta_{03})[3(\eta_{30}+\eta_{12})^2-(\eta_{21}+\eta_{03})^2] \qquad (2.27)
\end{aligned}
$$

Where η_{ij} is the normalized central moment calculated based on the central moment μ_{ij} as follows:

$$
\begin{aligned}
\eta_{ij} &= \frac{\mu_{ij}}{\mu_{00}^{(1+\frac{i+j}{2})}} \qquad (2.28)\\
\mu_{ij} &= \sum_x \sum_y I(x,y)(x-x_0)^i(y-y_0)^j \qquad (2.29)
\end{aligned}
$$

Where:

- (x_0, y_0) is the centroid of the image.
- $I(x,y)$ represents the gray level of the image at the position (x,y).

Zernike moments [125] To compute the Zernike moments [125], the Cartesian coordinates (x,y) of the image are first transformed into polar coordinates (ρ,θ) inside a unit disk (the centre of the image is the origin of the unit disk, $x^2+y^2 \leq 1$) as follows:

$$
\begin{aligned}
\rho &= \sqrt{x^2+y^2} \qquad (2.30)\\
\theta &= \arctan\left(\frac{y}{x}\right) \qquad (2.31)
\end{aligned}
$$

The complex Zernike moments A_{mn} of an image $I(x,y)$ are built based on the set of complex Zernike polynomials $V_{mn}(x,y)$ as follows:

$$
A_{mn} = \frac{m+1}{\pi} \sum_x \sum_y I(x,y)[V_{mn}(x,y)]^* \qquad where\ x^2+y^2 \leq 1 \qquad (2.32)
$$

where:

- $m \geq 0$ defines the order of the moment, n is an integer, $|n| \leq m$ and $m-|n|$ is even.
- $I(x,y)$ represents the gray level of the image at the position (x,y).
- $[V_{mn}(x,y)]^*$ is the conjugate of the complex Zernike polynomial $V_{mn}(x,y)$.

The complex Zernike polynomial $V_{mn}(x,y)$ is calculated as:

$$
V_{mn}(x,y) = V_{mn}(\rho\cos\theta, \rho\sin\theta) = R_{mn}(\rho)e^{in\theta} \qquad (2.33)
$$

where $i^2=-1$, and $R_{mn}(\rho)$ is the orthogonal radial polynomial:

$$
R_{mn}(\rho) = \sum_{s=0}^{\frac{m-|n|}{2}} (-1)^s \frac{(m-s)!}{s!\left(\frac{m+|n|}{2}-s\right)!\left(\frac{m-|n|}{2}-s\right)!}\rho^{m-2s} \qquad (2.34)
$$

We can easily see that $R_{mn}(\rho) = R_{m,-n}(\rho)$ and therefore $[V_{mn}(x,y)]^* = V_{m,-n}(x,y)$.

2.2.2 Local approaches

Local descriptors are nowadays widely used in content-based image retrieval. Instead of calculating a signature on the entire image, local approaches characterize the local properties of the image region around fixed grid points or detected interest points. Thus, each image is characterized by a set of local feature vectors (one vector for each point).

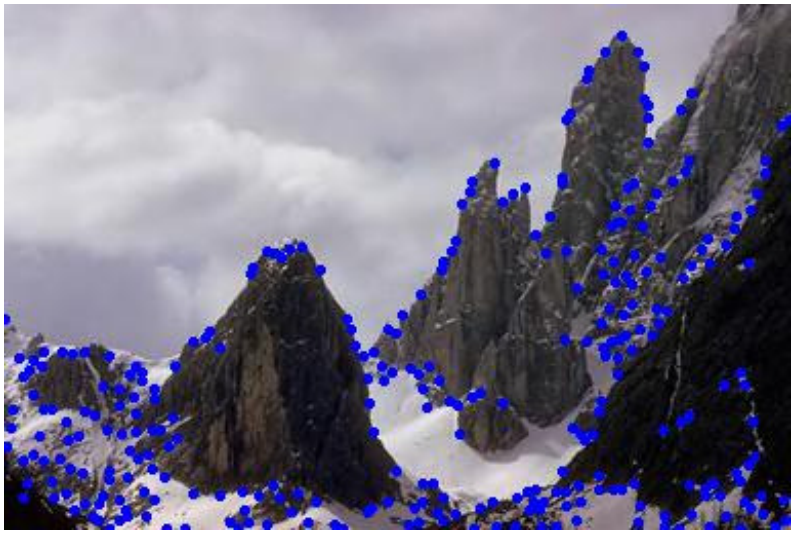


Figure 2.5 – Output interest points of the Harris detector.

Different detectors for identifying the interest points are proposed in the literature such as the Harris detector [46], the Harris-Laplace detector [90], the Difference of Gaussian (DoG) detector [81], the Laplacian of Gaussian (LoG) detector [78], etc. Figure 2.5 represents an example of the output interest points detected by the Harris detector.

For representing the local characteristics of the image around each point, various descriptors are proposed such as the local color histogram [130], Scale-Invariant Feature Transform (SIFT) [81], Speeded Up Robust Features (SURF) [11], color SIFT descriptors [1, 17, 130], etc. Among these descriptors, SIFT descriptors are very popular because of their very good performance.

SIFT (Scale-Invariant Feature Transform) [81] SIFT was proposed by Lowe [81]. The feature descriptor representing the characteristics of the region around each interest point (keypoint) is calculated based on the gradient distribution of this region. Figure 2.6 illustrates the computation of the SIFT descriptor around a detected interest point. The feature descriptor of each interest point is calculated on the Gaussian-smoothed image $L(x, y, \sigma)$ at the scale σ where the interest point is detected:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.35}$$

where $*$ is the convolution operation, $I(x, y)$ is the original image and $G(x, y, \sigma)$ is the 2D Gaussian kernel at scale σ:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.36}$$

Each image sample point $L(x, y)$ in a region around the interest point location is assigned a gradient magnitude $m(x, y)$ and an orientation $\theta(x, y)$ computed using pixel differences (see Figure 2.6(a)).

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (2.37)$$

$$\theta(x,y) = tan^{-1}\left((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))\right) \quad (2.38)$$

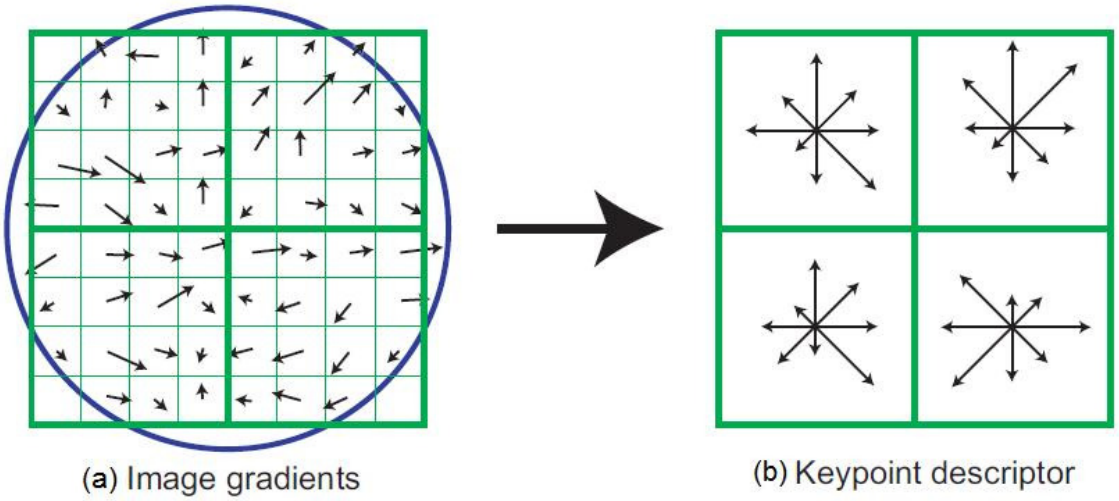


Figure 2.6 – Computation of a SIFT descriptor in a region of size 8x8 around an interest point. (Lowe, 2004)

The gradient magnitudes of the sample points in the region around the interest point are weighted by a Gaussian window, indicated by the circle in Figure 2.6(a), with a σ of 1.5 times the scale of the interest point. The feature descriptor of each interest point is concatenated from orientation histograms computed in a region around the interest point such that each histogram is computed from points of a 4×4 subregion. Each orientation histogram created on a 4×4 subregion has 8 bins corresponding to 8 orientations, as shown in Figure 2.6(b). The length of each arrow (bin) corresponds to the sum of the gradient magnitudes of the sample points within this 4×4 subregion having a similar orientation to this arrow. Figure 2.6 illustrates the computation of the SIFT descriptor concatenated from $2 \times 2 = 4$ histograms computed from a 8×8 region of sample points. In his work Lowe [81] used a region of size 16×16 for calculating the 4×4 orientation histograms, therefore the feature descriptor has $8 \times 4 \times 4 = 128$ elements.

Color SIFT descriptors As the SIFT descriptor of Lowe [81] is applied only on the intensity channel, the color information may be lost. Therefore, some color SIFT descriptors are proposed for being used in different color spaces:

- **RGB-SIFT [130]**: The local RGB-SIFT descriptor for each interest point is created by independently computing the SIFT descriptors for every channel of the RGB color space. The local feature descriptor has thus 3×128 elements.
- **HSV-SIFT [17]**: Same as RGB-SIFT, the HSV-SIFT descriptor for each interest point is the concatenation of the three SIFT descriptors computed

for each component of the HSV model. The HSV-SIFT descriptor for each interest point has 3×128 dimensions.

- **Hue-SIFT [130]**: The Hue-SIFT descriptor for each interest point is the concatenation of the Hue histogram and the original SIFT descriptor computed on the region around this interest point.
- **OpponentSIFT [130]**: OpponentSIFT of each interest point is computed as the concatenation of SIFT descriptors describing all the channels in the opponent color space (see Equation (2.2)).
- **rgSIFT [130]**: For the rgSIFT descriptor, the SIFT descriptors which are computed for the r and g components of the normalized RGB color space (see Equation (2.1)) are concatenated with the original SIFT descriptor in the intensity channel.
- **CSIFT (Colored SIFT) [1]**: For the CSIFT descriptor of each interest point, Abdel-Hakim and Farag [1] compute the SIFT descriptors for all color invariants of the color invariance model described in Equation (2.39)

$$\begin{pmatrix} E \\ E_\lambda \\ E_{\lambda\lambda} \end{pmatrix} = \begin{pmatrix} 0.06 & 0.63 & 0.27 \\ 0.3 & 0.04 & -0.35 \\ 0.34 & -0.6 & 0.17 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{2.39}$$

As local approaches represent each image by a set of local descriptors and considering the fact that the number of interest points may vary from one image to another, the calculation of the distance between two images is not straightforward. 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 [5]. For each interest point P of the query image, we consider, among all the interest points detected from 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 $D(P, P1) \leq distRatio \times D(P, P2)$, where D is the Euclidean distance between two points (computed using their n-dimensional feature vectors) and $distRatio$ is a fixed threshold, $distRatio \in (0, 1)$. By defining a *match* in this way, we can reject the ambiguity cases where the distances between P to the closest and the second closest interest points are similar. 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:

$$D_{i,j} = D_{j,i} = 100 * \left(1 - \frac{M_{ij}}{\min\{K_{A_i}, K_{A_j}\}}\right) \tag{2.40}$$

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 "bag of words" method [121] [7] which calculates, from a set of local descriptors, a global feature vector for each image. Figure 2.7 shows the "bag of words" approach used in an object recognition problem. First, we extract local descriptors of all the images in the database and learn to aggregate these local descriptors into groups in order to create a dictionary (also called codebook). The centre of each group is considered as a visual word in the dictionary. Unsupervised learning methods (generally k-means) are mostly used for constructing the dictionary, some other works learned the codebook by using supervised learning [18] [86]. Then, a "bag of words" is constructed for each image via two steps: coding and pooling. The coding step projects each local descriptor of the image onto the visual words of the codebook, and the pooling step combines the projected codes of all the local descriptors of the image into a vector. The projected code of a local descriptor x_j is a vector $\alpha_j = (\alpha_{1,j}, ..., \alpha_{n,j})$, where n is the number of visual words in the dictionary (*i.e.* the number of groups of local descriptors) and $\alpha_{m,j}$ is the m^{th} component of the encoded vector corresponding to the m^{th} visual word in the dictionary. Traditionally, the coding step activates $\alpha_{m,j} = 1$ only for the word which is closest to the descriptor x_j, and assigns zero to all other components. Another possibility for the coding step is the soft coding [131]. Many visual words in the codebook can be assigned to a same local descriptor, but with different weights corresponding to the distances between these visual words and the descriptor. The pooling step aggregates all encoded vectors of each image into a single vector $z = (z_1, ..., z_n)$. The sum pooling or the max pooling operators are usually used. For the sum pooling operator, each component z_m of the vector z is computed as $z_m = \sum_{j=1}^{M} \alpha_{m,j}$ (where M is the number of local descriptors of the image), while the sum pooling operator computes each component as $z_m = \max_{j \in 1..M} \alpha_{m,j}$. The tf-idf weighting method [123] is also used alternatively for the pooling step. By using the "bag of words" method, each image is characterized by a feature vector of size n and the distance between any two images can be easily calculated using the Euclidean distance or the χ^2 distance.

2.2.3 Spatial approaches

In spatial approaches, each image is considered as a set of visual objects which are often obtained by a preliminary stage of image segmentation. Objects/Regions and spatial relationships among objects/regions are then characterized for representing the content of the image. This kind of approaches is less used than the global and the local approaches because it requires the preliminary stage of image segmentation which is not straightforward, especially in the context of huge image databases where the contents may be very heterogeneous. The most widely used spatial feature descriptors are the 2D string [20] and its variants [19, 55, 59, 73].

2D String [20] Spatial representation by the 2D string was presented by Chang *et al.* [20]. The significant objects in an image are first segmented and recognized. The idea of the 2D string approach is to represent each image by a symbolic picture

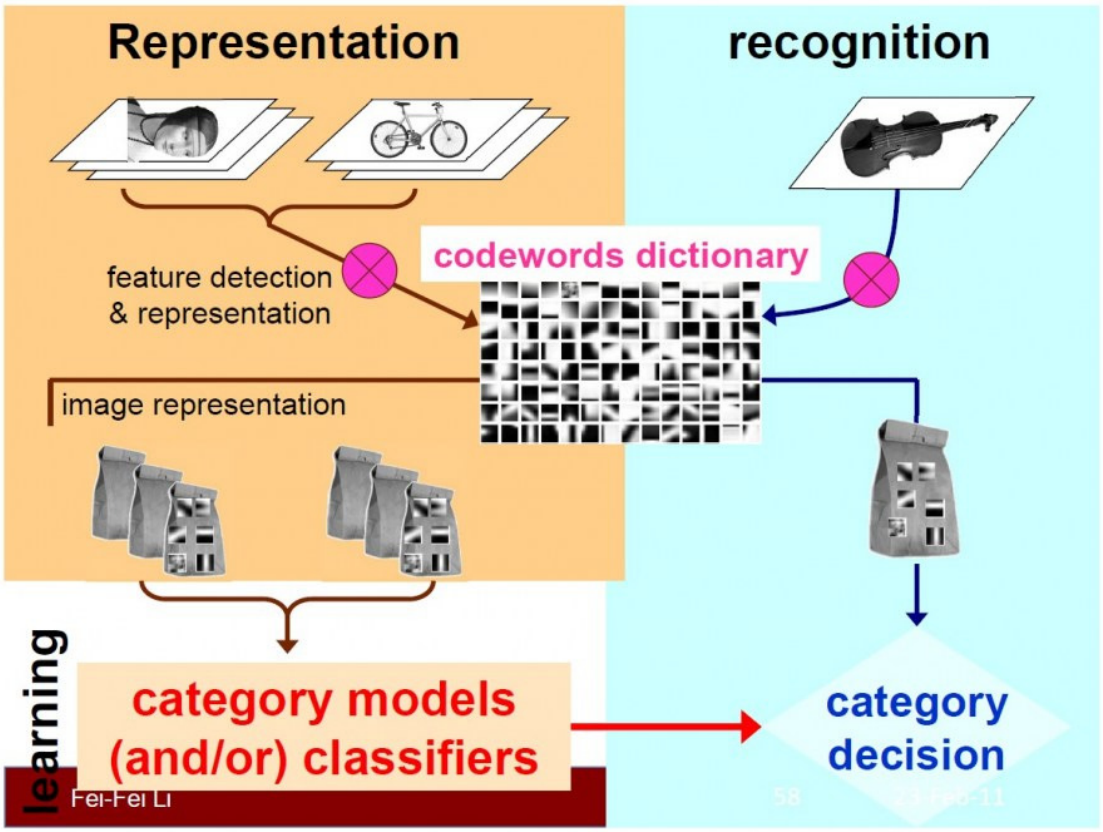
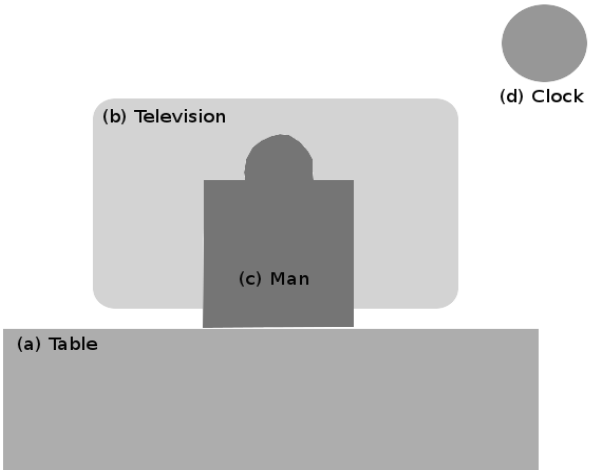


Figure 2.7 – Bags of words approach in an object recognition system (Source: `http://sensblogs.wordpress.com/page/2/`)

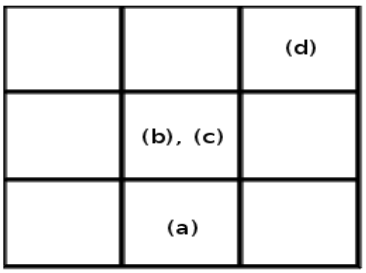
where each symbol represents an object in the image. A symbolic picture over a set V of symbols is an $m \times n$ matrix, where each symbol in V is assigned to a slot of the matrix corresponding to the position in the image of the centroid of the object represented by this symbol. Figure 2.8b shows an example of the symbolic picture corresponding to the image containing four segmented objects represented in Figure 2.8a. The spatial relationships among the symbols in a symbolic picture are expressed by using different spatial operators defined in the set $A = \{<, =, :\}$. The operator "$<$" specifies the *left-right* or *below-above* spatial relationship, the operator "$=$" signifies "at the same spatial location as" and the operator "$:$" denotes the relation "in the same set as". By orthogonally projecting the symbols in a symbolic picture by columns and by rows, the 2D string representing the corresponding image is defined as a pair of strings:

$$(u, v) = (o_1 x_1 o_2 x_2 ... x_{n-1} o_n, o_{p(1)} y_1 o_{p(2)} y_2 ... y_{n-1} o_{p(n)}) \tag{2.41}$$

where $o_1 o_2 ... o_n$ is a 1D string over the set of symbols V ($o_i \in V$), $o_{p(1)} o_{p(2)} ... o_{p(n)}$ is a permutation of $o_1 o_2 ... o_n$, $x_1 x_2 ... x_{n-1}$ and $y_1 y_2 ... y_{n-1}$ are 1D strings over the set of spatial operators A. The first string $o_1 x_1 o_2 x_2 ... x_{n-1} o_n$ represents the spatial information of n objects along the x-axis, while the second string $o_{p(1)} y_1 o_{p(2)} y_2 ... y_{n-1} o_{p(n)}$ represents the spatial information of these objects along the y-axis. For example, the symbolic picture in Figure 2.8b can be represented using the 2D string $\{a = b : c < d, a < b : c < d\}$.



(a) Segmented objects in an image.



(b) Symbolic picture representing the image.

Figure 2.8 – Examples of image representation using 2D string.

The similarity between two images represented by 2D strings can be determined by using exact or approximate matching techniques [20, 103]. In the case of exact matching, two images are similar if every object in the first image has at least one similar object (object of the same class) in the second image and the matched objects of these two images have exactly the same spatial relationships. In the case of approximate matching techniques, the similarity between two 2D strings represented two images can be determined based on the longest subsequence that the two 2D strings have in common.

2D string variants The 2D string approach gives a natural way for representing spatial relationships in 2D pictures. However, for images with large number of overlapping objects having complex shapes, the spatial operators $\{<, =, :\}$ are not sufficient for representing the spatial relationships among objects. Therefore, many variants of the 2D string approach have been proposed.

- Jungert [59] proposed an extended version of the 2D string approach which allows a more precise description of the spatial relationships among objects by using a new set of operators $\{=, <, \backslash, /, |=, =|, \%, |\}$ representing different relations as shown in Table 2.2.

Notation	Relation	Condition
$A = B$	equal	centroid(A)=centroid(B)
$A < B$	less than	$max(A) < min(B)$
B/A	overlap	$max(B) < max(A)$ and $length(B) \geq length(A)$
$A\backslash B$	overlap inverse	$min(A) < min(B)$ and $length(A) \leq length(B)$
$A \mid= B$	begin	$min(A) = min(B)$ and $length(A) < length(B)$
$B =\mid A$	end	$max(B) = max(A)$ and $length(B) > length(A)$
$B\%A$	contain	$min(B) < min(A)$ and $max(B) > max(A)$

Table 2.2 – Spatial operators of the extended 2D string approach.

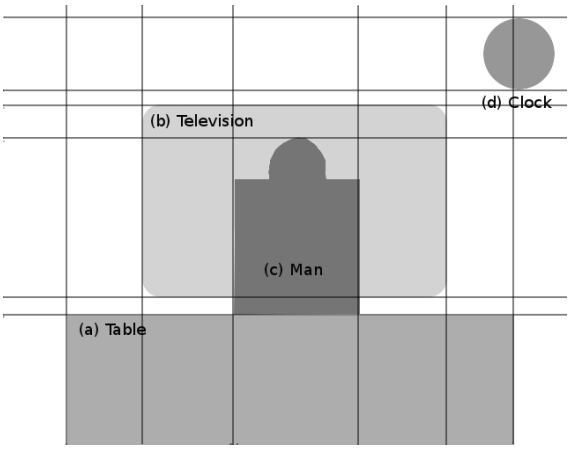


Figure 2.9 – Example of cutting mechanism in the 2D G-string.

- The generalized 2D string (2D G-string) was introduced by Chang *et al.* [19] with the cutting mechanism. The vertical and horizontal cutting lines are drawn through all the extremal points of all objects in the image as in Figure 2.9. By projecting the objects along the x-axis and the y-axis, the generalized 2D string corresponding to the example in Figure 2.9 is as follows:

$$u : a|ab|acb|ab|a|ad|d$$
$$v : a|c|bc|b < d$$

where "$<$" is the *less than* operation as in Table 2.2, and "$|$" is the *edge-to-edge* operator representing two objects touching together.

- The 2D C-string was proposed by Lee and Hsu [73] to reduce the number of cutting objects. The 2D C-string approach uses a set of seven spatial operators defined in Table 2.3. Instead of drawing cutting lines through all extremal points of all objects, the 2D C-string approach uses only cutting lines through partly overlapping objects. For a pair of overlapping objects, the cutting line is drawn through the end point of the first ending object. Therefore, one of the overlapping objects is split into two parts, the object that is not split is the dominating object. In Figure 2.10, a horizontal cutting line is drawn through the end point of object (c) and divides object (b) into two parts, and the vertical cutting line at the end of object (a) divides object (d) into two parts. The 2D C-string corresponding to the example in Figure 2.10 is as follows:

$$u : (a\%b\%c)]d \mid d$$
$$v : a|c]b \mid b < d$$

Notation	Meaning	Condition
$A < B$	A disjoint B	$end(A) < begin(B)$
$A = B$	A equals B	$begin(A) = begin(B)$ and $end(A) = end(B)$
$A \mid B$	A edge-to-edge B	$end(A) = begin(B)$
$A\%B$	A contains B	$begin(A) < begin(B)$ and $end(A) > end(B)$
$A[B$	A contains B with same begin	$begin(A) = begin(B)$ and $end(A) > end(B)$
$A]B$	A contains B with same end	$begin(A) < begin(B)$ and $end(A) = end(B)$
A/B	A partly-overlap B	$begin(A) < begin(B)$ and $end(A) < end(B)$

Table 2.3 – Spatial operators of the 2D C-string approach.

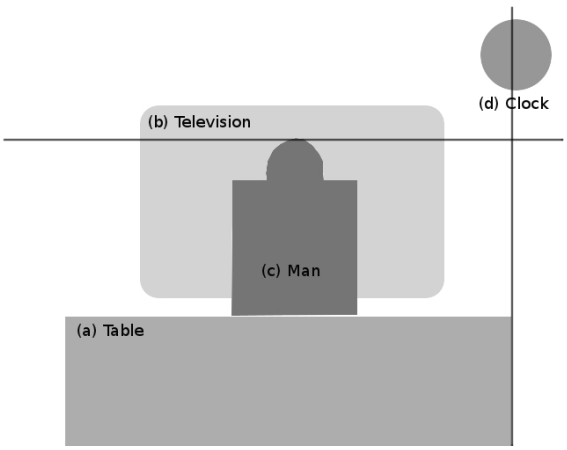


Figure 2.10 – Example of cutting mechanism in the 2D C-string. Object (a) dominates (d) along the x-axis while (c) dominates (b) along the y-axis.

2.3 Indexing

By using the feature extraction methods presented in the previous section, the visual characteristics of the images in the database are extracted in the form of feature vectors. With the development of many large image databases, indexing methods are needed to organize the feature vectors in an index structure that supports a fast and efficient retrieval in the feature database. Index structures can be created based on one or more features of the feature vector. Indexing methods can support point databases in which data are points in an N-dimensional space and/or spatial databases where data are lines, rectangles or other geometric objects in N-dimensional space [34]. In this section, we present successively two main types of traditional indexing methods: space partitioning methods and data partitioning methods.

2.3.1 Space partitioning methods

Space partitioning approaches generally partition the feature space into disjoint (or non-overlapping) cells (sometimes also called "buckets") by one or more hyperplanes. Each point of the database lies in only one of the cells. Some techniques of this kind are Quadtree [28, 116], k-d-tree [14], k-d-b-tree [112], grid file [98], LSD tree [48], etc.

Quadtree [28, 116] One of the most famous space partitioning structures is the quadtree [28,116]. The quadtree is usually used to organize two dimensional data in a tree structure by recursively subdividing the space into $2^2 = 4$ quadrants or cells, each of which corresponding to a rectangle. Each internal node has therefore four children which are respectively the NW (north-west), NE (north-east), SW (south-west) and SE (south-east) quadrants. The decomposition of the tree is continued until the number of objects in each leaf node is below a given threshold, or until the similarity of the points in different quadrants is greater than a given threshold, etc. Note that the basic idea of the quadtree can be easily generalized to an arbitrary dimension d in which each internal node has 2^d descendants.

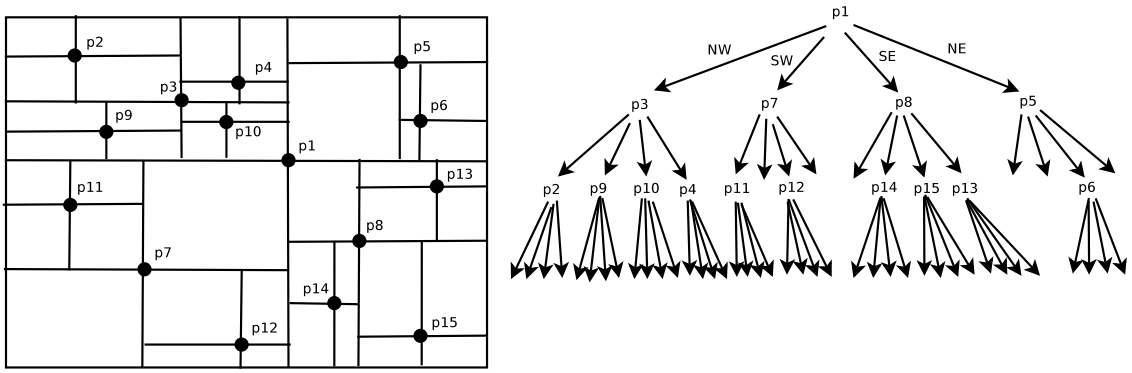


Figure 2.11 – Point quadtree.

Two main variants of the quadtree are the point quadtree [28] and the region quadtree [116]. The point quadtree is constructed by successively inserting each point in the database into the tree. For each point, we first go down from the root

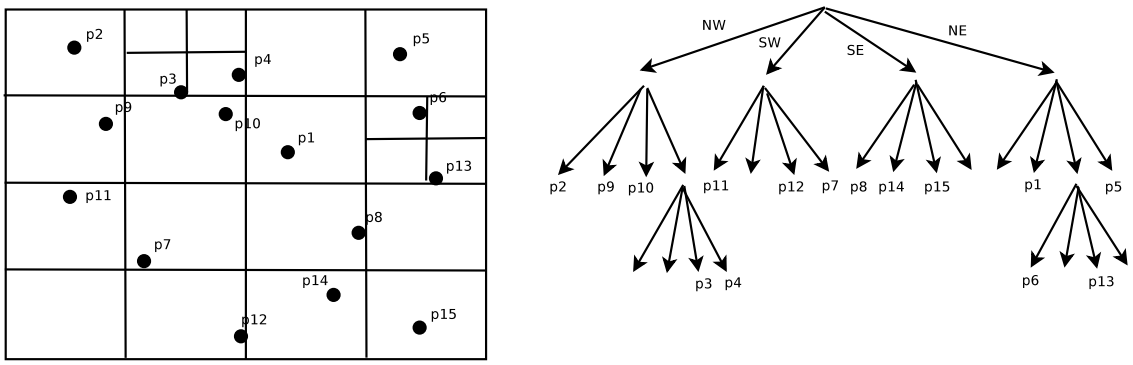
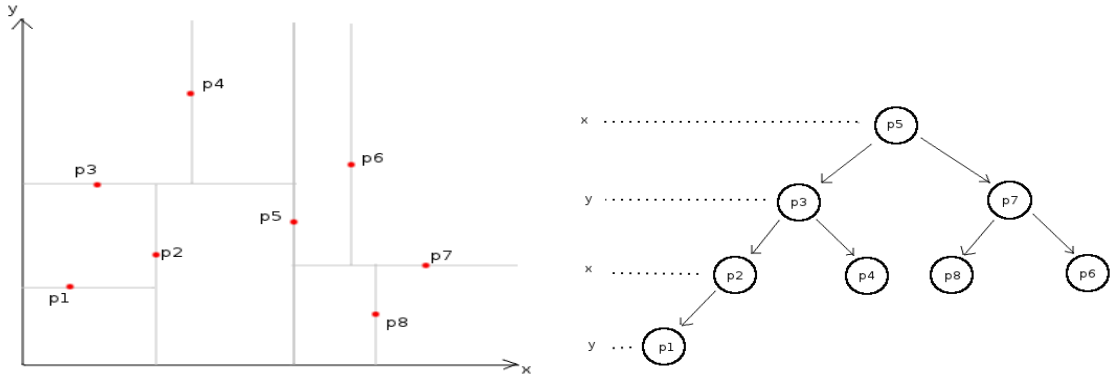


Figure 2.12 – Region quadtree.

to find the leaf at which the point should be added. Then, the point is added into the leaf and the region of the corresponding leaf is divided into 2^d subregions with the new point at the centre. Figure 2.11 shows on the right-hand side an example of a 2-dimensional point quadtree corresponding to the space decomposition described on the left-hand side. The region quadtree decomposes successively the space corresponding to each node into 2^d subspaces of equal size until the number of objects in each cell is below or equal a given threshold. Figure 2.12 shows an example of a region quadtree with threshold=1. We can see that the point quadtree is sensitive to the order in which the points are inserted while the region quadtree is not. The quadtree allows point queries, nearest neighbour queries and range queries.

k-d-tree [14] Another widely known space partitioning structures is the multidimensional binary search tree (k-d-tree) proposed by Bentley, in 1975 [14]. A k-d-tree is a binary tree which aims at organizing points in a k-dimensional space by recursively subdividing the space into two subspaces by a (k-1)-dimensional hyperplane at each non-leaf node. Each node of the k-d-tree contains a k-dimensional point $(d_1, d_2, ..., d_k)$, two pointers referring to the left and right sons, and a discriminator which is an integer varying from 1 to k specifying one of the k-dimensions. At each non-leaf node, the space is divided into two subspaces by an implicitly associated hyperplane passing through the point specified in this node and perpendicular to the axis of the dimension specified by the discriminator of this node. So, for any non-leaf node x which discriminator specifies dimension j, all points in the left subtree of x have their d_j values less than the d_j value of x, and all points having greater d_j values will appear in the right subtree. The original algorithm to construct the k-d-tree selects the splitting hyperplanes by cycling through the d dimensions. For example, in a $(d = 3)$-dimensional space, when moving down the tree, the splitting hyperplanes are first perpendicular to the x-axis, then y-axis and z-axis before cycling back to the x-axis. According to the selected dimension, the splitting plane can cross through the median point of all points included in the current node or through the median point of some randomly selected points. Figure 2.13b shows a k-d-tree constructed in a 2-dimensional space corresponding to the space decomposition in Figure 2.13a. The first splitting plane is the vertical line crossing $p5$. The splitting planes in the next level are horizontal lines crossing $p3$ for the left subtree and $p7$ for the right subtree, and so on. In this k-d-tree structure, data points are scattered all over the tree and the directions of the splitting hyperplanes have to be

alternating in a fixed order which is not always the best choice. Bentley and Friedman [15] proposed an extension of the k-d-tree which stores data points only in the leaves. Each non-leaf node contains the discriminator specifying the dimension and the split value corresponding to this dimension; points in the left subtree are less or equal to the split value while points having greater values are in the right subtree. In this extension, the splitting hyperplane directions do not need to be chosen in a fixed order. At each internal node, the attribute j is chosen as discriminator if it maximizes the spread of attribute values (variance or distance from minimum to maximum) of points contained in this node. The k-d-tree structure is used for point data, it allows point queries, nearest neighbour queries and range queries, but it is sensitive to the order in which the points are inserted.



(a) Space decomposition by the k-d-tree. (b) The resulting k-d-tree.

Figure 2.13 – Examples of the k-d-tree.

k-d-b tree [112] The k-d-b-tree structure [112] combines the properties of the B-tree [12] and the k-d-tree [14] structures to organize k-dimensional data points in a balanced tree. Each node in the k-d-b-tree corresponds to a page. There are two kinds of pages in a k-d-b-tree: point pages corresponding to leaf nodes and region pages corresponding to internal nodes. Data points are located in point pages. Region pages contain entries in the form of (*region*, *childID*), where *childID* is the pointer pointing to a child page of this page, *region* in the form of $((min_1, max_1), ..., (min_i, max_i), ..., (min_k, max_k))$ describing the region of the corresponding child node. If the child node is a point page, all the points in the child page must be in *region*. If the child node is a region page, the union of the regions in the child page is *region*. Note that the regions in a region page are disjoint and their union is a region. The region corresponding to the root page covers all the space. As the k-d-b-tree is a balanced tree, the path length from root page to leaf page is the same for all leaf pages. Figure 2.14 shows an example of a 2-d-b-tree in which page 0, page 1, page 2, page 3 are region pages while page 4, page 5 and page 6 are point pages.

The k-d-b-tree is constructed by successively adding each point in the database into the tree. When inserting a new point into the tree, it goes down from the root until it reaches the point page it should be added to. The disjunction between nodes of the same level in the tree involves a single path in the search, and therefore

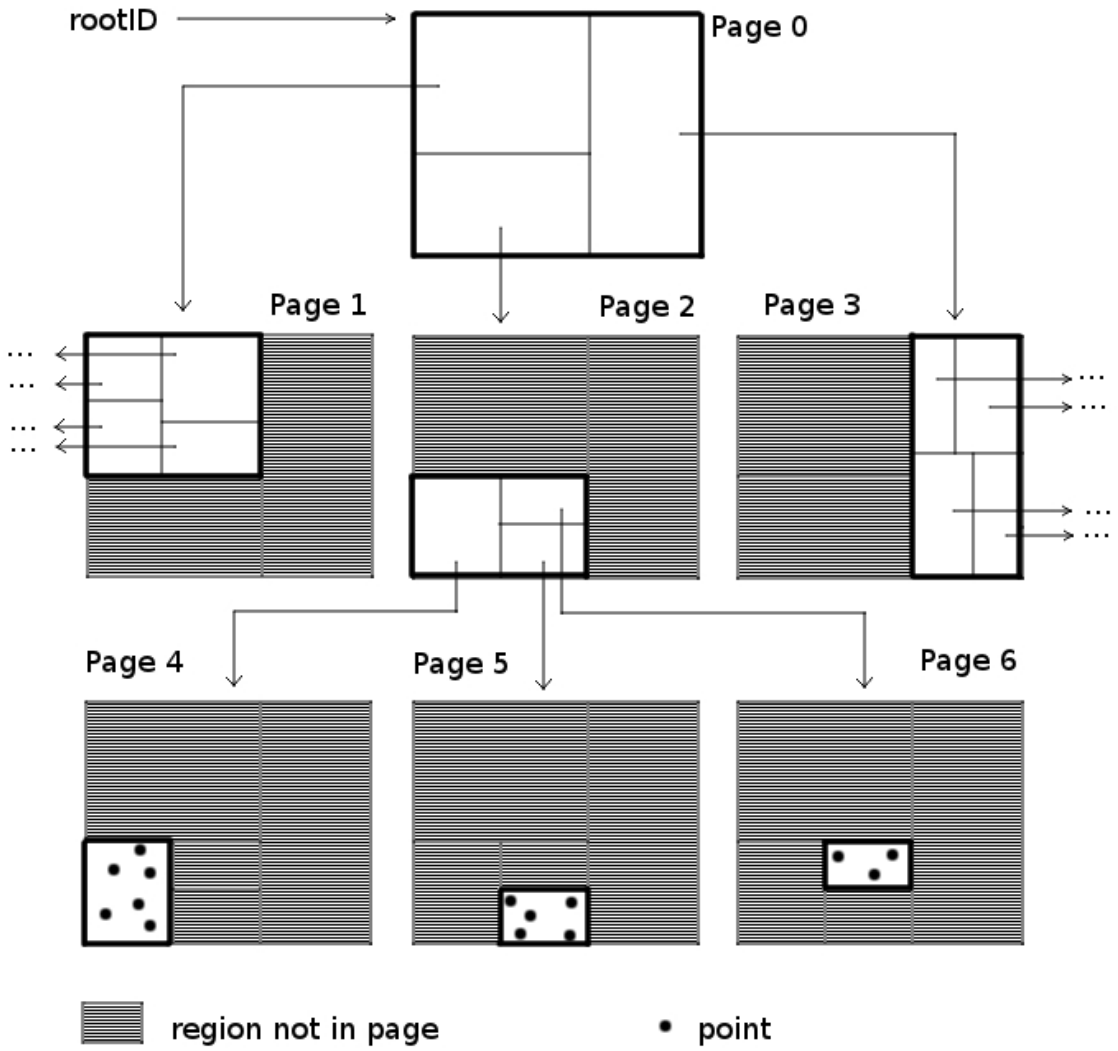


Figure 2.14 – Example of a 2-d-b-tree.

only one point page can be reached. Then, the point is added to that point page if the page is not full. Otherwise, a dimension d_i and an element x_i in this dimension are picked; and the point page is split by a hyperplane perpendicular to d_i at the position x_i. Like in a B-tree, the father node is updated by replacing the entry corresponding to the old point page by two new entries corresponding to the two new point pages. A new split of the father page may be needed if it overflows. As in the case of k-d-tree, the cyclic method can be used to choose the dimension d_i and the medoid value along this dimension can be chosen as x_i. Note that when we split a region of an internal node along an axis, we must also split the regions of the sub-nodes along this axis. Therefore, empty or almost empty nodes can be created. This may cause a degradation in the performances of the k-d-b-tree in the case of nearest neighbour queries and range queries.

Grid file [98] The grid file [98] is a space partitioning method based on hashing technique. A grid file uses a d-dimensional orthogonal grid to partition the d-dimensional space into a non-periodic grid of cells. The grid file does not contain any data, but data points are stored in data buckets which are associated with one

or more cells of the grid. Each data bucket is stored on one disk page and has a capacity of c records. Each cell refers to only one bucket, while a bucket may contain points of several adjacent cells.

The *Grid directory* is the data structure representing the grid file. A grid directory consists of two parts: d one-dimensional arrays called *linear scales* representing the partition of the space along each of d dimensions and a d-dimensional array called *grid array* whose elements point to data buckets and correspond to grid cells of this partition. Since the directory may grow large, the grid array is usually kept on secondary storage while the linear scales are kept in main memory.

Figure 2.15 shows an example of a grid file partitioning a 2-dimensional space into cells associated with buckets of capacity $c = 4$. We can see for instance that bucket C stores data points corresponding to four different cells in the upper right part of the grid.

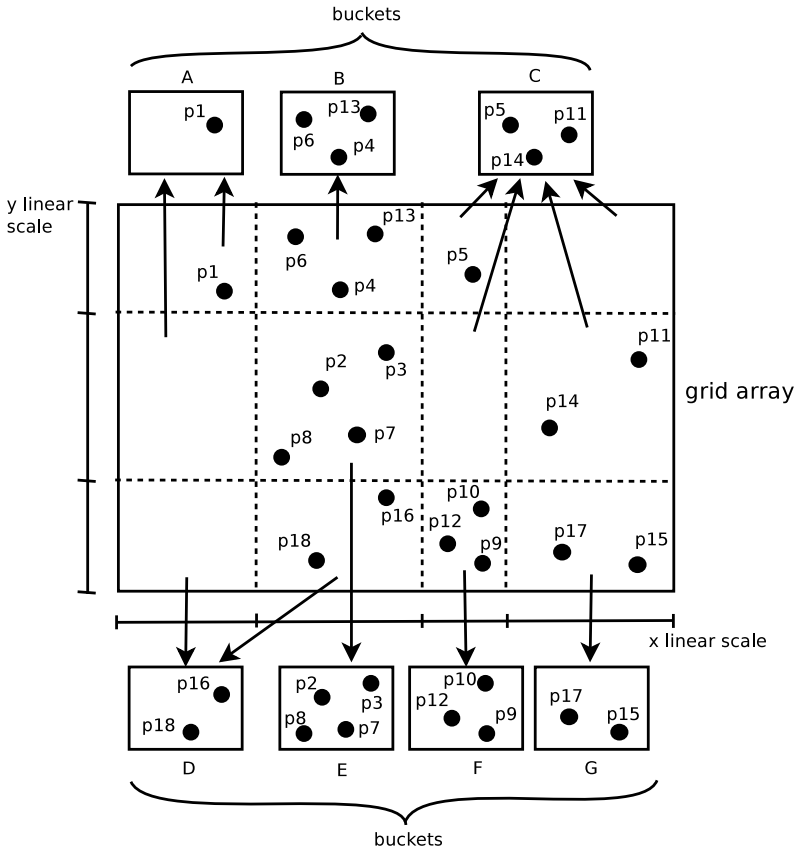


Figure 2.15 – Example of a grid file in 2-dimensional space.

To insert a new point in a grid file, the cell and the corresponding data bucket p_i where the point should be inserted are first located. If the bucket p_i is not full, then the new point is inserted in p_i. If p_i is full, then p_i is split into two data buckets by an existing hyperplane of the grid if it is possible (in the case that there are several grid cells that point to p_i) or by a new hyperplane H perpendicular to the axis of a chosen dimension d_i (all corresponding cells being also split according to H). There are several splitting policies that are compatible with the grid file. The simplest policy is to choose cyclically the dimension for splitting. Another policy is to choose dimensions corresponding to some favoured attributes more often than

others. The location of a split can be chosen at the midpoint of the corresponding linear scale range, or at any point around the midpoint. As all corresponding cells are split when splitting a data bucket by a new hyperplane, the grid can therefore considerably grow by having many empty cells, which makes it not very adapted to nearest neighbour queries or range queries.

2.3.2 Data partitioning methods

While space partitioning methods partition the space into disjoint cells, data partitioning methods partition the data objects into regions which may overlap, based on the distribution of the descriptors and their relative proximity in space. Similar points are generally enclosed by a bounding region (normally in the form of minimum bounding rectangle or minimum bounding sphere). Popular index structures of this kind are the R-tree family [13,42,117], SS-tree [139], SR-tree [60], X-tree [16], etc.

R-tree family [13,42,117] R-tree (Rectangle Tree) family consists of three index structures: R-tree [42], R+-tree [117] and R*-tree [13]. The basic idea of these approaches is to group data objects using multidimensional minimum bounding rectangles. These rectangles are organized in a height-balanced tree corresponding to the data distribution, where data objects are stored in the leaves and all leaves are at the same level. Each node of the tree corresponds to a disk page and a bounding rectangle representing the region of this node. The bounding rectangle of a leaf is the minimum bounding rectangle of the objects (point objects and/or spatial objects) stored in this leaf. 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. Note that every node contains at least m and at most M entries, unless it is the root. The lower bound m ensures an efficient storage utilization, while the upper bound M ensures that each node does not exceed one disk page size. Figure 2.16 shows an example of an R-tree with 2-dimensional point data objects. In the case of spatial data objects, each object can be represented by a minimum bounding rectangle.

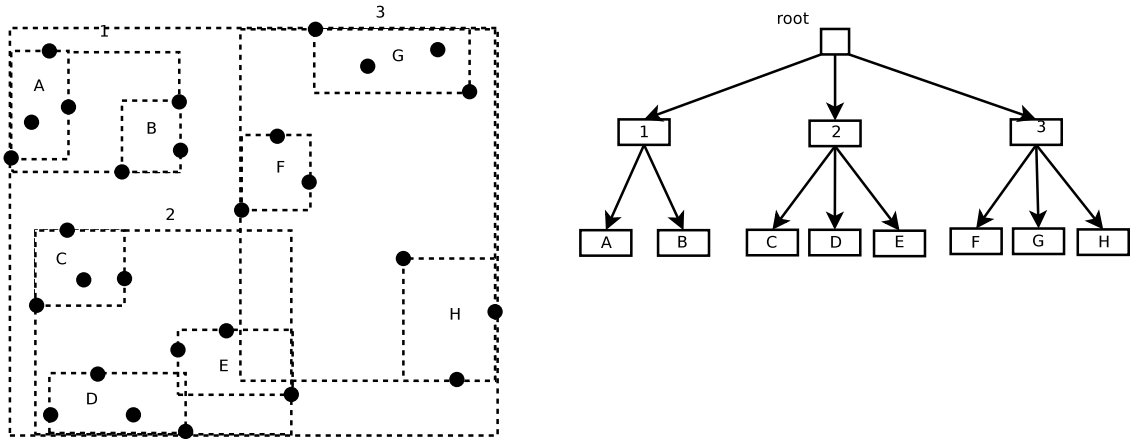


Figure 2.16 – Example of an R-tree in 2-dimensional space.

The R-tree can be built under repeated insertions of points in the databases.

To insert a new object o, we go down from the root by selecting, at each level, the child node that requires the least enlargement of its bounding rectangle to enclose the new object until a leaf L is reached. If the leaf L is not full, then the new object is inserted into L. If L is full, then L must be divided into two new leaves by minimizing the total volume of the two corresponding bounding boxes. After inserting the new object, the bounding rectangles of the corresponding nodes are adjusted; the change and also the split are propagated upward.

For searching by point query or range query in the R-tree, we have to go down from the root and visit, at each level, all child nodes having the bounding rectangles which contain the input point or intersect with the input interval. As the bounding rectangles in each level may overlap, we might have to visit many child nodes. The rectangle overlapping probability increases when the number of dimensions is high. In the worst case, we may have to visit every nodes of the tree. In this case, sequential scan may be more effective.

As the overlap between nodes is generally important in R-tree, the R+-tree [117] and R*-tree [13] structures have been developed with the aim of minimizing the overlap of bounding rectangles, in order to optimize the search in the tree. The R+-tree structure avoids the overlap of bounding rectangles by dividing each overlapping rectangle into smaller rectangles, until they no longer overlap. This may increase the height of the tree, but we can save time during retrieval by reducing the number of sub-trees to visit. The R*-tree structure tries to minimize the overlap among rectangles and also their volume by using the reinsertion mechanism. When a new entry is inserted into a full node R_i, we try to reinsert some entries of this node, and hopefully find better positions for these entries. If all these entries are reinserted in the same location, then R_i is split into two nodes as in R-tree. Experiments in [13] show that R*-tree is the best structure of the R-tree family for indexing multidimensional data. The structures in the R-tree family are all sensitive to the insertion order of the objects. On the other hand, since there is no empty cell, the R-tree family is more suitable for the nearest neighbour search.

SS-tree [139] The SS-tree (Similarity Search Tree) [139] is a similarity indexing structure which groups the objects based on their similarity in the space. The SS-tree structure is similar to that of the R-tree, but bounding spheres are used instead of bounding rectangles to enclose the entries of each node. This allows to offer an isotropic analysis of the feature space. Data objects are stored in the leaves. Each bounding sphere is represented by a centre and a radius. The centre of the bounding sphere of a node is the gravity centre of all the elements in the sub-tree of this node. The radius of a leaf bounding sphere is the distance between the centre and the farthest point in this leaf. The radius of an internal node bounding sphere is computed as $\max_{i=0}^{n}(D(c,c_i)+R_i)$ where n is the number of children of this node, $D(c,c_i)$ is the distance between the centre c of this node and the centre c_i of the child node i, and R_i is the radius of the corresponding child node. An example of an SS-tree in a 2-dimensional space is shown in Figure 2.17.

An SS-tree can also be constructed by successively inserting each data object into the tree. When a new entry goes down from the root, at each step the new element moves to the node whose centre is closest to the new element. The SS-tree also uses the reinsertion mechanism of the R*-tree in order to minimize the overlap

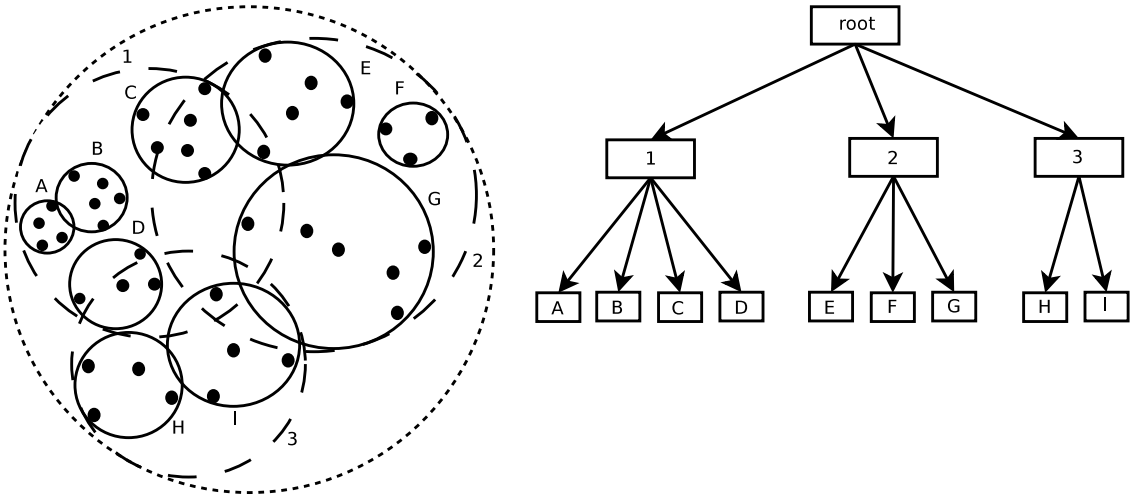


Figure 2.17 – Example of an SS-tree in 2-dimensional space.

among bounding spheres, as well as their volume. In comparison to the R*-tree, SS-tree has been shown to have better performance [60]. But the SS-tree structure is sensitive to the insertion order of the objects and the overlap between nodes is still high, when indexing high dimensional data.

SR-tree [60] SR-tree (Sphere/Rectangle Tree) [60] combines R*-tree and SS-tree by identifying the region of each node as the intersection of the bounding rectangle and the bounding sphere. The bounding rectangle and the bounding sphere of a leaf node are respectively the minimum bounding rectangle and the minimum bounding sphere containing all the data objects of this leaf. The bounding rectangle of an internal node covers all the bounding rectangles of its child nodes, while the bounding sphere covers all the bounding spheres of its child nodes. Figure 2.18 shows an SR-tree organizing 2-dimensional point data.

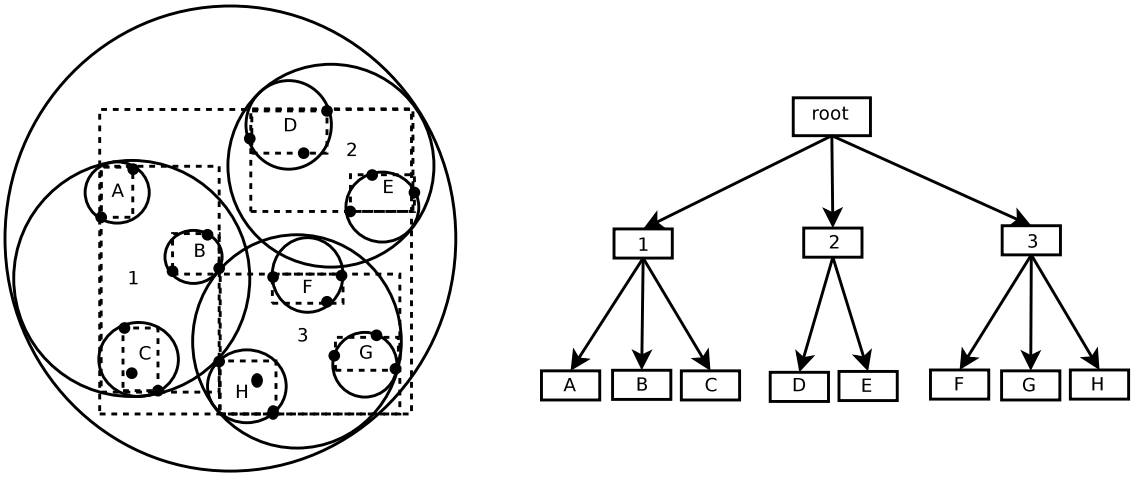


Figure 2.18 – Example of an SR-tree in 2-dimensional space.

In high dimensional space, bounding rectangles tend to have smaller volumes than bounding spheres, while bounding spheres have in general shorter diameters than bounding rectangles [60]. By combining the bounding rectangle and the

bounding sphere, SR-tree allows to create regions with small volumes and small diameters. This allows to reduce the overlap between nodes and thus enhances the performance of nearest neighbour search with high dimensional data. But SR-tree is still sensitive to the processing order of the data.

2.4 Retrieval

In the retrieval phase of a traditional CBIR system, the user is required to submit a query image to the CBIR system. The query image may be an existing image or a sketch made by the user providing a rough approximation of the image he is interested in. The CBIR system has to return result images which share common characteristics with the query image. Depending on the application, they may be the k images which are most similar to the query image, all the images within a given fixed distance ϵ from the query image or images containing similar objects to the query image. For searching suitable images, the system extracts the feature vector of the query image and then measures the similarity with the feature vectors previously extracted from the images in the database. The performance of the retrieval phase strongly depends on the chosen similarity measure; the choice of this measure plays therefore a crucial role. In practice, dissimilarity measures are usually used instead of similarity measures for comparing different images in the database. Moreover, as the feature vectors only capture low-level information such as color/texture or shape, there may be a "semantic gap" between high-level semantic concepts expressed by the user and these low-level features. Thus, the retrieval based on the similarities between those low-level features may produce images which do not fit the intent of the user. In this section, we present first the retrieval strategy, then different dissimilarity measures, and finally different strategies for solving the "semantic gap" problem.

2.4.1 Retrieval strategy

There are different kinds of queries which are used in CBIR, such as point query, range query, k nearest neighbour query, etc. Point query requires to find a vector (image) in the database. Range query requires to find all vectors in the database which are located in a pre-defined region (normally in the form of a rectangle). While k nearest neighbour query requires to find the k vectors which are most similar to the input vector among all the vectors of the database.

The simplest solution for retrieval is to use linear search or sequential search by comparing successively the feature vector of the query image with the feature vector of every other image in the database. Images in the database are then sorted in descending order of similarity and returned to the user. Linear search is very simple to implement, and is useful when the database has only few images. But it is not suitable for large databases due to the high number of comparisons needed.

The most frequently used solution in the case of large databases is to organize data objects by an indexing method (space partitioning method or data partitioning method (see Section 2.3)) and to navigate in the structured feature space for finding suitable images.

For finding a vector in an indexed structure, we have to locate the cells or

regions containing the input vector in this structure. For grid-based structure such as grid file, the cell containing the input vector is determined by scanning the linear scales. For tree-based structure such as k-tree, k-d-tree, k-d-b-tree, R-tree, SS-tree, SR-tree, we go down from the root and select at each level the nodes whose region contains the vector. As space partitioning methods divide the space into disjoint cells, only one cell has to be checked at each step of the retrieval phase, while many cells may be suitable in the case of data partitioning methods due to the overlapping between cells. The search ends when the vector is found or when there are no more cell or node to check.

The retrieval by range query is similar to the retrieval by point query, but we have to find the results in the cells or regions which overlaps with the region defined in the input query.

Searching for a k-nearest neighbour query in an indexed structure can proceed as follows:

1. Randomly select k points as the current k-nearest neighbours.
2. *Repeat*:
   (a) Compute the distance *Dist* between the input point and the farthest point among the current k-nearest neighbour points; and define the "result region" as the hyper-sphere of radius *Dist* around the search point.
   (b) The algorithm checks whether there could be any points in the indexed structure that are closer to the input point than the current k-nearest neighbours. Conceptually, these points may be found in the cells overlapping with the "result region". The algorithm locates in the indexed structure the "current cell" as the nearest cell which is not visited and overlaps with the "result region".
   (c) If there are any points in the "current cell" which are closer to any points in the current k-nearest neighbours $\rightarrow$ update the list of the current k-nearest neighbours.

   until there is no more cell to check.

Note that for tree-based structure, by going down from the root, the algorithm first locates the "current cell" as the leaf node which is nearest to the input point and set the "current node" as the father node of this leaf. Then, it finds the "current cell" among the leaves of the "current node" before walking up the tree to find in the other branch.

We can see that by organizing feature vectors in an indexed structure, we do not have to compare the input vector with every other vectors in the database, but in general only with the vectors in a small number of cells. Therefore, this strategy is more suitable to large databases than linear search.

2.4.2 Dissimilarity measures

The performance of the dissimilarity measure depends on the extracted features and may differ among different databases. Different dissimilarity measures have been presented in the literature [79, 107]. They are divided into geometric measures, information theory measures and statistical measures.

Geometric measures Geometric measures are among the most widely used in CBIR systems. They consider images as n-dimensional feature vectors and compute the dissimilarity between two images by successively measuring the difference between the two corresponding vectors in each dimension.

Having two vectors $X = (x_1, x_2, ..., x_n)$ and $Y = (y_1, y_2, ..., y_n)$, the dissimilarity between them can be calculated by different geometric measures:

- *Minkowski distance* [79]: the Minkowski distance of order p (or $\mathcal{L}_p$ norm) between X and Y is defined as:

$$D_p(X, Y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}} \tag{2.42}$$

  The Minkowski distance can be considered as a general form of the Manhattan distance ($\mathcal{L}_1$), the Euclidean distance ($\mathcal{L}_2$) and the Chebyshev distance ($\mathcal{L}_\infty$).

- *Canberra distance (d_{can})* [71]: the Canberra distance is a normalized variant of the Manhattan distance where the absolute difference between two vectors in each dimension is divided by the sum of the absolute values of two vectors in this dimension. The Canberra distance is computed as follows:

$$d_{can}(X, Y) = \sum_{i=1}^{n} \frac{|x_i - y_i|}{|x_i| + |y_i|} \tag{2.43}$$

- *Cosine based dissimilarity (d_{cos})* [79]: the Cosine similarity (s_{cos}) measure the similarity between X and Y by calculating the cosine of the angle between them, which determines whether two vectors point in the same direction.

$$s_{cos}(X, Y) = \cos\theta = \frac{X.Y}{|X|.|Y|} = \frac{\sum_{i=1}^{n} x_i \times y_i}{\sqrt{\sum_{i=1}^{n}(x_i)^2} \times \sqrt{\sum_{i=1}^{n}(y_i)^2}} \tag{2.44}$$

  and the corresponding dissimilarity is computed as:

$$d_{cos}(X, Y) = 1 - \cos\theta = 1 - \frac{\sum_{i=1}^{n} x_i \times y_i}{\sqrt{\sum_{i=1}^{n}(x_i)^2} \times \sqrt{\sum_{i=1}^{n}(y_i)^2}} \tag{2.45}$$

Information theory measures Information theory measures consider images as probabilistic distributions and use different divergence measures for calculating the dissimilarity between images. Thus, these measures can not be used with features having negative values. Moreover, for each probabilistic distribution $X = (x_1, x_2, ..., x_n)$, we have $\sum_{i=1}^{n} x_i = 1$.

- *Kullback-Leibler (K-L) divergence (d_{kl})* [99]: the K-L divergence is a non-symetric measure which is used to measure the dissimilarity between two probability distributions as follows:

$$d_{kl}(X, Y) = \sum_{i=1}^{n} x_i \log \frac{x_i}{y_i} \tag{2.46}$$

- *Jeffrey divergence (d_{jd})* [79]: the Jeffrey divergence is the symmetric form of the Kullback-Leibler divergence, it measures the difference between two probability distribution as follows:

$$d_{jd}(X,Y) = \sum_{i=1}^{n}(x_i \log \frac{x_i}{m_i} + y_i \log \frac{y_i}{m_i}) \tag{2.47}$$

where $m_i = \frac{x_i+y_i}{2}$.

Statistical measures Statistical measures consider image feature vectors as samples and compare them on the basis of distribution analysis.

- χ^2 *statistics* [108]: the χ^2 statistic measure computes the dissimilarity between two vectors based on the difference between one vector and the mean of both vectors as follows:

$$d_{\chi^2}(X,Y) = \sum_{i=1}^{n} \frac{(x_i - m_i)^2}{m_i} \tag{2.48}$$

where $m_i = \frac{x_i+y_i}{2}$.

- *Kolmogorov-Smirnov distance (d_{ks})* [36]: the Kolmogorov-Smirnov distance measures the dissimilarity between two probability distributions as the maximal absolute difference between two distribution functions:

$$d_{ks}(X,Y) = \sum_{1 \le i \le n} |F_X(i) - F_Y(i)| \tag{2.49}$$

where F_X and F_Y are respectively the probability distribution functions of the first and the second sample. This measure is defined only for one-dimensional probability distributions.

2.4.3 Semantic gap problem

The term "semantic gap" is used to characterize the difference between two levels of description of an object/image. The first description is done in CBIR using feature vectors which only capture low-level information of the image (color, texture, shape, etc.). The second description is done by the human user which uses high-level semantic concepts to retrieve images. As the retrieval in a traditional CBIR system is in general based on the similarities between the low-level features, the result images may not fit the intent of the user.

An approach to overcome the "semantic gap" problem is to combine the visual content information (low-level feature vectors) with other semantic information (high-level information) during the retrieval phase [72,77,97,126]. In his thesis, Nguyen [97] uses the multimodal search combining visual content information with textual information in the form of user annotations representing some semantic descriptions of the images. Combining these information allows to search images which are similar to the query image in both visual content aspect and semantic aspect.

Another approach which is most widely used for the "semantic gap" problem is the relevance feedback approach. This is an interactive technique which is used

to improve the result of a retrieval system according to the interactions between the user and the machine. In a CBIR system, the user is proposed to indicate the relevance of each image in the results to the query image (such as "very relevant", "relevant", "not relevant" or "neutral"). Based on the user feedback, the search is repeated by modifying, for example, the query vector or by updating the similarity measure in order to give results which are more suitable to the user's perception. Among the existing techniques for relevance feedback, we can cite:

- *Query point movement*: This kind of techniques tries to improve the retrieval performance by reformulating the query vector so that the query is moved toward the relevant images and at the same time away from the non-relevant images. The Rocchio's technique [113] is the most widely used for modifying the query vector:

$$q_m = \alpha q_o + \frac{\beta}{|S_r|} \sum_{x_i \in S_r} x_i - \frac{\gamma}{|S_{nr}|} \sum_{x_i \in S_{nr}} x_i \tag{2.50}$$

  where q_o is the original query vector, q_m is the modified query vector, S_r is the set of relevant points and S_{nr} is the set of non relevant points. α, β and γ are successively the weights corresponding to the original query vector, the relevant points and the non relevant points. Typically $\alpha = 1$ and $0 < \beta, \gamma < 1$. In general, the weight of relevant feedback is usually greater than the weight of non-relevant feedback ($\beta > \gamma$).

- *Distance function modification*: This kind of technique considers that certain elements of the feature vector may be more important to the user than others. Therefore, a weight is provided for each dimension in the distance metric and this technique is used to modify these weights according to the user's feedback. Ortega-Binderberger and Mehrotra [100] used in their work the weighted L_p metric to compute the distance between vectors:

$$L_p(X, Y) = \sqrt[p]{\sum_i \omega_i (x_i - y_i)^p} \tag{2.51}$$

  where $\sum_i \omega_i = 1$. Then, they tried to change the weights ω_i according to the user's feedback as follows:

  - Estimate the new weights w_i as the inverse of the standard deviation of dimension i of all relevant vectors given by user. The idea is that a large variation of the relevant vectors in a dimension means that this dimension poorly captures the user's need and should as a consequence have a smaller weight; and *vice versa*.

$$\omega_i^{est} = \frac{1}{\sigma(d_j[i] | d_j \in D_r)} \tag{2.52}$$

    where D_r is the set of relevant vectors.

  - Normalize the weights so that $\sum_i \omega_i^{est} = 1$.

- Combine the estimated weights with the original weights:

$$\omega_i^{new} = \alpha\omega_i + \beta\omega_i^{est} \tag{2.53}$$

where $\alpha + \beta = 1$.

- *Machine learning techniques*: The retrieval task of a CBIR system can be considered as a machine learning problem [22]. The aim is to classify the database into two classes: the relevant class containing images which are similar to the query image and the irrelevant class represents the class of images which are not relevant to the query image. The most relevant images, according to the learned classifier, are returned to the user. The user can annotate the resulted images as relevant or irrelevant images. The annotated images can then be added in the training set for the learning process in the next iteration. The learning process aims at estimating the parameters of the classifier in order to reduce the different between the user's labels and the classifier's labels, and thus reduce the "semantic gap". The learning process assigns also a degree of relevance to each image in the database in order to provide the most relevant images to the user. Supervised learning [118] [51] or semi-supervised learning [82] [50] techniques can be used for the learning process. While supervised learning used only labeled data for the training process, semi-supervised learning incorporate also unlabeled data into the training set. Nowadays, active learning techniques [104] [38] [50] [56] [21] [127] [39] are increasingly used for content-based image retrieval problem. In each interactive iteration, the active learning model returns to the user not only the list of the most relevant images, but also the *pool query set* containing unlabelled images which seem to be the most informative images that should be annotated by the user to improve the most the classification process. For instance, images in the pool query set can be the ones about which the system is most unsure (*e.g.* images which are close to the class boundary).

2.5 Discussion

With the aim of manipulating large image databases, feature space structuring methods are necessary for facilitating and accelerating further retrieval. Indeed, the complexity for searching similar images in the indexed structured is generally much less than in the case of the linear search as we do not have to compare the input vector with every other vectors in the database, but only with the vectors in a small number of cells of the indexed structured. As described in Section 2.3, traditional indexing methods can be divided into two categories: space partitioning methods and data partitioning methods. Space partitioning methods partition the feature space into disjoint cells of restricted cardinality (in terms of number of objects per cell) or into regular disjoint cells (in terms of size of cell) by different hyperplanes. Therefore, dissimilar points may be included in a same cell while similar points may end up in different cells. The resulting index is therefore not optimal for retrieval, as the user generally wants to retrieve similar images to the query image. Moreover, by dividing the whole space into cells, there may have many empty or almost empty cells, especially in the case of large dimensionality, which is generally the case of

image feature vectors that commonly count hundreds of elements. That leads to poor storage utilization of these indexed structures. Data partitioning methods partition the data objects into overlapping regions. There are no empty cells, and therefore, that helps to improve the storage utilization. However, the limitations on the cardinality of the cells remain; their parameters are difficult to tune causing the resulting index to be generally non-optimal for retrieval, especially in the case where different groups of similar objects are unbalanced, i.e. composed of very different numbers of images.

As the traditional indexing methods generally provide non-optimal indexed structured for retrieval due to the limitations on the cardinality of the cells, especially in the case where the database contains unbalanced groups of similar objects, this restricts the use of indexing methods in many applications. Our claim is that using clustering methods instead of traditional indexing methods to organize feature vectors results in indexed structures which may be more adapted to high dimensional and unbalanced data. Indeed, clustering aims at splitting a collection of data into groups (clusters) so that similar objects belong to the same group and dissimilar objects are distributed in different groups, without any constraint on the cluster size. As in traditional indexing, clustering methods also group objects into groups based on the similarity between them, therefore, during the retrieval phase, we do not have to compare the input vector with every other vectors in the database, but only with vectors in a small number of groups which are similar to the input vector. Clustering is thus suitable for indexing large databases. Moreover, while in traditional indexing methods it might be difficult to fix the number of objects in each bucket (especially in the case of unbalanced data), clustering methods have no limitation on the cardinality of the clusters, since objects in the database can be grouped into clusters of very different sizes. Therefore, clustering can be used for indexing both balanced or unbalanced data. Furthermore, by using clustering, we can avoid the empty cell problem of the space partitioning methods in the case of high dimensional data, which leads to poor storage utilization of the indexed structures. With these advantages of clustering methods compared to traditional indexing methods, in this thesis we have decided to use the clustering instead of traditional indexing for indexing database objects.

Because of the "semantic gap" between high-level semantic concepts expressed by the user and the low-level features extracted from the images, the clustering results and also the resulting images of the retrieval phase may therefore generally be quite different from the intent of the user. While existing CBIR systems allow the user to give relevance feedback about resulting images, in this work, we try to involve the user sooner in the clustering phase so that the user could interact with the system in order to improve the clustering results. The idea is that a good clustering structure may lead to high performance of the retrieval phase. Moreover, involving the user in the clustering phase requires less user's effort than involving the user later in the retrieval process, as the user might interact with a small number of cluster prototypes rather than with numerous single images. Therefore, the next chapter presents different unsupervised clustering methods and analyzes their possibilities to be used in an interactive context.

2.6 Summary of the chapter

This chapter presents a structured survey of the Content-Based Image Retrieval (CBIR) approaches. It presents the general model for a CBIR system, consisting of the three principal phases: feature extraction, feature space structuring and retrieval. The principal techniques used in different phases of a CBIR system are also presented in detail, especially the feature space structuring techniques. We also present in this chapter the "semantic gap" problem, which explains the difference between the retrieval results and the wishes of the user. We propose an overview of the existing approaches, generally used in the retrieval phase, for solving this semantic gap problem.

Our contribution in this chapter is an analysis of the advantages of using clustering methods instead of traditional feature space structuring methods in the structuring phase of large image databases. The aim is to obtain an indexed structure more adapted to the retrieval of high dimensional and unbalanced data. For reducing the semantic gap between the high-level semantic concepts expressed by the user and the low-level features extracted automatically from the images, we propose to involve the user, not in the retrieval phase, but in the clustering (indexing) phase so that he could interact with the system in order to obtain an indexed structure closer to the user wishes.

Chapter 3

Unsupervised clustering

3.1 Introduction

Clustering is one of the most popular techniques in data mining. The objective of the data mining process is to analyze a mass of data and to extract the hidden information in these data which is useful for further use. For the clustering problem, the hidden information is the groups of data objects. Clustering aims at splitting a collection of data into groups (clusters) so that similar objects belong to the same group and dissimilar objects are in different groups, without any constraint on the cluster size. As analyzed in Chapter 2, we aim at using clustering instead of traditional indexing to organize feature vectors; our objective is also to involve the user in the clustering phase in order to improve the clustering results according to the intent of the user. In our context where the user is involved in the clustering of large image database, we are interested by the following criteria of the clustering methods:

- *Hierarchical structure*: The clustering methods should produce a hierarchical cluster structure where the initial clusters may be easily merged or split.

- *Suitability for large image databases*: the system must be able to tackle large image databases.

- *Incrementality*: We are also interested by clustering methods which can be incrementally built in order to facilitate the insertion or deletion of images by the user. It can be noted that incrementality is also very important in the context of huge image databases, when the whole data set cannot be stored in the main memory.

- *Complexity*: The computational complexity is another very important point of the clustering algorithm, especially in an interactive context where the user is involved in the clustering of large databases.

In the context of large image database indexing, we may be interested by traditional clustering (unsupervised) [57,140] or semi-supervised clustering [9,10,26,136]. While no information about the ground truth is provided in the case of unsupervised clustering, a limited amount of knowledge is available in the case of semi-supervised clustering. In our interactive clustering loop, the objects in the database are first

organized into groups by an initial unsupervised clustering during which no information about any ground truth is provided. After receiving the feedbacks of the user, another semi-supervised clustering may be used to re-organize the objects according to the intent of the user.

In this chapter, we present a survey of unsupervised clustering techniques and analyze the advantages and drawbacks of different unsupervised clustering methods in the context of huge masses of data where incrementality and hierarchical structuring are needed; semi-supervised clustering techniques are studied in the next chapter. This chapter is organized as follows. Section 3.2 analyzes different clustering methods. A formal comparison of these methods is presented in Section 3.3. Section 3.4 presents some measures which are generally used for evaluating clustering results. An experimental comparison of some unsupervised clustering methods are analyzed in Section 3.5. Finally, a conclusion and some discussions are given in Section 3.6.

3.2 Major existing unsupervised clustering methods

There are currently many unsupervised clustering methods that allow us to aggregate data into groups based on the proximity between points (vectors) in the feature space. Unsupervised clustering can be divided in hard clustering and fuzzy clustering. Hard clustering methods assign each object to only one cluster, while with fuzzy clustering methods, an object can belong to one or more clusters with different degrees of membership. In this thesis, we are interested in only hard clustering methods. Indeed, as a first attempt to involve the user in the clustering process, we consider that the user assigns some well-chosen images to one cluster only, and therefore hard clustering is more adapted in that context. Fuzzy clustering might be studied in further work. Different kinds of hard clustering methods have been proposed in the literature:

- Partitioning methods partition the data set based on the proximities between the images in the feature space. These methods give in general a "flat" (*i.e.* non hierarchical) organization of clusters.

- Grid-based methods *a priori* partition the space into cells without considering the distribution of the data and then group neighbouring cells to create clusters. The cells may be organized in a hierarchical structure or not.

- Density-based methods aim at partitioning a set of points based on their local densities. These methods may give a "flat" or hierarchical organization of clusters.

- Hierarchical methods organize the points in a hierarchical structure of clusters.

The most famous methods of each of these categories are presented in this section. For being used in our interactive context, each method is analyzed according to the criteria presented in Section 3.1. We use the following notations:

- $X = \{x_i | i = 1, ..., N\}$: the set of input feature vectors for clustering, where N is the number of vectors.
- $K = \{K_j | j = 1, ..., k\}$: the set of clusters, where k is the number of clusters.

3.2.1 Partitioning methods

Partitioning methods are intended to partition the data set into k partitions corresponding to k clusters, where k is usually predefined. In general, these methods give a "flat" organization of clusters (no hierarchical structure). Some methods of this kind are: k-means [85], k-medoids [63], CLARA [62], CLARANS [96], ISODATA [8], SOM [66].

K-means [85] 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 (according to the Euclidean distance). The idea is to minimize an objective function which is the within-cluster sum of squares computed as:

$$J_{obj} = \sum_{j=1}^{k} \sum_{x_i \in K_j} \|x_i - \mu_j\|^2 \tag{3.1}$$

where μ_j is the mean of points in cluster K_j:

$$\mu_j = \frac{1}{|K_j|} \sum_{x_i \in K_j} x_i \tag{3.2}$$

where $|K_j|$ is the number of points assigned to the cluster K_j.

The k-means algorithm has the following steps:

1. Initialize k means μ_j of clusters (*e.g.* by randomly choosing k points in the data set).
2. *Repeat*:
   - Assign each point x_i to the cluster with the nearest mean μ_j.
   - Recalculate the means μ_j of the k clusters using Equation (3.2).

   until there is no change in any mean, or until a predefined number of iterations has been reached.

K-means is very simple to implement. It works well for compact and hyperspherical clusters (partly because of the Euclidean distance being used) and it does not depend on the processing order of the data. Moreover, it has a 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. And the complexity for finding the cluster to which a new point should be assigned is O(k). In practice, l and k are usually very small compared to N, so that k-means can be considered as linear to the number of elements. K-means is therefore effective for large databases.

On the other hand, k-means does not work well when clusters have non-globular shapes and/or when data points of different clusters are overlapping. It can produce almost empty clusters when outliers are chosen as initial means. Furthermore, k-means is not incremental, does not produce any hierarchical structure, and it is very sensitive to the initial means and to the outliers; it can converge to a local optimum and its results depend on the value of k.

There are several variants of k-means such as k-harmonic means [142], global k-means [76], 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 the number of clusters $i = 1, ..., k$. In step i, we set the i initial centroids of clusters as follows:

- *i-1* centroids returned by the k-means algorithm in step *i-1* are considered as the first *i-1* initial centroids in step i.
- The position of the i^{th} centroid is initialized at the point x_h of the database that maximizes b_n:

$$b_n = \sum_{j=1}^{N} \max(d_{i-1}^j - \|x_n - x_j\|^2, 0) \tag{3.3}$$

$$h = \underset{n}{\operatorname{argmax}}\, b_n \tag{3.4}$$

  where d_{i-1}^j is the minimum squared distance between x_j and one of the *i-1* centroids found in the previous iteration. Thus, b_n measures the possible reduction of the error obtained by inserting a new centroid at position x_n.

After having i initial centroids, the traditional k-means algorithm is executed to obtain the solution with i clusters.

Global k-means is not sensitive to initial conditions, it is more effective than k-means [76], 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 Section 3.4).

K-medoids [63] The k-medoids method is similar to the k-means method, but instead of using centroids 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 and outliers. The k-medoids algorithm can be used with an arbitrary distance, most commonly Euclidean distance or Manhattan distance. This method and other methods using medoids are expensive because the calculation phase of medoids has a quadratic complexity. Thus, it is not compatible in 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) [63] is the most common realisation of k-medoids clustering. Starting with an initial set of medoids, we iteratively replace

[1]The medoid of a cluster is defined as the object in the cluster which has the minimal average distance with the other objects in the cluster.

one medoid by a non-medoid point if that operation decreases the *overall distance*, *i.e.* the sum of distances between each point in the database and the medoid of the cluster to which it belongs. PAM therefore contains the following steps:

1. Randomly select k points as k initial medoids.
2. *Repeat:*
   (a) Associate each point to its nearest medoid.
   (b) For each pair $\{m, o\}$ (m is a medoid, o is a point in the database which is not a medoid):
      - Exchange the role of m and o and calculate the overall distance of the new configuration when m is a non-medoid and o is a medoid.
   (c) Select the configuration with the minimum *overall distance*.

   until there is no change in the medoids.

The results of PAM do not depend on the processing order of the data. But because of its high complexity $O(k(N-k)^2 l)$, where l is the number of iterations, PAM is not suitable to the context of large image databases. Similar to k-means, PAM has a low complexity of $O(k)$ for finding the cluster to which a new point is assigned. Like every variant of the k-medoids algorithm, PAM is not compatible with the incremental context either, and provides flat clustering (*i.e.* no hierarchical structure). Its results depend on the value of k. Outliers in a cluster can be defined as points having the distance from the medoid greater than 1.5 times the average distance between the medoid and each point of the cluster [95].

CLARA [62] The idea of Clustering LARge Applications (CLARA) is to apply the PAM algorithm with only a small sample of the data set instead of with the entire data set to avoid the high complexity of PAM. The objects in the sample are randomly chosen. After applying PAM for finding optimal medoids for the sample, the other points which are not in this sample will be assigned to the cluster with the closest medoid. The idea is that, when the sample is chosen randomly, the medoids of this sample would approximate the medoids of the entire data set. PAM is applied several times, each time with a different part of the data set, to avoid the dependence of the algorithm on the selected part. The set of medoids with the lowest overall distance computed from the whole database is chosen as the final clustering result. Assume N_S to be the number of objects in a sample and q to be the number of samplings. The CLARA algorithm is as follows:

1. For i from 1 to q, do:
   (a) Create a sample S by choosing N_S objects randomly from the data set.
   (b) Apply PAM on the sample S.
   (c) Assign each point in the database to the cluster with the closest medoid.
   (d) Compute the overall distance of the current solution.
2. Select the partition with the lowest average distance.

Due to its lower complexity of $O(q(kN_s^2 l + k(N-k)))$, where l is the number of iterations when applying PAM on the sample S, CLARA is more suitable than PAM in the context of large image databases. The complexity for finding the cluster for assigning a new point is also $O(k)$. Its result is dependent on the randomly selected samples, the number of iterations q and the value of k. Moreover, it may converge to a local minimum. Similar to other partitioning methods, CLARA does not produce any hierarchical structure. It is more suitable to the incremental context because when there are new points added to the system, these points can be directly assigned to the cluster with the closest medoid, considering they are outside of the sample S. The results of CLARA do not depend on the processing order of the data and outliers can be detected by the same strategy used for PAM.

CLARANS [96] Clustering Large Application based upon RANdomize Search (CLARANS) is based on the search through a graph $G_{N,k}$ for finding k medoids. In this graph, each node corresponds to a set of k objects (O_{m_1},O_{m_2},...,O_{m_k}) of the data set representing k selected medoids. Each node is associated with a cost representing the average distance (between all points in the database and their closest medoids) corresponding to the partition where the k points of this node are selected as k medoids. Two nodes are neighbours if they differ by only one medoid, each node has therefore $k(N-k)$ neighbours. The whole set of nodes of the graph represents the set of all possible choices of k points in the database as k medoids. Similar to CLARA, CLARANS does not search on the entire graph, but in the neighbourhood of some chosen nodes. Beginning from a randomly selected node, CLARANS checks at most *maxneighbour* randomly selected neighbours of this node, and if a better neighbour (neighbour with lower cost) is found, it moves to this neighbour. The process continues until finding a local minimum which is the node having the lowest cost among its selected neighbours. Then, the algorithm continues to search for another local minimum from a new randomly selected node. After finding a number of local minimums, the local minimum having the lowest cost will be returned as the clustering result.

Assume *numlocal* to be the number of local minimums to consider and *maxneighbour* the maximum number of neighbours to be checked of each node. The CLARANS algorithm is as follows:

1. Initialize $mincost = \infty$.
2. For i=1 to *numlocal* do:
   (a) Randomly set *current* to a node in $G_{N,k}$.
   (b) Set j=1
   (c) While $j <$ *maxneighbour* do
      i. Randomly choose a neighbour S of *current*.
      ii. If S has a lower cost than *current* then
         - current = S
         - j=1

      else $j = j + 1$
   (d) If the cost of *current* is less than *mincost* then

- $mincost = cost(current)$
- $bestNode = current$

3. Return $bestNode$ as result.

We can see that steps 2.a to 2.c search a local minimum from a randomly selected node, while step 2.d compares the cost of the current local minimum with the best cost obtained so far.

CLARANS has been shown to be more effective than both PAM and CLARA [96]. It is also more robust for outlier detection than PAM and CLARA [95]. Its complexity for finding a cluster for assigning a new object is $O(k)$. However, its time complexity is $O(N^2)$. Therefore, it is not quite effective in very large data set. Furthermore, the clustering result depends on the randomly selected neighbours, the value of k, and the other parameters (*numlocal* and *maxneighbour*). CLARANS is sensitive to the processing order of the data. It is not suitable to the incremental context because the graph changes when new elements are added. As a partitioning method, CLARANS produces a "flat" structure of clusters.

ISODATA [8] The Iterative Self-Organizing Data Analysis Techniques (ISODATA) is an iterative clustering method which is an extension of the k-means method with some heuristics to revise the number of clusters. At first, it randomly selects k initial cluster centres and assigns all the points in the database to the nearest centre using the k-means method. Then, it can eliminate clusters with too few items, split clusters whose points are sufficiently dissimilar or merge clusters which are sufficiently close. Further iterations can be performed with new cluster centres.

Assume N_{minEx} to be the minimum number of points per cluster, N_C to be the desired number of clusters, D_{merge} to be the threshold for merging cluster and σ_{split} to be the threshold for splitting. The ISODATA algorithm is as follows:

1. Initialize the number of clusters k (normally less or equal to the desired number of clusters N_C specified by the user) and randomly select k points as cluster centres μ_j, $j = 1, ..., k$.

2. Assign points to clusters using the k-means algorithm.

3. Eliminate clusters containing less than N_{minEx} points, either consider their points as outliers or assign their points to the closest cluster among the remaining clusters and decrease k accordingly.

4. Compute, for each cluster j:

   - The mean μ_j
   - The average distance d_{avg_j} between points in cluster j and μ_j.
   - The standard deviation of each axis (attribute in the feature space), and d^* is the axis with the maximum deviation $\sigma_j(d^*)$.

   and compute the overall average distance d_{avg} between all objects in the database and the mean of their clusters.

5. Split each cluster j having the maximum standard deviation ($\sigma_j(d^*)$) greater than the threshold σ_{split} if (a) or (b) is true and update k accordingly.
   (a) there are too few clusters ($k < N_C/2$).
   (b) the number of objects in cluster j exceeds ($2N_{minEx}+1$) and the average distance between its centre and its objects is greater than the overall average distance between all objects in the database and their cluster centres ($d_{avg_j} > d_{avg}$).
6. Merge the closest clusters if the distance between them is less than the threshold D_{merge} and update k accordingly. Note that the merge cannot be done if there are two few clusters ($k < N_C/2$).
7. Go to 2.

The algorithm stops when the maximum number of iterations is reached or when the average distance between cluster centres does not significantly change between iterations.

The advantage of ISODATA is that it is not necessary to permanently set the number of clusters. Similar to k-means, ISODATA has a low storage (space) complexity of $O(N+k)$ and a low computational (time) complexity of $O(Nkl)$, where N is the number of objects and l is the number of iterations. The complexity for finding the cluster for a new point is $O(k)$. It is therefore compatible with large databases. Furthermore, the clustering result does not depend on the processing order of the data. ISODATA can also detect outliers as points in clusters with too few items. But its drawback is that it relies on numerous thresholds which are highly dependent on the size of the database and the data distribution and which are therefore difficult to settle. As other partitioning methods, ISODATA does not produce any hierarchical structure.

SOM [66] In Self-Organizing Map (SOM) (also called Kohonen map), similar points are grouped by a mono-layer neural network the output layer of which contains nodes (neurons) representing the clusters. The neurons are connected to each other via a neighbourhood topology (or structure) of the map. Usually, these neurons are organized in the form of rectangular or hexagonal topology. Each output neuron is associated with a weight vector of the same dimension as the input data, representing the centroid of a cluster, and a position in the map space. SOM aims at mapping from high dimensional input data space to the map space of lower dimensional by finding, for each input data, the output node associated with the closest weight vector.

The SOM clustering algorithm is as follows:

1. Randomly initialize the weight vectors of all the output neurons.
2. Repeat until the weight vectors stop changing:
   (a) Randomly choose an input vector in the database and compute the distances between the input vector and all the nodes in the output layer, the node associated with the nearest weight vector being called the winner (or Best Matching Unit (BMU)).

(b) Update the weight vectors of the BMU and the neurons in its neighbourhood to move them towards the input vector. The weight vector w_i of the neuron i is updated as follows:

$$w_i(t+1) = w_i(t) + \alpha(t)\phi_{ci}(t)[x(t) - w_i(t)] \tag{3.5}$$

where t is the step index, c is the index of the BMU for the input vector $x(t)$, $\alpha(t)$ is the learning coefficient which monotonically decreases with time and $\phi_{ci}(t)$ is the neighbourhood function.

The neighbourhood function $\phi_{ci}(t)$ decreases with the distance on the map between the BMU c and the neuron i so that weight vectors of the neurons which are distant from the BMU are less moved than the neurons which are near to the BMU. The neighbourhood function also shrinks with time so as to re-estimate the weight of the neurons in a large neighbourhood at the beginning and in a small neighbourhood in further learning iterations.

SOM is incremental, as the weight vectors can be updated when new data arrive. But for this method, we have to *a priori* fix the number of neurons, and the rules of influence of a neuron on its neighbours. The result depends on the initialization values and also the rules of evolution concerning the size of the neighbourhood 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 $O(Nkl)$, and its complexity for finding a cluster for a new point is $O(k)$, where k is the number of neurons (or clusters), l is the number of training iterations and N is the number of objects. As l and k are usually much smaller than the number of objects, SOM can be considered as linear with the number of objects, and therefore is adapted to large databases. But, it does not provide any hierarchical structure and the clustering result depends on the value of k.

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

3.2.2 Grid-based methods

Grid-based clustering partitions the space into cells which form a grid structure and then groups neighbouring cells to form clusters. In some case clusters may be organized in a hierarchical structure. Some methods of this kind are: STING [137], CLIQUE [3], WaveCluster [120], etc.

STING [137] STING (STatistical INformation Grid) is a grid-based clustering technique. It divides the feature space into rectangular cells and organizes them according to a hierarchical structure, where each cell (except the leaves) is divided

into a fixed number of cells. For instance, for two-dimensional data, each cell at a higher level is partitioned into 4 smaller cells at the lower level (see Figure 3.1). The root cell corresponds to the whole portion of the feature space containing data, while data objects are stored in leaf cells. The size of the leaf level cells and also the number of layers depend on the density of objects. In general, the size of the leaf cell is determined such that the average number of objects in each leaf is in a range determined by the user.

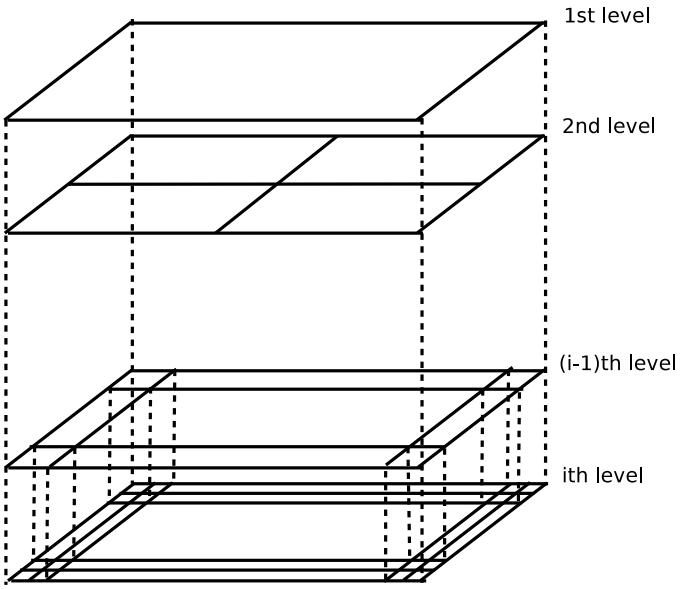


Figure 3.1 – Hierarchical structure in STING clustering.

The idea of STING is to pre-compute and to store statistical information associated with each grid cell in order to further answer data mining queries and clustering problems without recourse to the individual objects. For the clustering problem, an independent parameter n which is the number of objects in each grid cell is needed to be stored. In each grid cell, STING could also capture, for each attribute, some attribute-dependent parameters describing this attribute (such as mean, standard deviation, maximum and minimum values, etc.) for answering data mining queries.

The statical information (parameters) of the grid cells is computed when loading the objects from the database. For the cells at the lowest level (leaves), we directly calculate the parameters from the data. Parameters of the cells at higher level can be derived from parameters of lower level cells as in [137].

With the hierarchical structure of grid cells, a top-down approach can be used to answer the data mining queries or to detect dense cells which form clusters. Starting with a high level layer (normally the root), we examine if each cell of the current layer is *relevant* or not. For the clustering problem, STING uses a density parameter *dent*, and labels each cell as *relevant* (dense) if $n \geq S \times dent$, where n and S are, respectively, the number of points in this cell and the area of a leaf cell. After finishing the current layer, we go down to the next lower layer and repeat the same process, but only for cells which are children of the cells labelled as relevant in the previous step. This procedure continues until the bottom layer is examined. Then, regions are formed based on the relevant (dense) leaf cells and are

returned as clusters as follows. For each relevant leaf cell which is not examined, STING examines cells in an area within a certain distance from the current relevant cell. If the average density within this area is greater than the density *dent*, this area is marked and all non-examined relevant cells within this area are put into a queue. Then, STING takes each relevant cell from the queue and repeat the same procedure. When the queue is empty, we obtain a region which forms a cluster. The distance used above is calculated as $d = \max(l, \sqrt{\frac{f}{dent \times \pi}})$, where l and f are respectively the side length of the leaf cell, and a small constant number fixed by STING.

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.* [137] demonstrated that STING outperforms the partitioning method CLARANS (see page 52) as well as the density-based method DBSCAN (see page 59) when the number of points is large. As the grid can cover the whole feature space, points could be inserted or deleted by updating the parameters of the corresponding cells in the tree and STING can be used in an incremental context. The complexity for finding the cluster for assigning a new point is $O(B)$ where B is the number of cells at the bottom level. STING does not depend on the processing order of the data and it is able to detect outliers based on the number of objects in each cell. The result of STING does not depend on the value of k, but on the density parameter *dent*, the size of the leaf cell and also the constant f which are difficult to fix.

CLIQUE [3] CLustering In QUEst (CLIQUE) is dedicated to high dimensional databases. In this algorithm, the d-dimensional feature space is partitioned into cells of the same size by a grid, each dimension is partitioned into ξ intervals of equal lengths, where ξ is a constant and corresponds to an input parameter. Then clusters are determined by maximal sets of connected dense cells in d-dimensions. A cell is dense if the number of points in this cell is greater than a *density threshold* τ which is another input parameter. Two cells c_1 and c_2 are connected if they have a common face, or if there is another cell c_3 such that c_1 is connected to c_3 and c_2 is connected to c_3.

CLIQUE uses a bottom-up algorithm for determining dense cells in d-dimensional space. The idea 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 and then all 2-dimensional dense cells, etc., as follows. Having obtained (*k-1*)-dimensional dense cells, the set C_k of k-dimensional candidate dense cells are determined by joining the (*k-1*)-dimensional dense cells. Then, this algorithm discards from C_k the cells which have a projection in (*k-1*)-dimensions which is not included in C_{k-1}. Finally, by parsing all data, only those candidates that are really dense are kept in C_k. In Figure 3.2, the two dimensional space (x,y) has been partitioned by a 6×6 grid. Assuming $\tau = 8$, if the points are projected on the x dimension, there are two 1-dimensional dense cells $X1 = (0.2 \leq x < 0.3)$ and $X2 = (0.4 \leq x < 0.5)$. And if the points are projected on the y dimension, there are also two 1-dimensional dense cells $Y1 = (0.1 \leq y < 0.2)$ and $Y2 = (0.4 \leq y < 0.5)$. By joining the 1-dimensional dense cells, we obtain four 2-dimensional candidate

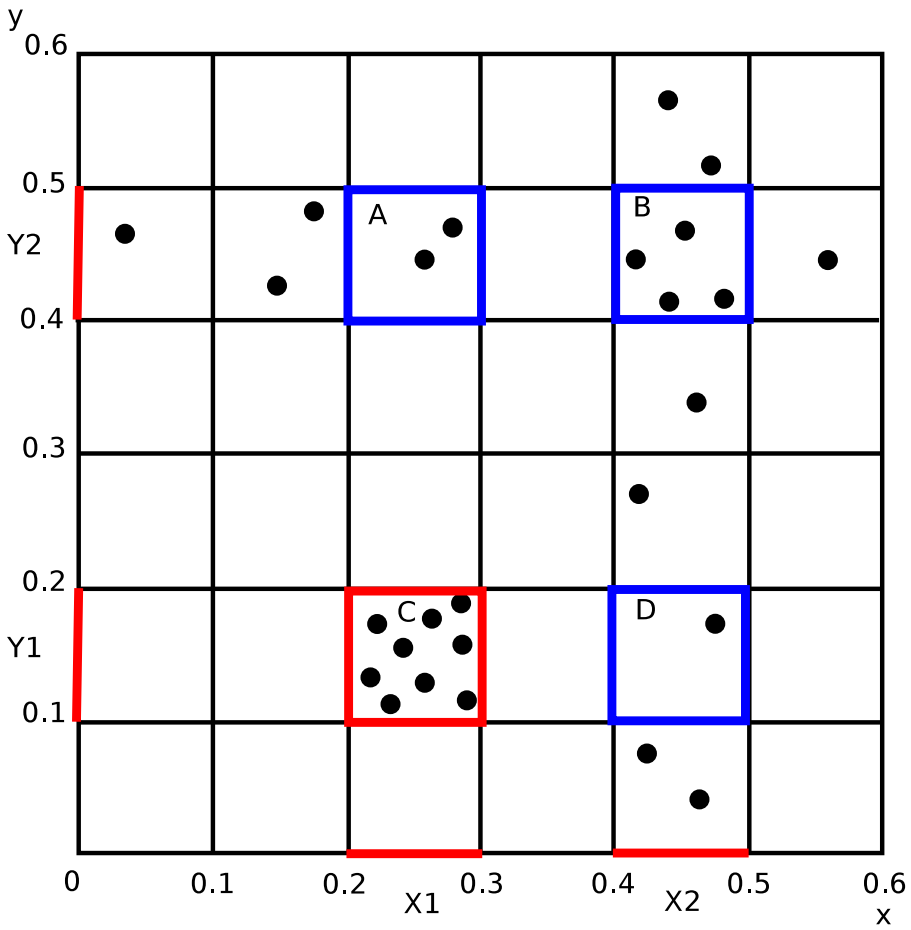


Figure 3.2 – CLIQUE clustering example.

dense cells $A = (0.2 \leq x < 0.3, 0.4 \leq y < 0.5)$, $B = (0.4 \leq x < 0.5, 0.4 \leq y < 0.5)$, $C = (0.2 \leq x < 0.3, 0.1 \leq y < 0.2)$ and $D = (0.4 \leq x < 0.5, 0.1 \leq y < 0.2)$. By parsing all data for verifying the four 2-dimensional candidate dense cells, only the cell C which is really dense is kept.

CLIQUE 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. If they are dense, we try to connect these cells with existing clusters or create new cluster if they correspond to isolate dense cells. Therefore, CLIQUE could be used in an incremental context. Its computational complexity is $O(Nd + k^d)$, where d is the number of dimensions. It is thus suitable for large databases but not suitable for very high dimensional data. The complexity for finding cluster for a new point is $O(1)$. The outliers may be detected by determining the cells which are not dense. However, CLIQUE does not produce any hierarchical structure of clusters. The result of CLIQUE does not depend on the value of k but depends on the interval length ξ, which highly depends on the data distribution.

Conclusion for the grid-based methods 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. However, in high dimensional context, data is generally extremely sparse, and therefore it is difficult to specify the density parameter for

determining dense cells. In this case, the hierarchical methods are better than grid-based methods.

3.2.3 Density-based methods

Density-based clustering methods aim at partitioning a set of vectors based on the local density of these vectors. Each vector group which is locally dense is considered as a cluster. Points in sparse areas are normally considered as noise points or border points. Some methods of this kind are DBSCAN [27], OPTICS [4], DENCLUE [49] and EM [89].

DBSCAN [27] Density Based Spatial Clustering of Applications with Noise (DBSCAN) is the most popular density-based clustering method. 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-neighbourhood* of a point p, denoted as $N_\varepsilon(p)$, contains all the points q, whose distance $D(p,q) < \varepsilon$.
- *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-neighbourhood*.
- *Directly density-reachable*: a point p is directly density-reachable from a point q if q is a core point and $p \in N_\varepsilon(q)$.
- *Density-reachable*: a point p is density-reachable from a core point q if there is a chain of points p_1,...,p_n such that $p_1 = q$, $p_n = p$ and for $\forall i = 1, ..., n-1$, p_{i+1} is directly density-reachable from p_i.
- *Density-connected*: two point p and q are density-connected 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. Label all points in the database as unclassified.
2. For each unclassified point x_i:
   (a) If x_i is a core point:
      i. Assign x_i to a new cluster.
      ii. Expand the new cluster of x_i so that it contains all points which are density-reachable from x_i.
   (b) If x_i is not a core point, we label x_i as NOISE.

We can see that a point which has been marked as NOISE may be assigned later to a cluster if it becomes density-reachable to a point in this cluster.

Differing from k-means for instance, DBSCAN 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. The computational complexity of this method being low: $O(NlogN)$, DBSCAN is suitable to large data sets. The complexity for finding a cluster for assigning a new point is $O(logN)$. But on the other hand, the parameters ε and $MinPts$ are difficult to adjust and this method does not generate clusters of different levels of scatter because of the parameters being fixed. The points which are on the edge of two clusters may change their cluster membership depending on the processing order of the points. Moreover, the DBSCAN fails for identifying clusters if the density varies inside the cluster or if the data set is too sparse. This method is therefore not adapted to high dimensional data. DBSCAN does not produce any hierarchical structure of clusters. And it is difficult to used this method 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 [4] Ordering Points To Identify the Clustering Structure (OPTICS) is an extension of the DBSCAN algorithm. As DBSCAN, OPTICS has two parameters ε and $MinPts$, but instead of working with a single value ε, it works with an infinite number of distance parameters ε_i ($0 \leq \varepsilon_i \leq \varepsilon$), which allows to obtain clusters with different scatters (densities). We first introduce two definitions:

- *Core_ distance*: the *core_ distance* of a point p is the minimum distance $\varepsilon' \leq \varepsilon$ such that p would be a core point with respect to ε'.

$$core\_distance(p) = \begin{cases} UnDefined,\ if\ |N_\varepsilon(p)| < MinPts \\ MinPts\_distance(p),\ otherwise \end{cases}$$

  where $|N_\varepsilon(p)|$ is the cardinality of $N_\varepsilon(p)$ and $MinPts\_distance(p)$ is the distance from p to its $MinPts^{th}$ nearest neighbour.

- *Reachability_ distance*: the *reachability_ distance* of a point p with respect to another point o is the minimum distance $\varepsilon' \leq \varepsilon$ such that p is directly density-reachable from o with respect to ε'. In this case, the reachability_distance cannot be smaller than the core_distance of o because for $\varepsilon' < core\_distance(o)$, o is not a core point with respect to ε' and no point is directly density-reachable from o.

$$reachability\_distance(p,o) = \begin{cases} Undefined,\ if\ |N_\varepsilon(o)| < MinPts \\ max(core\_distance(o), D(o,p)),\ otherwise \end{cases}$$

  where $D(o,p)$ is the distance between o and p.

The idea of the OPTICS algorithm is to create an ordering of the database objects according to their smallest reachability_distance such that the closest points in the feature space are neighbours in the output ordered list. In this ordered list, each object is stored with the core_distance and the smallest reachability_distance to its predecessors in the ordered list.

The OPTICS algorithm is as follows:
OPTICS(DB, ε, $MinPts$, $OrderedFile$)

1. For each point $p \in DB$: $p.reachability\_distance \leftarrow UNDEFINED$
2. For each unprocessed point $p \in DB$:
   (a) $ListNeighbours \leftarrow$ list of points in the ε-neighbourhood of p.
   (b) Mark p as processed.
   (c) Compute the $core\_distance$ of p.
   (d) Write p to the $OrderedFile$.
   (e) Initialize $OrderedSeeds \leftarrow$ empty queue.
   (f) If $p.core\_distance \neq UNDEFINED$ then // if p is a core point
      i. Update($ListNeighbours$, p, $OrderedSeeds$)
      ii. While $OrderedSeeds$ is not empty do:
      - $q \leftarrow OrderedSeeds.next()$
      - $newListNeighbours \leftarrow$ list of points in the ε-neighbourhood of q.
      - Mark q as processed.
      - Compute the $core\_distance$ of q.
      - Write q to the $OrderedFile$.
      - If $q.core\_distance \neq UNDEFINED$ then Update($newListNeighbours$, q, $OrderedSeeds$)

For each unprocessed object p, if p is a core object, all points which are directly density-reachable from p are collected and inserted into the list $OrderedSeeds$ for further expansion. Objects in the $OrderedSeeds$ are ascendantly sorted by their reachability_distance to the closest processed core object from which they are directly density-reachable. In each step of the WHILE loop, OPTICS chooses to process an object q having the smallest reachability_distance in the $OrderedSeeds$ list. If q is also a core object, its directly density-reachable points are also collected and inserted into the list $OrderedSeeds$. Each time a point is processed, it is written into the output $OrderedFile$. Note that the object with the lowest reachability_distance in the $OrderedSeeds$ list is always selected for processing in order to ensure that clusters of higher density are finished first.

Insertion of all points in the ε-neighbourhood of a core point into the seed-list $OrderedSeeds$ is managed by the method **Update**($listNeighbours$, $coreObject$, $OrderedSeeds$) described below. For each unprocessed point o in the ε-neighbourhood of the $coreObject$, its reachability_distance from the $coreObject$ is first determined. o is inserted into the $OrderedSeeds$ list with its reachability_distance if it is not yet in the list. Otherwise, if o is already in the list and its new reachability_distance is smaller than the previous reachability_distance, it is moved up to the the suitable position in the list.

Update(*listNeighbours*, *coreObject*, *OrderedSeeds*)

1. $core\_dist \leftarrow coreObject.core\_distance$.
2. For each unprocessed point $o \in listNeighbours$:
   (a) $new\_reachability\_dist \leftarrow \max(core\_dist, D(coreObject, o))$
   (b) if $o.reachability\_distance = UNDEFINED$ // o is not in the $OrderedSeeds$
      i. $o.reachability\_distance \leftarrow new\_reachability\_dist$
      ii. $OrderedSeeds.insert(o, new\_reachability\_dist)$

   else if $new\_reachability\_dist < o.reachability\_distance$ then
      i. $o.reachability\_distance \leftarrow new\_reachability\_dist$
      ii. $OrderedSeeds.move\_up(o, new\_reachability\_dist)$

The ordered list produced by the OPTICS algorithm can then be used to detect the cluster memberships of objects. A "flat" or hierarchical clustering structure can be extracted from this ordered list depending on the algorithm used. The ExtractDBSCAN-Clustering algorithm described in [4] allows to determine a "flat" clustering structure with respect to $MinPts$ and a distance parameter $\varepsilon' \leq \varepsilon$ by scanning the ordered list and assigning the cluster-memberships for the objects depending on their reachability_distance and core_distance. For each current object o, if its reachability_distance is smaller than ε', o is inserted into the current cluster because it is directly density-reachable with respect to $MinPts$ and ε' from a preceding core object in the ordered list. Otherwise, it is not density-reachable from any of the preceding objects, and we start a new cluster if o is a core object, or assign o as a NOISE point if o is not a core object. Another way to identify the clustering structure is to use the reachability-plot which is a 2D plot with the ordering of the point on the x-axis and the corresponding reachability_distance on the y-axis as shown in Figure 3.3. As points having low reachability_distance should be in the same cluster with precedent points in the ordered list, the clusters can be detected as valleys in the reachability-plot. By using the reachability-plot, we can detect clusters in the form of "flat" or hierarchical structures.

As DBSCAN, the number of clusters does not have to be fixed *a priori*. OPTICS is robust to outliers, but may not be applied to high-dimensional data or in an incremental context. The time complexity of this method is about $O(NlogN)$, it is thus suitable to large data sets. The cluster for assigning a new point can be found in $O(logN)$ time. The clustering result also depends on the processing order of the data and on the parameters ϵ and $MinPts$.

EM [89] For the Expectation Maximization (EM) algorithm, we assume that the vectors of the data set are independent and identically distributed according to a mixture of k Gaussians, the j^{th} Gaussian generates the points from the j^{th} cluster. The Gaussian distribution of a cluster K_j is characterized by the mean μ_j and the covariance matrix Σ_j. EM algorithm aims at estimating the optimal parameters of the mixture of Gaussians by iteratively assigning the data points to clusters (soft assignment) and updating the Gaussian parameters to maximize the likelihood of

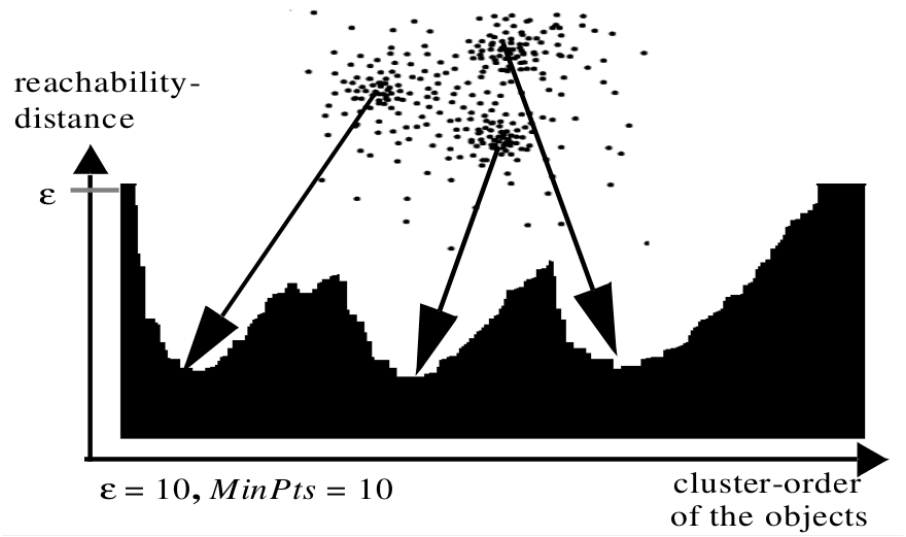


Figure 3.3 – Reachability-plot example [4].

these parameters, given the cluster assignment of the points.

The EM algorithm consists of the two following steps:

1. Randomly initialize the parameters of the Gaussian mixture model. Assume that we have k Gaussian components, the parameters of the model are:

$$\theta^{(0)} = \{\mu_1^{(0)}, \mu_2^{(0)}, ..., \mu_k^{(0)}, \Sigma_1^{(0)}, \Sigma_2^{(0)}, ..., \Sigma_k^{(0)}, \alpha_1^{(0)}, \alpha_2^{(0)}, ..., \alpha_k^{(0)}\} \tag{3.6}$$

   where α_j represents the occurrence probability of each cluster.

2. $t \leftarrow 0$;

3. Repeat until convergence:

   (a) *E-step:* Having the parameters of the Gaussian mixture model, we compute, for each object x_i and each cluster K_j, the probability that x_i belongs to K_j:

$$\begin{aligned} P(x_i \in K_j | x_i, \theta^{(t)}) &= \frac{P(x_i | x_i \in K_j, \theta^{(t)}).P(K_j|\theta^{(t)})}{P(x_i|\theta^{(t)})} \\ &= \frac{P(x_i | x_i \in K_j, \mu_j^{(t)}, \Sigma_j^{(t)})\alpha_j^{(t)}}{\sum_{j=1}^{k} P(x_i | x_i \in K_j, \mu_j^{(t)}, \Sigma_j^{(t)})\alpha_j^{(t)}} \end{aligned} \tag{3.7}$$

   (b) *M-step:* Given the cluster assignment of the points, we update the parameters of the mixture of Gaussians so that it maximizes the likelihood of the parameters.

$$\alpha_j^{(t+1)} = P(K_j) = \frac{1}{N} \sum_{i=1}^{N} P(x_i \in K_j | x_i, \theta^{(t)}) \tag{3.8}$$

$$\mu_j^{(t+1)} = \frac{\sum_{i=1}^{N} P(x_i \in K_j | x_i, \theta^{(t)}) x_i}{\sum_{i=1}^{N} P(x_i \in K_j | x_i, \theta^{(t)})} \tag{3.9}$$

$$\Sigma_j^{(t+1)} = \frac{\sum_{i=1}^{N} P(x_i \in K_j | x_i, \theta^{(t)})(x_i - \mu_j^{(t+1)})(x_i - \mu_j^{(t+1)})^T}{\sum_{i=1}^{N} P(x_i \in K_j | x_i, \theta^{(t)})} \quad (3.10)$$

After setting all the parameters of the Gaussian mixture model, EM assigns each data point x_i to the most likely cluster K_j (associated with the maximum probability $P(x_i \in K_j | x_i, \theta)$). EM is simple to apply and does not depend on the processing order of the data. 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. And the complexity for finding a cluster for assigning a new point is about $O(k^2)$. EM is thus suitable to large databases when k is small enough. However, if the data is not distributed according to a mixture of Gaussian distributions, the results are often poor, while it is very difficult to determine the distribution of high dimensional data. Moreover, the clustering result depends on the value of k, 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, and it does not produce any hierarchical structure.

Conclusion on the density-based methods The density-based clustering methods are in general suitable to large databases and are able to detect outliers. But they are not adapted to high dimensional (sparse) data and to an incremental context.

3.2.4 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 [62], DIANA [62], AHC [70], BIRCH [145], CURE [41].

DIANA [62] DIvisive ANAlysis (DIANA) is a divisive (or top-down) approach which constructs the hierarchy by successively dividing 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 split into two smaller clusters, until all clusters contain only one singleton. Once the whole hierarchy is built, we can choose the clustering solution based on the desired number of clusters. The diameter of a cluster is the largest distance between any two objects of this cluster. It is easy to see that DIANA builds the hierarchy of N objects in $N-1$ steps.

A cluster K is split into two as follows:

1. Identify, in cluster K, the object x^* which has the largest average dissimilarity with all the other objects of cluster K, then x^* initializes a new cluster K^*.

2. For each object $x_i \in K \setminus K^*$, compute:

$$d_i = \underset{x_j \in K \setminus K^*}{\text{average}}[D(x_i, x_j)] - \underset{x_j \in K^*}{\text{average}}[D(x_i, x_j)] \quad (3.11)$$

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 distance or dissimilarity measures (see Section 2.4.2). The hierarchical structure of DIANA does not depend on the processing order of the data or on any parameter. But it is sensitive to outliers and 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 from 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 with 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. At each node of the hierarchy, for determining the child node to which a new point should be assigned, we have to calculate the average distance between the new point and every points of each child node. Therefore, the complexity for assigning a new point to the tree is at least $O(N)$.

Minimum Spanning Tree based clustering [57] Minimum Spanning Tree based clustering is a simple divisive algorithm. It starts by constructing a Minimum Spanning Tree (MST) [35]. Assume G_N is a complete graph in which each vertex represents a point in the database and every pair of vertices is connected by an edge having the weight corresponding to the dissimilarity between the two corresponding points. A spanning tree for G_N is a sub-graph which connects all vertices of G_N either by an edge or by a path, but no cycles are formed. A minimum spanning tree for G_N is the spanning tree with the lowest total weight. A MST can be constructed using different algorithms, among which Kruskal's algorithm [68] and Prim's algorithm [106] for instance. For clustering, the algorithm removes, at each iteration, the longest edge of the MST to obtain the clusters. The process continues until k clusters are found or until there is no more edge to eliminate. Figure 3.4 represents a MST corresponding to a set of 9 points $A, ..., I$. The longest edge of weight 6 is removed first for having two clusters $K_1 = \{A, B, C\}$ and $K_2 = \{D, E, F, G, H, I\}$. The process continues by removing the edge of weight 3.8 for dividing the cluster K_2 into two smaller clusters $K_3 = \{D, E, F, G\}$ and $K_4 = \{H, I\}$.

The MST based clustering does not depend on the processing order of the data. There is no parameter to be fixed. When new elements are added into the database, the minimum spanning tree has to be reconstructed, therefore it may be difficult to use this method in an incremental context. Moreover, this method is very sensitive to outliers and it has a high computational complexity of $O(N^2)$, it is therefore not compatible for clustering large databases. For simply assigning a new point for the existing clusters without reconstructing the minimum spanning tree, we can assign this point to the cluster containing the point which is nearest to the new point. Therefore, the complexity for finding a cluster for a new point is $O(N)$.

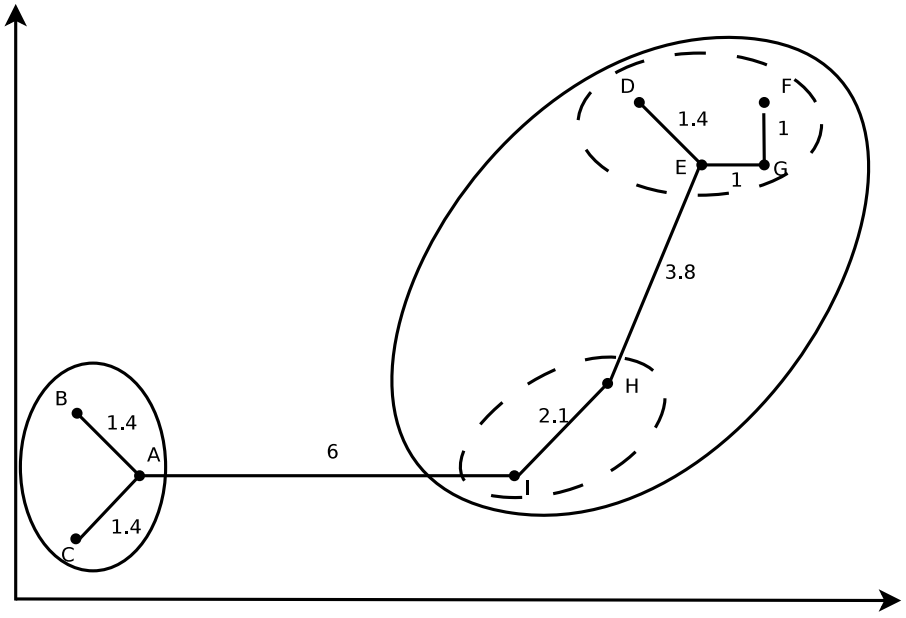


Figure 3.4 – Example of minimum spanning tree based clustering.

Agglomerative Hierarchical Clustering (AHC) [70] AHC is a bottom-up clustering method. It starts by assigning each object to a separate cluster. Then, it merges at each iteration the two closest clusters, until it remains only one cluster containing all data objects. The algorithm consists of the following steps:

1. Assign each object to a single cluster. Thus, we obtain N clusters corresponding to N objects of the database.
2. Compute the distance matrix containing distances between all pairs of clusters.
3. Merge the two closest clusters.
4. Compute the distances from the new cluster to all other clusters and update the matrix.
5. Repeat steps 3 and 4 until there is only one cluster left.

The distance between two objects $D(x, x')$ can be computed by different measures described in Section 2.4.2, Chapter 2. For evaluating the distance between any two clusters K_i and K_j, different approaches are proposed:

- *Single-linkage*: the distance between two clusters K_i and K_j is the minimum distance between an object in cluster K_i and another object in cluster K_j:

$$D(K_i, K_j) = \min_{x \in K_i, x' \in K_j} D(x, x') \tag{3.12}$$

- *Complete-linkage*: the distance between two clusters K_i and K_j is the maximum distance between an object in cluster K_i and another object in cluster K_j:

$$D(K_i, K_j) = \max_{x \in K_i, x' \in K_j} D(x, x') \tag{3.13}$$

- *Average-linkage*: the distance between two clusters K_i and K_j is the average distance between an object in cluster K_i and another object in cluster K_j:

$$D(K_i, K_j) = \frac{1}{|K_i||K_j|} \sum_{x \in K_i, x' \in K_j} D(x, x') \tag{3.14}$$

- *Centroid-linkage*: the distance between two clusters K_i and K_j is the distance between the centroids μ_i, μ_j of these two clusters:

$$D(K_i, K_j) = D(\mu_i, \mu_j) \tag{3.15}$$

- *Ward's distance* [138]: 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:

$$\begin{aligned} D(K_i, K_j) &= \sum_{x \in K_i \cup K_j} (x - \mu_{K_i \cup K_j})^2 - \sum_{x \in K_i} (x - \mu_{K_i})^2 - \sum_{x \in K_j} (x - \mu_{K_j})^2 \\ &= \frac{|K_i||K_j|}{|K_i| + |K_j|} \left(\mu_{K_i} - \mu_{K_j}\right)^2 \end{aligned} \tag{3.16}$$

where μ_i, μ_j, $\mu_{K_i \cup K_j}$ are respectively the centres of clusters K_i, K_j, $K_i \cup K_j$, and $|K_i|$, $|K_j|$ are respectively the numbers of points in clusters K_i and K_j.

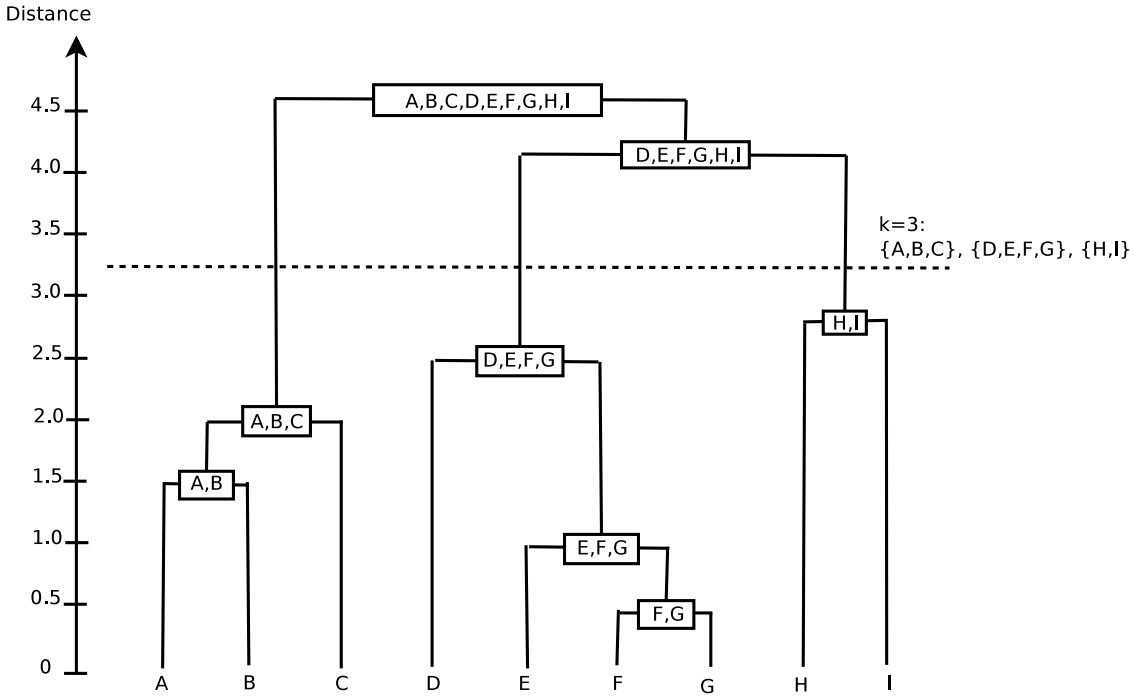


Figure 3.5 – Cluster tree constructed by the AHC clustering.

Figure 3.5 shows a cluster tree (dendrogram) constructed using the AHC approach. Data points are at the bottom row. Higher nodes represent clusters which are formed by joining individual points or existing clusters. The height of each cluster node corresponds to the distance value between the two sub-clusters. The

higher the height value is, the greater the distance. A partition can be constructed by cutting horizontally the tree. For example, the partition shown in Figure 3.5 gives three clusters $\{A, B, C\}$, $\{D, E, F, G\}$ and $\{H, I\}$. The dendrogram can be cut either at a prespecified distance, or at the position where the gap between two successive distances for merging clusters is largest, or at the cutting point that produces k clusters when the number of clusters k is predefined.

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 clustering result does not depend on the processing order of the data or on any parameter, but on the measure chosen for calculating the distance between clusters. The other disadvantages of this method is that it has a high computational complexity of $O(N^2 logN)$ and a storage complexity for the distance matrix of $O(N^2)$, and therefore AHC is not really adapted to large databases. It is also sensitive to noise and outliers. For assigning a new point to the existing clusters without reconstructing the hierarchical structure, we go down from the root and assign the new point to the nearest child node. Depending on the distance measure between clusters, the complexity for finding cluster for a new point may be $O(k)$ (in the case of centroid-linkage) or $O(N)$ (for other measures).

An incremental variant of the AHC method was proposed in [110]. When there is a new element NE, it determines its location in the tree by going down from the root. At each node R which has two children C_1 and C_2, the new element NE will be merged with R if $D(C_1, C_2) < D(NE, R)$; otherwise, it goes down to the child whose region of influence contains NE. The new element NE belongs to the region of influence of C_1 if $D(NE, C_1) \leq D(C_1, C_2)$ and $D(NE, C_1) < D(NE, C2)$. For calculating D, any of the distances between clusters (3.12) to (3.16) may be used.

BIRCH [145] Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) is developed to partition very large databases that can not be stored in the main memory. The idea is to scan the database and organize points in a Clustering Feature Tree (CF-tree).

A Clustering Feature (CF) vector summarizes information of a cluster including M data points $\{x_i | i = 1, ..., M\}$. It is defined as a triplet $CF = (M, LS, SS)$ where LS and SS are respectively the linear sum and the square sum of M data points in this cluster ($LS = \sum_{i=1}^{M} x_i$; $SS = \sum_{i=1}^{M} x_i.x_i$). From the CF vector of a cluster, we can simply compute the mean, the average radius (average distance from a point to the mean of the cluster) and the average diameter (average distance between two points of the cluster) of a cluster, and also some distance between two clusters (*e.g.* the Euclidean distance between their means). The CF vector of a cluster which is formed by merging two disjoint clusters respectively represented by $CF_1 = (M_1, LS_1, SS_1)$ and $CF_2 = (M_2, LS_2, SS_2)$ can be simply computed as:

$$CF_1 + CF_2 = (M_1 + M_2, LS_1 + LS_2, SS_1 + SS_2) \tag{3.17}$$

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

- Each internal node contains at most B elements $[CF_i, child_i]$, where CF_i summarizes the information of the sub-cluster represented by its i^{th} child pointed by the pointer $child_i$.

- Each leaf node contains at most L elements $[CF_i]$; it also contains two pointers *prev* and *next* to link all leaf nodes together.
- Threshold condition: each element CF_i of a leaf must have a diameter (or radius depending on the version of the algorithm) lower than a threshold value T.

So, each node in the CF-tree represents a cluster which is formed by merging all the sub-clusters represented by its entries. An example of a CF-tree with $B = 5$ and $L = 4$ is shown in Figure 3.6.

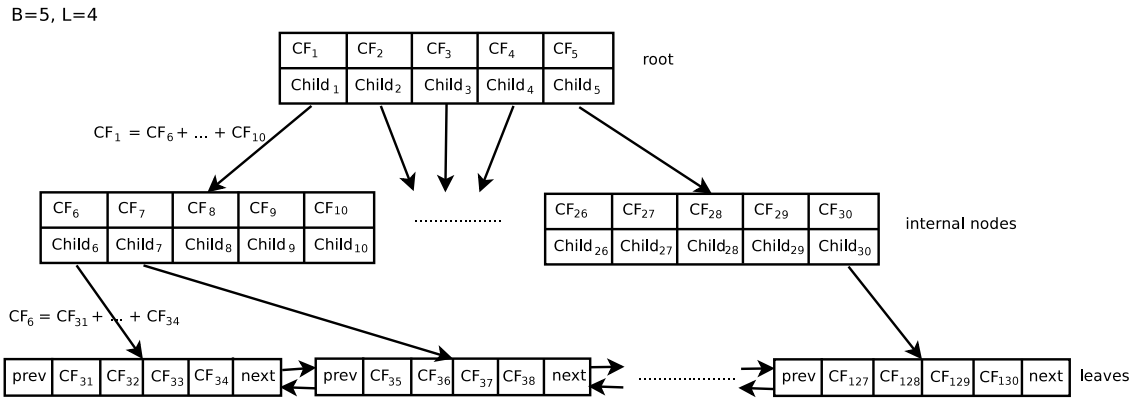


Figure 3.6 – Example of a CF-tree with B=5, L=4.

The CF-tree can be created by inserting successively points into the tree. To insert a new entry Ent, we go down in the tree from the root by selecting, at each level, the closest child. When a leaf L^* is reached, we find the closest leaf entry of L^*, say CF^*. If CF^* can absorb the new entry Ent without violating the threshold condition, then CF^* is updated to take into account the insertion of Ent into this entry. Otherwise, a new entry CF_{new} is created for Ent. If the leaf L^* is not full, then CF_{new} is inserted into L^*. If L^* is full, then L^* must be split into two new leaves. The change of CF information is then propagated upwards in the concerned branch nodes; nodes in higher levels should also be split if needed.

The size of the tree depends on the threshold T. The smaller T is, the larger the tree. At first, we create the tree with a small value of T, then if it exceeds the maximum allowed size of the memory, T is increased and the tree is reconstructed. This process is iterated until the tree can be stored in the main memory. During reconstruction, vectors that are already inserted will not be reinserted because they are already represented by CF vectors. Only these CF vectors will be reinserted. We must increase T so that the two closest micro-clusters (leaf node entries) could be merged. After creating the CF-tree, we can use any clustering method (AHC, k-means, etc.) for clustering the CF entries 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 complexity of $O(N)$ for constructing the CF-tree with a specified value of T. When we have to reconstruct the CF-tree with a higher value of T, the complexity is about $O(L)$ where L is the number of CF entries at the leaf level. In general, $L << N$. And the complexity for finding a cluster for assigning a new point is about $O(k)$. BIRCH can be applied to large databases. BIRCH is

incremental; the CF-tree can be updated easily when new points are inserted into the system. Outliers can be eliminated by identifying the objects that are sparsely distributed in isolated CF entries. But, it is sensitive to the data processing order, and it depends on the choice of its three parameters.

CURE [41] Similar to the AHC clustering, CURE (Clustering Using REpresentative) initializes each object as a separate cluster, and successively merges the two closest clusters at each step. But, the distance between two clusters is computed based on some representative objects of these two clusters instead of using all their objects, which leads to a lower computational complexity.

For each cluster K_i, CURE stores a set of c representative objects. For determining the representative set of K_i, we firstly select a set $tmpSet$ of c well scattered points within K_i (more details are given in the next paragraph), and then each point $p \in tmpSet$ is shrank toward the mean of the cluster by a fraction α ($0 \leq \alpha \leq 1$) before being added into the representative set of K_i as in Equation (3.18):

$$K_i.rep = K_i.rep \cup \{p + \alpha * (K_i.mean - p)\} \tag{3.18}$$

where $K_i.mean$ and $K_i.rep$ store respectively the mean of all objects in K_i and the set of representative points of K_i. By shrinking selected points towards the mean, we can reduce the effects of outliers, as outliers are typically moved more towards the centre.

The set $tmpSet$ of c well scattered points within K_i is formed as follows: the farthest point from the mean μ_i is chosen as the first scattered point, and then each new scattered point is defined as the farthest point from the previously chosen scattered points. Similar to the AHC clustering, the distance between a point and the previously chosen scattered points can be calculated using Equations (3.12) to (3.16).

The time complexity of CURE is $O(N^2 logN)$, so that CURE cannot be directly applied to large databases. For a large database, a random sampling strategy is used to select a random subset of the database, then CURE is applied with this subset rather than the entire database. The modified version of CURE clustering for large database is as follows:

1. Randomly select a sample set containing N_S points of the database.

2. Partition this subset into p partitions of size N_S/p and partially realize CURE clustering for each partition, until having $\frac{N_S}{pq}$ clusters in each partition (where q is a constant, $q > 1$), or the distance between two closest clusters is greater than a given threshold.

3. Perform another CURE clustering where input data are all clusters of all partitions resulting from the previous step.

4. Associate each point which is not in the sample set with the cluster having the closest representative points.

Outliers can be filtered by identifying clusters which contain only a few points. CURE does not depend on the processing order of the data. Any new point can be directly associated with the clustering having closest representative points in

$O(k.c)$ time (where c is the number of representative objects of each cluster). The execution time of this modified version of CURE is relatively low: $O(N_S^2 \log N_S)$. So, it can be applied on a large image database. However, it is difficult to fix the input parameters (shrink factor α, number of representative points c, number of partitions p, sample size N_S). CURE relies on a trade-off between the effectiveness and the complexity of the overall method. Selecting too few samples may reduce the effectiveness, while the complexity increases with the number of samples. This trade-off 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 where users are involved.

R-tree family [13, 42, 117], SS-tree [139], SR-tree [60] These index structures aim at grouping data vectors in a balanced tree corresponding to the data distribution. The region of each node in the R-tree family, SS-tree and SR-tree are, respectively, a bounding rectangle, a bounding sphere and a region which is the intersection of a bounding rectangle and a bounding sphere in the feature space (see Chapter 2). The data objects are stored in the leaves, and all leaves are at the same level. The region of a leaf covers the objects belonging to it. The region of an internal node covers the union of the regions of its descendant nodes. And the root node therefore covers all objects in the database. Note that every node contains at least m and at most M entries, unless it is the root. The lower bound m ensures an efficient storage utilization, while the upper bound M ensures that each node does not exceed one disk page size. As in BIRCH clustering, after creating the tree, we can use any existing clustering algorithms (AHC, k-means, etc.) for clustering the leaf nodes or the nodes at any level of the tree. These tree structures are incrementally constructed by inserting iteratively the objects into the corresponding leaves. They are sensitive to the insertion order of the objects and to the outliers. Moreover, the resulting trees of these methods depend on their parameters m and M, and the clustering results depend on the input number of clusters k. The computational complexity of these method is about $O(NlogN)$, thus they are suitable to large databases. The cluster for assigning a new point can be found in $O(k)$. Among these structures, SR-tree is reported to give the best results [60] as it allows to create regions with small volumes and small diameters, by combining the bounding rectangle and the bounding sphere, thus reducing the overlap between nodes.

Conclusion for the hierarchical methods The advantage of hierarchical methods is that they organize data in a hierarchical structure. Therefore, by considering the structure at different levels, we can obtain different numbers of clusters. DIANA, MST and AHC are not adapted to large databases, while the others are suitable. In BIRCH, the CF-tree is built by incrementally adding the records; it is by nature incremental. But because of this incremental construction, it depends on the processing order of the input data. CURE is able to add new points, but all the records have to be reassigned whenever the number of clusters k is changed. CURE is therefore not suitable to the context where users are involved. Similar to

BIRCH, R-tree, SS-tree and SR-tree are also by nature incremental, but depend on the processing order of the input data.

3.3 Formal comparison of unsupervised clustering methods

Table 3.1 formally compares the different clustering methods presented in the last section (partitioning methods (P), hierarchical methods (H), grid-based methods (G) and density-based methods (D)) based on different criteria (complexity, adapted to large databases, incrementality, 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_S: the number of the sample set (if any).
- d: the number of dimensions size of the feature vector.

The partitioning methods (k-means, k-medoids (PAM), CLARA, CLARANS, ISODATA, SOM) are generally not incremental (except SOM); they do not produce any hierarchical structure. Most of them are independent of the processing order of the data (except CLARANS and SOM). K-means, CLARA, ISODATA and SOM are suitable to large databases, while PAM, CLARA, CLARANS and ISODATA are able to detect the outliers. Among different partitioning methods, k-means is the baseline method because of its simplicity and its effectiveness for large databases.

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 produces hierarchical structure, but it is not suitable to high dimensional data such as feature image space. Moreover, in high dimensional context, data is generally extremely sparse, so it is difficult to specify the density parameter for determining dense cells. In this case, the hierarchical methods are better than grid-based methods.

The density-based methods (DBSCAN, OPTICS, EM) are in general suitable to large databases and are able to detect outliers. But they are dependent on their parameters and on the processing order of data (except EM). Moreover, they do not produce any hierarchical structure, are not adapted to high dimensional data and are not incremental.

The hierarchical methods (DIANA, MST, AHC, BIRCH, CURE, R-tree, SS-tree, SR-tree) organize data in a hierarchical structure. Therefore, by considering the structure at different levels, we can obtain different numbers of clusters, which are useful in the context where users are involved and can change the number of clusters by merging or splitting clusters. DIANA, MST and AHC are not suitable to the incremental context and are not adapted to large databases because of their high complexities. BIRCH, R-tree, SS-tree and SR-tree are by nature incremental,

because they are built incrementally by adding the records; but because of this incremental construction, they depend on the processing order of the input data. BIRCH, R-tree, SS-tree and SR-tree are also adapted to large databases because of their relatively low computational complexity. CURE realizes the hierarchical clustering using only a random subset containing N_S points of the database, the other points being associated to the closest cluster. Its computational complexity is thus relatively low and CURE is adapted to large databases. It is able to add new points but the results of CURE clustering strongly depend 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.

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 the most suitable to our context.

We choose to present, in Section 3.5, an experimental comparison of five different clustering methods: global k-means [76], AHC [70], R-tree [42], SR-tree [60] and BIRCH [145]. Global k-means is a variant of the famous and the most used clustering method (k-means) which is partition based. 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.

Table 3.1 – Formal comparison of different clustering methods (partitioning methods (P), hierarchical methods (H), grid-based methods (G) and density-based methods (D)) based on different criteria (complexity, adapted to large databases, incrementality, hierarchical structure, data order dependence, sensitivity to outliers, parameters dependence). Methods in bold are chosen for experimental comparison.

Methods	**Complexity**	**Adapted to large database**	**Incrementality**	**Hierarchical structure**	**Data order dependence**	**Sensitivity to outliers**	**Parameters Dependence**
k-means (P)	$O(Nkl)$ (time) $O(N+k)$ (space)	Yes	No	No	No	Sensitive	No
k-medoids (PAM) (P)	$O(k(N-k)^2l)$ (time)	No	No	No	No	Enable outliers detection	No
CLARA (P)	$O(q(kN_S^2l+k(N-k)))$ (time)	Yes	Able to add new points	No	No	Enable outliers detection	Yes
CLARANS (P)	$O(N^2)$ (time)	No	No	No	Yes	Enable outliers detection	Yes
ISODATA (P)	$O(Nkl)$ (time) $O(N+k)$ (space)	Yes	No	No	No	Enable outliers detection	Yes
SOM (P)	$O(Nkl)$ (time)	Yes	Yes	No	Yes	Sensitive	Yes
STING (G)	$O(N)$ (time)	Yes	Yes	Yes	No	Enable outliers detection	Yes
CLIQUE (G)	$O(Nd+k^d)$ (time)	Yes	Able to add new points	No	No	Enable outliers detection	Yes
DBSCAN (D)	$O(NlogN)$ (time)	Yes	No	No	Yes	Enable outliers detection	Yes
OPTICS (D)	$O(NlogN)$ (time)	Yes	No	No/Yes	Yes	Enable outliers detection	Yes
EM (D)	$O(Nk^2l)$ (time)	Yes	No	No	No	Enable outliers detection	Yes
DIANA (H)	$O(N^2)$ (time)	No	No	Yes	No	Sensitive	No
Simple Divisitive Algorithm (MST) (H)	$O(N^2)$ (time)	No	No	Yes	No	Sensitive	No
AHC (H)	$O(N^2logN)$ (time) $O(N^2)$ (space)	No	Have incremental version	Yes	No	Sensitive	No
BIRCH (H)	$O(N)$ (time)	Yes	Yes	Yes	Yes	Enable outliers detection	Yes
CURE (H)	$O(N_S^2logN_S)$ (time)	Yes	Able to add new points	Yes	No	Enable outliers detection	Yes
R-tree (H)	$O(NlogN)$ (time)	Yes	Yes	Yes	Yes	Sensitive	Yes
SS-tree (H)	$O(NlogN)$ (time)	Yes	Yes	Yes	Yes	Sensitive	Yes
SR-tree (H)	$O(NlogN)$ (time)	Yes	Yes	Yes	Yes	Sensitive	Yes

3.4 Clustering evaluation measures

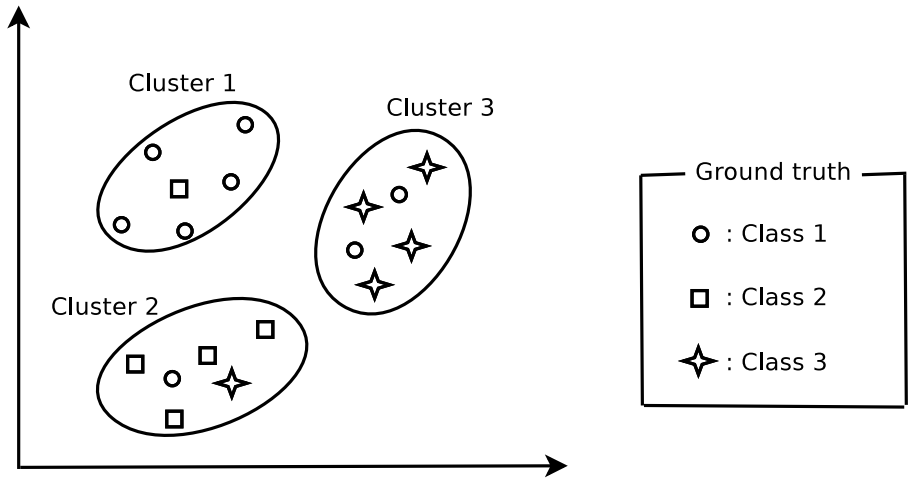


Figure 3.7 – Clustering result *vs.* Ground truth.

As it has been discussed in previous parts, the clustering methods use the low-level features extracted from the images to group images into clusters while the images in the ground truth are classified by the user using high-level semantic concepts. There may be a "semantic gap" between high-level semantic concepts and low-level features, and therefore, the clusters given by the clustering algorithm are often different from the classes of images in the ground truth, as shown in Figure 3.7.

In order to evaluate the clustering results, there are currently two main kinds of measures:

- *Internal 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 vectors 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 measures.

- *External measures*: 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.

3.4.1 Internal measures

Internal measures are numerical (unsupervised) measures which do not consider the semantic point of view while evaluating the clustering results, but consider only the distribution of the points in the clusters and the balance of these clusters.

As the goal of clustering is to split a collection of data into groups (clusters) so that similar objects belong to the same cluster and dissimilar objects are in different clusters, internal measures often evaluate the clustering results based on two criteria: *compactness* and *separation*. While the compactness measures how closely the objects in a cluster are, the separation measures how separate different clusters are.

We present in this section some widely used internal clustering evaluation measures. For this, we use the following notations:

- $X = \{x_i | i = 1, ..., N\}$: the set of points in the database, where N is the number of points.
- $K = \{K_i | i = 1, ..., k\}$: the set of clusters, where k is the number of clusters.
- $|K_i|$: the number of points in cluster K_i.
- $K(x_i)$: the cluster containing the point x_i.
- μ_i: the centre of cluster K_i.
- $\mu(x_i)$: the centre of the cluster containing the point x_i.
- $D(x_i, x_j)$: the distance between two points x_i and x_j.
- $M = N(N-1)/2$: the maximum number of pairs of points in the database.

Compactness or homogeneity Shamir and Sharan [119] evaluates the compactness (or homogeneity) of the clustering results by computing the average distance between each point and the centre of the corresponding cluster:

$$H = \frac{1}{N} \sum_{x_i \in X} D(x_i, \mu(x_i)) \tag{3.19}$$

The smaller the values of this measure are, the higher the average compactness of the clusters, and the better the clustering results.

Another measure to compute the compactness is presented in [47]. We first define the variance of a data set as:

$$v(X) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} D^2(x_i, \overline{x})} \tag{3.20}$$

where $\overline{x} = \frac{1}{N} \sum_i x_i$ is the mean of the set X. The compactness of the clustering results is defined as:

$$C = \frac{1}{k} \sum_{i}^{k} \frac{v(K_i)}{v(X)} \tag{3.21}$$

where $v(K_i)$ is the variance of the cluster K_i and $v(X)$ is the variance of the data set X. A smaller value of this measure indicates a higher compactness of the clustering results.

Separation For measuring how separate different clusters are, Shamir and Sharan [119] compute the separation as the weighted average distance between cluster centres. The higher the values of this measure, the better the clustering results.

$$S = \frac{1}{\sum_{i \neq j} |K_i||K_j|} \sum_{i \neq j} |K_i||K_j| D(\mu_i, \mu_i) \tag{3.22}$$

He *et al.* [47] compute the separation of the clustering results as follows:

$$S = \frac{1}{k(k-1)} \sum_{i=1}^{k} \sum_{j=1, j \neq i}^{k} exp(-\frac{D^2(\mu_i, \mu_j)}{2\sigma^2}) \tag{3.23}$$

where σ is the standard deviation of a Gaussian. In [47], He *et al.* fix $2\sigma^2 = 0.25$ for the ease of evaluation. A smaller value of separation indicates a larger dissimilarity between clusters, and a better clustering result.

Overall cluster quality [47] He *et al.* [47] combine the compactness in Equation 3.21 and the separation in Equation 3.23 into one measure called *overall cluster quality* as follows:

$$Ocq = \beta.C + (1 - \beta).S \tag{3.24}$$

where $\beta \in [0, 1]$ is the weight controlling the trade-off between the compactness and the separation. The smaller the overall cluster quality values, the better the clustering result.

Silhouette Width [115] The Silhouette Width evaluates the clustering results by measuring at the same time the compactness and the separation. The silhouette width for each point x_i measures how much x_i belong to its cluster; it is computed as follows:

$$SW(x_i) = \frac{b(x_i) - a(x_i)}{max\{a(x_i), b(x_i)\}} \tag{3.25}$$

where:

- $a(x_i)$: the average distance between x_i and the other points in the same cluster. $a(x_i)$ represents thus the compactness between x_i and the other points in the same cluster.

$$a(x_i) = \frac{1}{|K(x_i)| - 1} \sum_{x_j \in K(x_i), x_j \neq x_i} D(x_i, x_j) \tag{3.26}$$

- $b(x_i)$: the average distance between x_i and the points in the cluster $\neq K(x_i)$ which is the closest to x_i. $b(x_i)$ represents thus the separation between x_i and the closest cluster.

$$D(x_i, K_l) = \frac{1}{N_l} \sum_{x_j \in K_l} D(x_i, x_j) \tag{3.27}$$

$$b(x_i) = min_{K_l \neq K(x_i)}\{D(x_i, K_l)\} \tag{3.28}$$

The average Silhouette Width of all the points in the data set represents the quality of the clustering result. The higher the values are, the better the results.

$$SW = \frac{1}{N} \sum_{x_i \in X} SW(x_i) \tag{3.29}$$

Hubert Γ statistic [43] The Hubert Γ statistic measures the similarity between two matrices P and Q as follows:

$$\Gamma = \frac{1}{M} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} P(i,j)Q(i,j) \tag{3.30}$$

In order to use the Hubert Γ statistic for evaluating the clustering result, P is the proximity matrix of the points in the data set and Q is an $N \times N$ matrix whose element $Q(i,j)$ is the distance between the centres of clusters $K(x_i)$ and $K(x_j)$. The higher the value of this measure is, the better the clustering results.

Normalized Hubert Γ statistic The normalized Hubert Γ statistic measures also the similarity between two matrices P and Q as follows:

$$\bar{\Gamma} = \frac{\frac{1}{M} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} (P(i,j) - \mu_P)(Q(i,j) - \mu_Q)}{\sigma_P \sigma_Q} \tag{3.31}$$

where μ_P, μ_Q, σ_P, σ_Q are respectively the means and the variances of P and Q. For evaluating the clustering result, the two matrices P and Q are defined as in the case of the Hubert Γ statistic. The value of the normalized Hubert Γ statistic is between -1 and 1. The higher the value is, the better the clustering result.

3.4.2 External measures

External measures are semantic-based (supervised) measures which evaluate the clustering result by comparing it to the distribution of the data in the ground truth (classes), which is often created by humans or by a source of knowledge "validated" by humans. External measures often compare the clustering result and the classes in the ground truth based on two criteria:

- *Homogeneity*: a clustering solution satisfies the homogeneity criteria if all clusters contain only points of a single class.
- *Completeness*: a clustering solution satisfies the completeness criteria if all the points that are members of a single class are assigned to a single cluster.

In this section, we present some widely known external measures. We use the following notations:

- N: the number of points in the data set.
- $C = \{C_i | i = 1, ..., c\}$: the set of classes in the ground truth, where c is the number of classes.
- $K = \{K_j | j = 1, ..., k\}$: the set of clusters produced by the clustering algorithm, where k is the number of clusters.
- n_{ij}: the number of points of the class C_i which are assigned to the cluster K_j.
- $M = N(N-1)/2$: the maximum number of pairs of points in the database.

Purity [146] The purity measure evaluate the homogeneity of the clustering result. The purity of each cluster K_j is defined as the most represented class among the points assigned to this cluster:

$$P(K_j) = \frac{1}{|K_j|} \max_{C_i \in C} \{n_{ij}\} \tag{3.32}$$

The overall purity of the clustering result is computed as the weighted sum of the purities of all clusters. Higher values of the overall purity indicate better clustering results.

$$Purity = \sum_{j=1}^{k} \frac{|K_j|}{N} P(K_j) \tag{3.33}$$

Entropy [146] Entropy measures also the homogeneity of the clustering result by evaluating the distribution of different classes of points within each cluster. The entropy of each cluster K_j is defined as:

$$Entropy(K_j) = -\frac{1}{\log c} \sum_{i=1}^{c} \frac{n_{ij}}{|K_j|} \log \frac{n_{ij}}{|K_j|} \tag{3.34}$$

The overall entropy of the clustering solution is then computed as the weighted sum of the entropies of all clusters. The smaller the value of the overall entropy is, the better the clustering result.

$$Entropy = \sum_{j=1}^{k} \frac{|K_j|}{N} Entropy(K_j) \tag{3.35}$$

Overall entropy [47] Overall entropy measure evaluates both the homogeneity and the completeness criteria based on the cluster entropy and the class entropy.

The cluster entropy measures the homogeneity of the data points in each individual cluster. The cluster entropy $E(K_j)$ is defined for each cluster K_j as follows:

$$E(K_j) = -\sum_{i=1}^{c} \frac{n_{ij}}{|K_j|} \log \frac{n_{ij}}{|K_j|} \tag{3.36}$$

The smaller the value of $E(K_j)$, the higher the homogeneity of cluster K_j. The overall cluster entropy is defined as the weighted sum of the entropies of all clusters:

$$E_K = \frac{1}{N} \sum_{j=1}^{k} |K_j| E(K_j) \tag{3.37}$$

The class entropy measures how the data points of a same class are represented by different clusters, it thus measures the completeness criteria. The class entropy $E(C_i)$ of each class C_i is defined as follows:

$$E(C_i) = -\sum_{j=1}^{k} \frac{n_{ij}}{|C_i|} \log \frac{n_{ij}}{|C_i|} \tag{3.38}$$

Similar to the overall cluster entropy, the overall class entropy E_C is defined as the weighted sum of the individual class entropies. The smaller the value of the overall class entropy, the higher the completeness of the clustering solution.

$$E_C = \frac{1}{N}\sum_{i=1}^{c}|C_i|E(C_i) \tag{3.39}$$

The overall entropy combines the cluster entropy and the class entropy as follows:

$$E_{CK} = \beta * E_K + (1-\beta) * E_C \tag{3.40}$$

where β is the weight defining the importance of these two entropies. The smaller the value of E_{CK}, the better the clustering solution.

F-measure [33] The F-measure evaluates both the homogeneity and the completeness criteria of the clustering solution. The *Recall* and *Precision* for a class C_i and a cluster K_j are defined as:

$$Recall(C_i, K_j) = \frac{n_{ij}}{|K_j|} \tag{3.41}$$

$$Precision(C_i, K_j) = \frac{n_{ij}}{|C_i|} \tag{3.42}$$

The $Recall(C_i, K_j)$ is computed as the proportion of the points of the class C_i in cluster K_j, it thus measures the homogeneity of the cluster K_j regarding to the class C_i. The $Precision(C_i, K_j)$ is computed as the proportion of the points of the class C_i which are assigned to the cluster K_j, it thus measures the completeness of the class C_i regarding to the cluster K_j. F-measure for a class C_i and a cluster K_j is computed as the harmonic mean of the corresponding *Recall* and *Precision*:

$$F(C_i, K_j) = \frac{2 * Recall(C_i, K_j) * Precision(C_i, K_j)}{Recall(C_i, K_j) + Precision(C_i, K_j)} \tag{3.43}$$

$F(C_i, K_j)$ evaluates the quality of cluster K_j for describing the class C_i. The success of a clustering solution for describing a class C_i is measured by the max value of $F(C_i, K_j)$ among clusters K_j, $j = 1, ..., k$. The overall F-measure for evaluating the clustering solution is measured by the weighted sum of the maximum F-measures for all the classes:

$$F\text{-}measure = \sum_{C_i \in C}\frac{|C_i|}{N}\max_{K_j \in K}\{F(C_i, K_j)\} \tag{3.44}$$

The value of the overall F-measure is in the range $[0, 1]$. The higher the value is, the higher the accuracy of the clustering result.

More generally, $F_\beta(C_i, K_j)$ measures can be computed as in [111]:

$$F_\beta(C_i, K_j) = \frac{(1+\beta^2) * Recall(C_i, K_j) * Precision(C_i, K_j)}{Recall(C_i, K_j) + \beta^2 * Precision(C_i, K_j)} \tag{3.45}$$

where β defines the importance of *Recall* and *Precision* in the calculation. Equation (3.43) therefore gives the F_1 measure.

V-measure [114] V-measure is an entropy-based measure which evaluates both the homogeneity and the completeness criteria of the clustering solution. The higher the value of the V-measure is, the better the clustering solution. V-measure is defined as follows:

$$\text{V-measure} = \frac{(1+\beta) * homogeneity * completeness}{\beta * homogeneity + completeness} \tag{3.46}$$

where:

- *homogeneity* measures the homogeneity of the clustering solution (see Equation (3.47) hereafter).
- *completeness* measures the completeness of the clustering solution (see Equation (3.50) hereafter).
- β is a variable which controls the contributions of homogeneity and completeness.

The homogeneity is defined as:

$$homogeneity = 1 - \frac{H(C|K)}{H(C)} \tag{3.47}$$

where $H(C|K)$ is the conditional entropy of the class distribution given the proposed clustering:

$$H(C|K) = -\sum_{j=1}^{k}\sum_{i=1}^{c}\frac{n_{ij}}{N}\log\frac{n_{ij}}{\sum_{i=1}^{c} n_{ij}} \tag{3.48}$$

and $H(C)$ is the entropy of the classes and is defined as:

$$H(C) = -\sum_{i=1}^{c}\frac{\sum_{j=1}^{k} n_{ij}}{N}\log\frac{\sum_{j=1}^{k} n_{ij}}{N} \tag{3.49}$$

Symmetrically, the completeness is defined as:

$$completeness = 1 - \frac{H(K|C)}{H(K)} \tag{3.50}$$

where $H(K|C)$ is the conditional entropy of the cluster distribution given the classes:

$$H(K|C) = -\sum_{i=1}^{c}\sum_{j=1}^{k}\frac{n_{ij}}{N}\log\frac{n_{ij}}{\sum_{j=1}^{k} n_{ij}} \tag{3.51}$$

and $H(K)$ is the entropy of the clusters:

$$H(K) = -\sum_{j=1}^{k}\frac{\sum_{i=1}^{c} n_{ij}}{N}\log\frac{\sum_{i=1}^{c} n_{ij}}{N} \tag{3.52}$$

Γ statistic [6] For evaluating the clustering result, the Γ statistic measures the correlation between the produced clustering and the classification (classes) given by the ground truth. Γ statistic is computed as follows:

$$\Gamma = \frac{M.a - m_1.m_2}{[m_1.m_2.(M - m_1).(M - m_2)]^{\frac{1}{2}}} \tag{3.53}$$

where:

$$a = \frac{1}{2}\sum_{j=1}^{k}\sum_{i=1}^{c} n_{ij}^2 - \frac{N}{2} \tag{3.54}$$

$$b = \frac{1}{2}\sum_{i=1}^{c} |C_i|^2 - \frac{1}{2}\sum_{j=1}^{k}\sum_{i=1}^{c} n_{ij}^2 \tag{3.55}$$

$$d = \frac{1}{2}\sum_{j=1}^{k} |K_j|^2 - \frac{1}{2}\sum_{j=1}^{k}\sum_{i=1}^{c} n_{ij}^2 \tag{3.56}$$

$$m_1 = a + b \tag{3.57}$$

$$m_2 = a + d \tag{3.58}$$

The value of the Γ statistic measure ranges between -1 and 1. The higher the value is, the better the clustering result.

Combinatorial approaches Combinatorial approaches evaluate the clustering result by examining the number of pairs of data points which are clustered similarly or differently in the clustering solution and in the ground truth. We define:

- N_{11}: the number of pairs of points which are clustered together in the clustering solution and in the ground truth.
- N_{00}: the number of pairs of points which are clustered separately in the clustering solution and in the ground truth.
- N_{01}: the number of pairs of points which are clustered together in the ground truth but not in the clustering solution.
- N_{10}: the number of pairs of points which are clustered together in the clustering solution but not in the ground truth.

Some measures of this kind are:

- **Rand Index [109]**: The Rand Index measures the similarity between the produced clustering and the target clustering determined by the ground truth by examining the ratio of agreements between these two clusterings (*i.e.* the ratio of the pairs of points that are clustered similarly together or separately in the two clusterings). The value of Rand Index ranges between [0,1]. The higher the value is, the higher the similarity between the clustering and the ground truth, and the better the clustering result.

$$Rand(C, K) = \frac{N_{00} + N_{11}}{N_{00} + N_{11} + N_{10} + N_{01}} = \frac{N_{00} + N_{11}}{N(N-1)/2} \tag{3.59}$$

- **Jaccard Index [91]**: Similar to the Rand Index, the Jaccard Index measures the similarity between two clusterings by examining the agreements between these two clusterings, but it ignores the pairs of points which are grouped separately in both the clustering and the ground truth.

$$Jaccard(C,K) = \frac{N_{11}}{N_{11}+N_{10}+N_{01}} \tag{3.60}$$

- **Fowlkes-Mallows Index [32]**: Let $P(C,K) = \frac{N_{11}}{N_{11}+N_{01}}$ be the probability that a pair of points of the same class are also in the same cluster, and $P(K,C) = \frac{N_{11}}{N_{11}+N_{10}}$ be the probability that a pair of point which are clustered in the same cluster belongs to a same class. The Fowlkes-Mallows Index measures the similarity between two clusterings by the geometric mean of $P(C,K)$ and $P(K,C)$:

$$Fowlkes\text{-}Mallows(C,K) = \sqrt{\frac{N_{11}}{N_{11}+N_{01}} . \frac{N_{11}}{N_{11}+N_{10}}} \tag{3.61}$$

- **Mirkin metric [94]**: The Mirkin metric evaluates the clustering result by measuring the number of disagreements between the clustering solution and the ground truth (*i.e.* the number of pairs of points which are clustered together in the clustering solution but separately in the ground truth or vice versa).

$$Mirkin(C,K) = 2(N_{01}+N_{10}) = N(N-1)(1-Rand(C,K)) \tag{3.62}$$

Discussion about clustering evaluation measures Internal measures do not consider the semantic point of view, they can therefore be applied automatically by the machine, but they only represent a numerical evaluation. External measures force the human to provide a ground truth. They are therefore more difficult to be done automatically, but this evaluation protocol is closer to the wishes of the user. It is thus a semantic evaluation.

3.5 Experimental comparison of unsupervised clustering methods

Following our theoretical analysis (Section 3.2) and formal comparison (Section 3.3) of unsupervised clustering methods, we present in this section an experimental comparison of five different methods: global k-means [76], AHC [70], R-tree [42], SR-tree [60] and BIRCH [145]. 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 (see page 50). 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.

3.5.1 Experiment protocol

In order to compare the five selected clustering methods, we use different image databases of increasing sizes (Wang[2], PascalVoc2006[3], Caltech101[4], Corel30k). Some examples of these databases are shown in Figures 3.8, 3.9, 3.10 and 3.11. Small databases are intended to verify the performance of different feature descriptors and also different clustering methods. Large databases are used to evaluate the clustering methods for structuring large amount of data.

- The Wang database (Figure 3.8) is a small and simple database which contains 1000 images of 10 different classes (100 images per class).
- The PascalVoc2006 database (Figure 3.9) contains 5304 images of 10 different classes, each image containing one or more objects of different classes. In this thesis, 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 (3885 images in total).
- The Caltech101 database (Figure 3.10) contains 9144 images of 101 classes, with 40 up to 800 images per class.
- The Corel30k database (Figure 3.11) is the largest database used, it contains 31695 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 k as the number of classes c in the ground truth.

Figure 3.8 – Example of Wang image database.

Concerning the image descriptors, we implement one global and five local descriptors. Because our study focuses on the clustering methods and not on the features (descriptors), we choose some descriptors that are widely used in the literature for our experiments (see Chapter 2).

- As a global descriptor, we use a descriptor of size 103 which is built as the concatenation of three different global descriptors:
  - *RGB histobin*: we quantize each channel of the RGB color space into 16 ranges (bins) for having, for each image, a histobin of size $3 \times 16 = 48$.

[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/`

Figure 3.9 – Example of PascalVoc2006 image database.

Figure 3.10 – Example of Caltech101 image database.

Figure 3.11 – Example of Corel30k image database.

- *Gabor filters*: we use 24 Gabor filters on 4 orientations and 6 frequencies. The statistical measure associated with each output image is the mean and standard deviation. We obtain thus a vector of size $24 \times 2 = 48$ for the texture.
- *Hu's moments*: we implement the 7 invariant moments of Hu for describing the shape of the image.

- For local descriptors, we use the SIFT and some color SIFT (CSIFT, rgSIFT, RGBSIFT, OpponentSIFT) descriptors. These descriptors are widely used nowadays for their high performance. We use the SIFT descriptor code of David Lowe[5] and the color SIFT descriptors codes 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 Section 2.2.2).

As mentioned above, we implement five different clustering methods: global k-means [76], AHC [70], R-tree [42], SR-tree [60] and BIRCH [145]. For the Agglomerative Hierarchical Clustering (AHC), we have tested the five distances described in Section 3.2.4 and Equations (3.12) to (3.16) (single-linkage, complete-linkage, average-linkage, centroid-linkage and Ward's method) in our experiments.

[5] `http://www.cs.ubc.ca/*lowe/keypoints/`.
[6] `http://koen.me/research/colordescriptors/`.

We found that the Ward's distance gives the best results among the five. Therefore, to simplify the presentation of results in this section, we present only AHC clustering with the Ward's method for measuring distance between clusters.

In order to evaluate our clustering results, we use both internal and external measures. As external measures compare the results of the clustering to what the human wants, they are more suitable to evaluate the clustering results in our interactive context where the user is involved. Therefore, external measures are mostly used in our evaluations. Internal measures are also used for evaluating the differences between numerical evaluation and semantic evaluation. Among internal measures, there are simple measures which evaluate only one aspect (homogeneity, separation or completeness) of the clustering solution, and also more complex measures which evaluate simultaneously many aspects of the clustering results. In this thesis, we do not study how to evaluate the different clustering evaluation measures, but we choose here five different complex measures that seem representative and that are widely used nowadays: Silhouette Width (SW) [115] as internal measure and V-measure [114], Rand Index [109], Jaccard Index [91] and Fowlkes-Mallows Index [32] as external measures. Higher values of these measures indicate better clustering results.

3.5.2 Experimental results

3.5.2.1 Clustering methods and feature descriptors analysis

The first set of experiments evaluates the performances of different clustering methods and different feature descriptors. Another objective is to 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 is 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 $c = 10$ classes).

Method analysis Figure 3.12 shows the results 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 the BIRCH methods give in general the best results, while R-tree, SR-tree and AHC give similar results. Moreover, the internal measure SW appreciates the global k-means more than the BIRCH method, while external measures appreciate BIRCH more than the global k-means.

We 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 Section 3.2.4, 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 leaf entry, and thus on the results of the clustering. Figure 3.13 shows the influence of

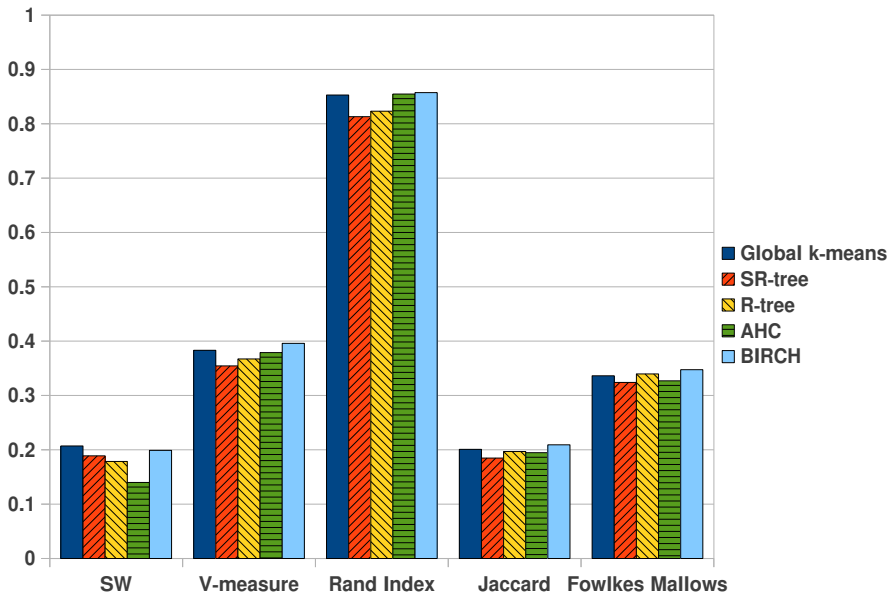


Figure 3.12 – Clustering methods (global k-means, SR-tree, R-tree, AHC, BIRCH) comparison using global feature descriptor on the Wang image database. Five measures (Silhouette Width, V-measure, Rand Index, Jaccard Index, Fowlkes–Mallows Index) are used. The vertical axis represents the values of the measures. The higher the values of these measures are, the better the results.

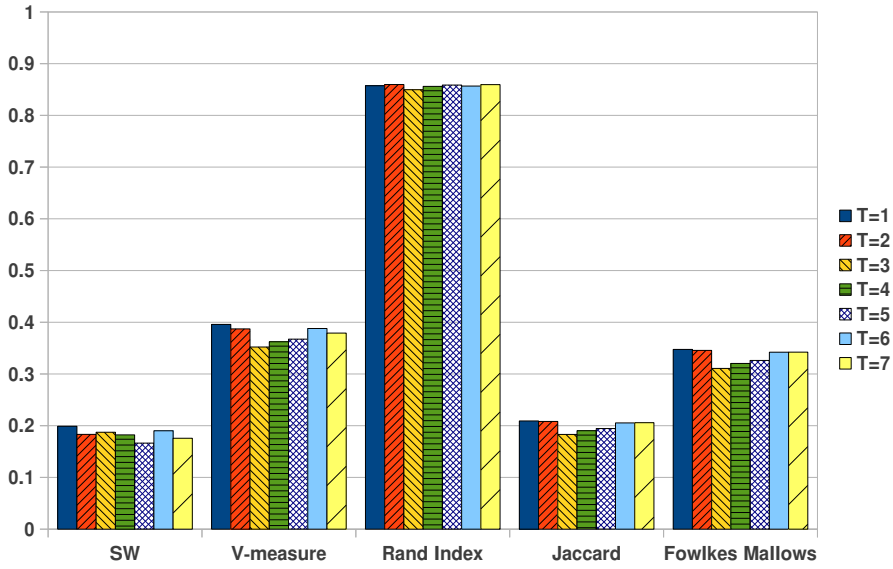


Figure 3.13 – Influence of the threshold T (T = 1,...,7) on the BIRCH clustering results using the Wang image database and global features. Five measures (Silhouette Width, V-measure, Rand Index, Jaccard Index, Fowlkes–Mallows Index) are used. The vertical axis represents the values of the measures.

T on the results of BIRCH clustering. 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 initialization. We can see that for the Wang image database and for the value of

T in our experiments, the results are very stable when the values of T are varying, even though $T = 1$ gives slightly better result. For the R-tree and the SR-tree clustering methods, 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 branch node if the number of children is high enough. Figure 3.14 shows the influence of these parameters to the result of these two methods. We can see that these parameters have a large influence 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 (according to the Rand Index measure). 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 vector 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.

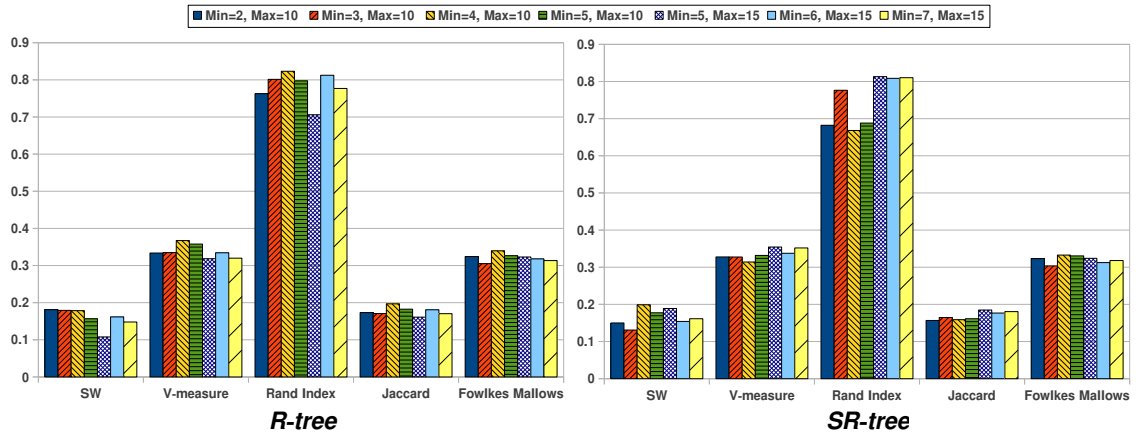


Figure 3.14 – Influence of minimum and maximum numbers of children on R-tree and SR-tree results, using the Wang image database and global features. Five measures (Silhouette Width, V-measure, Rand Index, Jaccard Index, Fowlkes–Mallows Index) are used. The vertical axis represents the values of the measures.

Feature descriptor analysis Figure 3.15 shows the results of 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, 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. Among local descriptors, SIFT gives the worst results while rgSIFT gives the best results. This may be explained by the fact that in the Wang database, color is very important. Van de Sande *et al.* [130] show, in their evaluation of different local color descriptors, the high performance of the rgSIFT and CSIFT descriptors. Thus, in the remaining experiments, we use three feature descriptors (global feature descriptor, CSIFT and rgSIFT).

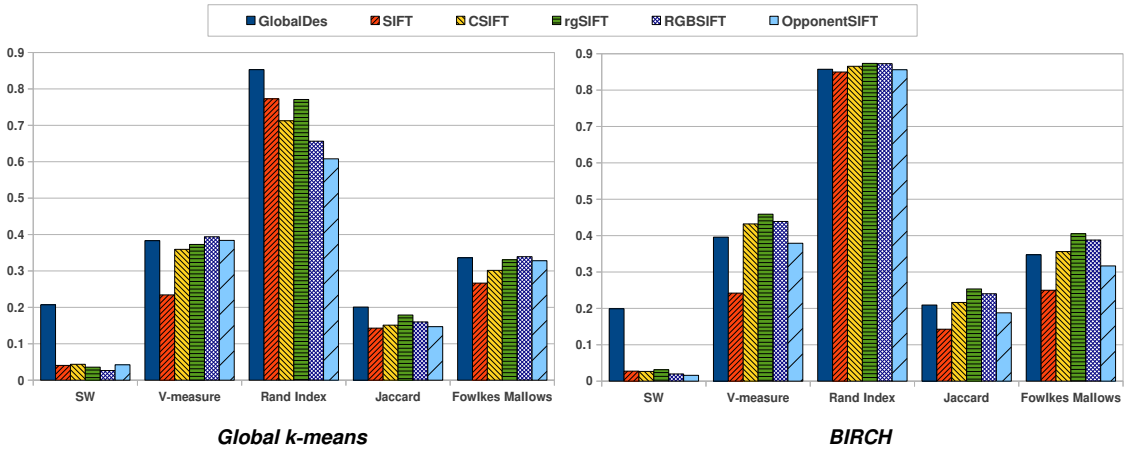


Figure 3.15 – Feature descriptor (global feature descriptor, local feature descriptors (SIFT, CSIFT, rgSIFT, RGBSIFT, OpponentSIFT)) comparison using the global k-means and BIRCH clustering methods on the Wang image database. Five measures (Silhouette Width, V-measure, Rand Index, Jaccard Index, Fowlkes–Mallows Index) are used. The vertical axis represents the values of the measures.

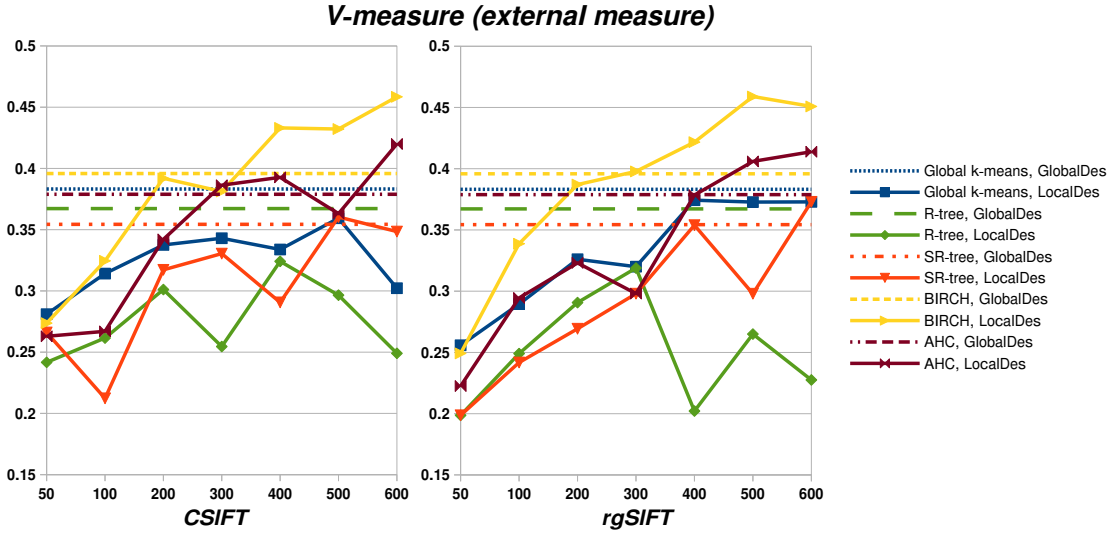


Figure 3.16 – Clustering method (global k-means, R-tree, SR-tree, BIRCH and AHC) and feature descriptor (global k-means, CSIFT and rgSIFT) comparisons using V-measure on the Wang image database. The dictionary size of the "Bag of Words" approach used jointly with the local descriptors is from 50 to 600. The vertical axis represents the values of the V-measure.

Method and feature descriptor comparisons As stated in Section 2.2.2, the dictionary size determines the size of the feature vector of each image. Given the problems and difficulties related to large dimension spaces, we prefer to choose a relatively small size of 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 in the clustering phase. But, a too small dictionary may not be sufficient to accurately describe the database. Therefore, we must find the best trade-off between

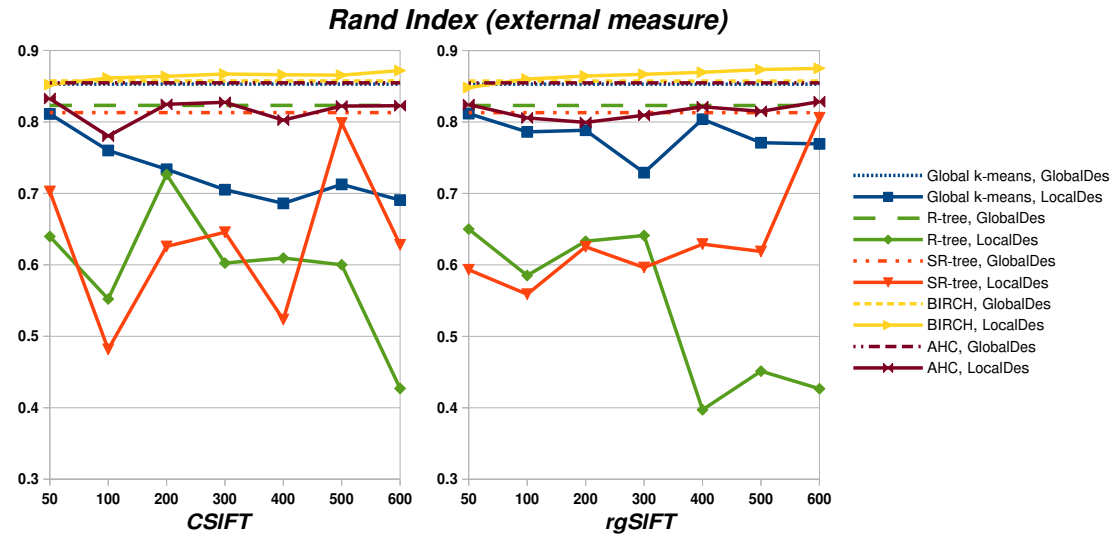


Figure 3.17 – Clustering method (global k-means, R-tree, SR-tree, BIRCH and AHC) and feature descriptor (global k-means, CSIFT and rgSIFT) comparisons using Rand Index on the Wang image database. The dictionary size of the "Bag of Words" approach used jointly with the local descriptors is from 50 to 600. The vertical axis represents the values of the Rand Index.

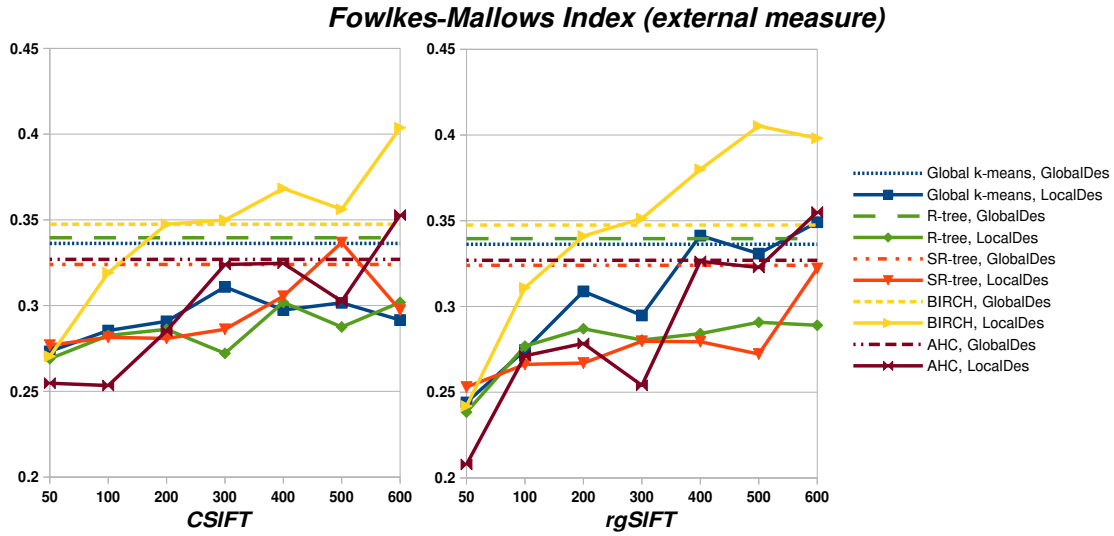


Figure 3.18 – Clustering method (global k-means, R-tree, SR-tree, BIRCH and AHC) and feature descriptor (global k-means, CSIFT and rgSIFT) comparisons using Fowlkes-Mallows Index on the Wang image database. The dictionary size of the "Bag of Words" approach used jointly with the local descriptors is from 50 to 600. The vertical axis represents the values of the Fowlkes-Mallows Index.

the performance and the size of the dictionary. Figures 3.16, 3.17, 3.18, 3.19 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 600). The measures used for evaluation are respectively the V-measure, Rand Index, Fowlkes-Mallows Index and Silhouette Width measures. The Jaccard Index is not used for these analysis because it is

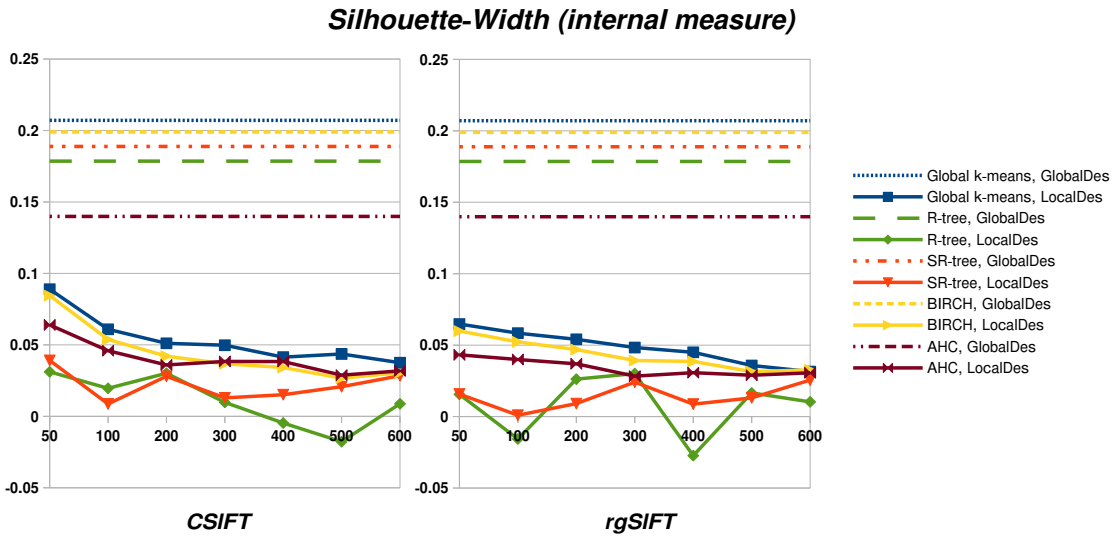


Figure 3.19 – Clustering method (global k-means, R-tree, SR-tree, BIRCH and AHC) and feature descriptor (global k-means, CSIFT and rgSIFT) comparisons using Silhouette Width on the Wang image database. The dictionary size of the "Bag of Words" approach used jointly with the local descriptors is from 50 to 600. The vertical axis represents the values of the Silhouette Width measure.

very similar with the Fowlkes-Mallows Index (we can see that they logically give similar evaluations in the previous results). Note that *LocalDes* means the local descriptor (CSIFT on the left-hand side of the figures or rgSIFT on the right-hand side of the figures). And *GlobalDes* means the global descriptor, which is not influenced by any size of the 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 methods. When using external measures (V-measure, Rand Index and Fowlkes-Mallows Index), BIRCH using rgSIFT with a dictionary of 400 to 600 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 supervision, the compactness and the separation of clusters based on the Euclidean distance between elements in a same cluster and in different clusters (which favors k-means), 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 further experiments in this chapter because it is a good trade-off between the size of the feature vector and the performance (for both internal and external measures). The value 500 for the dictionary size which give in general the best results is also used for further experiments. 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 according to the internal measure.

3.5.2.2 Scalability study

Because the Wang image database is very simple and small, the results of clustering obtained on it may not be representative of what happens with huge masses of data. Thus, in the following experiments, we analyze the clustering results with larger image databases (PascalVoc2006, Caltech101 and Corel30k). Global k-means, BIRCH and AHC are used because of their high performance and stability. In the case of the Corel30k image database, the AHC method is not used because of the lack of the RAM memory. In fact, the AHC clustering requires a large amount of memory when processing more than 10000 elements, while the Corel30k contains more than 30000 images. This problem could be solved by using the incremental version [110] 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 tested. Note that the size of the dictionary used for both local descriptors is 200 and 500.

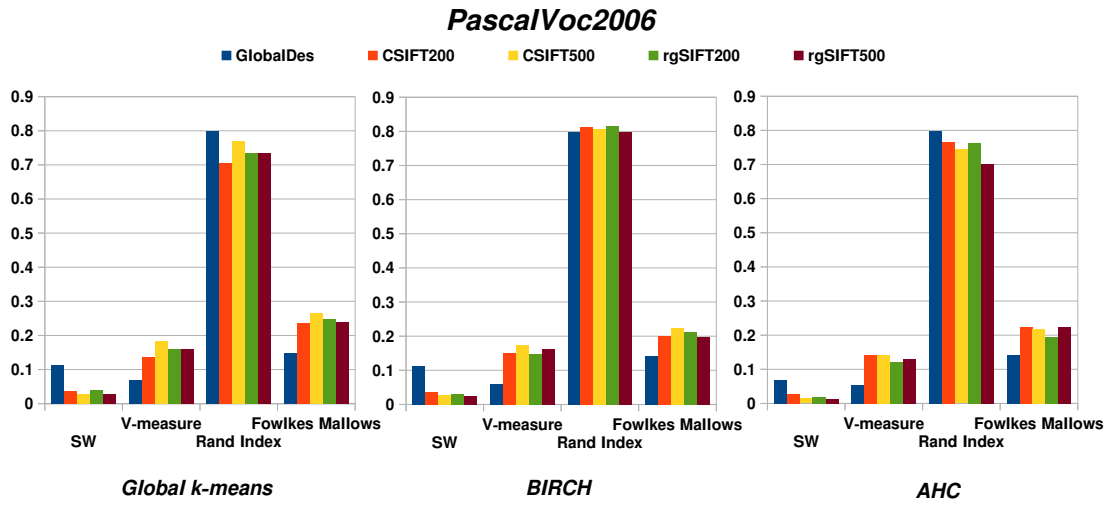


Figure 3.20 – Global k-means, BIRCH and AHC clustering methods and features descriptor (global descriptor, CSIFT, rgSIFT) comparisons using Silhouette Width (SW), V-measure, Rand Index and Fowlkes–Mallows measures on the PascalVoc2006 image database. The vertical axis represents the values of the Fowlkes-Mallows Index.

Figures 3.20 and 3.21 show the results of the global k-means, BIRCH and AHC clustering methods respectively on the PascalVoc2006 and Caltech101 image databases. The corresponding processing time for these experiments are shown in Table 3.2. Internal measures (Silhouette Width (SW)) and external measures (V-measure, Rand Index and Fowlkes-Mallows Index) are used for these experiments. We can see that the numerical evaluation (internal measure) always appreciates the global descriptor but the semantic evaluations (external measures) appreciate the local descriptors in almost all cases. Concerning different clustering methods, we can see that the AHC clustering gives the worst results. And in general, the global k-means and the BIRCH clustering methods give similar results. Concerning the size of the dictionary in the case of the local descriptors, we can see that the dictionary of size 200 give slightly better results than the dictionary of size 500. Moreover,

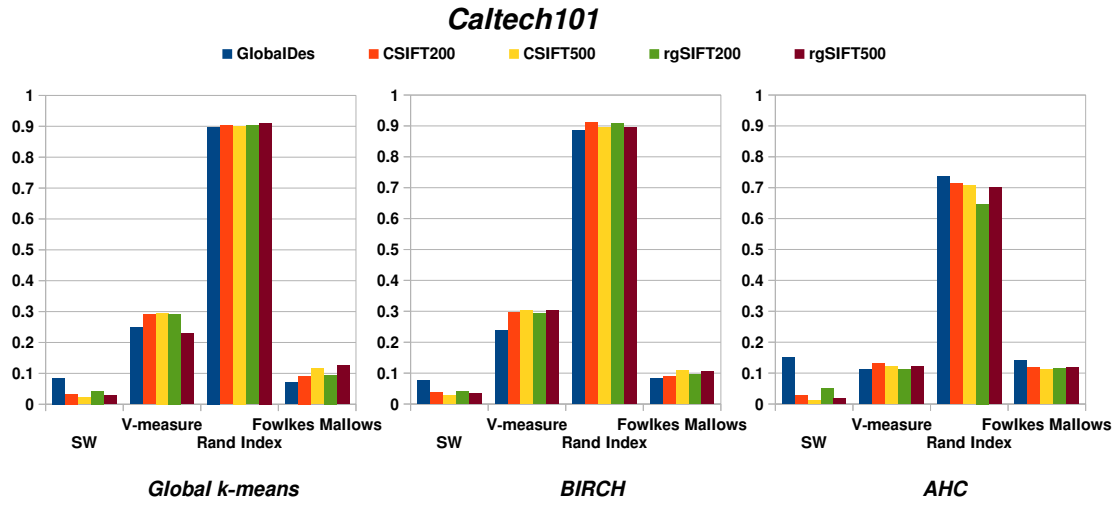


Figure 3.21 – Global k-means, BIRCH and AHC clustering methods and features descriptor (global descriptor, CSIFT, rgSIFT) comparisons using Silhouette Width (SW), V-measure, Rand Index and Fowlkes–Mallows measures on the Caltech101 image database. The vertical axis represents the values of the Fowlkes-Mallows Index.

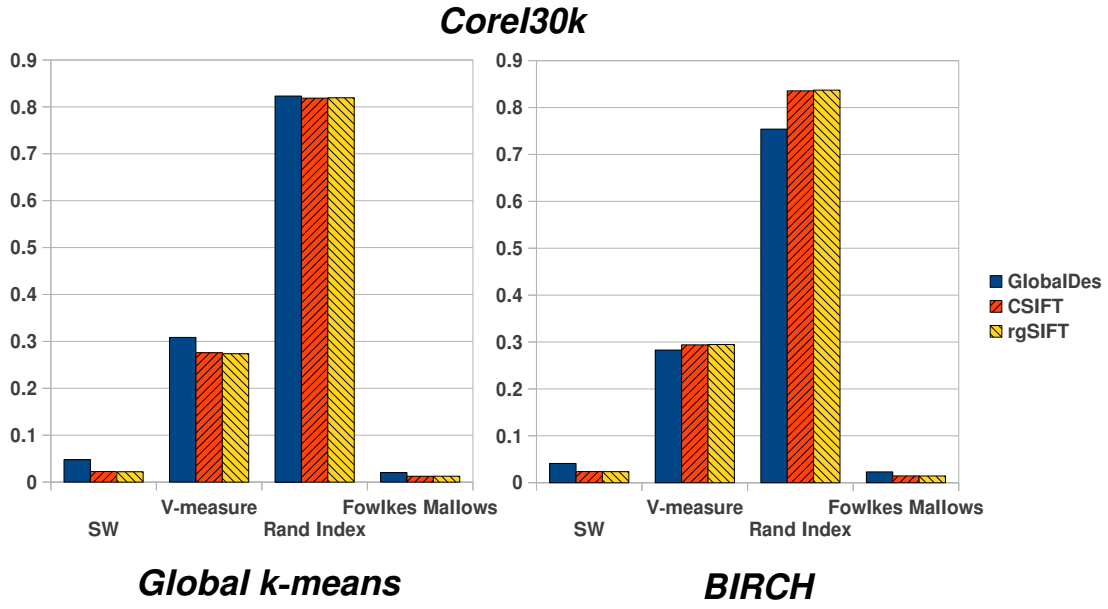


Figure 3.22 – Global k-means and BIRCH clustering methods and features descriptor (global descriptor, CSIFT, rgSIFT) comparisons using Silhouette Width (SW), V-measure, Rand Index and Fowlkes–Mallows measures using the Corel30k image database. The vertical axis represents the values of the Fowlkes-Mallows Index.

lower dictionary can improve the execution time. Therefore, the dictionary the size of which is 200 is used for the experiments on the Corel30k image database shown in Figure 3.22. We can see that for the Corel30k image database, the rgSIFT gives better results than the CSIFT. Moreover, BIRCH using rgSIFT gives in general the best result. Concerning the execution time, we notice that the clustering using the

Table 3.2 – Processing time of the Global k-means, BIRCH and AHC clustering methods on the PascalVoc2006, Caltech101 and Corel30k image databases corresponding to experiments shown in Figures 3.20 and 3.21

		Global k-means	BIRCH	AHC
PascalVoc2006	GlobalDes	23'52"	0'20"	27'06"
	CSIFT200	30'12"	0'26"	28'49"
	CSIFT500	49'59"	0'55"	31'47s
	rgSIFT200	29'53"	0'25"	28'47"
	rgSIFT500	49'42"	0'50"	31'38"
Caltech101	GlobalDes	52h06'	9'07"	5h59'
	CSIFT200	62h26'	12'09"	6h09'
	CSIFT500	95h31'	17'58"	6h24'
	rgSIFT200	62h10'	11'53"	6h08'
	rgSIFT500	95h19'	17'04"	6h22'

global descriptor is faster than that using the local descriptors. This is due to the fact that the global feature descriptor has a dimension of 103 while the dimension of the local descriptors is 200 and 500. The higher the size of the dictionary, the higher the processing time. Moreover, among the local descriptors, the clustering using rgSIFT is faster. And in comparison with global k-means and AHC, BIRCH is much faster.

3.5.3 Discussion on the experimental comparison

Concerning the different feature descriptors, the global descriptor is appreciated by the internal measure (numerical evaluation) but the external measures (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 experiments. This might come from the fact that global features are computed based on the color, shape and/or texture which may have no concern with the semantic point of view, while local features are computed by detecting, in the image, the interest points which may be more related to the user perception.

Concerning the stability of the different clustering methods, global 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 for R-tree and SR-tree or the threshold T determining the density of each leaf entry for BIRCH). 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 because there is no parameter to tune according to the interaction of the user, 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 modify the value of the input parameters in order to improve the final results. R-tree and SR-tree have a very unstable behaviour when varying their parameters or the number of visual words (in the case of the local descriptor) compared to BIRCH. AHC is also much more sensitive than BIRCH 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, especially in our case where the best values of the parameters highly depend on the database characteristics.

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 according to external measures. 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 parse the data many times (one time for each iteration). Therefore, global k-means clustering is much more computationally complex than BIRCH, especially when the number of clusters k is high, because we have to execute the k-means clustering k times (with the number of clusters varying from 1 to k), while in BIRCH, we have to execute the k-means (if it is used) only one time for clustering all CF entries of the leaf nodes. Additionally, the AHC clustering costs much time and memory when the number of images is high, because of its time complexity $O(N^2 \log N)$ and its space complexity $O(N^2)$. The incremental version [110] 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, and are much more unstable when varying their parameters than BIRCH. **For all these reasons, we consider that BIRCH+rgSIFT is the best choice in our context.**

3.6 Discussion

Regarding clustering evaluation measures, while internal measures are numerical evaluations which do not consider the semantic point of view, but which can be computed automatically, external measures require the ground truth provided by the user in order to provide the evaluation which is closer to the wishes of the user. In the context of interactive clustering, internal measures can be used for evaluating the results of the initial unsupervised clustering, but external measures may be more suitable for measuring the performance of the clustering involving the interactions of the user.

This chapter compares both formally and experimentally different unsupervised clustering methods in the context of multi-dimensional data with image databases of large size. As the final objective of this thesis is to allow the user to interact with the system in order to improve the results of the clustering, it is therefore important that a clustering method is incremental and that the clusters are hierarchically structured (so that it is easy to merge or split clusters according to the wishes of the users). Formally, the hierarchical methods (BIRCH, R-tree, SS-tree and SR-tree) are the most suitable to our context because they give hierarchical structures of clusters, and they are by nature incremental and adapted to large databases. Experimentally, we compare some hierarchical 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 hierarchical methods. Our results indicate that BIRCH is less sensitive to variations in its parameters and/or to the dictionary size in the case of local descriptors than AHC, R-tree and SR-tree. Moreover, BIRCH may be more promising to be used in the incremental context than global k-means. BIRCH is also more efficient than global k-means and AHC in the context of large image databases. The experimental results show that BIRCH+rgSIFT is the best choice in our context.

Unsupervised clustering methods are suitable for the initial clustering when we know nothing about the ground truth (*i.e.* before the user is involved). In the context of an interactive clustering system, the user can be involved in the further steps for providing feedback to the system. Then, another clustering is done by considering the user feedback. This clustering may then be semi-supervised. Semi-supervised clustering is studied in the next chapter.

3.7 Summary of the chapter

This chapter presents a survey of the most principal unsupervised clustering methods, divided into four types: partitioning methods, grid-based methods, density-based methods and hierarchical methods. Moreover, different internal and external measures (also considered respectively as unsupervised and supervised measures) for evaluating the clustering results are also presented.

The first contribution of this chapter lies in respectively analyzing the advantages and drawbacks of different unsupervised clustering methods in a context in which the user is involved in the clustering of huge masses of data and where incrementality and hierarchical structuring are needed. Based on a formal comparison of the analyzed methods, the hierarchical methods (BIRCH, R-tree, SS-tree and SR-tree) are determined as the most suitable to our context.

The second contribution of this chapter is an experimental comparison, using both internal and external measures, of some unsupervised 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. Based on this experimental comparison, the BIRCH+rgSIFT, using the Bags of Words approach, is the best choice in our context.

Part II

Interaction

Chapter 4

Semi-supervised clustering

4.1 Introduction

Because of the "semantic gap" between high-level semantic concepts expressed by the user and the low-level features extracted from the images, the results of unsupervised clustering methods are generally different from the intent of the user. Therefore, in some cases unsupervised clustering can be replaced by semi-supervised clustering where a small amount of supervised information is available in order to guide the clustering of unlabelled data. Supervised information can be provided by the user or another system at the beginning of the clustering in the form of prior knowledge (traditional semi-supervised clustering) or can be provided progressively during each interactive iteration in the form of user's feedback useful for further clustering steps (interactive semi-supervised clustering). Note that the prior knowledge or the feedback provided by the user in each step is too poor to be used with supervised learning. While only similarity information is used in the case of unsupervised clustering, it is used in conjunction with the provided supervised information in the case of semi-supervised clustering. In the latter case, supervised information is used to guide the structuring process. In general, semi-supervised clustering aims at maximizing intra-cluster similarity, minimize inter-cluster similarity while keeping a high consistency between the clustering results and the domain knowledge.

Different semi-supervised clustering methods are presented and analyzed in this chapter. Different kinds of supervised information which are usually used in semi-supervised clustering are presented in Section 4.2. Section 4.3 presents a brief survey of the semi-supervised clustering methods (traditional methods as well as interactive methods) and analyzes the possibility to use these methods in an interactive context in which users are involved for providing supervised information in the form of feedback at each interactive iteration. An experimental comparison of some semi-supervised clustering methods in an interactive context is presented in Section 4.4. Finally, Section 4.5 gives some conclusions and discussions.

4.2 Different kinds of supervised information

In general, supervised information for semi-supervised clustering is based on some knowledge given by the user or another system, either in the form of class labels (for some objects) or pairwise constraints between objects. For class label super-

vised information, class labels are assigned to a small amount of objects in the data set. In general, it requires at least one labelled object for each class. Pairwise constraints specify whether two objects should be in the same cluster (must-link) or in different clusters (cannot-link). Depending on the moment when supervised information is provided, semi-supervised clustering methods can be divided into traditional methods and interactive methods. In the case of traditional semi-supervised clustering, supervised information is given at the beginning of the clustering and is not changed depending on the clustering results. Whereas in the case of interactive semi-supervised clustering, supervised information is progressively provided by the user based on the clustering results at each interactive iteration.

As unsupervised clustering produces clusters based on the low-level features extracted from the images, the produced clusters may not be the ones wished by the user. Therefore, supervised information is needed to guide the clustering process for giving clusters which are more adapted to the user's wishes. For instance, for clustering a database with thousands of animal images, a user may want to cluster by animal species or by background landscape types. An unsupervised clustering method may give, as a result, a cluster containing images of horses with a grass background together with some images of elephants with a grass background and another cluster containing images of elephants with a sand background. These results are ideal when the user wants to cluster by background landscape types. But they are poor when the user wants to cluster by animal species. In this case, the user can provide some supervised information to guide the clustering process. When using supervised information in the form of class labels, the user can assign the label "elephant" for some images of elephants with a grass background and for some images of elephants with a sand background, and the label "horse" for some images of horses with a grass background. By using these labels, the user specifies that he wants to cluster by animal species and thus avoids the merge of images of elephants and horses with a grass background into a same cluster. With the same purpose, when using supervised information in the form of pairwise constraints between images, the user can specify must-link constraints between images of elephants with a grass background and images of elephants with a sand background as well as cannot-link constraints between images of elephants with a grass background and images of horses with a grass background.

4.3 Semi-supervised clustering methods

This section presents a brief overview of different semi-supervised clustering methods (traditional semi-supervised methods as well as interactive semi-supervised methods), in which a small amount of supervised information is provided by the user or another system in order to guide the clustering of unlabelled data. Note that semi-supervised clustering is different from semi-supervised classification. Semi-supervised classification trains labelled data for inducing the classification model, by exploiting some additional unlabelled data. Whereas semi-supervised clustering exploits additional labelled data (labelled objects or pairwise constraints) to aid and bias the clustering of unlabelled data. Moreover, in semi-supervised classification, the set of all the data categories (data classes) has to be given in the labelled data

and is not changed during the classification process. While the category set can be extended in the case of semi-supervised clustering (*i.e.* the number of obtained clusters may be different from the number of categories given in labelled data). Therefore, semi-supervised clustering is more compatible than semi-supervised classification when knowledge of data categories is incomplete.

Semi-supervised clustering (traditional or interactive) has been developed in the last decade and some methods have been published in the literature. Similar to unsupervised clustering, semi-supervised clustering can be divided in semi-supervised hard clustering and semi-supervised fuzzy clustering. As in this thesis, we are interested in only hard clustering methods, semi-supervised fuzzy clustering methods [40] [80] [84] are not studied in this chapter. Hard semi-supervised clustering can be divided into semi-supervised clustering with class labels or semi-supervised clustering with pairwise constraints corresponding to the two kinds of supervised information. As far as we know, semi-supervised clustering methods with class labels currently use supervised information in the form of prior knowledge or user feedback, while the current methods with pairwise constraints use only supervised information in the form of prior knowledge. In the experimental section of this chapter, we adapt the semi-supervised methods with pairwise constraints into the interactive context so that supervised information is not provided *a priori* but during different interactive iterations based on the clustering results. Most of the semi-supervised clustering methods presented in this section are partitioning methods, only few of them are hierarchical methods. Depending on the final application, we can choose the suitable kind of supervised information as well as the suitable semi-supervised clustering method.

In this chapter, we use the following notations:

- $X = \{x_i | i = 1, ..., N\}$: the set of input feature vectors for clustering, where N is the number of vectors.
- $K = \{K_j | j = 1, ..., k\}$: the set of clusters, where k is the number of clusters.
- μ_j: the centre of cluster K_j.
- $|K_j|$: the number of points in cluster K_j.
- $K(x_i)$: the cluster containing the point x_i.
- $\mu(x_i)$: the centre of the cluster containing the point x_i.

4.3.1 Semi-supervised clustering with class labels

Semi-supervised clustering using labelled objects as supervised information has both traditional and interactive forms. Traditional methods use supervised information in the form of prior knowledge (labelled objects) provided at the beginning while interactive methods use supervised information in the form of feedback progressively provided in each interactive iteration, depending on the clustering results.

4.3.1.1 Traditional semi-supervised clustering with class labels

Some traditional semi-supervised clustering methods using labelled objects as prior knowledge have been published such as seeded-kmeans [9], constrained-kmeans [9].

Seeded-kmeans [9] : Seeded-kmeans is a partitioning method which is a variant of the k-means algorithm that uses seeds to guide the k-means algorithm. Given a data set X that should be partitioned into k clusters $\{K_j\}_{j=1}^{k}$. Prior knowledge for this method is a small subset of the input data set $S \subseteq X$, called *seed set*, containing user-specified labelled objects of k different clusters. Assume that for each cluster K_j, there is at least one seed point $x_i \in S$. Therefore, the *seed set* contains k disjoint partitions $\{S_j\}_{j=1}^{k}$, each partition containing points of one class. Unlike the k-means algorithm which initializes the clustering from k random means, the seeded-kmeans uses the *seed set* for initializing clustering, so that the mean of the j^{th} cluster is initialized as the mean of the partition S_j. Following this we repeat, until convergence, the re-assignment of each object to the nearest mean and the re-estimation of the means, given the assigned objects. The clustering does not make any difference between the points in the *seed set* and other points in the data set. Note that the distance used for the seeded-kmeans is the Euclidean distance. The algorithm of the seeded-kmeans is as follows:

1. Initialize the mean μ_j of each cluster K_j as the mean of each partition S_j $(j = 1, ..., k)$ of the *seed set*:

$$\mu_j = \frac{1}{|S_j|} \sum_{x_i \in S_j} x_i \tag{4.1}$$

2. Repeat until *convergence*

   (a) **Re-assignment**: assign each data point $x_i \in X$ to the nearest cluster K_{j^*}:

$$j^* = \underset{j}{\operatorname{argmin}} ||x_i - \mu_j||^2 \tag{4.2}$$

   (b) **Re-estimation**: re-estimate the mean of each cluster K_j:

$$\mu_j = \frac{1}{|K_j|} \sum_{x_i \in K_j} x_i \tag{4.3}$$

As a partitioning method, the seeded-kmeans gives a "flat" organization of clusters. The supervised information is provided in the form of prior knowledge. However, the seeded-kmeans is still very basic in the way of handling prior knowledge, as supervised information is not used during the re-clustering phase, but only used for initializing the cluster centres. Therefore, the seeded-kmeans allows the violation of the provided class labels. It can be adapted to the interactive context but the user should have prior knowledge about the data set in order to provide at least one labelled object for each of the k clusters.

Constrained-kmeans [9] Another variant of the k-means algorithm that uses seeds to guide the k-means algorithm is the constrained-kmeans algorithm. Prior knowledge of constrained-kmeans is also a *seed set* containing user-specified labelled objects of k different clusters. As in seeded-kmeans, the *seed set* is used for initializing clustering so that the mean of the j^{th} cluster is initialized as the mean of the

partition S_j using equation (4.1). However, in the subsequent steps, while seeded-kmeans re-estimates the labels of all objects in the data set X, constrained-kmeans maintains the data points of the *seed set* S in their initial clusters and re-estimates only the labels of the non-seed objects $x_i \in X \setminus S$. Similar to the seeded-kmeans, the Euclidean distance is used for calculating the distance between objects. The algorithm of the constrained-kmeans is as follows:

1. Initialize the mean μ_j of each cluster K_j as the mean of each partition S_j $(j = 1, ..., k)$ of the *seed set* using equation (4.1).

2. Repeat until *convergence*

   (a) **Re-assignment**:

   - For $x_i \in S$: if $x_i \in S_j$, assign x_i to the cluster K_j which contains all the points of S_j.
   - For $x_i \in X \setminus S$, assign x_i to the nearest cluster K_{j^*} using equation (4.2).

   (b) **Re-estimation**: re-estimate the mean of each cluster K_j using equation (4.3).

Similar to the seeded-kmeans, constrained-kmeans is a partitioning method and gives a "flat" organization of clusters. Supervised information is also in the form of prior knowledge. As the data points of the seed set are maintained in their initial clusters, constrained-kmeans does not allow constraint violation. As seeded-kmeans, the prior knowledge of the user about the data set is required in order to implement the constrained-kmeans in the interactive context.

4.3.1.2 Interactive semi-supervised clustering with class labels

Semi-supervised clustering methods using feedback as supervised information are defined as *interactive semi-supervised methods*. Among the interactive semi-supervised clustering methods using class labels, we can cite: cluster-level semi-supervised clustering [26], clustering based on relevance feedback [64] and the Rocchio formula [113] based clustering method.

Interactive cluster-level semi-supervised clustering [26] Dubey *et al.* proposed an interactive cluster-level semi-supervised clustering framework for document analysis. In this model, knowledge is not provided *a priori*, but is progressively provided as user feedback during different interactive iterations. In each interactive iteration, the user provides a set of *assignment feedback* and a set of *cluster description feedback*. The numbers of *assignment feedback* and *cluster description feedback* given in different interactive iterations can be different. Using *assignment feedback*, the user moves an object from one cluster to another. Using *cluster description feedback*, the user modifies the feature vector of any current cluster to make it more meaningful. Note that the method proposed in [26] is dedicated to document analysis, in which the feature vector of a cluster or a document is a vector representing the weights of different text words in the dictionary. For modifying the feature vector of a cluster, the user can, for example, click and drag a weight curve over different

text words. In each interactive iteration, the system re-organizes the objects in the data set, not only based on the feedback provided in this iteration but also on the feedback accumulated from all the previous iterations. The re-clustering minimizes the average distance between points and their cluster centres while minimizing the violation of constraints corresponding to feedback. The interactive process continues until the clustering result satisfies the user. This interactive framework is useful for large databases when supervised information is not provided in advance.

In each interactive iteration, let us consider that l and m are respectively the total numbers of assignment feedback and cluster description feedback provided by the user over all different interactive iterations, supervised information for the re-clustering phase of this interactive iteration are:

- Let $F^a = \{f_i^a\}_{i=1}^l$ be the set of l assignment feedback provided by the user over all different interactive iterations (the letter 'a' means assignment feedback). The i^{th} assignment feedback $f_i^a = \{x_i^a, M_i^a, \mu_i^a\}$ indicates that, having the set of cluster centres $M_i^a = \{\mu_{ij}^a\}_{j=1}^k$, the user wants to assign the data point x_i^a to the cluster corresponding to the centre μ_i^a from M_i^a. Note that the set of cluster centres M_i^a indicates the cluster centres at the moment when the assignment feedback f_i^a is provided. As the cluster centres generally change after each interactive iteration, the set of cluster centres at the current iteration may be different from M_i^a if the assignment feedback f_i^a is provided at a previous iteration.

- Let $F^d = \{f_i^d\}_{i=1}^m$ be the set of m cluster description feedback provided by the user over all different interactive iterations (the letter 'd' means cluster description feedback). For the i^{th} cluster description feedback f_i^d, the user observes the feature vector of a cluster centre and provides a new feature vector for this cluster centre. Cluster description feedback is suitable for the case of document clustering where the user could assign more important weights for some text words which he thinks they better describe the cluster. But this kind of feedback is not compatible in the case of image clustering as it is difficult for the user to understand the image feature descriptors which are low-level features. Therefore, we do not describe here the details of neither the cluster description feedback nor the processing concerning cluster description feedback. See [26] for more information about this kind of feedback.

The interactive cluster-level semi-supervised clustering proposed in [26] is based on the k-means clustering. Without feedback, the objective function of the traditional k-means (using Euclidean distance) is as follows:

$$E^x = \sum_i (x_i - \mu(x_i))^2 \tag{4.4}$$

where $\mu(x_i)$ is the cluster centre corresponding to the data point x_i. In order to incorporate the user feedback in the objective function, a constraint and a corresponding penalty for violating this constraint are associated for each feedback.

The constraint associated with an assignment feedback $f_i^a = \{x_i^a, M_i^a, \mu_i^a\}$ is that every time the current set of cluster centres $\{\mu_j\}_{j=1}^k$ exactly matches $M_i^a = \{\mu_{ij}^a\}_{j=1}^k$, x_i^a always has to be assigned to the cluster corresponding to the centre μ_i^a. If the

current set of cluster centres matches M_i^a but x_i^a is currently assigned to $\mu(x_i^a) \neq \mu_i^a$, then this constraint is violated and the penalty for this violation is calculated as the distance between μ_i^a and $\mu(x_i^a)$. Of course, the set of cluster centres at the current interactive iteration can be different from M_i^a. In order to deal with this problem, the relevance $R_i^a(K)$ of an assignment constraint f_i^a, given the current set of cluster centres $\{\mu_j\}_{j=1}^k$, is defined as the similarity of $\{\mu_j\}_{j=1}^k$ with the set of cluster centres M_i^a specified in the feedback. This relevance is measured using the best mapping $Map_i^a(K)$ between the current set of cluster centres $\{\mu_j\}_{j=1}^k$ and the set of cluster centres M_i^a specified in the feedback. The best mapping between the two sets of cluster centres is defined as the mapping having the smallest sum of distances between mapped cluster centres from the two sets. The relevance is computed as follows:

$$R_i^a(K) = exp(-(\sum_{j=1}^{k}(\mu_j - Map_i^a(\mu_j))^2)) \tag{4.5}$$

where $Map_i^a(\mu_j)$ is, among the set of cluster centres $M_i^a = \{\mu_{ij}^a\}_{j=1}^k$, the mapped cluster centre, defined by the best mapping, corresponding to the cluster centre μ_j of the current set of clusters. For an assignment feedback f_i^a, given the current set of cluster centres $\{\mu_j\}_{j=1}^k$, Dubey *et al.* [26] assume that there is no penalty when x_i^a is currently assigned to the mapped cluster centre $Map_i^a(\mu_i^a)$ of μ_i^a in the current set of cluster centres. If, however, $\mu(x_i^a) \neq Map_i^a(\mu_i^a)$, then the constraint corresponding to f_i^a is violated and the penalty for this violation is the squared Euclidean distance between $\mu(x_i^a)$ and $Map_i^a(\mu_i^a)$. The assignment error takes into account both penalty and the current relevance of the constraint:

$$E_i^a = (\mu(x_i^a) - Map_i^a(\mu_i^a))^2 \times R_i^a(K) \tag{4.6}$$

and the total error for all assignment feedback is computed as follows:

$$E^a = \sum_{i=1}^{l} E_i^a \tag{4.7}$$

A constraint and a penalty for violating this constraint are also associated with each cluster description feedback f_i^d. Let E_i^d is the error associated with the cluster description feedback f_i^d (see [26] for more information), the total error for all cluster description feedback is computed as follows:

$$E^d = \sum_{i=1}^{m} E_i^d \tag{4.8}$$

Finally, the total error (or the objective function) to be minimized is computed as:

$$E = E^x + E^a + E^d \tag{4.9}$$

In each interactive iteration, after receiving the user feedback, the algorithm repeats, until convergence, the re-estimation of cluster centres and the re-assignment of points to clusters. The objective is to minimize the total error in Equation (4.9).

- *Cluster centre re-estimation*: In this step, the cluster centres are updated based on the current assignment of points to clusters and the current relevance of the feedback in F^a and F^d. As the feedback represents the dependencies across clusters, different cluster centres cannot be updated independently. Dubey *et al.* [26] cyclically update each cluster centre μ_j while keeping the other $k-1$ cluster centres fixed, until all cluster centres are stable. Each cluster centre μ_j is updated as follows:

$$\begin{aligned}\mu_j &= \frac{1}{|K_j|}\sum_{x_i \in K_j} x_i \\ &+ \sum_{i=1}^{l} R_i^a(K)[\delta(x_i^a, \mu_j) Map_i^a(\mu_i^a) + \sum_{\mu'} I(\mu_j, Map_i^a(\mu_i^a))\delta(x_i^a, \mu')\mu'] \\ &+ \sum_{i=1}^{m} Move(f_i^d) \end{aligned} \tag{4.10}$$

  where $I()$ is the indicator function ($I(s_1, s_2)$ equals 1 if $s_1 = s_2$, and equals 0 otherwise), $\delta(x, \mu)$ is the assignment function ($\delta(x, \mu)$ equals 1 if x is assigned to the cluster corresponding to the centre μ, and equals 0 otherwise). The first term of Equation (4.10) represents the movement of μ_j towards the centroid of the data points currently assigned to the cluster K_j. The second and the third terms of Equation (4.10) reflect the effects of the set of assignment feedback F^a in the update step. According to the second term, if the data point x_i^a specified in the feedback f_i^a is currently assigned to μ_j, we move μ_j towards the cluster centre $Map_i^a(\mu_i^a)$ which is the preferred cluster centre for x_i^a defined by the feedback f_i^a. According to the third term, if μ_j is the preferred cluster centre $Map_i^a(\mu_i^a)$ for the feedback f_i^a, we move μ_j towards the cluster centre μ' to which x_i^a is currently assigned. Finally, the fourth term of Equation (4.10) represents the movement of μ_j based on the set of cluster description feedback F^d, where $Move(f_i^d)$ represents the movement according to the feedback f_i^d (see [26] for more details).

- *Point re-assignment*: In this step, points are re-assigned to clusters based on the current cluster centres and the current relevance of assignment feedback. The cluster description feedback does not influence the re-assignment of data points. A point $x_i \in X$ is assigned to the cluster K_{j^*} which minimizes the contribution of its cluster centre μ_{j^*} to the objective function:

$$j^* = \underset{j}{\operatorname{argmin}}(||x_i - \mu_j||^2 + \sum_{i=1}^{l} R_i^a(K) I(x_i, x_i^a)(\mu_j - Map_i^a(\mu_i^a))^2) \tag{4.11}$$

  where $I()$ is the indicator function.

The cluster-level semi-supervised clustering is a partitioning method. It does not give any hierarchical structure. This method is interactive. By minimizing the objective function, including the penalty cost for the violation of the constraint corresponding to each feedback, the violation of the constraints corresponding to the set of feedback is minimized during the re-clustering process. However, the satisfaction of all the constraints is not ensured after the re-clustering phase.

Clustering based on relevance feedback [64] Kinoshita *et al.* present in their work the clustering of video packets based on relevance feedback. Relevance feedback approach is usually used in CBIR that allows the user to indicate the relevance of the retrieved images for a particular query. For the clustering problem, relevance feedback is used for specifying if an image is relevant or non-relevant to the cluster it belongs to. Therefore, in that case the relevance feedback can be seen as a class label information incrementally added on some objects, at each interactive iteration.

In [64], the k-means algorithm is used for clustering a set of video packets represented by their feature vectors. In each interactive iteration, the user gives feedback specifying whether each video packet is relevant or not to the cluster it belongs to. In order to satisfy the user feedback, the system updates the feature vectors of all video packets based on the feedback and then re-clusters the set of updated feature vectors by the k-means algorithm. Note that the satisfaction of the user feedback is not guaranteed after the re-clustering process.

The feature vectors in each cluster K_j are updated based on the feedback as follows. For the set of feature vectors which are marked as relevant to the cluster K_j, the system considers that the l^{th} feature is not important and gives it a small weight w_l if the standard deviation of this feature is large in K_j, and vice versa a large weight w_l is given to the feature whose standard deviation is small in K_j. The weight w_l is defined as $1/\sigma_l$, where σ_l is the standard deviation of the l^{th} feature computed based on all $x_i \in K_j$ that user considers as relevant. The system normalizes the weights $W_l = w_l / \sum_{m=1}^{d} w_m$ and updates each feature vector $x_i = \{x_{i_1}, ..., x_{i_d}\}$ of cluster K_j as $x_i = \{W_1 x_{i_1}, ..., W_d x_{i_d}\}$, where d is the number of dimensions of the feature vectors. The updated feature vectors are then used in the re-clustering phase using the k-means algorithm.

The interactive clustering based on relevance feedback proposed by Kinoshita et al. [64] originally uses the k-means partitioning method because of its simplicity. As the feature vectors are updated after each interactive iteration for being used in the next re-clustering phase, any other clustering method (partitioning, hierarchical, grid-based or density-based methods) can also be used instead of k-means. This method allows the violation of the feedback. However, as all feature vectors have to be updated in each iteration, it may not compatible for performing with large database. Moreover, as the clusters change between different interactive iterations, the feedback given in previous iterations may be not compatible to be used in the current iteration. Furthermore, only relevant objects are used for updating the feature vectors, whereas non-relevant objects are omitted.

Rocchio formula [113] based clustering The Rocchio formula [113] is the most popular technique for implementing relevance feedback in information retrieval systems. The idea is that the user specifies, among the retrieved documents of each search query, the documents which are relevant or non-relevant for the input query. The relevance feedback information is then used for updating the search query, in order to move it closer to the centroid of relevant documents and farther away from the centroid of non-relevant documents in the feature space. The search query

vector is updated as follows:

$$q_m = \alpha q_o + \frac{\beta}{|S_r|} \sum_{x_i \in S_r} x_i - \frac{\gamma}{|S_{nr}|} \sum_{x_i \in S_{nr}} x_i \tag{4.12}$$

where q_m is the modified query vector, q_o is the original query vector, S_r and S_{nr} are the sets of feature vectors of respectively relevant and non-relevant documents specified in the feedback. α, β and γ are the weights corresponding to respectively the original query, relevant feedback and non-relevant feedback. Typically $\alpha = 1$ and $0 < \beta, \gamma < 1$. In general, the weight of relevant feedback is usually greater than the weight of non-relevant feedback ($\beta > \gamma$).

Originally, the Rocchio formula is used for implementing relevance feedback in information retrieval systems. But we can easily extend it for incorporating relevance feedback in clustering systems. To the best of our knowledge, such a method has not been implemented yet. We propose to apply the Rocchio formula in the interactive clustering as follows. The user can specify, in each interactive iteration, whether each document is relevant or not to the cluster it belongs to. Then, the centre μ_j of each cluster K_j can be altered by incorporating the feedback information as in Equation (4.13):

$$\mu_j = \frac{\alpha}{|K_j|} \sum_{x_i \in K_j} x_i + \frac{\beta}{|S_{n_j}|} \sum_{x_i \in S_{n_j}} x_i - \frac{\gamma}{|S_{nr_j}|} \sum_{x_i \in S_{nr_j}} x_i \tag{4.13}$$

where S_{n_j} and S_{nr_j} are respectively the set of relevant and non-relevant objects of cluster K_j. Then, updated centres can be used as initial centres for clustering using k-means in further interactive iteration.

The Rocchio formula based clustering is a partitioning method, it does not give any hierarchical structure of cluster. It is interactive and uses not only relevant but also non-relevant objects for updating the cluster centres. Moreover, the violation of the constraints is allowed. However, supervised information is not used during the re-clustering of the data set, but only used for initializing the cluster centres for the next re-clustering step.

4.3.2 Semi-supervised clustering with pairwise constraints

Pairwise constraint supervised information does not exactly specify the cluster to which each object should belong, but specifies whether two objects should be in the same cluster (must-link) or in different clusters (cannot-link). Among the semi-supervised clustering methods using pairwise constraints between objects, we can cite COP-kmeans (constrained-kmeans) [136], constrained complete-link [65], constrained agglomerative clustering [25], COP-COBWEB [135], HMRF-kmeans (Hidden Markov Random Fields Kmeans) [10], semi-supervised kernel-kmeans [69], etc. According to our knowledge, all semi-supervised clustering methods using pairwise constraints which are published to date are traditional methods, in which pairwise constraints are provided as prior knowledge at the beginning of the clustering. Therefore, all methods presented in this section are traditional methods (*i.e.* not interactive).

Supervised information for these methods are:

- ML: the set of all must-link constraints. A must-link constraint (sometimes referred to as must-link) $(x_i, x_j) \in ML$ specifies that x_i and x_j should be in a same cluster.

- CL: the set of all cannot-link constraints. A cannot-link constraint (sometimes referred to as cannot-link) $(x_i, x_j) \in CL$ specifies that x_i and x_j should be in different clusters.

Constrained Complete-Link [65] is a traditional hierarchical semi-supervised clustering method. It is a variant of the complete-link Agglomerative Hierarchical Clustering (AHC) [70] which uses pairwise constraints as supervised information. Similar to the AHC clustering, the constrained complete-link is an ascendant clustering method. It starts by assigning each object to a separate cluster. Then, it merges at each iteration the two closest clusters until it remains only one cluster containing all data objects. Therefore, the closest clusters are merged earlier, whereas distant clusters are merged later. Note that the constrained complete-link [65] computes the distance between two clusters K_i and K_j as the maximum distance between an object in cluster K_i and another object in cluster K_j. In order to ensure the must-link entries are merged before the cannot-link entries, Klein *et al.* [65] set the distance between entries of each must-link in the proximity matrix to 0, and the distance between entries of each cannot-link to $(max_{x_i \in X, x_j \in X} D(x_i, x_j) + 1)$. As the distance between must-link entries are the minimum distance and the distance between cannot-link entries are the maximum distance in the proximity matrix, must-link entries will be merged first while cannot-link entries will be merged last when applying the complete-link clustering on the updated proximity matrix. Similar to the AHC clustering, the dendrogram created by the constrained complete-link can be cut either at the cutting point producing k clusters when k is predefined, or at a prespecified distance, or at the position where the gap between two successive distances for merging clusters is largest.

The constrained complete-link produces a hierarchical structure of clusters. It allows the violation of the constraints, but it ensures that cannot-link objects are the last ones which are merged. In the interactive context where pairwise constraints are progressively provided by the user, the user constraints of different iterations can be in contradiction, especially in the case of large image database. For example, the user gives at iteration i a cannot-link (A, B) where A and B are respectively an image of a dog and an image of a cat; then at iteration $j > i$ when seeing that there are many images containing both dogs and cats, he decides to give must-links between all images containing a dog or a cat, including a must-link (A, B), for specifying a cluster of domestic animals. In this case, the must-link (A, B) given in iteration j is in contradiction with the cannot-link (A, B) given in iteration i. And therefore, the distance matrix cannot be computed because the distance between A and B cannot be determined. Therefore, the constrained complete-link is not compatible with the interactive context of large image database.

Constrained Agglomerative Clustering [25] A constrained agglomerative clustering which uses must-link and cannot-link constraints as supervised information is presented in [25]. The transitive closure of the must-link constraints is

used first for resulting in r connected components M_1, M_2, ..., M_r. For each connected component M_j, each point $x_i \in M_j$ is connected by must-link with at least another point $x_l \in M_j$. The constrained agglomerative clustering constructs then an initial clustering consisting of r clusters corresponding to the r connected components M_1,...,M_r and a singleton cluster for each point which is not included in any connected component. Then, it merges at each iteration the two closest "mergeable clusters", until there is no more pair of clusters which can be merged without violating any cannot-link constraint. Note that the term "mergeable clusters" denotes the clusters that can be merged without violating any cannot-link constraint. We can see that by grouping all points in a connected component into a single initial cluster, must-link constraints are not violated. The algorithm of the constrained agglomerative clustering is as follows:

1. Construct the r connected components M_1, M_2, ..., M_r by using the transitive closure of the must-link constraints given as prior knowledge by the user.

2. If there are both must-link and cannot-link constraints between any two points x_i, x_j then return "No Solution".

3. Let $X_1 = X - (\cup_{i=1}^{r} M_i)$. Let $k_{max} = r + |X_1|$.

4. Construct k_{max} initial clusters consisting of the r clusters corresponding to r connected components M_1,...,M_r and a singleton cluster for each point in X_1.

5. While (there exists a pair of "mergeable clusters")

   (a) Select the two closest "mergeable clusters" K_l and K_m.

   (b) Merge K_l and K_m.

Note that the number of clusters k does not need to be fixed *a priori*, but we can cut the dendrogram at any level for having clusters as described for the constrained complete-link, page 109.

The constrained agglomerative clustering is an hierarchical method organizing points into an hierarchical structure of clusters. It is a traditional semi-supervised clustering where supervised information is provided in the form of prior knowledge. The constrained agglomerative clustering is not compatible with the interactive context of large image database where the user constraints provided in different interactive iterations can be in contradiction as this method does not allow the violation of the constraints.

COP-kmeans [136] The COP-kmeans semi-supervised clustering method is a variant of the k-means algorithms which uses supervised information in the form of pairwise constraints between objects. In COP-kmeans, points are assigned to clusters without violating any constraint. A point x_i is assigned to the cluster K_j corresponding to the closest cluster centre μ_j, unless a constraint is violated. For instance, if (x_i, x_l) is a must-link constraint and x_l is already placed in another cluster $K(x_l) \neq K_j$, then x_i cannot be placed in its closest cluster K_j. Or if (x_i, x_l) is a cannot-link constraint and x_l is already placed in the closest cluster K_j of x_i, then x_i cannot be placed in K_j any more. If x_i cannot be placed in the nearest

cluster K_j, we continue attempting to assign x_i to the next cluster in the sorted list of clusters by ascending order of distances with x_i until a suitable cluster K_h, such that x_i can be assigned to K_h without violating any constraint, is found. The clustering fails if no solution respecting the constraints is found. The algorithm of the COP-kmeans is as follows:

1. Let μ_1, μ_2, ..., μ_k be the initial cluster centres randomly chosen from the objects of the data set.

2. For each point $x_i \in X$, assign x_i to the cluster corresponding to the closest centre μ_j such that no constraint is violated. If no such cluster is found, then the clustering fails, return NULL.

3. Re-estimate the mean of each cluster K_j using the re-estimation equation (4.3) of the seeded-kmeans, page 102.

4. Iterate steps (2) and (3) until convergence.

The COP-kmeans is a partitioning method, it does not provide any hierarchical structure of clusters. As a traditional semi-supervised clustering, supervised information is provided in the form of prior knowledge. The constraint violation is not allowed, therefore COP-kmeans is not compatible with the interactive context of large image database where pairwise constraints provided in different interactive iterations can be in contradiction.

COP-COBWEB [135] COP-COBWEB [135] is a constrained partitioning version of the hierarchical clustering COBWEB [29] which was designed for categorical attributes (*i.e.* the values for each data feature are discrete). As COBWEB, it employs the concept of Category Utility (CU) [37] for evaluating the quality of a partition. Assuming $x_i = \{x_{i_1}, ..., x_{i_d}\}$ is an object in the data set, and x_{i_l} $(l = 1, ..., d)$ can take a discrete value V_{l_m} from the set V_l of the discrete values for the l^{th} feature, the Category Utility of a partition $K = \{K_1, ..., K_k\}$ is measured as:

$$CU(K) = \frac{1}{k}((\sum_{j=1}^{k} P(K_j) \sum_{l=1}^{d} \sum_{m=1}^{|V_l|} P(x_{i_l} = V_{l_m}|K_j)^2) - \sum_{l=1}^{d} \sum_{m=1}^{|V_l|} P(x_{i_l} = V_{l_m})^2) \quad (4.14)$$

where $|V_l|$ is the cardinality of the set V_l. The term $P(K_j)$ denotes the probability that an object x_i belongs to the cluster K_j, the term $P(x_{i_l} = V_{l_m})$ designates the probability that the l^{th} feature takes on value V_{l_m}, and the term $P(x_{i_l} = V_{l_m}|K_j)$ denotes the probability that the l^{th} feature takes the value V_{l_m} given that the object x_i belongs to the cluster K_j.

For each point x_i in the data set, COP-COBWEB considers all the ways to incorporate x_i into the existing partition K by considering five operators (**Must-link check**, **New**, **Add**, **Merge**, **Split**) presented in the following pseudo code. Then, the best partition among the partitions created by the five previous operators is kept for x_i. It first checks the must-link constraints. If there exists a must-link between x_i and another point x_l which is already in a cluster K_j of the partition, the constraint is enforced by adding x_i in K_j for creating the new partition K_{must} (step 2(a)). If not, another operator (**New**, **Add** or **Merge**) is applied to determine

the place of x_i. For the operator **New**, a new singleton cluster K^* is created for containing x_i (partition K_{new}, step 2(b)). Cannot-links are checked during the **Add** or **Merge** steps. x_i can be added in an existing cluster K_j for creating the partition K_{add_j} if no cannot-link is violated (step 2(c)). Similarly, the partition K_{merge} is created by merging the two best clusters for x_i if there is no cannot-link between points of these two clusters (step 2(d)). The split operator, which recurses the COP-COBWEB on the best host cluster for x_i, is applied for creating the new partition K_{split} (step 2(e)). Finally, the partition with the highest CU, among the partitions K_{must}, K_{new}, K_{add_j}, K_{merge} and K_{split}, is selected as the new partition K.

The pseudo code of the COP-COBWEB algorithm is as follows:

COP-COBWEB(data set X, must-link set ML, cannot-link set CL)

1. Let K be the set of clusters, initially $K = \{\}$ and $k = 0$.
2. For each point $x_i \in X$, consider all ways to incorporate x_i:
   (a) **Must-link check**: If there exists some must-link constraints $(x_i, x_l) \in ML$ such that x_l is already in a cluster $K_j \in K$, then add x_i to K_j and skip to (e). $K_{must} = (K - K_j) \cup \{K_j \cup \{x_i\}\}$.
   (b) **New**: Let $K_{new} = K \cup \{K^*\}$ where $K^* = \{x_i\}$ is a new cluster.
   (c) **Add**: For each cluster $K_j \in K$, create a new partition $K_{add_j} = (K - K_j) \cup \{K_j \cup \{x_i\}\}$ unless there exist a cannot-link $(x_i, x_l) \in CL$ where $x_l \in K_j$.
   (d) **Merge**: If there exist at least two clusters, let K_{max1} and K_{max2} be the two best hosts for x_i from step (c) as determined by the CU values of their resulting partitions. Merge these two clusters $K_{merge} = (K - K_{max1} - K_{max2}) \cup \{K_{max1} \cup K_{max2} \cup \{x_i\}\}$ unless there exist a cannot-link $(x_l, x_m) \in CL$ where $x_l \in K_{max1}$ and $x_m \in K_{max2}$.
   (e) **Split**: If there exist at least two clusters, let K_{max} be the best host for x_i as determined by the CU values. Let $K_{split} = (K - K_{max}) \cup COP - COBWEB(K_{max} \cup \{x_i\}, ML, CL)$.
   (f) Let $m = \arg\max_l CU(K_l)$ for $l \in \{must, new, add_j, merge, split\}$.
   Update $K = K_m$
   If $m = new$, then $k = k + 1$
   else if $m = split$, then k=*the number of clusters in* K_m.
3. Return K.

The COP-COBWEB is a partitioning method, it does not produce any hierarchical structure of clusters. As a traditional semi-supervised clustering, supervised

information is provided as prior knowledge. As all constraints have to be satisfied when using the COP-COBWEB clustering, this method is not compatible to be used in the interactive context of large image database where pairwise constraints are progressively provided by the user and the constraints of different interactive iterations can be in contradiction.

CVQE [24] The Constrained Vector Quantization Error (CVQE) algorithm is a semi-supervised clustering technique which penalizes the pairwise constraint violation using distance. The objective function to be minimized is as follows:

$$\begin{aligned} J_{obj_{CVQE}} &= \frac{1}{2}\sum_{x_i \in X} D(x_i, \mu(x_i))^2 \\ &+ \frac{1}{2}\sum_{(x_i,x_j)\in ML, K(x_i)\neq K(x_j)} D(\mu(x_i), \mu(x_j))^2 \\ &+ \frac{1}{2}\sum_{(x_i,x_j)\in CL, K(x_i)=K(x_j)} D(\mu(x_i), \mu_{h(\mu(x_i))})^2 \end{aligned} \tag{4.15}$$

where $h(\mu(x_i))$ refers to the label of the cluster whose centre is the nearest to the centre of the cluster $\mu(x_i)$. The first term in Equation (4.15) is the distance between points and the corresponding cluster centres, the second term represents the penalty of must-link violation, while the third term represents the penalty of cannot-link violation. The penalty for violating a must-link constraint $(x_i, x_j) \in ML$ is the distance between the two corresponding cluster centres $\mu(x_i)$ and $\mu(x_j)$. The penalty for violating a cannot-link constraint $(x_i, x_j) \in CL$ is the distance between the cluster centre $\mu(x_i)$ to which these two points are assigned and the nearest cluster centre $\mu_{h(\mu(x_i))}$ to which the point x_i or x_j could be moved in order to satisfy the cannot-link constraint $(x_i, x_j) \in CL$.

The CVQE algorithm repeats until convergence the re-assignment step of objects to clusters and the re-estimation step of cluster centres.

- The re-assignment step assigns objects to clusters so as to minimize the objective function in Equation (4.15) as follows:
  - For each object x_i which is not part of any constraint, we assign x_i to the closest cluster K_{j^*}:

$$j^* = \underset{j}{\operatorname{argmin}}\, D(x_i, \mu_j)^2 \tag{4.16}$$

  - For each must-link constraint $(x_i, x_j) \in ML$, all possible combinations of cluster assignments for the two objects x_i and x_j are considered, and x_i and x_j are assigned to respectively the clusters μ_{l^*} and μ_{m^*} which minimize the increase of the objective function:

$$l^*, m^* = \underset{l,m}{\operatorname{argmin}}\, D(x_i, \mu_l)^2 + D(x_j, \mu_m)^2 + I(l \neq m) * D(\mu_l, \mu_m)^2 \tag{4.17}$$

    where $I()$ is the indicator function ($I(true) = 1$, $I(false) = 0$).

- For each cannot-link constraint $(x_i, x_j) \in CL$, this method verifies all possible combinations of cluster assignments for the two objects x_i and x_j, and assigns x_i and x_j to respectively the clusters μ_{l^*} and μ_{m^*} which minimize the increase of the objective function:

$$l^*, m^* = \underset{l,m}{\operatorname{argmin}}\ D(x_i, \mu_l)^2 + D(x_j, \mu_m)^2 + I(l = m) * D(\mu_l, \mu_{h(\mu_l)})^2 \quad (4.18)$$

  where $I()$ is the indicator function.

- The re-estimation step updates the cluster centres so as to minimize the objective function. Each cluster centre μ_j is updated as in Equation (4.19).

$$\mu_j = \frac{\sum_{x_i \in K_j} [x_i + \sum_{(x_i,x_l) \in ML, K(x_i) \neq K(x_l)} \mu(x_l) + \sum_{(x_i,x_l) \in CL, K(x_i) = K(x_l)} \mu_{h(\mu(x_i))}]}{|K_j| + \sum_{x_i \in K_j, (x_i,x_l) \in ML, K(x_i) \neq K(x_l)} 1 + \sum_{x_i \in K_j, (x_i,x_l) \in CL, K(x_i) = K(x_l)} 1} \quad (4.19)$$

The CVQE algorithm is a partitioning method, it does not produce any hierarchical structure of clusters. By allowing the violation of the constraints with the corresponding penalty costs, this method is compatible with the interactive context where constraints of different iterations can be in contradiction. However, for each pairwise constraint (x_i, x_j), we have to verify $k \times k$ possible combinations of cluster assignments for the two objects x_i and x_j. Therefore, this method is not compatible for large databases, especially when the number of clusters is high.

Semi-supervised kernel-kmeans [69] Similar to the CVQE algorithm, the semi-supervised kernel-kmeans (SS-Kernel-Kmeans) [69] allows the violation of the constraints with a violation cost. But instead of calculating the objective function in the original feature space, the SS-Kernel-Kmeans computes the objective function in a transformed space, by using a kernel function mapping ϕ as follows:

$$\begin{aligned} J_{obj_{SS-Kernel-Kmeans}} &= \sum_{x_i \in X} ||\phi(x_i) - \overline{\phi(K(x_i))}||^2 \\ &- \sum_{(x_i,x_j) \in ML, K(x_i) = K(x_j)} w_{ij} \\ &+ \sum_{(x_i,x_j) \in CL, K(x_i) = K(x_j)} \overline{w_{ij}} \end{aligned} \quad (4.20)$$

where $\phi(x_i)$ is the kernel function mapping, $K(x_i)$ refers to the cluster containing x_i, w_{ij} and $\overline{w_{ij}}$ are respectively the penalty costs for violating a must-link and a cannot-link constraint between x_i and x_j, and $\overline{\phi(K(x_i))}$ is the centre in the transformed space of the cluster containing x_i:

$$\overline{\phi(K(x_i))} = \frac{\sum_{x_j \in K(x_i)} \phi(x_j)}{|K(x_i)|} \quad (4.21)$$

In the second term of Equation (4.20), instead of adding a penalty cost for a must-link violation if the two points are in different clusters, Kulis *et al.* [69] give a reward for a must-link constraint satisfaction if the two points are in the same cluster, by subtracting the corresponding penalty term from the objective function.

We can expand the distance computation $||\phi(x_i) - \overline{\phi(K(x_i))}||^2$ as follows:

$$\phi(x_i).\phi(x_i) - \frac{2\sum_{x_j \in K(x_i)} \phi(x_i).\phi(x_j)}{|K(x_i)|} + \frac{\sum_{x_j, x_l \in K(x_i)} \phi(x_j).\phi(x_l)}{|K(x_i)|^2} \tag{4.22}$$

We can see that if the dot product $\phi(x_i).\phi(x_j)$ is known, the distance between points in the transformed space can be computed without knowing the mapping of x_i and x_j in the transformed space. Therefore, instead of explicitly defining a mapping function, only a kernel matrix $K = \{K_{ij}\}$, where $K_{ij} = \phi(x_i).\phi(x_j)$, is usually needed (the same trick is used like for kernel SVMs for instance).

The semi-supervised kernel-kmeans is a partitioning method, it produces a "flat" organization of clusters. Supervised information is provided in the form of prior knowledge. It allows the violation of the constraints, and therefore is compatible to the interactive context where pairwise constraints of different interactive iterations can be in contradiction. However, it is difficult to tune the parameters of the kernel matrix, while the selection of these parameters influences much on the clustering results.

HMRF-kmeans [10] The constraint violation is not strictly prohibited, but it is allowed with a violation cost (penalty) in Hidden Markov Random Fields Kmeans (HMRF-kmeans). The Hidden Markov Random Field (HMRF) probabilistic framework for semi-supervised pairwise constrained clustering is presented in [10]. In this framework, the set of given data points X corresponds to the set of observable variables while the unobserved cluster labels of the points are hidden variables for the HMRF. Each hidden random variable representing the cluster label of $x_i \in X$ is associated with a set of neighbours N_i containing the cluster labels of the points to which x_i is must-linked or cannot-linked. And the value of the hidden variable representing the cluster label of x_i, given the set of constraints, depends only on the cluster labels of the observable variables x_j's that are must-link or cannot-link to x_i.

HMRF-kmeans aims at partitioning the data set into k clusters so as to minimize the sum of distances between the points and the corresponding cluster centres, while minimizing the total cost of constraint violation. The objective function to be minimized in the semi-supervised HMRF-kmeans is as follows:

$$\begin{aligned} J_{obj_{HMRF-Kmeans}} &= \sum_{x_i \in X} D(x_i, \mu(x_i)) \\ &+ \sum_{(x_i,x_j) \in ML, K(x_i) \neq K(x_j)} w_{ij} f_{ML}(x_i, x_j) \\ &+ \sum_{(x_i,x_j) \in CL, K(x_i) = K(x_j)} \overline{w_{ij}} f_{CL}(x_i, x_j) \end{aligned} \tag{4.23}$$

where:

- $D()$: the distance measure, *e.g.* the Euclidean distance.
- $f_{ML}(x_i, x_j)$: the must-link penalty function measuring the penalty cost for violating the must-link constraint between x_i and x_j.

- $f_{CL}(x_i, x_j)$: the cannot-link penalty function measuring the penalty cost for violating the cannot-link constraint between x_i and x_j.

- w_{ij}: the weight corresponding to the must-link constraint (x_i, x_j). In general, w_{ij}'s are constant ($w_{ij} = w$) for all must-link constraints.

- $\overline{w_{ij}}$: the weight corresponding to the cannot-link constraint (x_i, x_j). In general, $\overline{w_{ij}}$'s are constant ($\overline{w_{ij}} = \overline{w}$) for all cannot-link constraints.

We can see that, the first term of the objective function in Equation (4.23) measures the sum of distances between the points and the corresponding cluster centres, the second term measures the total cost of must-link constraint violation while the third term measures the total cost of cannot-link constraint violation. Note that the violation cost of a pairwise constraint may be either a constant or a function of the distance between the two points specified in the pairwise constraint as follows:

$$f_{ML}(x_i, x_j) = D(x_i, x_j) \tag{4.24}$$

$$f_{CL}(x_i, x_j) = D_{max} - D(x_i, x_j) \tag{4.25}$$

where D_{max} is the maximum value of the distance measure $D(.,.)$ for the data set. We can see that, to ensure that the most "difficult" constraints are respected, higher penalties are assigned to violations of must-link constraints between points which are distant and to violations of cannot-link constraints between points which are close to each other. The term D_{max} in Equation (4.25) ensures that the cannot-link penalty remains non-negative. However, it can make the cannot-link penalty term sensitive to extreme outliers, but all cannot-link are processed in the same way, so even in the presence of extreme outliers, there would be no cannot-link constraint favoured compared to the others. The objective function in Equation (4.23) is also sensitive to outliers. We could reduce this sensitivity by using an outlier filtering technique or by replacing the term D_{max} by the maximum distance between two clusters and specifying a lower bound for the term $D_{max} - D(x_i, x_j)$. These sensitivity reduction techniques are aimed in our future works.

The algorithm of the HMRF-kmeans is as follows:

1. Initialize the k cluster centres $\{\mu_j\}_{j=1}^{k}$ based on the set of pairwise constraints as presented in the following paragraph.

2. Repeat until *convergence*:

   (a) **E-step**: Given the cluster centres $\{\mu_j\}_{j=1}^{k}$, re-assign the points $\{x_i\}_{i=1}^{N}$ to clusters $\{K_j\}_{j=1}^{k}$ so as to minimize the objective function $J_{obj_{HMRF-Kmeans}}$ (Equation (4.23)).

   (b) **M-step (A)**: Given the cluster assignments $\{K(x_i)\}_{i=1}^{N}$, re-estimate the cluster centres $\{\mu_j\}_{j=1}^{k}$ so as to minimize the objective function $J_{obj_{HMRF-Kmeans}}$.

   (c) **M-step (B)**: Re-estimate the distance measure $D(.,.)$ in order to reduce the objective function $J_{obj_{HMRF-Kmeans}}$.

HMRF-kmeans first initializes the k cluster centres based on the set of must-link constraints as follows. The transitive closure of the must-link constraints is used first for resulting in λ connected components (further referred to as neighbourhoods) so that points in a neighbourhood are connected by must-link constraints. Then, k cluster centres are initialized as follows:

- If $\lambda = k$, then k cluster centres are initialized as the $\lambda = k$ centroids of all the neighbourhoods.
- If $\lambda < k$, then λ clusters centres are initialized as the centroids of the λ neighbourhoods, and the remaining $k - \lambda$ cluster centres are randomly initialized.
- If $\lambda > k$, k neighbourhoods are selected for initializing k cluster centres as follows. The largest neighbourhood (in number of points) is chosen as the first selected component. Then, the neighbourhood with the highest average weighted distance from the selected set is chosen, until having k selected neighbourhoods. The weighted distance between two neighbourhoods corresponds to the distance between the centroids of these two neighbourhoods multiplied by the weights of the two components, where the weight of each neighbourhood is proportional to the number of points included in this component.

After the initialization step, an iterative relocation approach is applied to minimize the objective function. The iterative algorithm represents the repetition of the E-step, M-step(A) and M-step(B). The E-step (or re-assignment step) re-assigns each data point to the cluster which minimizes its contribution to the objective function $J_{obj_{HMRF-Kmeans}}$. In the M-step(A), given the cluster assignments $\{K(x_i)\}_{i=1}^{N}$ of the points $\{x_i\}_{i=1}^{N}$, the cluster centres $\{\mu_j\}_{j=1}^{k}$ are re-estimated so as to minimize the objective function for the current assignment. In the M-step(B), the distance measure $D(.,.)$ is re-estimated to reduce the objective function $J_{obj_{HMRF-Kmeans}}$. In general, the M-step(B) minimizes the objective function by modifying the parameters (the weights) of the distance measure. Therefore, this step is only needed when a parameterized distance measure is used for the clustering. The details of these steps are described in [10].

The HMRF-kmeans is a partitioning method, it does not produce any hierarchical structure of clusters. Supervised information is provided in the form of prior knowledge. By allowing the constraint violation with a corresponding penalty cost, the HMRF-kmeans can be used in the interactive context where pairwise constraints provided by the user in different interactive iterations can be in contradiction.

4.3.3 Discussion

Table 4.1 resumes some characteristics (supervised information used, constraint violation tolerated or not, adaptability to the interactive context, hierarchical structure produced or not) of different semi-supervised clustering methods presented in this chapter. We can see that most of them use supervised information in the form of prior knowledge, whereas only few methods use supervised information in the form of feedback in the interactive context to guide the clustering process. Moreover, most of them (except the clustering based on relevance feedback, the constrained

complete-link and the constrained agglomerative clustering) do not produce any hierarchical structure of clusters.

Among semi-supervised clustering methods which use class labels of some objects as supervised information, the seeded-kmeans and the constrained-kmeans are traditional methods (using prior knowledge) while the others are interactive methods (using user feedback on the clustering results). The constrained-kmeans method does not allow any constraint violation, as the data points of the seed set are maintained in their initial clusters. The seeded-kmeans allows constraint violation, but it is still very basic in the way of handling prior knowledge, as supervised information is used only for initializing the cluster centres. Both constrained-kmeans and seeded-kmeans can be used in the interactive context, but the user should have prior knowledge about the data set in order to provide the class labels for the objects.

Among the different interactive semi-supervised clustering methods with class labels, the interactive cluster-level semi-supervised clustering uses assignment feedback and cluster description feedback, while the others use relevance feedback. For incorporating the feedback in the clustering phase, the cluster-level semi-supervised clustering incorporates, in the objective function, a penalty cost for the violation of the constraint corresponding to each feedback. The objective function is then minimized during the clustering process, but the satisfactions of the constraints are not ensured. Concerning the methods using relevance feedback, the constraint violation is allowed. The clustering based on relevance feedback proposed by Kinoshita *et al.* [64] may produce a "flat" or hierarchical structure of clusters based on the used clustering method. However, by updating, in each iteration, all the feature vectors based on user feedback, it may not compatible for performing with large database. Moreover, it uses only relevant objects. Clustering based on the Rocchio formula [113] is more compatible for large databases, by updating only cluster centroids in each iteration, and it uses both relevant and non-relevant objects for guiding the clustering. However, for both clusterings using relevance feedback, the relevance feedback given in previous interactive iterations may not be compatible to be used in the current iteration, as the cluster centres change between different iterations.

All the semi-supervised clustering methods which use supervised information in the form of pairwise constraints between objects presented in this chapter are traditional semi-supervised methods, in which supervised information is provided *a priori* at the beginning of the clustering. But some methods can be compatible with an interactive context, which means that they can be modified and adapted for an interactive usage. Semi-supervised methods using pairwise constraints allow constraint violation or not. Constrained agglomerative clustering, COP-kmeans and COP-COBWEB do not allow constraint violation and therefore are not compatible with the interactive context of large image database where pairwise constraints are provided progressively by the user and the user constraints of different iterations can be in contradiction. The constrained complete-link method allows the violation of the cannot-link constraints, but it ensures that cannot-link objects are the last ones which are merged. Despite this, it is also not compatible with the interactive context of large image database as when the user constraints of different iterations are contradictory, the distance matrix could not be constructed. CVQE, HMRF-kmeans and semi-supervised kernel-kmeans assign a corresponding penalty cost for

the violation of each constraint and realize the clustering by minimizing the objective function including the sum of distances between objects and the corresponding cluster centroids as well as the penalty of constraint violation. By allowing the constraint violation, these methods are compatible with the interactive context where constraints of different iterations may be contradictory. However, CVQE is not compatible for large databases, especially when the number of clusters k is high, as we have to verify, for each pairwise constraint (x_i, x_j), $k \times k$ possible combinations of cluster assignments for the two objects x_i and x_j. Concerning the semi-supervised kernel kmeans, it is difficult to tune the parameters of the kernel matrix which influence much on the clustering results.

Table 4.1 – Semi-supervised clustering methods. Interactivity adaptability is the possibility to adapt the method (if not yet) for an interactive context.

	Methods	**Supervised information**	**Constraint violation**	**Interactivity adaptability**	**Hierarchical structure**
Class label	Seeded-kmeans	Prior knowledge	Yes	Yes	No
	Constrained-kmeans	Prior knowledge	No	Yes	No
	Interactive cluster-level semi-supervised clustering	Feedback	Yes	Yes	No
	Clustering based on relevance feedback	Feedback	Yes	Yes	Yes/No
	Rocchio formula based clustering	Feedback	Yes	Yes	No
Pairwise constraint	Constrained Complete-Link	Prior knowledge	Yes	No	Yes
	Constrained Agglomerative Clustering	Prior knowledge	No	No	Yes
	COP-kmeans	Prior knowledge	No	No	No
	COP-COBWEB	Prior knowledge	No	No	No
	CVQE	Prior knowledge	Yes	Yes	No
	Semi-supervised kernel-kmeans	Prior knowledge	Yes	Yes	No
	HMRF-kmeans	Prior knowledge	Yes	Yes	No

4.4 Experiments

In this section, we present some experimental results on the Wang image database (see page 84) of some semi-supervised clustering methods (traditional or interactive methods) implemented in the interactive context.

Among different semi-supervised clustering methods presented in the above section, we choose to implement the interactive cluster-level semi-supervised clustering method [26], the algorithm we proposed based on the Rocchio formula [113] and the HMRF-kmeans (Hidden Markov Random Fields Kmeans) method [10]. The cluster-level semi-supervised clustering method [26] is one of the semi-supervised clustering methods which use class labels between objects as supervised information. It is interactive and the feedback information is not only used for initializing the cluster centres as in the seeded-kmeans method, but also used during the re-clustering phase by including, in the objective function, the corresponding penalty costs for the violation of the constraints. Note that there are two kinds of feedback (assignment feedback and cluster description feedback) used for the interactive cluster-level semi-supervised clustering. For the cluster description feedback, the user has to provide the new feature vector for some cluster centres. It is possible in the case of document clustering where the user could assign more important weights for some text words which he thinks they better describe the cluster. But cluster description feedback is not compatible in the case of image clustering as it is difficult for the user to understand the image feature descriptors which are low-level features. The algorithm we proposed based on the Rocchio formula [113] uses relevance feedback instead of assignment feedback as for the cluster-level semi-supervised clustering method. It is more compatible for large database than the clustering based on relevance feedback proposed by Kinoshita *et al.* [64] and it uses both relevant and non-relevant feedback for updating the cluster centroids. The HMRF-kmeans [10] is the most popular method among different semi-supervised clustering methods using pairwise constraints as supervised information. Moreover, it is also compatible with the interactive context where constraints of different iterations may be contradictory by allowing the constraint violations with the corresponding penalty costs. Therefore, the traditional semi-supervised clustering HMRF-kmeans is also applied in our experiments where supervised information is provided as feedback in different interactive iterations. Note that the M-step (B) of the HMRF-kmeans (page 115) is only needed when a parameterized distance measure is used for the clustering. In our system, we use the Euclidean distance, therefore the M-step (B) is not implemented.

In our interactive context, the initial clustering is realized without any prior knowledge. At each interactive iteration, the user views the clustering results and provides feedback to the system. Then, the semi-supervised clustering is applied to re-cluster the whole dataset using the feedback provided by the user. The interactive process is repeated until the clustering result satisfies the user.

4.4.1 Interactive interface

In order to allow the user to view the clustering results and to provide feedback to the system, we implement an interactive interface as shown in Figure 4.1. The rectangle at the bottom right corner of Figure 4.1 is the principal plane representing

all clusters by their prototype images (one prototype image for each cluster). The prototype image of each cluster is the most representative image of that cluster chosen as presented in the next paragraph.

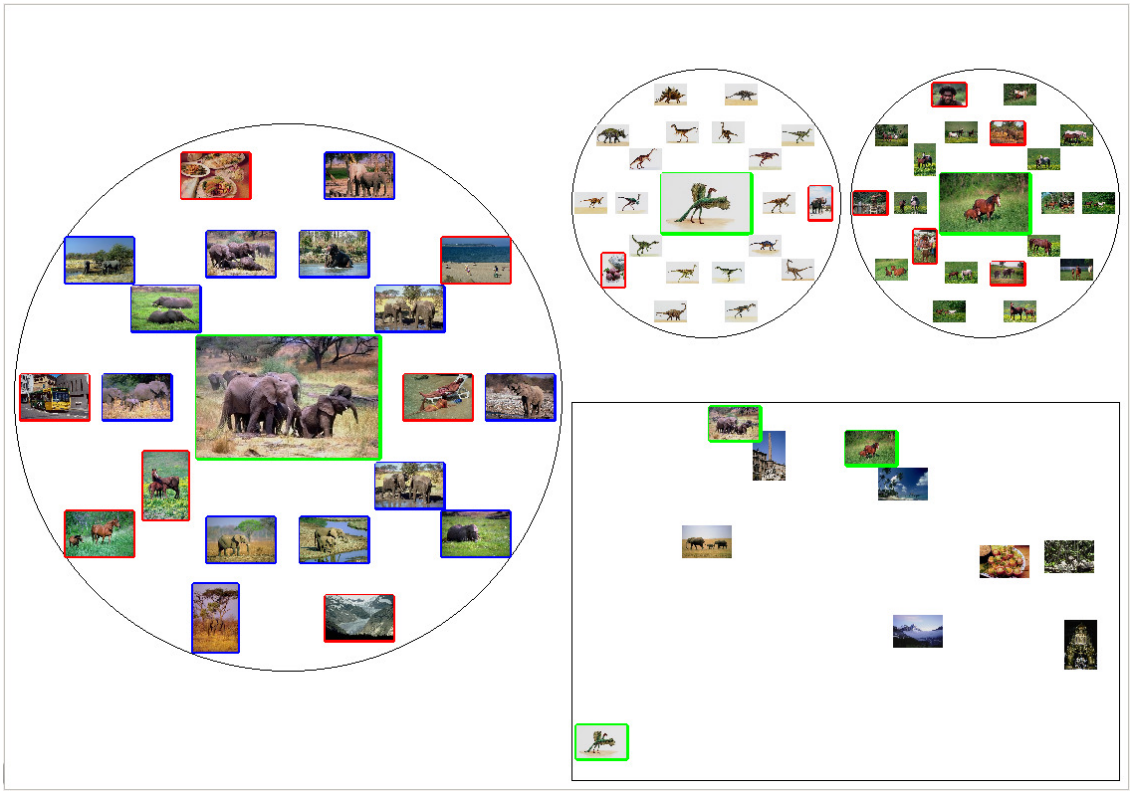


Figure 4.1 – 2D interactive interface. The rectangle at the bottom right corner represents the principal plane consisting of the two first principal axes (obtained by PCA) of the prototype images of all clusters. Each circle represents the details of a given cluster. If needed, the user can zoom in a specific cluster (see the "elephant" cluster in the left-hand part of the interface).

In our model, we use the internal measure Silhouette Width (SW) [115] to estimate the quality of each image in a cluster. The higher the SW value of an image in a cluster, the more representative this image is for the cluster. The prototype image of a cluster is thus the image with the highest SW value in the cluster. Any other internal measure could be used instead but here we choose the SW measure because, based on preliminary experiments, the SW measure was the most correlated with external measures and therefore to the user's wishes. The position of the prototype image of each cluster in the principal plane represents the position of the corresponding cluster centre. It means that, if two cluster centres are close (respectively distant) in the n-dimensional feature space, their prototype images are close (respectively distant) in the 2D principal plane.

For representing the cluster centres (which are n-dimensional feature vectors) in the 2D plane for user visualization, we use Principal Component Analysis (PCA) [101]; the principal plane consists of the two principal axes associated with the highest eigenvalues. The importance of an axis is represented by its inertia (the sum of the squared elements of this axe [2]) or by the percentage of its inertia in the total inertia of all axes. In general, if the two principal axes explain (cumulatively)

greater or equal to 80% of the total inertia, the PCA approach could lead to a nice 2D-representation of the prototype images. In our case, the accumulated inertia explained by the two first principal axes is about 65% for the Wang databases (note that the accumulated inertia does not depend on the clustering algorithm, but on the feature vectors of the images in the database). The quality of the 2D-representation is therefore acceptable to give to the user a rough idea of the pairwise distances between clusters.

By clicking on a prototype image in the principal plane, the user can view the corresponding cluster. In Figure 4.1, each cluster selected by the user is represented by a circle:

- The prototype image of this cluster is located at the centre of the circle.
- The 10 most representative images (images with the highest SW values), which have not yet received feedback from the user in the previous iterations, are located in the first circle of images around the prototype image, near the centre.
- The 10 least representative images (images with the smallest SW values), which have not received feedback from the user in the previous iterations, are located in the second circle of images around the prototype image, close to the cluster border.

By showing, for each iteration, the images which have not received user feedback in previous iterations, we wish to obtain feedback for different images. Note that the largest circle in Figure 4.1 represents the last cluster selected by the user. Therefore, if needed, the user can zoom in a specific cluster by a simple click on the corresponding cluster prototype image.

The user can specify positive feedback and negative feedback (images in Figure 4.1 with blue and red borders respectively) for each cluster. The user can also change the cluster assignment of a given image by dragging and dropping the image from the original cluster to another cluster.

4.4.2 Experiment protocol

In order to analyze the performance of the selected semi-supervised clustering methods in the interactive context, we use the Wang image database which contains 1000 images divided into 10 classes. Note that in our experiments, we use the same number of clusters as the number of classes in the ground truth. As there are only 10 classes in the Wang image database, all the cluster prototype images can be shown to the user on the principal plane.

The external measures compare the clustering results with the ground truth, thus they are compatible for estimating the quality of the interactive clustering involving user interaction. Therefore, in order to evaluate the clustering results, we use for these experiments only external measures (V-measure [114], Rand Index [109] and Fowlkes-Mallows Index [32]). The greater the values of these measures are, the better the results (compared to the ground-truth).

Concerning feature descriptors, we use the local descriptor rgSIFT [130] which gives the best results for the experiments in Chapter 3. The "Bag of words" approach [121] is used. The size 200 of the visual word dictionary, which determines

the dimension of the feature vector of each image, is chosen because it gives a good trade-off between the size of the feature vector and the performance (see Section 3.5).

In order to automatically undertake the interactive tests, we implement a software agent, later referred to as "user agent" that simulates the behaviour of the human user when interacting with the system (assuming that the agent knows all the ground truth, *i.e.* the class label for each image). At each interactive iteration, clustering results are returned to the user agent by the system; the agent simulates the behaviour of the user giving feedback to the system. For simulating the user behaviour, we suggest some rules:

- At each interactive iteration, the user agent interacts with a fixed number of c clusters.
- The user agent uses two strategies for choosing clusters for interacting in each iteration: (1) randomly choose c clusters, or (2) iteratively choose pairs of closest clusters until there are c clusters. The closest clusters are chosen by the user agent in the second strategy because when there are different clusters which are close together, the human user may want to separate these clusters or at least view them to check if some of their images should be moved from one cluster to another.
- The user agent determines the image class (in the ground truth) corresponding to each cluster by the most represented class among the 21 presented images (in the interactive interface) of the cluster. The number of images of this class in the cluster must be greater than a threshold $MinImages$. If this is not the case, this cluster can be considered as a noise cluster, but the user can still re-assign some images from this cluster to another cluster. For the experiments in this chapter, $MinImages = 5$.
- When several clusters (among chosen clusters) corresponding to a same class, the cluster in which the images of this class are the most numerous (among the 21 presented images of the cluster) is chosen as the principal cluster of this class. The classes of the other clusters are redefined as usual, but without taking into account the images from this class.
- For simulating the human user's action of giving positive and negative feedback, in each chosen cluster, all of the 21 displayed images where the result of the algorithm corresponds to the ground truth are labelled as positive samples of this cluster, while the others are negative samples of this cluster. All negative samples are moved to the cluster (among the selected clusters) corresponding to their class in the ground truth. This corresponds to the drag and drop action of the human user for changing the cluster assignment of images.

As the chosen semi-supervised clustering algorithms (the interactive cluster-level semi-supervised clustering method [26], the clustering based on the Rocchio formula [113] and the HMRF-kmeans [10]) use different kinds of supervised information for re-clustering, we have to deduce, for each method, the corresponding kind of supervised information based on the user feedback in each interactive iteration. Note that in each interactive iteration, the user (or user agent) can specify positive

and negative feedback for each cluster, he can also change the cluster assignment of some images. For the experiments in this chapter, we use some simple strategies for deducing the corresponding supervised information from the user feedback as follows:

- In our experiments, the interactive cluster-level semi-supervised clustering method uses supervised information in the form of assignment feedback (assignment of images to clusters). Therefore, in each interactive iteration, for each positive image of a cluster, we create an assignment feedback of this positive image to this cluster. Moreover, when the user moves an image from the original cluster A to another cluster B, an assignment feedback of this image to the cluster B is created. The feedback of different interactive iterations is accumulated for being used in the current iteration.

- The clustering based on the Rocchio formula uses supervised information in the form of relevant and non-relevant feedback. Therefore, in each interactive iteration, all positive and negative feedback given by the user is used as relevant and non-relevant feedback. Note that only the feedback of the current iteration is used for the re-clustering phase, the feedback of the previous iterations being ignored because the cluster centres are changed after each iteration.

- The HMRF-kmeans method uses pairwise constraints (must-link and cannot-link) between images as supervised information. Note that when the user changes the cluster assignment of an image by moving it from the original cluster A to another cluster B, the image is considered as negative feedback for cluster A and positive feedback for cluster B. Therefore, in each interactive iteration, we obtain a positive image list and a negative image list for each cluster with which the user has interacted. For the experiments in this chapter, in each iteration, must-link constraints are created between each pair of positive images of each cluster, and cannot-link constraints are created between each pair of a positive image and a negative image of each cluster. The feedback of different interactive iterations is accumulated for being used in the current iteration.

4.4.3 Scenarios

For evaluating the performance of chosen semi-supervised clustering method in the interactive context, we propose three test scenarios corresponding to different strategies used by the user agent when choosing clusters for interacting in each interactive iteration. Note that c specifies the number of clusters which are chosen for interacting in each iteration. The three scenarios are as follows:

- **Scenario 1**: $c = 5$ closest clusters are chosen.

- **Scenario 2**: $c = 5$ clusters are randomly chosen.

- **Scenario 3**: $c = 10$, all cluster are chosen (as there are 10 classes of images in the Wang image database).

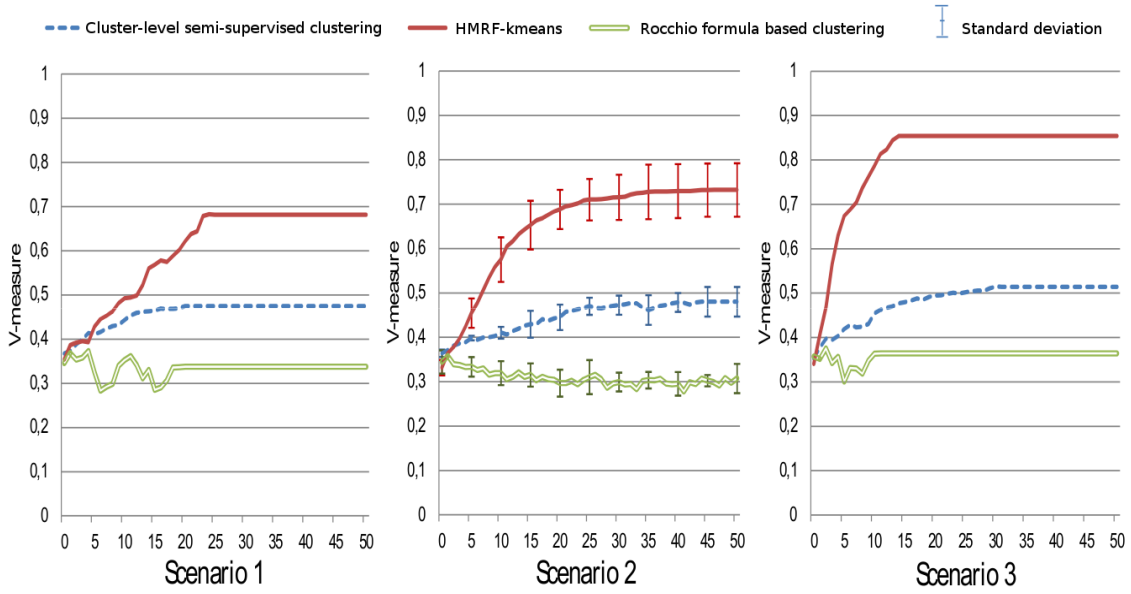


Figure 4.2 – Comparison of different semi-supervised clustering methods (cluster-level semi-supervised clustering, HMRF-kmeans, Rocchio formula based clustering) in the interactive context using the external measure *V-measure*.

4.4.4 Results and discussion

Figures 4.2, 4.3 and 4.4 compare the results during 50 interactive iterations of the three chosen semi-supervised clustering methods (cluster-level semi-supervised clustering, Rocchio formula based clustering, HMRF-kmeans) on the Wang image database by using, respectively, the V-measure, Rand Index and Fowlkes-Mallows Index external measures. The vertical axis specifies the values of an external measure (V-measure, Rand Index or Fowlkes-Mallows Index) corresponding to each figure, while the horizontal axis specifies the number of iterations. The results at iteration 0 are the results of the initial unsupervised clustering (k-means) without any supervised information. The results at iteration i ($i > 0$) are the results of the corresponding semi-supervised clustering after i interactive iterations using supervised information in the form of feedback provided by the user agent. The experiments shown in these figures are performed according to the three scenarios proposed in section 4.4.3. As in the second scenario, 5 among 10 clusters are randomly chosen for being interacted by the user agent, we realize this scenario 10 times for each semi-supervised clustering model. The curves corresponding to the scenario 2 shown in Figures 4.2, 4.3 and 4.4 represent the mean values of respectively the V-measure, the Rand Index and the Fowlkes-Mallows Index over these 10 executions at each iteration. The corresponding standard deviations from the mean values of the 10 executions are also presented by vertical lines in Figures 4.2, 4.3 and 4.4. The average standard deviation after 50 iterations corresponding to each semi-supervised clustering model is presented in Table 4.2. And the corresponding execution time for these experiments is presented in Table 4.3 (note that for the scenario 2, the average execution time of the 10 executions are shown). The experiments are executed using a normal PC with 2GB of RAM.

Note that in our system, since 21 images are displayed for each selected cluster,

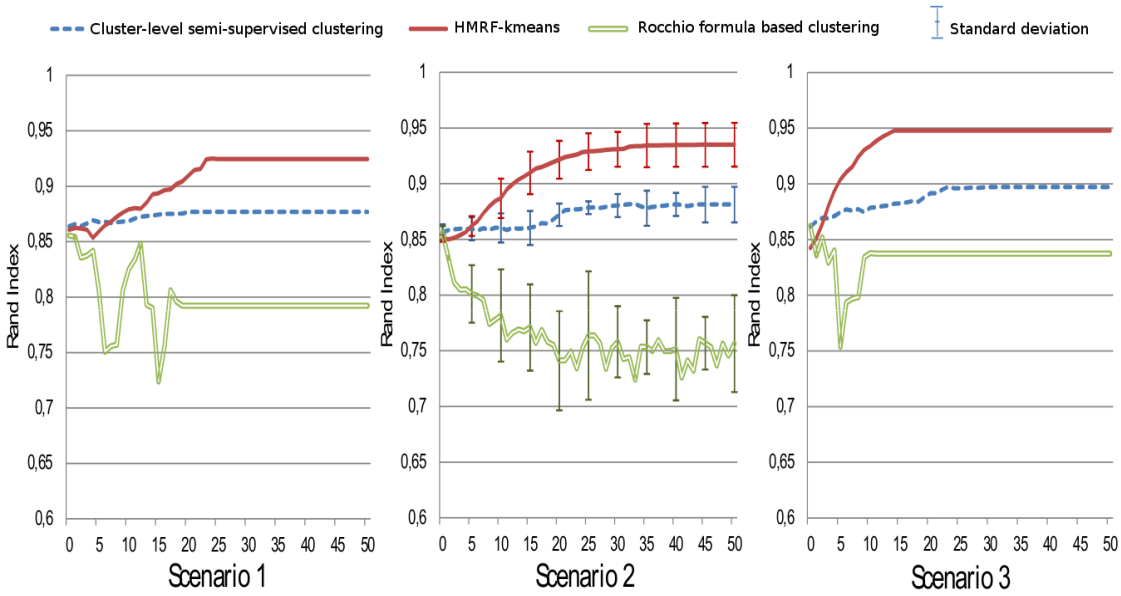


Figure 4.3 – Comparison of different semi-supervised clustering methods (cluster-level semi-supervised clustering, HMRF-kmeans, Rocchio formula based clustering) in the interactive context using the external measure *Rand Index*.

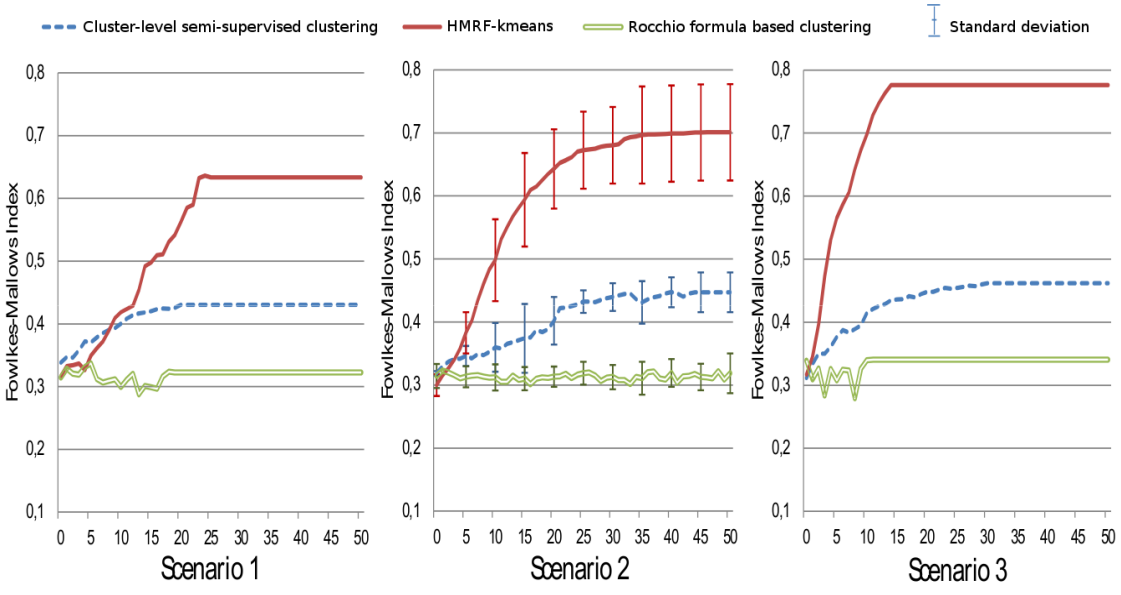


Figure 4.4 – Comparison of different semi-supervised clustering methods (cluster-level semi-supervised clustering, HMRF-kmeans, Rocchio formula based clustering) in the interactive context using the external measure *Fowlkes-Mallows Index*.

Table 4.2 – Average standard deviation of 10 executions of the scenario 2 after 50 interactive iterations corresponding to the experiments of different semi-supervised clustering method on the Wang image database shown in Figures 4.2, 4.3 and 4.4.

	Average standard deviation (Scenario 2)		
	V-measure	Rand Index	Fowlkes-Mallows Index
Cluster-level semi-supervised clustering	0.0232	0.0122	0.0283
HMRF-kmeans	0.049	0.0163	0.0623
Rocchio formula based clustering	0.0251	0.0381	0.0191

Table 4.3 – Processing time after 50 interactive iterations of the experiments of different semi-supervised clustering methods on the Wang image database shown in Figures 4.2, 4.3 and 4.4.

	Wang database		
	Scenario 1	Scenario 2	Scenario 3
Cluster-level semi-supervised clustering	39h55'	137h03'	165h04'
HMRF-kmeans	18'	12'35"	23'
Rocchio formula based clustering	9'9"	7'24"	10'30"

the maximum number of user clicks in each interactive iteration for interacting with 5 clusters (scenarios 1 and 2) or 10 clusters (scenario 3) are respectively 105 or 210. These upper bounds do not depend on the clustering methods, and in practice the number of clicks that the human user has to provided is far lower. In our experiments, with each selected cluster, the user agent gives all possible feedback (choose all possible positive and negative images of each selected cluster, move all possible negative images to the right cluster (among the c selected clusters)). Therefore, for each scenario, the numbers of user feedback (or user clicks) are comparable between different iterations and between different semi-supervised clustering methods.

We can see that the clustering comparison results evaluated by different external measures (V-measure, Rand Index and Fowlkes-Mallows Index) are similar (beware of the scale of the vertical axis which is not the same from one plot to another). The results change, in general, after each interactive iteration, as the system re-clusters the dataset by considering supervised information deduced from the new feedback provided by the user agent when interacting with the system. In most cases, the clustering results converge after some interactive iterations. This may be due to the fact that no new knowledge is provided to the system, as the supervised information deduced from the feedback of the current iteration is already deduced from the feedback of previous iterations. Moreover, we can easily see that the clustering results are better and converge more quickly when the number of chosen clusters (and therefore the number of user feedback) in each interactive iteration is higher (scenario 3 gives better results and converges more quickly than scenarios 1 and 2). In addition, scenario 2, in which clusters are randomly chosen for interacting, gives in general better results than scenario 1, in which the closest clusters are chosen. In fact, when selecting the closest clusters, there may be only a few clusters that

always receive user feedback; thus the supervised information is less "global" than when all clusters could receive user feedback (when we randomly select the clusters).

Regarding the comparison of different interactive semi-supervised clustering models, we can see that:

- The semi-supervised clustering method based on the Rocchio formula gives the worst results. The clustering results are in general not improved when having supervised information in the form of feedback provided by the user agent during different interactive iterations. And a lot of oscillations exist between different iterations. However, its processing time is low. This may be due to the fact that the Rocchio formula based clustering method uses supervised information only for updating the cluster centroids which are used as initial cluster centroids for re-clustering in the next interactive iteration. Therefore, supervised information does not have much influence on guiding the re-clustering phase and the clustering results cannot be improved.

- The interactive cluster-level semi-supervised clustering model has better performance than the Rocchio formula based clustering. The clustering results are improved after each interactive iteration and there is no oscillation. This can be explained by the fact that, supervised information has much more influence on the re-clustering phase of the cluster-level semi-supervised clustering model. Indeed, for the cluster-level semi-supervised clustering, supervised information guides the cluster prototype update step and the re-assignment step of points to clusters via the constraint violation penalty cost included in the objective function. However, the processing time of the cluster-level semi-supervised clustering model is huge. Indeed, for calculating the penalty costs of violating the constraints, the system has to find the best mapping between the set of the current cluster centroids and the set of the cluster centroids specified in each assignment feedback. Each time the cluster centroids are updated, these mappings should be re-computed. As the number of feedback accumulated during different interactive iterations is high and the cluster centroids change after each cluster update step, the processing time for calculating the mappings and therefore the processing time of the method is very high. Due to its high processing time, the cluster-level semi-supervised clustering is not compatible to be used in an interactive context.

- The semi-supervised HMRF-kmeans method gives better results than both the cluster-level semi-supervised clustering model and the Rocchio formula based clustering model. The clustering results of the HMRF-kmeans are much more improved after each interactive iteration and no oscillation occurs. In fact, with the same number of user clicks in each interactive iteration, the number of pairwise constraints created between pairs of positive/negative images which are used by the HMRF-kmeans is normally greater than the cluster assignment constraints which are used by the cluster-level semi-supervised clustering. And therefore, the same number of user clicks has more influence on the re-clustering phase of the HMRF-kmeans clustering than on the re-clustering phase of the cluster-level semi-supervised clustering. The clustering results of the HMRF-kmeans is thus better. Regarding the processing time, the HMRF-kmeans runs a little slower than the Rocchio formula based

clustering method, but much faster than the cluster-level semi-supervised clustering model. According to the processing time presented in Table 4.3, the HMRF-kmeans clustering is still compatible to be used in the interactive context (especially considering that it converges in 15-20 iterations whereas the computational times in Table 4.3 are given after 50 iterations).

Some preliminary experiments on the databases of larger size also give similar results. We can conclude, from this analysis, that among different experimented semi-supervised clustering methods, the HMRF-kmeans is the best method for being used in the interactive context, as it gives good performance in a reasonable processing time.

4.5 Discussion

This chapter presents a brief state of the art of different semi-supervised clustering methods which can help to reduce the "semantic gap" between high-level semantic concepts expressed by the user and the low-level features extracted from the images. Based on the kind of supervised information used for guiding the clustering process (see Section 4.2), we divide semi-supervised clustering methods into semi-supervised clustering with class labels (see Section 4.3.1) and semi-supervised clustering with pairwise constraints (see Section 4.3.2). Each kind of semi-supervised clustering method can be divided into traditional methods, which use supervised information in the form of prior knowledge, and interactive methods, which use supervised information in the form of feedback. It has to be noted that to the best of our knowledge, there is no interactive method based on pairwise constraints. We also analyze the possibility to use these methods in an interactive context where the user is involved during each interactive iteration for providing feedback.

According to the analysis in this chapter, we choose to compare experimentally the interactive cluster-level semi-supervised clustering method [26], the clustering based on the Rocchio formula [113] and the HMRF-kmeans method [10] in Section 4.4. In order to automatically undertake the interactive experiments and avoid the subjective dependence of the experimental results on a human user, a software agent which simulates the behaviour of a human user is used for providing feedback in each interactive iteration using ground truth. The experimental results of these methods after 50 interactive iterations show that: the Rocchio formula based clustering does not help to improve the clustering result, the cluster-level semi-supervised clustering improves the clustering results after each interactive iteration but its processing time is huge, while the HMRF-kmeans method gives the best result in a reasonable processing time. Moreover, with the same number of user clicks in each interactive iteration, the number of pairwise constraints created between pairs of images, which are used by the HMRF-kmeans, is normally greater than the number of cluster assignment constraints or relevance feedback, which are used respectively by the cluster-level semi-supervised clustering and the Rocchio formula based clustering. Thus, the same number of user clicks has more influence on the re-clustering phase of the HMRF-kmeans than on the re-clustering phase of the cluster-level semi-supervised clustering and the Rocchio formula based clustering. Therefore, the HMRF-kmeans is the most suitable to be used in our interactive context. However,

it is not based on a hierarchical method. Moreover, the pairwise constraints used for the HMRF-kmeans are between images. If similar images are grouped together and pairwise constraints are given between groups of images without reducing the supervised information quality, we could reduce the number of constraints, and therefore reduce the processing time, without decreasing the performance.

In the next chapter, we develop a new interactive semi-supervised clustering model in which the BIRCH unsupervised clustering method is used as initial clustering and a new semi-supervised clustering using pairwise constraints is used for re-clustering the data set in each interactive iteration. Instead of using pairwise constraints between images, the new semi-supervised clustering uses pairwise constraints between the leaf entries (CF entries) of the BIRCH tree. Note that each CF entry of the BIRCH tree represents a group of similar images. The integration of the pairwise constraints in the re-clustering of the BIRCH leaf entries is inspired from the HMRF-kmeans. Different strategies for deducing pairwise constraints from the cluster level feedback given by the user are proposed and experimented.

4.6 Summary of the chapter

In this chapter, a brief survey of the principal methods for semi-supervised clustering is presented. Depending on the kind of supervised information, they are divided into semi-supervised with class labels and semi-supervised with pairwise constraints. Depending on the moment when supervised information is provided, they are divided into traditional methods in which supervised information is provided *a priori* and interactive methods in which supervised information is progressively provided in the form of user feedback. The contribution of this state of the art lies in the analysis of the possibility to use different semi-supervised clustering methods in an interactive context in which the user is involved for providing feedback to the system.

The second contribution of this chapter is the proposition of a framework for implementing any semi-supervised clustering method in an interactive context as well as a 2D interactive interface allowing the user to provide some feedback to the system. We proposed to use internal measures for evaluating the quality of the images in their clusters, which helps to select the images to be displayed for each selected cluster on the interactive interface. Via the interactive interface, the user can give cluster-level feedback (*i.e.* positive and/or negative images for each selected cluster, drag and drop images from a cluster to another), without having any prior knowledge about the database.

The third contribution of this chapter is an experimental comparison of different semi-supervised clustering methods (the interactive cluster-level semi-supervised clustering, the Rocchio formula based clustering and the HMRF-kmeans), which are the most suitable for the interactive context, according to our theoretical analysis. Note that the Rocchio formula was initially proposed for updating the search query based on the relevance feedback. In our experiments, our proposed semi-supervised clustering relies on the Rocchio formula for initializing the cluster centres for the re-clustering phase, based on the current cluster centres and the positive/negative images of the clusters. Even if the HMRF-kmeans was not initially proposed for an

interactive context, in our experiments, we also adapt HMRF-kmeans for dealing with user feedback in our interactive model. In order to automatically undertake the interactive experiments and avoid the subjective dependence of the experimental results on a human user, we implemented a software agent simulating the behaviour of the human user for providing feedback in each interactive iteration. The experimental comparison shows a high performance of the HMRF-kmeans compared with the other methods.

Chapter 5

Proposed interactive semi-supervised clustering model

5.1 Introduction

In Chapter 4, we present a brief state of the art of semi-supervised clustering methods and experimentally compare some semi-supervised clustering methods (interactive cluster-level semi-supervised clustering [26], clustering based on the Rocchio formula [113] and the HMRF-kmeans [10]) in the interactive context. In these experiments, a software agent is used for simulating the behaviour of the human user when giving feedback to the system. The experimental analysis shows that the clustering result is improved with the user interaction and the HMRF-kmeans gives the best results. Moreover, the same number of user clicks gives more supervised information in the form of pairwise constraints than in the form of class labels (*i.e.* assignment constraints or relevance feedback in the interactive context).

However, the HMRF-kmeans is not based on a hierarchical method. Moreover, the pairwise constraints used for the HMRF-kmeans are between images. After a number of interactive iterations, the number of pairwise constraints could be very high, and the processing time of the re-clustering phase is thus of a high complexity. If similar images are grouped together, then pairwise constraints between images can be replaced by a less number of pairwise constraints between groups of images, without reducing the quality of supervised information. Therefore, the processing time could be reduced without decreasing the performance. In this chapter, we present our interactive semi-supervised clustering model which uses pairwise constraints between the leaf entries (CF entries) of the BIRCH tree as supervised information for guiding the clustering process. Note that each CF entry groups a set of similar objects. The integration of the pairwise constraints in the re-organization of the CF entries is inspired from the HMRF-kmeans. Section 5.2 presents our interactive semi-supervised clustering model as well as different strategies for deducing pairwise constraints to be used according to our model. Section 5.3 presents some experiments of our proposed interactive semi-supervised clustering model using different strategies for deducing pairwise constraints and databases of different sizes. The results of our proposed method are also compared with the results of the HMRF-kmeans in Section 5.3. Finally, some conclusions and discussion are presented in Section 5.4.

5.2 A new interactive semi-supervised clustering model

5.2.1 Model

In our model, the initial clustering is carried out without any supervised information, using an unsupervised clustering method. In Chapter 3, we discuss the adequation between different unsupervised clustering methods and our application context, with our willingness to involve the user in the loop; we also experimentally compare different unsupervised clustering methods such as global k-means [76], AHC [70], R-tree [13, 42, 117], SR-tree [60] and BIRCH [145]. Our conclusion is that BIRCH is the most suitable to our context. Indeed, BIRCH is incremental, it provides a hierarchical structure of clusters and it outperforms other methods in the context of a large database. Therefore, BIRCH is chosen for the initial unsupervised clustering in our model. After the initial clustering, the user views the clustering results and provides feedback to the system. The pairwise constraints (must-link and cannot-link) between the CF entries at the leaf level of the BIRCH tree are deduced, based on the user feedback. The system then re-organizes the leaf level CF entries of the BIRCH tree by considering the constraints, using the proposed semi-supervised clustering described in Section 5.2.6. The interactive process (user providing feedback and system re-organizing the clusters) is repeated until the clustering result satisfies the user. The proposed interactive semi-supervised clustering model is decomposed into the following steps:

1. Initial clustering using BIRCH unsupervised clustering (Section 5.2.2).
2. *Repeat*:
   (a) Receive feedback from the user (Section 5.2.3).
   (b) Deduce pairwise constraints from the user feedback (Sections 5.2.4 and 5.2.5).
   (c) Re-organize the leaf level CF entries of the BIRCH tree using the proposed pairwise constraint semi-supervised clustering method (Section 5.2.6) and provide the new clustering result to the user.

   until the clustering result satisfies the user.

Note that the deduction of pairwise constraints is not only based on the user feedback, but also on the image neighbourhoods. The notion of neighbourhood and the inference of the neighbourhoods from the user feedback is presented in Section 5.2.4, whereas the deduction of pairwise constraints is presented in Section 5.2.5.

5.2.2 Initial clustering

As mentioned above, the BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [145] is used as the initial unsupervised clustering in our interactive clustering model. The details of this method are presented in Chapter 3, Section 3.2.4 (page 64). The idea of BIRCH is to scan the database and organize feature vectors in a CF-tree. Each leaf node of the CF-tree contains a list of elements $[CF_i]$, where each element $CF_i = (N_i, LS_i, SS_i)$ summarizes the information of a cluster

including N_i data points, and where LS_i and SS_i are respectively the linear sum and the square sum of the data points in this cluster ($LS_i = \sum_{j=1}^{N_i} x_j$; $SS_i = \sum_{j=1}^{N_i} x_j.x_j$). Each internal node of the tree contains a list of elements $[CF_i, child_i]$, where CF_i summarizes the information of the sub-cluster represented by its i^{th} child pointed by the pointer $child_i$.

After creating the CF-tree, we can use any clustering method (AHC, k-means, etc.) for clustering all CF entries of the leaf nodes, where each CF entry is considered as a single point for clustering. The point corresponding to each entry $CF_i = (N_i, LS_i, SS_i)$ is computed as $\{\frac{LS_{i_1}}{N_i}, ..., \frac{LS_{i_d}}{N_i}\}$, where d is the number of dimensions of the feature vector. In our work, we use k-means for clustering the leaf entries, mainly because of its simplicity and its low computational complexity; it is furthermore suitable for being used with our interactive semi-supervised clustering inspired from the HMRF-kmeans, which gives the best results in our interactive experiments in the last chapter.

In the following sections, we explain the mechanisms to interactively improve this initial clustering, starting with the user feedback integration in the next section.

5.2.3 User feedback integration

In order to allow the user to view the clustering results and allow him/her to provide feedback to the system, we use the same interactive interface implemented for our experiments of different semi-supervised clustering methods presented in Chapter 4 (see Section 4.4.1 for more details about the interactive interface).

Using this interactive interface, all clusters are represented in the principal plane by their prototype images, where the prototype image of each cluster is the most representative image of the cluster (*i.e.* the image with the highest Silhouette Width (SW) value in the cluster).

By clicking on a prototype image in the principal plane, the user can view more details about the corresponding cluster (the prototype image, the 10 most representative images and the 10 least representative images which have not received any feedback from the user in the previous iterations). The user can select up to 5 clusters for visualizing and interacting simultaneously. But the user can interact with more than 5 clusters in each interactive iteration by alternatively selecting different groups of 5 clusters for visualizing at the same time. The user can specify positive feedback and negative feedback (relevant and non-relevant images) for each cluster. The user can also change the cluster assignment of a given image by dragging and dropping it from the original cluster to the new cluster. When an image is changed from cluster A to cluster B, it is considered as negative feedback for cluster A and positive feedback for cluster B. Therefore, in each interactive iteration, the process returns a positive image list and a negative image list for each cluster with which the user has interacted.

The user feedback presented here, combined with the neighbourhoods inferred from the feedback presented in the next section, will be used in Section 5.2.5 for deducing pairwise constraints. Pairwise constraints will then be used as supervised information for the re-clustering by the semi-supervised clustering method presented in Section 5.2.6, in order to improve the initial clustering of Section 5.2.2.

5.2.4 Neighbourhood inference and CF-tree update

As mentioned above, in each interactive iteration, we receive user feedback in the form of positive and negative images for each cluster with which the user interacts. According to the user feedback, all positive images should remain in "their" cluster while negative images should move to other clusters. Therefore, for each cluster, we consider that must-links exist between each pair of its positive images, and cannot-links exist between each negative image and each positive image of this cluster. There may have cannot-link constraints between images of the same entry CF_i. There may also exist simultaneous must-link and cannot-link between images of CF_i and images of CF_j. In such cases, these CF entries should be split into purer CF entries and the CF-tree should be accordingly updated, as the clustering is realized by considering each CF entry as a feature point.

If we assume that the user feedback is coherent between different interactive iterations, our objective is to group all images in a group called *neighbourhood*, according to the willingness of the user to group them in the same cluster (during the interactive session). The *neighbourhood* term is already used in the traditional HMRF-kmeans for grouping objects which are "must-linked" based on the pairwise constraints provided as prior knowledge. In this section, we explain how to extend the neighbourhoods for our interactive context where supervised information is progressively provided during different interactive iterations, not in the form of pairwise constraints, but in the form of cluster-level feedback (*i.e.* positive images and negative images for each cluster). We also define a new term called "seed" which helps to adapt the neighbourhoods with the CF entries of the BIRCH tree. The *seed* term is later presented in this section. The notion of "neighbourhood" helps to maximize the supervised information gained from a same number of user clicks, and is used in conjunction with the user feedback in the next section for deducing pairwise constraints for the re-clustering phase.

We define:

- $Np = \{Np_i\}$ is the neighbourhood list, each neighbourhood $Np_i = \{x_j\}$ includes a list of images which should be in a same cluster according to user feedback.

- Two neighbourhoods Np_i and Np_m are called cannot-link neighbourhoods if there is at least one cannot-link between a point of Np_i and a point of Np_m. We call $CannotNp$ the cannot-link neighbourhood matrix, where $CannotNp_{im} = 1$ specifying that Np_i and Np_m are cannot-link neighbourhoods according to user feedback, and $CannotNp_{im} = 0$ otherwise.

After receiving the list of feedback in the current iteration, the list Np and the matrix $CannotNp$ are updated as follows:

1. *Update based on positive feedback*: For each cluster K_h which receives interaction from the user:

   (a) If all positive images of K_h are not included in any existing neighbourhood, a new neighbourhood including all these positive images is created.

(b) If some positive images of K_h are already included in one or multiple neighbourhoods, these neighbourhoods (in the case of multiple neighbourhoods) are merged into one single neighbourhood, called $Np(K_h)$. The other positive images of K_h which are not included in any existing neighbourhood are inserted into the neighbourhood $Np(K_h)$. The matrix $CannotNp$ is updated by deleting the lines and columns corresponding to the neighbourhoods which have merged, and inserting a new line and column for the new neighbourhood $Np(K_h)$. Note that the new line and column for $Np(K_h)$ is updated for signifying that neighbourhoods, that had cannot-link with one of the neighbourhoods which have merged, now have cannot-link with the new neighbourhood $Np(K_h)$.

2. *Update based on negative feedback*: For each image x_j of each cluster K_h which receives negative feedback from the user:

(a) If x_j is not included in any existing neighbourhood, then a new neighbourhood is created for x_j. Assume that $Np(x_j)$ is the new neighbourhood for x_j, $Np(K_h)$ is the neighbourhood including the positives images of the cluster K_h, we update the corresponding entries of the matrix $CannotNp$ to signify that $Np(x_j)$ and $Np(K_h)$ have cannot-link between them.

(b) If x_j is already included in a neighbourhood $Np(x_j)$, the corresponding entries of the matrix $CannotNp$ are updated to signify that $Np(x_j)$ has cannot-link with the neighbourhood $Np(K_h)$ including positive images of the cluster K_h.

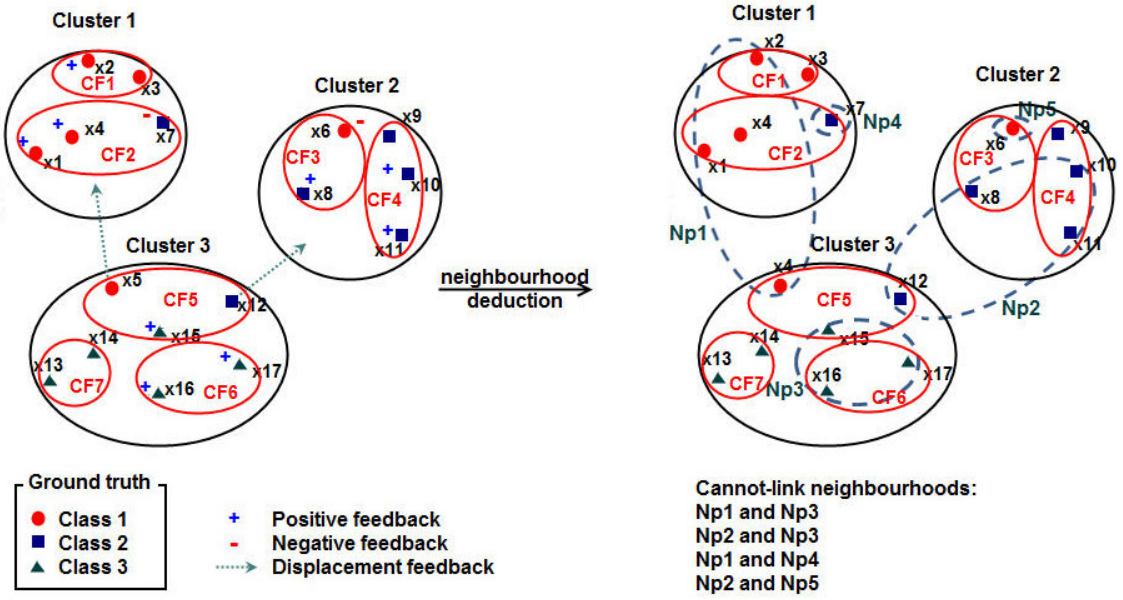


Figure 5.1 – Example of neighbourhood deduction based on user feedback (Np: neighbourhood). Left: user interaction. Right: deduced neighbourhoods and cannot-link neighbourhoods. The red ellipses represent the CF entries, the black ellipses represent the clusters, and the dashed ellipses represent the neighbourhoods.

Figure 5.1 shows an example of a neighbourhood deduction based on user feedback. User feedback is shown on the left hand-side of the figure while the resulting neighbourhoods are shown on the right hand-side of the figure. Note that our clustering organizes the CF entries into clusters by considering each CF entry as a feature point. In Figure 5.1, the CF entries are represented by the red ellipses, the clusters are represented by the black ellipses, and the neighbourhoods are represented by the dashed ellipses. We can see that cluster 1 contains two entries CF_1 and CF_2, cluster 2 contains two entries CF_3 and CF_4, while cluster 3 contains three entries CF_5, CF_6 and CF_7. While interacting with the system, the user visualizes the images in clusters and provides feedback with the images, not with the CF entries. For example, in cluster 1, the user marks x_1, x_2, x_4 as positive feedback and x_7 as negative feedback for the cluster. The user also changes the cluster assignment of x_5 from cluster 3 to cluster 1 and marks x_{15}, x_{16}, x_{17} as positive feedback of cluster 3. All positive points of cluster 1 are grouped in the neighbourhood Np_1 while all positive points of cluster 3 are grouped in the neighbourhood Np_3. As x_5 is changed from cluster 3 to cluster 1, it is considered as positive feedback of cluster 1 and negative feedback of cluster 3. Therefore, x_5 is added into the neighbourhood Np_1 corresponding to positive feedback of cluster 1. As a consequence, neighbourhood Np_1 has cannot-link with neighbourhood Np_3.

All images in a same neighbourhood should be in a same cluster and images of cannot-link neighbourhoods should be in different clusters. As mentioned above, we have to split into purer CF entries, the CF entries which include cannot-link images, or which simultaneously have must-link and cannot-link with another CF entry. For the example in Figure 5.1, three entries CF_2, CF_3 and CF_5, which include cannot-link images, should be split into purer CF entries. To do so, we define a *seed* of an entry CF_i as a subset of images of CF_i which are included in a same neighbourhood. Therefore, an entry CF_i may contain some seeds corresponding to different neighbourhoods, and additional images which are not included in any neighbourhood. Cannot-link may or may not exist between seeds of a given CF entry. For example, the points of CF_2 can be divided into two seeds. The first seed contains x_1 and x_4, which are included in the neighbourhood Np_1. The second seed contains x_7, which is included in the neighbourhood Np_4.

With each CF entry that should be split, we present the user with each pair of seeds which do not have cannot-link between them, in order to demand more information (for each seed, the image which is the closest to the centre of the seed is presented to the user):

- If the user indicates that there is must-link between these two seeds, then these two seeds and their corresponding neighbourhoods are merged. Note that the matrix $CannotNp$ should also be updated accordingly for specifying that each neighbourhood having cannot-link with one of the neighbourhoods containing these two seeds now has cannot-link with the new merged neighbourhood.

- If the user indicates that there is cannot-link between these two seeds, then the corresponding entries of the matrix $CannotNp$ are updated for specifying that the two neighbourhoods corresponding to these two seeds have cannot-link between them.

Any entry CF_i that should be split, either because it includes cannot-link images

or because it has simultaneously must-link and cannot-link with another CF entry, is split as follows:

1. For each seed of CF_i, create a new CF entry for including all points of this seed. Therefore, if CF_i has p seeds, p new CF entries are created (It has to be noted that all the seeds that could be merged were merged before).

2. Then, each point of CF_i that is not included in any seed is assigned to the new CF entry corresponding to the closest seed.

Given the inferred neighbourhoods and the user feedback from Section 5.2.3, the next section presents how to deduce pairwise constraints as supervised information for the re-clustering by the semi-supervised clustering presented in Section 5.2.6.

5.2.5 Pairwise constraints deduction strategies

As the clustering is done by considering each CF entry as a point, pairwise constraints (supervised information for the semi-supervised clustering method described in Section 5.2.6) should be deduced for pairs of CF entries at each interactive iteration, after user feedback integration, neighbourhood inference and CF-tree update.

We can see that, by splitting the necessary CF entries into purer CF entries as described above in Section 5.2.4, we can eliminate the case where cannot-link exists between images of a same CF entry or where must-link and cannot-link simultaneously exist between images of two different CF entries. Subsequently, pairwise constraints between CF entries can be deduced based on pairwise constraints between images as follows:

- If there is a "must-link" between an image of CF_i and another image of CF_j, a "must-link" is created between CF_i and CF_j.

- If there is a "cannot-link" between an image of CF_i and another image of CF_j, a "cannot-link" is created between CF_i and CF_j.

The problem now is how can we deduce pairwise constraints between images based on the user feedback of each interactive iteration, and also on the neighbourhood information. As mentioned in Section 5.2.3, user feedback is received in each interactive iteration under the form of positive and negative images of each cluster. The neighbourhood information is in the form of the list $Np = \{Np_i\}$ and the matrix $CannotNp$, where each neighbourhood Np_i contains images which should be in a same cluster and $CannotNp_{ij}$ specifies if two neighbourhoods Np_i and Np_j are cannot-link neighbourhoods or not ($CannotNp_{ij} = 1$ if Np_i and Np_j are cannot-link neighbourhoods, and $CannotNp_{ij} = 0$ otherwise). A simple and complete way to deduce pairwise constraints between images is to create must-link between each pair of images of a same neighbourhood, and to create, for each pair of cannot-link neighbourhoods (Np_i, Np_j), cannot-link between each image of Np_i and each image of Np_j. This strategy for deducing pairwise constraints between images is shown in Figure 5.2 and will be further called "Strategy 1". Note that in Figure 5.2, only cannot-link between objects of the two cannot-link neighbourhoods Np_1 and Np_3 are shown. Cannot-link between points of other pairs of cannot-link

neighbourhoods (Np_2 and Np_3, Np_1 and Np_4, Np_2 and Np_5) can be similarly deduced. We can see that by deducing pairwise constraints between images in this way, the number of constraints between images can be very high, and therefore the number of constraints between CF entries could also be very high. The processing time of the semi-supervised clustering in the next phase could thus be very high due to the high number of constraints. Therefore, in this section we propose some alternative strategies for deducing pairwise constraints between images that could reduce the number of constraints and also the processing time.

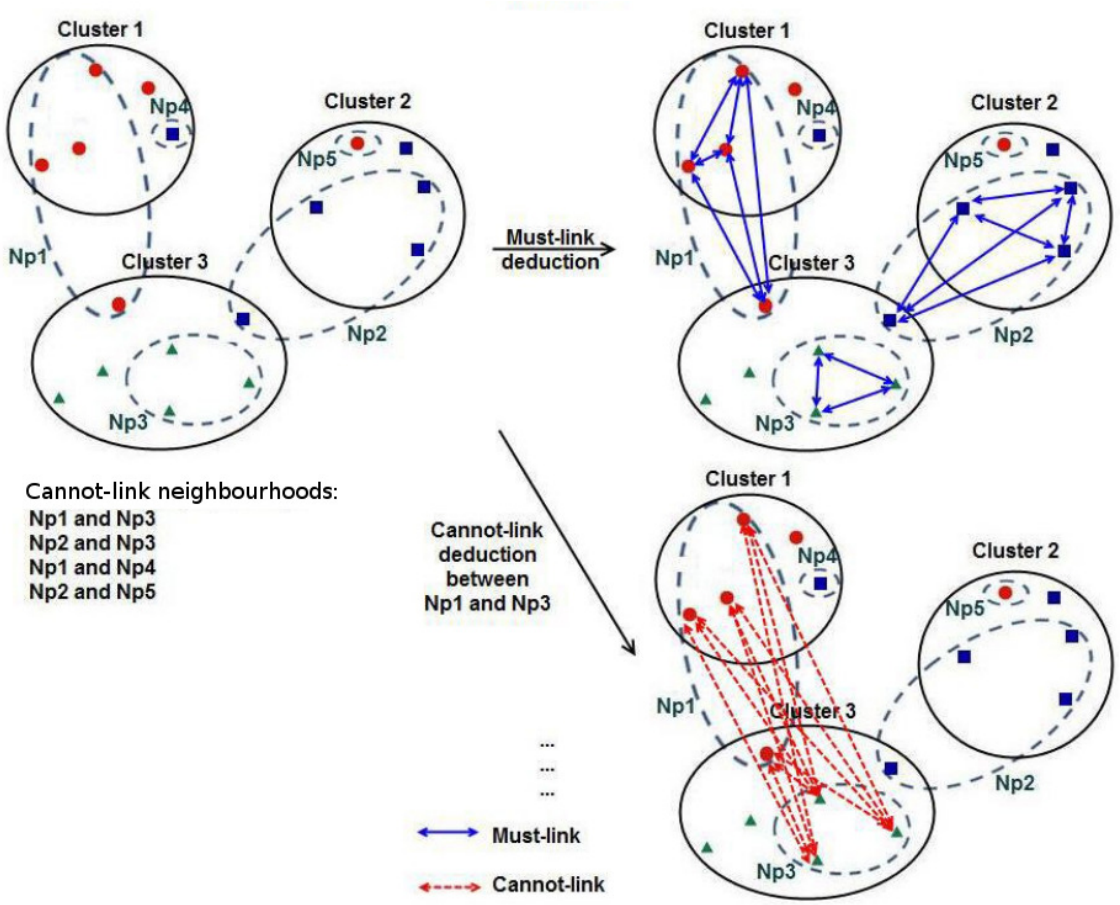


Figure 5.2 – Illustrative example of the Strategy 1 for deducing pairwise constraints between images based on the neighbourhood information deduced from the user interaction shown on Figure 5.1. Only cannot-link constraints between objects of the two cannot-link neighbourhoods $Np1$ and $Np3$ are shown in this figure. Cannot-link between points of other pairs of cannot-link neighbourhoods ($Np2$ and $Np3$, $Np1$ and $Np4$, $Np2$ and $Np5$) are deduced in a similar way.

We can divide pairwise constraints between images into two categories: *user constraints* and *deduced constraints*. User constraints are directly created, based on user feedback at each iteration, while deduced constraints are created by deduction rules. For instance, in the first iteration, the user marks x_1, x_2 as positive images and x_3 as a negative image of cluster K_i while in the second iteration, he marks x_1 and x_4 as positive images of cluster K_j. The created user constraints are: must-link between positive images in the first iteration $(x_1, x_2) \in ML$, must-link between positive images in the second iteration $(x_1, x_4) \in ML$, and cannot-link between

positive and negative images in the first iteration $(x_1, x_3) \in CL$, $(x_2, x_3) \in CL$. As there are must-links $(x_1, x_2) \in ML$ and $(x_1, x_4) \in ML$, there is also a deduced must-link $(x_2, x_4) \in ML$. In addition, deduced cannot-link $(x_3, x_4) \in CL$ is created, based on the must-link $(x_1, x_4) \in ML$ and the cannot-link $(x_1, x_3) \in CL$. We can see that deduced constraints can also be created based on neighbourhood information. Table 5.1 summaries the different strategies we experiment for deducing pairwise constraints between images. In Section 5.3, these strategies are evaluated and compared.

Table 5.1 – Different strategies for deducing pairwise constraints between images based on user feedback and on neighbourhood information.

N°	Take into account	Details
1	• ***All*** user constraints of ***all*** interactive iterations. • ***All*** deduced constraints of ***all*** interactive iterations.	All constraints are created based on the neighbourhood information: • Must-link between each pair of images of each neighbourhood $Np_i \in Np$. • Cannot-link between each image of each neighbourhood $Np_i \in Np$ and each image of each neighbourhood having cannot-link with Np_i (specified in the matrix $CannotNp$).
2	• ***All*** user constraints of ***all*** interactive iterations. • ***None*** of deduced constraints.	In each interactive iteration, all possible user constraints are created: • Must-link between each pair of positive images of each cluster. • Cannot-link between each positive image and each negative image of a same cluster.

Continued on next page

Table 5.1 – continued from previous page

N°	Take into account	Details
3	• ***All*** user constraints of ***all*** interactive iterations. • ***All*** deduced constraints of the ***current*** iteration (deduced constraints in the previous iterations are eliminated).	• In each interactive iteration, all possible user constraints are created as in Strategy 2. • Deduced constraints of the **current** interactive iteration are created while updating the neighbourhoods as follows: ◦ If there is a must-link (or cannot-link) (x_i, x_j) (user constraint of the **current** interactive iteration), $x_j \in Np_m$, deduced must-links (or cannot-links) (x_i, x_l), $\forall x_l \in Np_m$ are created. ◦ If there is a must-link (or cannot-link) (x_i, x_j) (user constraint of the **current** interactive iteration), $x_i \in Np_m$, $x_j \in Np_n$, deduced must-links (or cannot-links) (x_k, x_l), $\forall x_k \in Np_m$, $\forall x_l \in Np_n$ are created. Note that deduced constraints created in the previous iterations are omitted.

Continued on next page

Table 5.1 – continued from previous page

N°	Take into account	Details
4	• User constraints between images and cluster positive prototypes of ***all*** interactive iterations. • Deduced constraints between images and cluster positive prototypes in the ***current*** iteration (deduced constraints in the previous iterations are eliminated).	In each iteration, the positive image having the best internal measure (SW) value among all positive images of each cluster is called *positive prototype* image of this cluster. • User constraints are created in each interactive iteration as follows: ◦ Must-links are created between each positive image and the positive prototype image of the corresponding cluster. ◦ Cannot-links are created between each negative image and the positive prototype image of the corresponding cluster. • Deduced constraints in the current iteration are created while updating the neighbourhoods as follows: ◦ If there is a must-link (or cannot-link) (x_i, x_j) (user constraint of the current interactive iteration), $x_j \in Np_m$, deduced must-links (or cannot-links) are created between x_i and each positive prototype image of Np_m. ◦ If there is a must-link (or cannot-link) (x_i, x_j) (user constraint of the current interactive iteration), $x_i \in Np_m$, $x_j \in Np_n$, deduced must-links (or cannot-links) are created between x_i and each positive prototype image of Np_n and between x_j and each positive prototype image of Np_m. Note that deduced constraints created in the previous iterations are omitted.

Continued on next page

Table 5.1 – continued from previous page

N°	Take into account	Details
5	• Some selected user constraints (must-links between the most distant images and cannot-links between the closest images) of ***all*** iterations. • Some selected deduced constraints (must-links between the most distant images and cannot-links between the closest images) of ***all*** iterations.	• User constraints are created for each cluster in each iteration as follows: ◦ Must-links are successively created between two positive images (at least one of them is not selected by any must-link) that have the longest distance until all positive images of the cluster are connected by these must-links. ◦ Cannot-links are created between each negative image and the nearest positive image of the cluster. • Deduced constraints are created in each iteration as follows: ◦ Must-links for each neighbourhood are successively created between two images that have the longest distance until all images of this neighbourhood are connected by these must-links. ◦ Cannot-links are deduced, for each pair of cannot-link neighbourhoods (Np_i,Np_j), between each image of Np_i and the nearest image of Np_j and between each image of Np_j and the nearest image of Np_i.

Continued on next page

Table 5.1 – continued from previous page

N°	Take into account	Details
6	Same idea as in Strategy 5, but the sizes of the neighbourhoods are considered while creating deduced cannot-links.	• User constraints are created as in Strategy 5. • Deduced constraints are created in each iteration as follows: ◦ Deduced must-links are created as in Strategy 5. ◦ Deduced cannot-links, for each pair of cannot-link neighbourhoods, are only created between each image of the neighbourhood that has the minimum number of images and the nearest image of the neighbourhood that has the maximum number of images.

The first strategy (Figure 5.2), which is already described in the last paragraph, is the most complete one. By creating must-links between each pair of images of a same neighbourhood and cannot-links between images of neighbourhoods which have cannot-link between them, the strategy 1 covers all possible user constraints and deduced constraints that can be created based on feedback of all interactive iterations. But the number of constraints used (and therefore the execution time) may be huge when the size of the neighbourhoods is increased.

The second one is a very basic strategy, as it does not use deduced constraints, but only uses all the pairwise constraints which can be created at all interactive iterations based on the user feedback, and without taking into account the neighbourhood information. The user constraints of each interactive iteration are must-links between each pair of positive images of each cluster and cannot-links between each positive image and each negative image of a same cluster. This strategy helps to reduce the number of constraints compared with the strategy 1, but it does not use deduced constraints, which may also be important.

The third strategy is similar to the strategy 2. But along with all possible user constraints created in all interactive iterations, it also uses constraints deduced by using neighbourhood information in the current iteration (deduced constraints in the previous iterations are eliminated). Only the newest deduced constraints are kept, as they are considered as the most important deduced constraints. The number of constraints used in the strategy 3 is greater than in the strategy 2, but less than in the strategy 1.

Strategy 4 relies on the same idea of using user constraints of all interactive iterations and deduced constraints of the current iteration as Strategy 3. However, it does not create, in each interactive iteration, all possible user constraints and/or deduced constraints, but only a part of these constraints. The idea is to reduce the number of constraints used for re-clustering. In each interactive iteration, among all positive images of a cluster, this strategy considers the image having the best internal measure (SW) value as the positive prototype image of this cluster. Then, for each positive image of a cluster, instead of creating must-link user constraints with all other positive images of this cluster, only one must-link constraint between this positive image and the positive prototype image of this cluster is created. Similarly, instead of creating cannot-link user constraints between a negative image and all positive images of the same cluster, strategy 4 creates only one cannot-link between this negative image and the positive prototype image of this cluster. Regarding deduced constraints, if x_i and x_j must be in the same (or different) clusters (based on user feedback of the current iteration), deduced must-links (or cannot-links) are created between x_i and positive prototype images of the neighbourhood containing x_j, and *vice versa*, between x_j and positive prototype images of the neighbourhood containing x_i. By creating constraints in this way, we hope that the constraints between an image and the positive prototype images of a group (neighbourhood or cluster) can represent the constraints between this image and all the images of the group.

By considering that both constraints deduced in the current iteration and constraints deduced in older iterations are both important, the strategy 5 uses both user constraints and deduced constraints of all iterations, but reduces the number of constraints created in each iteration. The idea is to only keep the "difficult" con-

straints. For each cluster, must-link user constraints are not created between each pair of positive images of this cluster, but are successively created between two positive images of this cluster (at least one of them is not part of any created must-link user constraint of this cluster) that have the longest distance until all the positive images of this cluster are connected by these user must-links. Similarly, deduced must-link constraints for each neighbourhood are successively created, until all images of this neighbourhood are connected by these deduced must-links. Regarding cannot-link user constraints, for each negative image of a cluster, instead of creating multiple cannot-links between this negative image and each positive image of the cluster, only one cannot-link between this negative image and its nearest positive image is created. Similarly, for cannot-link deduced constraints, if Np_i and Np_j are cannot-link neighbourhoods, then deduced cannot-links are created between each image of Np_i and the nearest image of Np_j, and *vice versa* between each image of Np_j and the nearest image of Np_i.

Strategy 6 is similar to Strategy 5, but the number of deduced cannot-link constraints in each iteration is reduced. Indeed, if Np_i and Np_j are cannot-link neighbourhoods and $|Np_i| \leq |Np_j|$, where $|Np_i|$ is the number of images of Np_i, then deduced cannot-links are created only between each image of Np_i and the nearest image of Np_j.

Combining all the elements from this section and the previous ones, the next section is dedicated to our contribution related to semi-supervised clustering method based on pairwise constraints between CF entries.

5.2.6 Semi-supervised clustering method based on pairwise constraints between CF entries

After deducing pairwise constraints from the user feedback and the neighbourhood information, CF entries at the leaf level of the CF-tree have to be re-organized according to the wishes of the user. Our semi-supervised clustering method aims at re-organizing the set of all leaf entries $S_{CF} = (CF_1, ..., CF_m)$ of the CF-tree based on supervised information in the form of two sets of pairwise constraints between CF entries deduced from user feedback and neighbourhood information as described in Section 5.2.5. We call $ML_{CF} = \{(CF_i, CF_j)\}$ and $CL_{CF} = \{(CF_i, CF_j)\}$ the sets of respectively must-links and cannot-links constraints between CF entries. A must-link constraint $(CF_i, CF_j) \in ML_{CF}$ implies that CF_i, CF_j, and therefore all points which are included in these two entries, should belong to the same cluster, while a cannot-link constraint $(CF_i, CF_j) \in CL_{CF}$ implies that CF_i and CF_j should belong to different clusters. Inspired from the HMRF-kmeans method, our objective function to be minimized is as follows:

$$\begin{aligned} J_{obj} &= \sum_{CF_i \in S_{CF}} D(CF_i, \mu(CF_i)) \\ &+ \sum_{(CF_i,CF_j) \in ML_{CF}, K(CF_i) \neq K(CF_j)} w N_{CF_i} N_{CF_j} D(CF_i, CF_j) \\ &+ \sum_{(CF_i,CF_j) \in CL_{CF}, K(CF_i) = K(CF_j)} \overline{w} N_{CF_i} N_{CF_j} (D_{max} - D(CF_i, CF_j)) \end{aligned} \quad (5.1)$$

where:

- $K(CF_i)$ refers to the cluster containing CF_i.
- $\mu(CF_i)$ is the centre of the cluster containing CF_i.
- The first term of Equation (5.1) measures the distance between each leaf entry CF_i and its corresponding cluster centre $\mu(CF_i)$.
- The second and the third terms of Equation (5.1) represent the penalty costs for respectively violating the must-link and cannot-link constraints between CF entries. w and $\overline{w}$ are constant values specifying the violation cost of a must-link and a cannot-link between two points. As an entry CF_i represents the information of a group of N_{CF_i} points, a pairwise constraint between two entries CF_i and CF_j corresponds to $N_{CF_i} \times N_{CF_j}$ constraints between points of these two entries. The violation cost of a pairwise constraint between two entries CF_i, CF_j is thus a function of their distance $D(CF_i, CF_j)$ and of the number of points included in these two entries. D_{max} is the maximum distance between two CF entries in the set S_{CF}. Therefore, higher penalties are assigned to violations of must-link between entries that are distant and of cannot-link between entries which are close. As in HMRF-kmeans, the term D_{max} can make the cannot-link penalty term sensitive to extreme outliers. However, we could reduce this sensitivity by using an outlier filtering techniques or by replacing the term D_{max} by the maximum distance between two clusters and specifying a lower bound for the term $D_{max} - D(CF_i, CF_j)$. These sensitivity reduction techniques are considered in our future works.

In our case, we use the famous squared Euclidean distance as distortion measure. The distance between two entries $CF_i = (N_{CF_i}, LS_{CF_i}, SS_{CF_i})$, $CF_j = (N_{CF_j}, LS_{CF_j}, SS_{CF_j})$ is calculated as the distance between their means as follows:

$$D(CF_i, CF_j) = \sum_{p=1}^{d} \left(\frac{LS_{CF_i}(p)}{N_{CF_i}} - \frac{LS_{CF_j}(p)}{N_{CF_j}} \right)^2 \tag{5.2}$$

where d is the number of dimensions of the feature space.

Inspired from the HMRF-kmeans, our proposed pairwise constraint semi-supervised clustering is as follows:

Input: Set of leaf entries $S_{CF} = \{CF_i\}_{i=1}^{m}$ which are clustered into k clusters with the corresponding centres $\{\mu_h\}_{h=1}^{k}$,

set of must-link constraints $M_{CF} = \{(CF_i, CF_j)\}$

set of cannot-link constraints $C_{CF} = \{(CF_i, CF_j)\}$.

Output: New disjoint k clusters of S_{CF} such that the objective function in Equation (5.1) is locally minimized.

Method:

1. Set $t \leftarrow 0$.
2. *Repeat* until convergence:
   (a) *Re-assignment step*: Given $\{\mu_h^{(t)}\}_{h=1}^{k}$, re-compute the cluster assignments $\{K(CF_i)^{(t+1)}\}_{i=1}^{m}$ so as to minimize the objective function in Equation (5.1).

(b) *Re-estimation step*: Given the cluster assignments $\{K(CF_i)^{(t+1)}\}_{i=1}^{m}$, re-calculate the cluster centres $\{\mu_h^{(t+1)}\}_{h=1}^{k}$ to minimize the objective function in Equation (5.1).

(c) $t \leftarrow t+1$.

In the re-assignment step, given the current cluster centres $\{\mu_h^{(t)}\}_{h=1}^{k}$, each entry CF_i is re-assigned to the cluster $K_{h^*} \in K$ which minimizes the contribution of this re-assignment to the objective function in Equation (5.1). The contribution to the objective function of the re-assignment of an entry CF_i to a cluster K_h is computed as in Equation (5.3):

$$\begin{aligned} J_{obj}(CF_i, K_h) &= D(CF_i, \mu_h^{(t)}) \\ &+ \sum_{(CF_i,CF_j)\in ML_{CF}, K_h \neq K(CF_j)^{(t)}} wN_{CF_i}N_{CF_j}D(CF_i, CF_j) \\ &+ \sum_{(CF_i,CF_j)\in CL_{CF}, K_h = K(CF_j)^{(t)}} \overline{w}N_{CF_i}N_{CF_j}(D_{max} - D(CF_i, CF_j)) \end{aligned} \tag{5.3}$$

Therefore, each entry CF_i is re-assigned to the cluster K_{h^*} (*i.e.* $K(CF_i)^{(t+1)} = K_{h^*}$) such that:

$$h^* = \underset{h}{\operatorname{argmin}}\ J_{obj}(CF_i, K_h) \tag{5.4}$$

We can see that the optimal assignment of each CF entry also depends on the current assignment of the other CF entries, due to the violation cost of pairwise constraints in the second and third terms of Equation (5.3). Therefore, after all entries are re-assigned, they are randomly re-ordered, and the re-assignment process is repeated until no CF entry changes its cluster label between two successive iterations.

In the re-estimation step, given the cluster assignments $\{K(CF_i)^{(t+1)}\}_{i=1}^{m}$ of all CF entries, the cluster centres $\{\mu_h\}_{h=1}^{k}$ are re-calculated in order to minimize the objective function of the current assignment. For saving computational time, each cluster centre is also represented in the form of a CF-vector. By using the squared Euclidean measure, the CF-vector of each cluster centre μ_h is calculated based on CF entries which are assigned to this cluster as follows:

$$N_{\mu_h} = \sum_{K(CF_i)^{(t+1)}=K_h} N_{CF_i} \tag{5.5}$$

$$LS_{\mu_h} = \sum_{K(CF_i)^{(t+1)}=K_h} LS_{CF_i} \tag{5.6}$$

$$SS_{\mu_h} = \sum_{K(CF_i)^{(t+1)}=K_h} SS_{CF_i} \tag{5.7}$$

We can see that in each re-assignment step, each entry CF_i moves to a new cluster K_h if its contribution to the objective function is decreased with this re-assignment. Therefore, the objective function J_{obj} in Equation (5.1) is decreased or unchanged after the re-assignment step. In each re-estimation step, the mean of the CF-vector of each cluster centre μ_h (computed as $\frac{LS_{\mu_h}}{N_{\mu_h}}$) corresponds to the mean of the CF entries (and therefore the points) in this cluster. That minimizes the contribution of μ_h to the component $\sum_{CF_i \in S_{CF}} D(CF_i, \mu(CF_i))$ of the objective function

J_{obj} in Equation (5.1), where $K_h = K(CF_i)$. The penalty terms of the objective function J_{obj} do not depend on the cluster centres, thus they do not take part in cluster centre re-estimation. Therefore, the objective function J_{obj} of our proposed method will decrease or remain the same in the re-estimation step. Since J_{obj} is bounded below and decreases after each re-assignment and re-estimation steps, the proposed semi-supervised clustering converges to a (at least local) minimum in each interactive iteration.

After each interactive iteration, new constraints corresponding to new feedback are given to the system. These new constraints might be in contradiction with some of the ones previously deduced by the system from the earlier user interactive iterations. For this reason, and also for computational time matters, our system may at each step omit some of the constraints deduced at earlier steps. Therefore, the objective function J_{obj} in Equation (5.1) may be different between different interactive iterations. And the convergence of the interactive semi-supervised model is thus not guaranteed. But we can experimentally verify the convergence of the model by determining, at the end of all interactive iterations, the global objective function which considers all feedback given by the user in all interactive iterations and then by verifying if this global objective function has improved or not after several interactive steps.

5.3 Experiments

In this section, we present some experimental results of our interactive semi-supervised clustering model using the different strategies presented in Table 5.1 for deducing pairwise constraints between images. We also experimentally compare our semi-supervised clustering model with the semi-supervised HMRF-kmeans, which gives the best results in our experiments in Chapter 4. As in Chapter 4, when using the semi-supervised HMRF-kmeans in the re-clustering phase, the initial unsupervised clustering is k-means.

5.3.1 Experiment protocol

In order to analyze the performance of our interactive semi-supervised clustering model, we use the same image databases which are used in the experiments of the previous chapters:

- Wang (1000 images divided into 10 classes).
- PascalVoc2006 (5304 images divided into 10 classes).
- Caltech101 (9143 images divided into 101 classes).
- Corel30k (31695 images divided into 320 classes).

Note that in our experiments we use the same number of clusters as the number of classes in the ground truth. Concerning feature descriptors, the local descriptor rgSIFT with the size 200 of the visual word dictionary is also used in this chapter. As presented in Chapter 4, Section 4.4.1, the cluster prototype images (one image for

each cluster) are shown to the user on the principal plane. Users can choose to view and interact with any cluster in which they are interested. For databases which have a small number of classes, such as Wang and PascalVoc2006, all prototype images can be shown on the principal plane. For databases which have a large number of classes, such as Caltech101 and Corel30k, it may be difficult for the user to select the good clusters to interact with, if too many cluster prototypes are shown on the principal plane. Therefore, in the case of databases having a large number of classes, only a maximum of 30 cluster prototypes can be shown to the user on the principal plane in each interactive iteration. We use two simple strategies for choosing clusters to be shown in each iteration:

- Strategy 1: by considering that the user may be interested in any clusters, in this strategy, 30 clusters are randomly chosen to be shown on the principal plane.
- Strategy 2: by considering that the user may want to separate the clusters which are close together, in this strategy, the system iteratively chooses pairs of closest clusters until reaching 30 clusters.

As the external measures compare the clustering results with the ground truth, they are compatible for estimating the quality of the interactive clustering involving user interaction. Therefore, external measures are used for evaluating the experiments in this chapter. Experimental results in Chapter 4 show that different external measures (V-measure [114], Rand Index [109] and Fowlkes-Mallows Index [32]) evaluate the interactive clustering results in a similar way. Thus, in this chapter, only the V-measure is used for evaluating the experiments. The greater the V-measure values are, the better the results (compared to the ground-truth).

In order to automatically undertake the interactive tests and to avoid the subjective dependence of experimental results on the human user, the software agent simulating the behaviour of the human user when interacting with the system is always used for experiments in this chapter. For every interactive iteration, the user agent always chooses to interact with a fixed number of c clusters (random clusters or nearest clusters depending on the used strategy). In order to provide the feedback to the system, the software agent should know which image class in the ground truth corresponds to each chosen cluster. Note that for each chosen cluster, only 21 images (the prototype image, the 10 most represented images and the 10 least represented images of the cluster) can be viewed by the user in each interactive iteration. The image class corresponding to each cluster is determined by the most represented class among 21 presented images of the cluster. If the number of images of the most represented class of a cluster is not greater than a threshold $MinImages$, this cluster can be considered as a noise cluster, and the user can still re-assign some images from this cluster to another cluster. In our experiments, $MinImages = 5$ for databases having a small number of classes (Wang, PascalVoc2006), and $MinImages = 2$ for databases having a large number of classes (Caltech101, Corel30k). When several clusters (among the selected clusters) correspond to a same class C_j in the ground truth, the cluster in which the images of C_j are the most numerous (among the 21 shown images of the cluster) is chosen as the principal cluster of C_j. For the other clusters also corresponding to the class C_j, another image class has to be reassigned for each of these clusters.

The new image class is redefined for each cluster as usual, but without considering the images of C_j. In our experiments, we assume that the user agent gives all possible feedback in each interactive iteration. Therefore, for each chosen cluster, the user agent labels all images, where the result of the algorithm corresponds to the ground truth, as positive samples of this cluster, and labels all the others images (among 21 presented images) as negative samples of this cluster. Moreover, the user agent moves all possible negative images to the cluster (among the c chosen clusters) corresponding to their class in the ground truth.

As presented in Section 5.2.5, there are different strategies for deducing pairwise constraints between images, which further result in pairwise constraints between CF entries, to be used as supervised information in our proposed semi-supervised clustering. All the six pairwise constraint deduction strategies presented in Section 5.2.5 are analyzed in our experiments.

5.3.2 Experimental analysis of our proposed interactive semi-supervised clustering model

The first set of experiments aims at evaluating the performance of our interactive semi-supervised clustering model. The six pairwise constraint deduction strategies presented in Section 5.2.5 are used for deducing pairwise constraints between images from the user feedback. Note that constraints between CF entries should be deduced from constraints between images (see Section 5.2.5), before being used as supervised information in the re-clustering phase using our proposed semi-supervised clustering method. We use the Wang and the PascalVoc2006 image databases for these experiments. These two databases both have 10 classes of images (and therefore 10 clusters), thus all prototype images can be shown to the user on the principal plane. For these two databases, we use three test scenarios which are already used for the experiments of different semi-supervised clustering methods in Chapter 4. Note that c specifies the number of clusters which are chosen for interacting in each interactive iteration, the three scenarios are as follows:

- Scenario 1: $c = 5$ closest clusters are chosen.
- Scenario 2: $c = 5$ clusters are randomly chosen.
- Scenario 3: $c = 10$, all clusters are chosen.

Note that our experiments are carried out automatically, *i.e.* the feedback is given by a software agent simulating the behaviours of the human user when interacting with the system, but the proposed scenarios are also feasible by a human user. In fact, the human user can give feedback by clicking for specifying the positive and/or negative images of each cluster or by dragging and dropping the image from one cluster to another cluster. For each cluster selected by the user, only 21 images of this cluster are displayed. Therefore, for interacting with 5 clusters (scenarios 1 and 2) or 10 clusters (scenario 3), the user has to respectively realize a maximum of 105 or 210 mouse clicks in each interactive iteration. **These upper bounds do not depend neither on the size of the database nor on the pairwise constraint deduction strategy, and in practice the number of clicks that**

the user has to provide is far lower. However, the number of deduced constraints may be much greater than the user clicks (and this number depends on the database size and on the pairwise constraint deduction strategy). When applying the proposed interactive semi-supervised clustering model in the indexing phase, the user is generally required to provide as much feedback as possible for having a good indexing structure which could lead to better results in the further retrieval phase. Therefore, in the case of the indexing phase, the number of clicks proposed in these scenarios seems tractable.

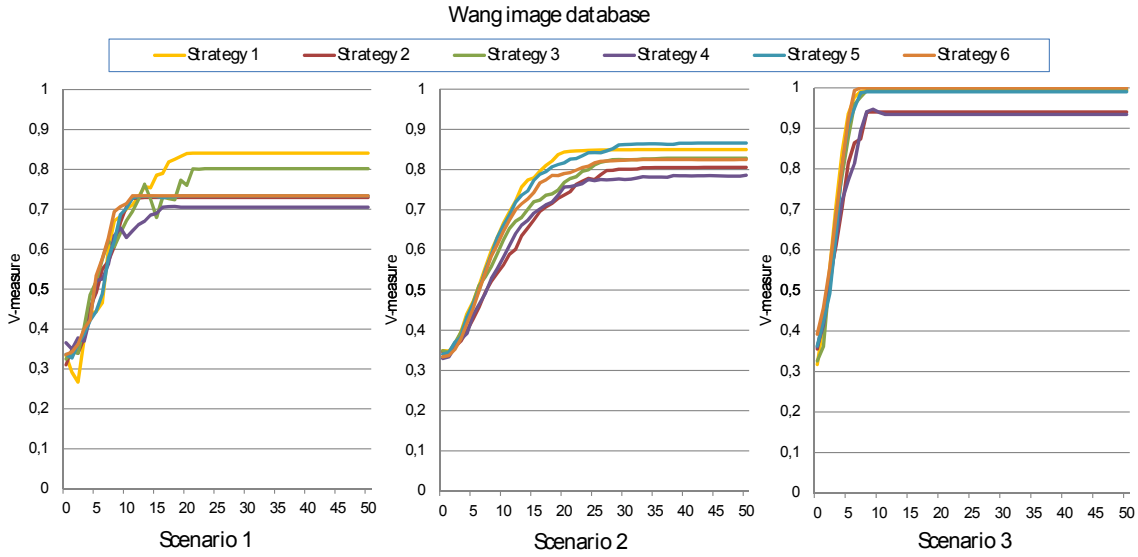


Figure 5.3 – Results of our proposed semi-supervised clustering model during 50 interactive iterations on the Wang image database, using 6 strategies for deducing pairwise constraint. The vertical axis specifies the V-measure values and the horizontal axis specifies the number of iterations.

Figures 5.3 and 5.4 show the results of our proposed interactive semi-supervised clustering model on respectively the Wang and PascalVoc2006 image databases, using 6 strategies for deducing pairwise constraints between images presented in Table 5.1. The experiment for each of the three proposed scenarios is realized in 50 interactive iterations in which the user agent is involved for providing feedback to the system. The vertical axis specifies the V-measure values, while the horizontal axis specifies the number of iterations. The results of the initial unsupervised clustering (BIRCH) without any supervised information is presented at iteration 0. The results at iteration i ($0 < i \leq 50$) represents the results of the re-clustering phase, using the proposed semi-supervised clustering method presented in Section 5.2.6, after i interactive iterations using supervised information in the form of pairwise constraints between CF entries deduced from the provided user feedback. Note that in our experiments, with each selected cluster, the user agent gives all possible feedback (chooses all possible positive and negative images of each selected cluster, moves all possible negative images to the right cluster (among the c selected clusters)). Therefore, for each scenario, the numbers of user feedback (user clicks) are similar between different interactive iterations, and between different pairwise constraint deduction strategies. As in scenario 2, the user agent randomly chooses 5 among 10 clusters for viewing and giving feedback, we realize this scenario 10

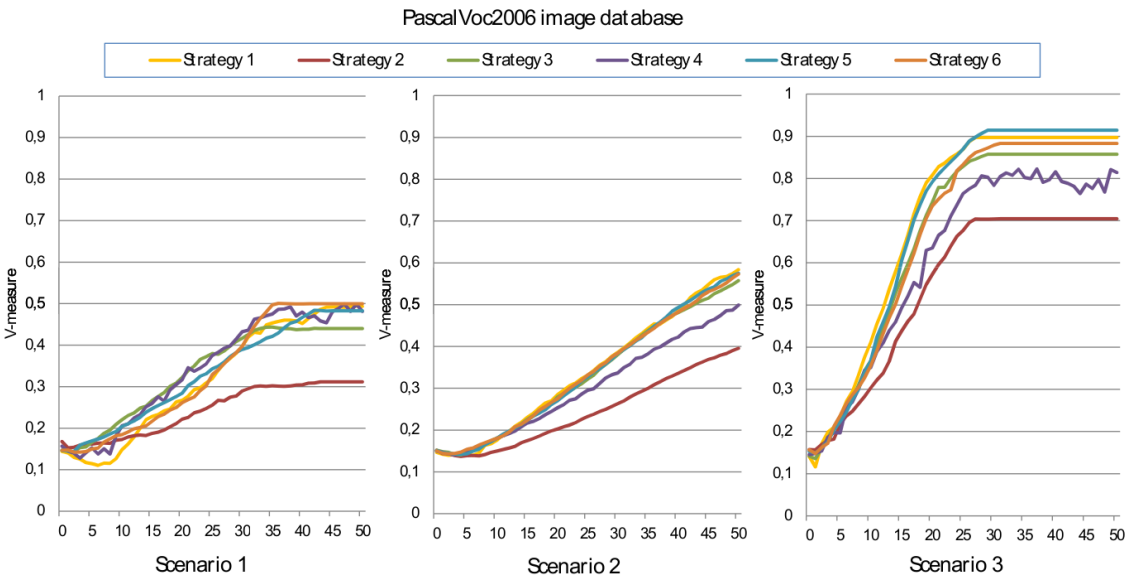


Figure 5.4 – Results of our proposed semi-supervised clustering model during 50 interactive iterations on the PascalVoc2006 image database, using 6 strategies for deducing pairwise constraint. The vertical axis specifies the V-measure values and the horizontal axis specifies the number of iterations.

times for each image database. The curves of the scenario 2 shown in Figure 5.3 and 5.4 represent, at each interactive iteration, the mean values of the V-measure over these 10 executions. As in Figures 5.3 and 5.4, the curves corresponding to the 6 pairwise constraint deduction strategies are close each other, it is not visible if the corresponding standard deviation are shown at each iteration in Figures 5.3 and 5.4. Instead, the average standard deviation of each strategy after 50 interactive iterations is presented in Table 5.2. And the corresponding processing times over all 50 interactive iterations for these experiments are shown in Table 5.3. Note that the processing times for the scenario 2 presented in Table 5.3 are the average processing times of 10 executions. As in the previous chapters, the experiments are executed using a normal PC with 2GB of RAM.

The results in Figures 5.3 and 5.4 show that the clustering results progress, in general, after each interactive iteration, in which the system re-clusters all the dataset by considering the supervised information in the form of pairwise constraints deduced from feedback provided by the user agent when interacting with the system. Similar to the experiments of different semi-supervised clustering model presented in Chapter 4, in most cases, the clustering results of our proposed semi-supervised clustering model converge after only a few iterations. This may be explained by the fact that after some interactive iterations, the constraints deduced from new provided feedback and neighbourhood information are similar to the constraints deduced in previous iterations, and therefore, no new knowledge is provided to the system and the clustering results cannot be improved further.

Regarding to the three proposed interactive scenarios, we can easily see that for both Wang and PascalVoc2006 image databases and for all the 6 pairwise constraint deduction strategies, the clustering results are better and converge more quickly when the number of chosen clusters in each interactive iteration is higher (scenario

Table 5.2 – Average standard deviation of 50 interactive iterations over 10 executions of the scenario 2 corresponding to the experiments of our proposed interactive semi-supervised clustering model on the Wang and PascalVoc2006 image databases shown in Figures 5.3 and 5.4. 6 strategies for deducing pairwise constraints between images are used.

	Average standard deviation (Scenario 2)	
	Wang database	PascalVoc2006 database
Strategy 1	0.033	0.022
Strategy 2	0.044	0.017
Strategy 3	0.045	0.025
Strategy 4	0.047	0.022
Strategy 5	0.036	0.024
Strategy 6	0.044	0.026

Table 5.3 – Processing times over 50 interactive iterations of the experiments shown in Figures 5.3 and 5.4 of our proposed interactive semi-supervised clustering model. These experiments are on the Wang and PascalVoc2006 image databases, according to 3 interactive scenarios and 6 strategies for deducing pairwise constraints between images.

	Wang database		
	Scenario 1	Scenario 2	Scenario 3
Strategy 1	1h58'	2h24'	1h41'
Strategy 2	9'	12'	10'
Strategy 3	31'	19'	47'
Strategy 4	8'	9'	8'
Strategy 5	8'	9'	9'
Strategy 6	6'	8'	8'
	PascalVoc2006 database		
	Scenario 1	Scenario 2	Scenario 3
Strategy 1	16d12h	14d11h	14h9h
Strategy 2	2h55'	4h02'	5h6'
Strategy 3	3h23'	6h39'	6h22'
Strategy 4	1h9'	1h33'	2h17'
Strategy 5	3h33'	4h42'	3h10'
Strategy 6	1h3'	1h21'	2h

3 gives better results and converges more quickly than scenarios 1 and 2). This is due to the fact that the higher the number of clusters chosen for interacting in each iteration, the higher the number of constraints used for re-clustering by the proposed semi-supervised method, and the better the results. And the higher the number of constraints deduced in each interactive iteration, the sooner no more new knowledge is provided and the sooner the clustering converges. In addition, for both image databases, scenario 2, in which clusters are randomly chosen for interacting, gives better results than scenario 1, in which the closest clusters are chosen. In fact, when closest clusters are always selected for interacting in every iteration, there may be only some clusters that always receive user feedback while others never receive any feedback. Therefore, the constraint information deduced from user feedback and neighbourhood information is more limited than when all the clusters could receive user feedback when clusters are randomly selected.

Regarding the different strategies for deducing pairwise constraints between images, we can see that, for the scenario 2, the average standard deviations of 50 interactive iterations over 10 executions shown in Table 5.2 are similar for all strategies. Therefore, different strategies can be analyzed based on the mean values shown in Figures 5.3 and 5.4. We can see that:

- Strategy 1 shows, in general, very good performance but the processing time is huge. This can be explained by the fact that all possible user constraints and deduced constraints created based on the user feedback accumulated during all interactive iterations are used as supervised information for the re-clustering process. When using a high number of pairwise constraints as supervised information, the clustering result can be considerably improved, but the processing time is very high, as all constraints are considered when minimizing the objective function in Equation 5.1. Due to the huge processing time, strategy 1 is not suitable to be used in the interactive context where the user is involved.

- Strategy 2, which does not use any deduced constraints but all possible user constraints created during all interactive iterations for re-clustering the dataset, generally gives the worst results. In fact, among 6 strategies for deducing pairwise constraints between images used in our experiments, only this strategy does not use any deduced constraint. Thus, we can conclude that deduced constraints are needed for improving the performance. The processing time of this strategy shown in Table 5.3 is still high due to the large number of user constraints. Therefore, the number of user constraints used for re-clustering needs to be reduced.

- Strategy 3 shows good or very good performance. By also using constraints deduced in the current iteration along with all possible user constraints created in all interactive iterations, strategy 3, in comparison with strategy 2, gives better results but with a higher processing time. The results shown in Figures 5.3 and 5.4 corresponding to strategy 3 show some oscillations between different iterations. This may be due to the fact that, when withdrawing deduced constraints created in previous iterations, some important constraints are omitted.

- Strategy 4, with the same idea of using user constraints of all interactive iterations and deduced constraints of the current iteration, generally gives better results than strategy 2, but the results are unstable due to the lack of constraints deduced in previous iterations. This strategy has good execution time while reducing the number of user constraints and deduced constraints created in each iteration. The worse performance of the strategy 4 in comparison to the strategy 3 shows that the substitution of the constraints between an image and all the images of a cluster (a neighbourhood) by the constraint between this image and the centre image of the cluster (or neighbourhood) is not a good strategy for reducing constraints created in each iteration, as it probably provides a too limited amount of information.

- Strategy 5, which uses both user constraints and deduced constraints of all interactive iterations, generally gives good or very good results. Its processing time is still high, but less than the processing time of the strategy 1, by reducing the number of pairwise constraints created in each iteration. However, strategy 5 and strategy 1 show similar performance. This may be due to the fact that, while reducing the number of constraints in each iteration, strategy 5 keeps the most important (*i.e.* the most "difficult") constraints (must-links between the most distant images and cannot-links between the closest images). Therefore, this strategy can be used in place of the strategy 1 for reducing the processing time.

- Strategy 6, by reducing the number of deduced cannot-link constraints in comparison to the strategy 5, gives in general very good results. Its performance is a little less than the performance of the strategy 5, but its execution time is considerably lower than the execution time of the strategy 5.

From this analysis, we can conclude that strategy 6 shows the best trade-off between performance and processing time. This strategy is suitable to be used in the interactive context and therefore it is used in our further experiments with bigger image databases.

5.3.3 Comparison of the proposed semi-supervised clustering and HMRF-kmeans

The experiments in this section aims at comparing our interactive semi-supervised clustering model with the semi-supervised HMRF-kmeans, the latter giving the best results in our experimental comparison of different semi-supervised clustering models presented in Chapter 4. Note that when using the semi-supervised HMRF-kmeans in the re-clustering phase, the initial unsupervised clustering is k-means. Our experiments of both semi-supervised clustering models are automatically carried out using the same software agent for simulating the behaviour of the human user when interacting with the system. In each interactive iteration, for each selected cluster, the user agent gives all possible feedback (chooses all possible positive and negative images of each selected cluster, moves all possible negative images to the right clusters). Therefore, for each interactive scenario, the numbers of user feedback (user clicks) are similar for both our interactive semi-supervised clustering

and the HMRF-kmeans clustering. The clustering results of the two methods are thus comparable. Note that the constraints between images deduced by using one of the deduction strategies presented in Table 5.1 are directly used by the HMRF-kmeans clustering, while they should be deduced into pairwise constraints between CF entries for being used by our semi-supervised clustering method.

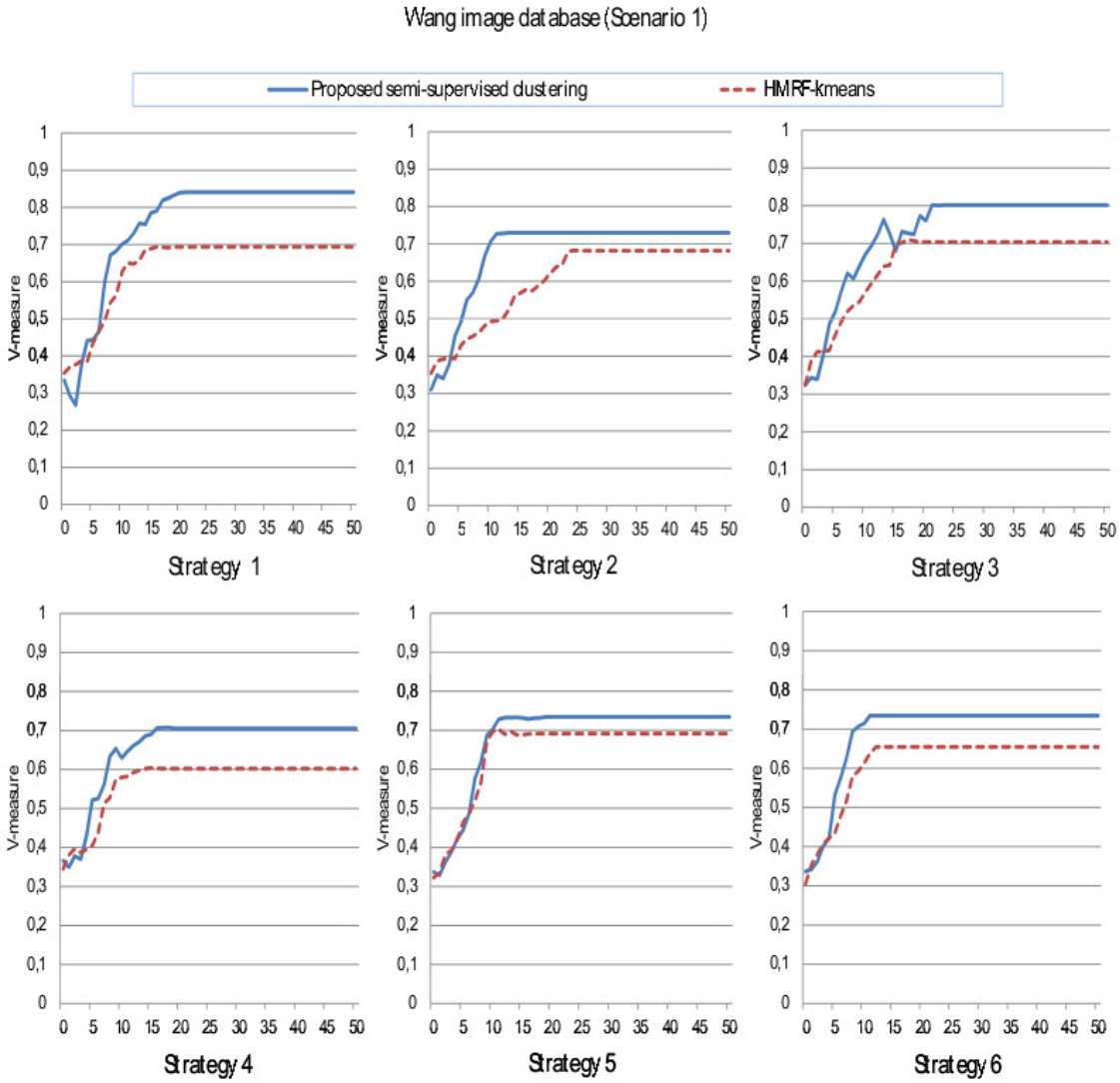


Figure 5.5 – Comparison of the proposed semi-supervised clustering and the HMRF-kmeans clustering on the Wang image database, according to the scenario 1. The 6 strategies for deducing pairwise constraints between images presented in Table 5.1 are used.

Comparison using the 6 pairwise constraint deduction strategies We first compare our proposed interactive semi-supervised clustering model with the HMRF-kmeans clustering, by using the experiments corresponding to the six strategies presented in Table 5.1 for deducing pairwise constraints between images. These experiments are carried out on the Wang image database using the three scenarios used in the previous experiments.

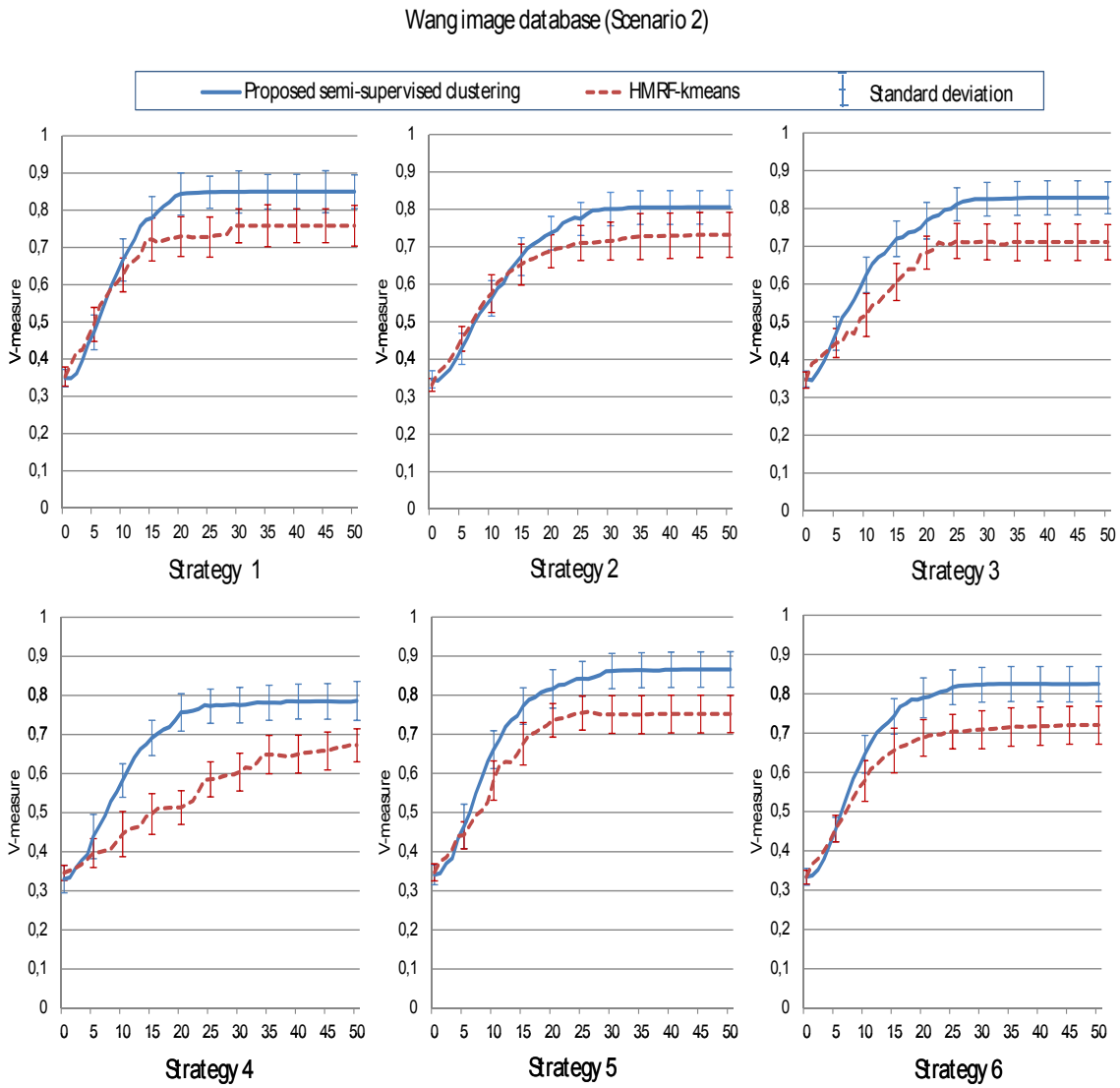


Figure 5.6 – Comparison of the proposed semi-supervised clustering and the HMRF-kmeans clustering on the Wang image database, according to the scenario 2. The 6 strategies for deducing pairwise constraints between images presented in Table 5.1 are used.

Figures 5.5, 5.6 and 5.7 represent the clustering results for 50 interactive iterations on the Wang image database respectively corresponding to the scenarios 1, 2 and 3 when using our semi-supervised clustering and the semi-supervised HMRF-kmeans in the re-clustering phase. The vertical axis specifies the V-measure values and the horizontal axis specifies the number of iterations. Note that the results of scenario 2, in which the user agent randomly chooses 5 among 10 clusters of the database for visualizing and giving feedback, represent, at each interactive iteration, the mean value of the V-measure over 10 executions, along with the corresponding standard deviation from the mean. The corresponding processing times over all 50 interactive iterations for these experiments are presented in Table 5.4. As usual, the average processing times of 10 executions are presented in Table 5.4 for the experiments of scenario 2.

We can see, from the results shown in Table 5.4 and Figures 5.5, 5.6 and 5.7,

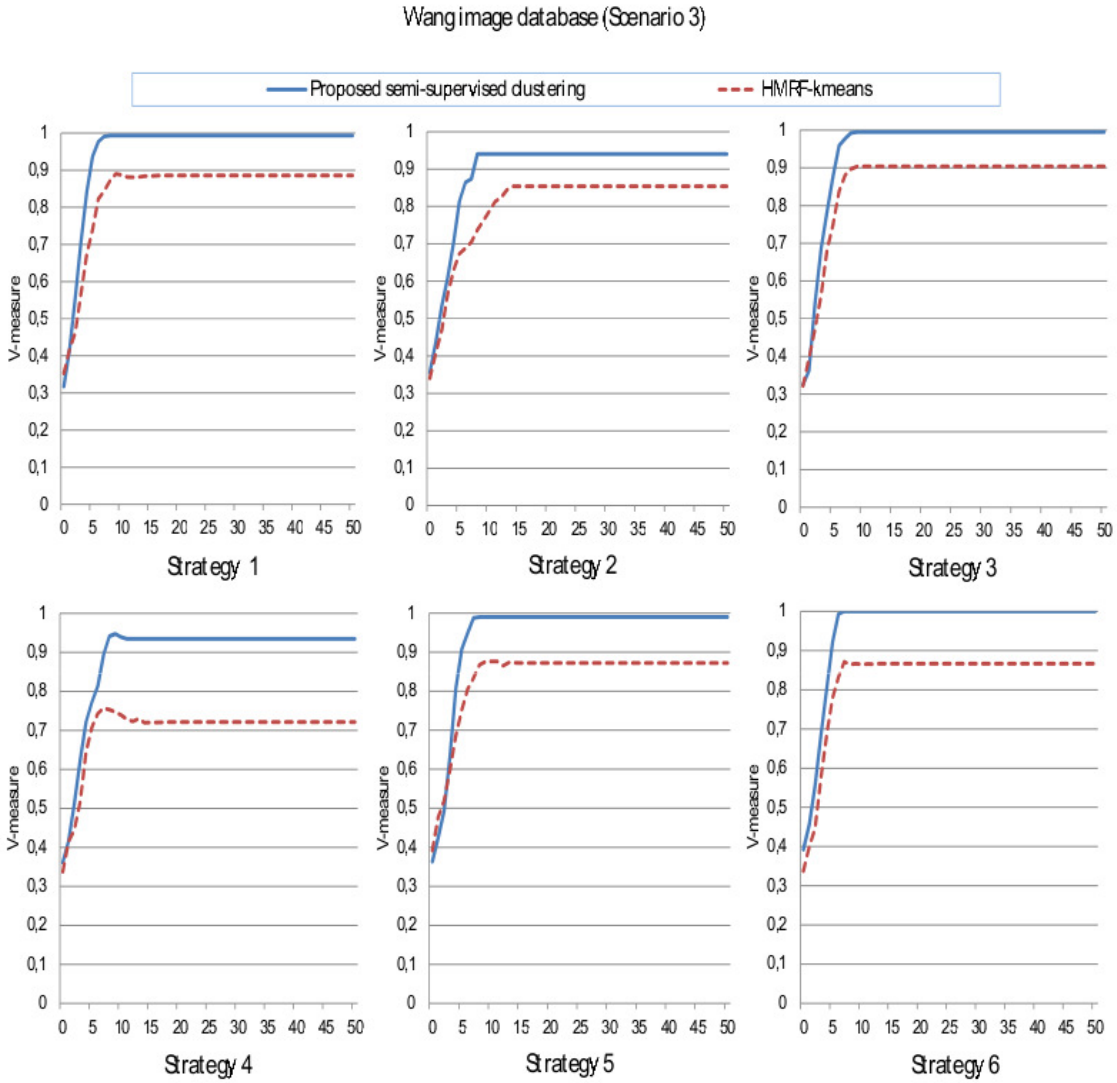


Figure 5.7 – Comparison of the proposed semi-supervised clustering and the HMRF-kmeans clustering on the Wang image database, according to the scenario 3. The 6 strategies for deducing pairwise constraints between images presented in Table 5.1 are used.

that in all scenarios and for all different strategies for deducing pairwise constraints between images, our semi-supervised clustering method always gives better results, in a lower processing time than the HMRF-kmeans. We can see that the results of the initial clustering (k-means for the HMRF-kmeans and BIRCH for our method) shown in Figures 5.5, 5.6 and 5.7 are generally similar. Therefore, the better performance of our model is mainly due to the interactive phase. In fact, for each scenario, the numbers of user feedback (user clicks) given by the software agent in each interactive iteration are similar for both our method and the HMRF-kmeans method. Therefore, when using a same pairwise constraint deduction strategy, the number of pairwise constraints between images created in each iteration are equivalent for both methods. While the pairwise constraints between images are directly used by the HMRF-kmeans, they are deduced into pairwise constraints between CF entries for being used by our semi-supervised clustering. As a CF entry groups a

Table 5.4 – Processing times over 50 interactive iterations of the experiments shown in Figures 5.5, 5.6 and 5.7 comparing our proposed interactive semi-supervised clustering model with the HMRF-kmeans clustering. These experiments are on the Wang image database, according to 3 interactive scenarios and 6 strategies for deducing pairwise constraints between images.

	Wang database					
	Proposed model			**HMRF-kmeans**		
	Scenario 1	Scenario 2	Scenario 3	Scenario 1	Scenario 2	Scenario 3
Strategy 1	1h58'	2h24'	1h41'	7h10'	4h25'	30h56'
Strategy 2	9'	12'	10'	18'	12'35"	23'
Strategy 3	31'	19'	47'	38'	53'	44'
Strategy 4	8'	9'	8'	9'	12'	12'
Strategy 5	8'	9'	9'	14'	17'	15'
Strategy 6	6'	8'	8'	7'	11'	10'

list of similar images (images which are close in the feature space), many pairwise constraints between some images of entry CF_i and some images of entry CF_j can result in only one pairwise constraint between CF_i and CF_j. Thus, with a same set of user feedback (user clicks), the number of pairwise constraints between images, which are used as supervised information in the HMRF-kmeans clustering, is generally greater than the number of pairwise constraints between CF entries, which are used as supervised information in our method. Due to the higher number of pairwise constraints used for re-clustering, the processing time of the HMRF-kmeans is much higher than the processing time of our method. However, HMRF-kmeans was not initially proposed for dealing with user feedback in an interactive context. Moreover, in our method, when a must-link (or cannot-link) (CF_i, CF_j) is deduced from the must-link (or cannot-link) of the corresponding images (x_k, x_l), $x_k \in CF_i$, $x_l \in CF_j$, the constraint (CF_i, CF_j) forces the grouping (or separating) of not only the two images x_k and x_l but also the other images, which do not appear in any pairwise constraints between images, but are included in the entries CF_i and CF_j. We can say that the set of pairwise constraints between CF entries deduced from the set of pairwise constraints between images are stronger and more informative than the set of pairwise constraints between images itself. That explains why our proposed clustering, which uses pairwise constraints between CF entries, gives better performance than the HMRF-kmeans clustering, which uses pairwise constraints between images.

Regarding the different strategies for deducing pairwise constraints between images, we can see for both our semi-supervised clustering method and the HMRF-kmeans method, the strategy 6 gives very good results in low processing time in comparison to the other strategies. This strategy provides a good trade-off between performance and processing time, and therefore is suitable to be used in the interactive context. Thus, in further experiments, we only use strategy 6 for deducing pairwise constraints between images.

Comparison using strategy 6 for deducing pairwise constraints These experiments aim at comparing the performance of our proposed semi-supervised

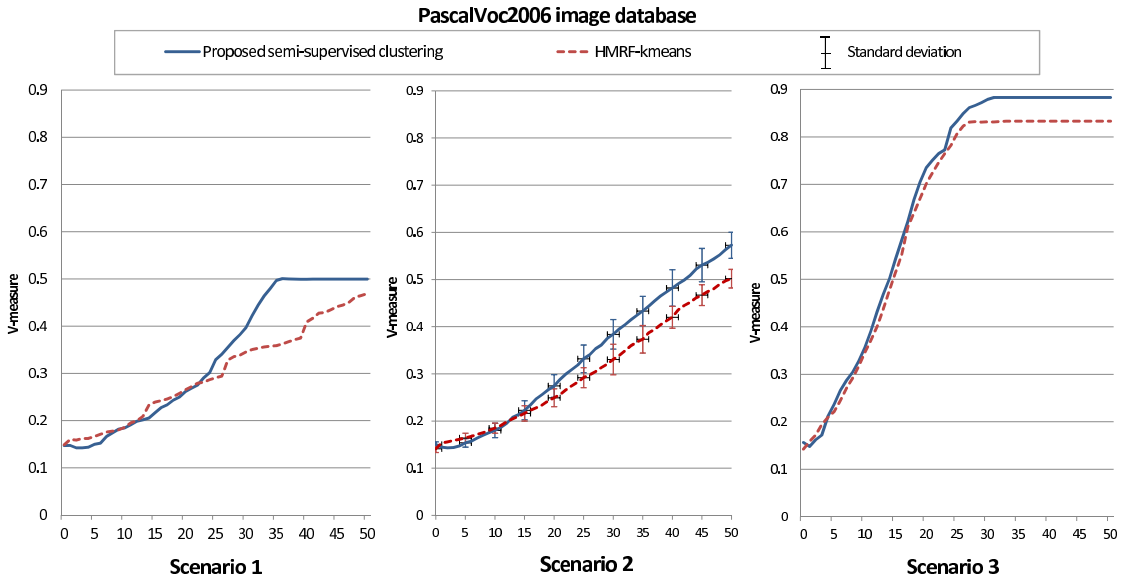


Figure 5.8 – Comparison of the proposed semi-supervised clustering and the HMRF-kmeans clustering on the PascalVoc2006 image database according to the scenario 1, 2 and 3. Strategy 6 for deducing pairwise constraints between images is used (see Table 5.1).

clustering model and the semi-supervised HMRF-kmeans in larger image databases (PascalVoc2006, Caltech101 and Corel30k). For the PascalVoc2006 which has 10 classes of images, the three scenarios 1, 2 and 3 described above are still used for comparing the results of the two methods. For databases which have a large number of classes: Caltech101 (101 classes) and Corel30k (320 classes), we do not show all cluster prototype images to the user on the principal plane, but only the prototype images of a small number of clusters, as explained in 5.3.1. The maximum number of clusters that could be shown on the principal plane in each interactive iteration is fixed to 30. The set of 30 clusters chosen to be shown on the principal plane of different interactive iterations may be slightly different or quite different, depending on the strategy used for choosing the clusters. As presented in Section 5.3.1, two strategies are proposed for choosing clusters to be shown on the principal plane: either 30 clusters are randomly chosen or the 30 closest clusters are chosen. Both these two strategies are analyzed experimentally in this section. Regarding the strategies used by the user agent for choosing the c clusters (among the displayed clusters), for interacting in each interactive iteration, the results of previous experiments show that the scenario 2 gives better results than the scenario 1. Note that the clusters are randomly chosen in the scenario 2, while the closest clusters are chosen in the scenario 1. Therefore, in the following experiments on the Caltech101 and the Corel30k image databases, we present only the clustering results when c among the 30 presented clusters are randomly chosen by the software agent simulating the behaviour of the human user when interacting with the system. We propose the 4 following scenarios for the experiments on the Caltech101 and Corel30k image databases in addition of the first three scenarios presented at the beginning of Section 5.3.2:

- **Scenario 4**: 30 closest clusters are chosen to be shown to the user on the

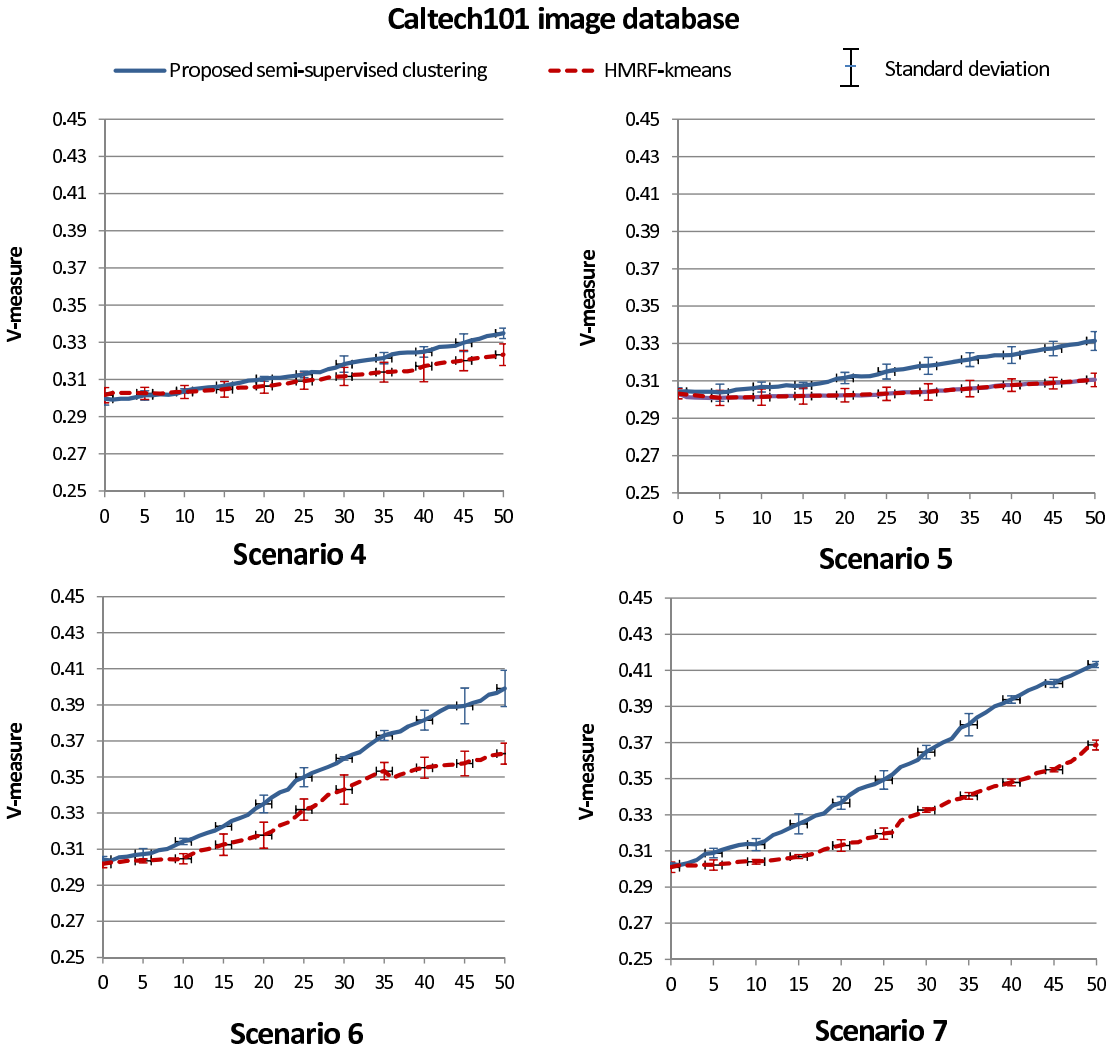


Figure 5.9 – Comparison of the proposed semi-supervised clustering and the HMRF-kmeans clustering on the Caltech101 image database according to the scenario 4, 5, 6 and 7. Strategy 6 for deducing pairwise constraints between images is used (see Table 5.1).

principal plane in each interactive iteration, $c = 5$ clusters are randomly chosen by the user agent for interacting.

- **Scenario 5**: 30 clusters are randomly chosen to be shown to the user on the principal plane in each interactive iteration, $c = 5$ clusters are randomly chosen by the user agent for interacting.

- **Scenario 6**: 30 closest clusters are chosen to be shown to the user on the principal plane in each interactive iteration, $c = 10$ clusters are randomly chosen by the user agent for interacting.

- **Scenario 7**: 30 clusters are randomly chosen to be shown to the user on the principal plane in each interactive iteration, $c = 10$ clusters are randomly chosen by the user agent for interacting.

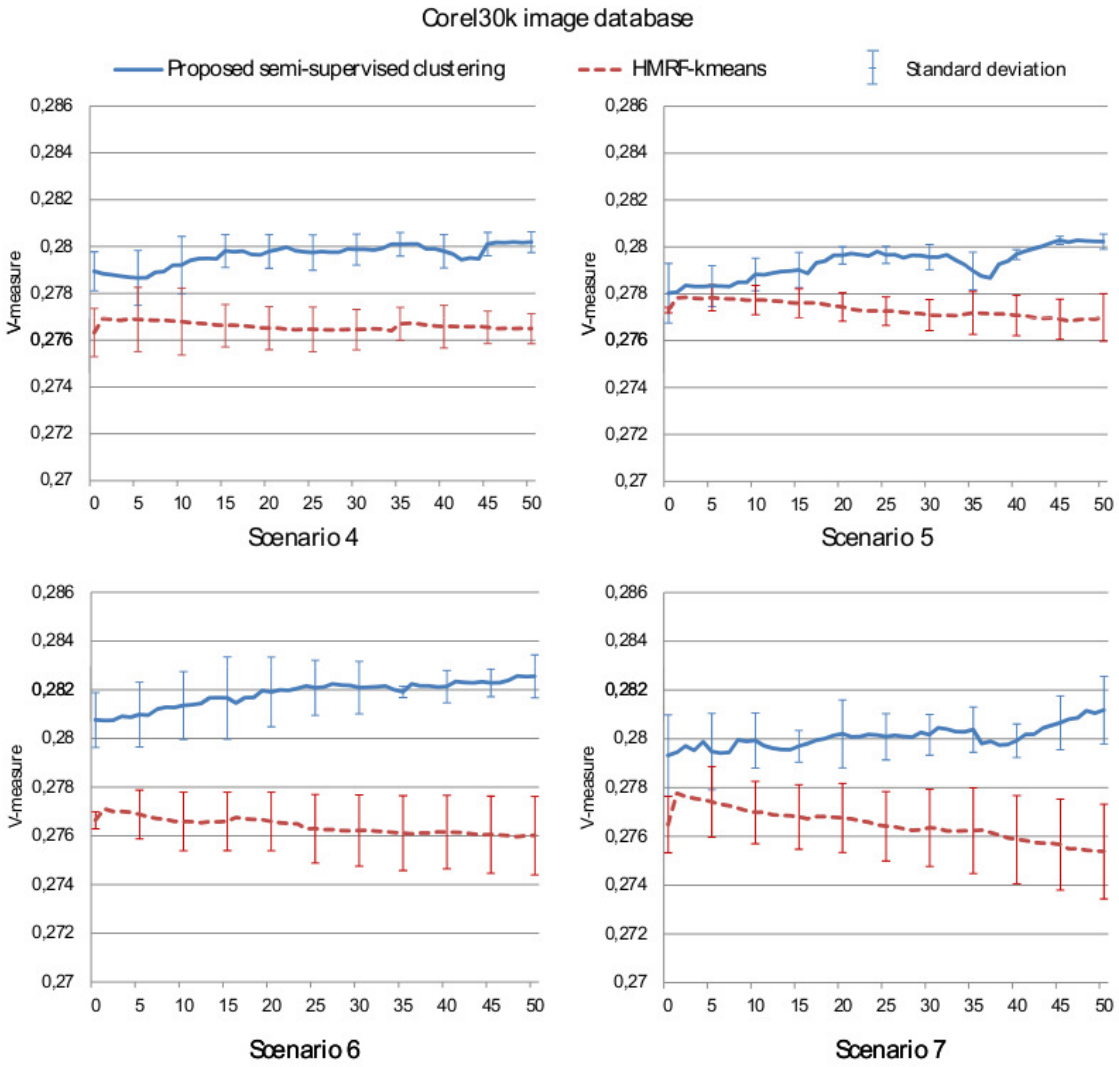


Figure 5.10 – Comparison of the proposed semi-supervised clustering and the HMRF-kmeans clustering on the Corel30k image database according to the scenario 4, 5, 6 and 7. Strategy 6 for deducing pairwise constraints between images presented in Table 5.1 is used.

Figure 5.8 compares our proposed semi-supervised clustering and the HMRF-kmeans method during 50 interactive iterations on the PascalVoc2006 image database, according to the three interactive scenarios 1, 2 and 3. Figures 5.9 and 5.10 present the clustering results of the two methods on respectively the Caltech101 and Corel30k image databases, using the four scenarios 4, 5, 6 and 7. As for previous experiments, the results of the scenario 2 shown in Figure 5.8 represent, at each interactive iteration, the mean value over 10 executions along with the corresponding standard deviation. Moreover, in all the scenarios 4, 5, 6 and 7 used for the experiments on the Caltech101 and Corel30k database, the clusters are also randomly chosen for the user agent interaction. Therefore, each scenario is executed 5 times and Figures 5.9 and 5.10 show the mean values and also the corresponding standard deviations over 5 executions of the clustering results for these scenarios. The processing time (or average processing time) over all 50 interactive iterations

Table 5.5 – Processing times over 50 interactive iterations of the experiments shown in Figures 5.8, 5.9 and 5.10 comparing our proposed interactive semi-supervised clustering model with the HMRF-kmeans clustering. These experiments are on the PascalVoc2006, Caltech101 and Corel30k image databases, using strategy 6 for deducing pairwise constraints between images.

	PascalVoc2006 database			
	Scenario 1	Scenario 2	Scenario 3	
Proposed model	1h3'	1h21'	2h	
HMRF-kmeans	2h16'	3h10'	2h49'	
	Caltech101 database			
	Scenario 4	Scenario 5	Scenario 6	Scenario 7
Proposed model	13h26'	8h4'	33h34'	50h12'
HMRF-kmeans	48h33'	33h45'	157h26'	101h11'
	Corel30k database			
	Scenario 4	Scenario 5	Scenario 6	Scenario 7
Proposed model	45h05'	94h04'	101h47'	217h38'
HMRF-kmeans	154h37'	159h33'	109h26'	629h13'

corresponding to the experiments shown in Figures 5.8, 5.9 and 5.10 are presented in Table 5.5.

The results shown in Figures 5.8, 5.9, 5.10 and Table 5.5 show that for all scenarios and for all image databases, our proposed interactive semi-supervised clustering model outperforms the HMRF-kmeans in both the performance and the processing time. Moreover, the clustering results are also better when the numbers of feedback for each iteration is higher (scenarios 6 and 7, where the number of clusters used for interaction is higher, give better results than scenarios 4 and 5). We can also see that the clustering results progress less when the size of the database increases. Note that the better performance of our semi-supervised clustering model can also be explained by the higher performance of the BIRCH in comparison to the k-means clustering in the initial step, especially for the Corel30k image database.

5.4 Discussion

A new interactive semi-supervised clustering model is presented in this chapter. After receiving user feedback for each interactive iteration, the proposed semi-supervised clustering re-organizes the dataset by considering the pairwise constraints between CF entries deduced from the user feedback. Experimental analysis, using a software agent for simulating human user behaviour, shows that our model helps to improve the clustering results when using user feedback. Note that our experimental scenarios are realistic, and can be realized by a real user as the number of clicks required is tractable. The experiments on different image databases (Wang, PascalVoc2006, Caltech101, Corel30k), presented in this chapter, also show that our semi-supervised clustering method outperforms the semi-supervised HMRF-kmeans in both performance and processing time. Note that the processing times of the experiments presented in this chapter include also the times for analyzing the per-

formance by the external measures, for simulating the behaviour of the human user by the user agent. Therefore, the real processing time of the clustering phase is lower. Moreover, our code is not optimized. The optimization of our code is a part of our future work.

Furthermore, we propose and compare, experimentally, different strategies for deducing pairwise constraints from the user feedback accumulated from all interactive iterations. The experimental results show that strategy 6 in Table 5.1, which keeps only the most important constraints (must-links between the most distant images and cannot-links between the closest images), provides the best trade-off between performance and processing time. Strategy 6 is therefore the most suitable, in our context involving the user in the indexing phase by clustering.

5.5 Summary of the chapter

This chapter presents the main contribution in this thesis: a new interactive semi-supervised clustering model which helps to reduce the semantic gap between the high-level semantic concepts expressed by the user and the low-level features extracted from the images, by involving the user for progressively providing feedback to the system. Our semi-supervised clustering method is developed from the hierarchical BIRCH unsupervised clustering, and is inspired from the integration of the pairwise constraints in the re-clustering of the HMRF-kmeans.

Different of the HMRF-kmeans, our semi-supervised clustering method uses pairwise constraints between CF entries which are more significant than the pairwise constraints between images. Using pairwise constraints between CF entries can help to reduce the number of constraints, since a pairwise constraints between two CF entries corresponds to multiple pairwise constraints between images of these two CF entries. As a consequence, it contributes to a significant reducing of the processing time.

Our semi-supervised clustering proposes to use the neighbourhood information in conjunction with the user feedback for deducing pairwise constraints for the re-clustering phase. The "neighbourhood" is already used in the traditional HMRF-kmeans for grouping objects which are "must-link", based on the basis the pairwise constraints provided as prior knowledge. In this chapter, we extend the neighbourhood for our interactive context where supervised information is progressively provided during different interactive iterations, not in the form of pairwise constraints, but in the form of cluster-level feedback (*i.e.* positive images and negative images of each cluster, displacement of images between clusters). We also define a new term called "seed" which helps to adapt the neighbourhood with the CF entries of the BIRCH tree.

Another important contribution of this chapter is the proposition of 6 different strategies for deducing pairwise constraints. These 6 strategies are both theoretically and experimentally analyzed in this chapter. From these analyses, the strategy 6 which keeps only the most important constraints (must-link between the most distant objects and cannot-links between the closest objects) gives the best trade-off between the performance and the processing time.

The final contribution of this chapter is an experimental comparison of our

method with the HMRF-kmeans, using different image databases of increasing sizes (Wang, PascalVoc2006, Caltech101 and Corel30k). Experimental results show that our semi-supervised clustering outperforms the HMRF-kmeans, in both performance and processing time.

Chapter 6

Conclusions

In this thesis, we are interested in the structuring phase of the Content-Based Image Retrieval (CBIR) model. We assume that the feature extraction phase is completed and the image feature descriptors are available. Instead of traditional indexing methods, we propose to use clustering methods in the structuring phase. The aim is to obtain an indexed structure more adapted to the retrieval of high dimensional and unbalanced data. The objective of this thesis is to implement the clustering in an interactive context where the user is involved to interact with the system in order to reduce the "semantic gap" between the high-level semantic concepts expressed by the user and the low-level features extracted from the images. That may help to improve the clustering results and therefore improve the performance of the further retrieval.

In this chapter, the Section 6.1 summarizes the contributions of this thesis on the interactive clustering problem of image databases of increasing sizes. Section 6.2 presents a discussion of the proposed approach as well as some future directions for this thesis.

6.1 Contributions

In this section, we summarize the contributions of this thesis for involving the user in the structuring phase, by using clustering methods instead of traditional indexing methods, in order to reduce the semantic gap between high-level semantic concepts expressed by the user and the low level features extracted from the images.

Our first contribution, associated with a structured survey of the principal techniques used in different phases of CBIR, lies in analyzing the advantages of using clustering methods instead of traditional feature space structuring method for structuring large image databases. The usefulness of involving the user in the clustering phase in order to reduce the semantic gap is also analyzed.

The second contribution of this thesis is a survey of the main unsupervised clustering methods as well as different measures (internal measures and external measures) for evaluating the clustering results. Different of other surveys in the literature, we respectively analyze the advantages and drawbacks of different unsupervised clustering methods in a context in which the user is involved in the clustering of large image databases and where incrementality and hierarchical structuring are needed. Our formal comparison of the analyzed methods shows that the hier-

archical methods (BIRCH, R-tree, SS-tree and SR-tree) are the most suitable to our context because of their hierarchical structures, their incrementality and their adaptability to large databases.

Our third contribution is an experimental comparison, using different internal and external measures, of some unsupervised clustering methods (global k-means, AHC, R-tree, SR-tree and BIRCH) with different image databases of increasing sizes (Wang, PascalVoc2006, Caltech101 and Corel 30k). The aim is to study the scalability of these approaches when the size of the database is increasing. In order to evaluate these unsupervised clustering approaches in the context of high-dimensional data, different feature descriptors of different sizes are used for these experiments. Based on this experimental comparison, the BIRCH+rgSIFT, using the Bags of Words approach, appears to be the best choice in our context because it gives high performance in low processing time.

Our fourth contribution is a brief state of the art of the principal semi-supervised clustering methods (traditional methods using prior knowledge as well as interactive methods using feedback). The contribution of this state of the art lies in the analysis of the possibility to use different semi-supervised clustering methods in an interactive context in which the user is involved for progressively providing feedback to the system.

The fifth contribution in this thesis is the proposition of a framework for implementing any semi-supervised clustering method in the interactive context. A 2D interactive interface is proposed, allowing the user to provide feedback to the system. Via this interactive interface, the user can give cluster-level feedback (positive images, negative images of each cluster, displacement of images between clusters), without having any prior knowledge about the image database. We proposed to use internal measures for evaluating the quality of the images in their clusters, in order to select the representative images that are shown for each selected cluster on the interactive interface. Internal measures are proposed to be used because they do not need any ground truth about the image database.

Based on our theoretical analysis of different semi-supervised clustering algorithms, our sixth contribution lies in experimentally comparing some semi-supervised clustering methods (the interactive cluster-level semi-supervised clustering, the Rocchio formula based clustering and the HMRF-kmeans), which are the most suitable for our interactive context. External measures, which provide evaluation close to the wishes of the user by using the ground truth, are used for evaluating these experiments. The Rocchio formula based clustering is our proposed interactive semi-supervised clustering, based on the use of the Rocchio formula for incorporating user feedback in initializing the cluster centres for the re-clustering phase. We adapt the traditional semi-supervised HMRF-kmeans for dealing with user feedback in our interactive model. With the aim of automatically undertaking these interactive experiments and avoiding the subjective dependence of the experimental results on a human user, a software software agent simulating the behaviour of the human user is implemented for providing feedback in each interactive iteration. The experimental results show an improvement of the clustering results when involving user feedback. Moreover, the HMRF-kmeans which uses pairwise constraints between images gives the best result in a reasonable processing time.

The seventh contribution, which is our main contribution in this thesis, is a

new interactive semi-supervised clustering model involving the user for progressively providing feedback to the system. Our semi-supervised clustering method is developed from the hierarchical BIRCH unsupervised clustering, and is inspired from the integration of the pairwise constraints in the re-clustering of the HMRF-kmeans. Different of the HMRF-kmeans, our semi-supervised clustering method uses pairwise constraints between CF entries which are more significant than the pairwise constraints between images. Using pairwise constraints between CF entries can help to reduce the number of constraints, and therefore improve the processing time, since multiple pairwise constraints between images of two CF entries can be replaced by only one pairwise constraint between these two CF entries. We propose to use the neighbourhood information in conjunction with the user feedback for deducing pairwise constraints for the re-clustering phase. The "neighbourhood" is already used in the traditional HMRF-kmeans for grouping objects which are "must-link" based on the pairwise constraints provided as prior knowledge. In our work, we explain how to extend the neighbourhood to our interactive context, where supervised information is progressively provided not in the form of pairwise constraints, but in the form of cluster-level feedback (*i.e.* positive images and negative images of each cluster, displacement of images between clusters). We also define a new term called "seed" which helps to adapt the neighbourhood with the CF entries of the BIRCH tree.

The eighth contribution of this thesis is the proposition of 6 different strategies for deducing pairwise constraints, based on the user feedback as well as the neighbourhood information. Both theoretical and experimental analyses of these 6 strategies are proposed. From these analyses, the strategy 6 which keeps only the most "difficult" constraints (must-link between the most distant objects and cannot-links between the closest objects) gives the best trade-off between the performance and the processing time. This strategy is therefore the most suitable to be used in the interactive context in which the user is involved.

The ninth and final contribution of this thesis is an experimental comparison of our semi-supervised clustering with the HMRF-kmeans, using different image databases of increasing sizes (Wang, PascalVoc2006, Caltech101 and Corel30k). According to this comparison, our semi-supervised clustering outperforms the HMRF-kmeans, in both performance and processing time.

6.2 Discussion and future directions

Since interactive clustering for feature space structuring of large image databases is still a difficult problem, there are numerous directions for improving our model.

The first direction, a theoretical perspective, concerns the clustering evaluation measures. As far as we know, there exist only internal measures which do not use any semantic information and external measures which require the ground truth specifying the class labels of all the images in the database. In our system, since the clustering system does not know the ground truth, we use internal measures for evaluating the representativity of the images in each cluster in order to select the images which should be shown to the user on the interactive interface. As only 21 images of each selected cluster are displayed for the user in each interactive

iteration, the choice of these images is very important. It influences the quality of the user feedback, especially in the case of large image databases in which each cluster generally contains a high number of images. In fact, the system receives a small amount of supervised information in the form of user feedback. It may be more efficient if there are some measures which can evaluate the quality of the images in each cluster with a small amount of knowledge provided by the user. Moreover, for huge image databases, the ground truth is generally difficult to be generated. The measures which evaluate the clustering results using a small amount of knowledge are therefore more suitable than external measures. The development of this kind of measures is considered in our future lines of work. This kind of measures will be valuable not only for our context, but also for evaluating the result of any unsupervised or semi-supervised clustering method, based on a small amount of supervised information which is generally easier to have than a complete ground truth.

The next directions concern the experimental aspects. Until now, our experimental analyses generally uses a software agent for simulating the behaviour of the human user for providing feedback to the system. Indeed, the user interface is already implemented and some preliminary experiments with a real user are performed but a complete experimental analysis involving a human user was not performed at a large scale yet. The experiments with different human users in the clustering of different image database are envisaged in the near future for studying the scalability of our model according to the interactions of different users. Our current experiments were only performed with a maximum of 30 000 images. Larger image databases such as MIRFLICK including about one million images are also envisaged for further experiments for analyzing the computational complexity as well as the performance of the proposed model in the case of huge image databases. Moreover, our model is actually only experimented in the interactive clustering context. It was not applied yet in the structuring phase of a real CBIR system. In the near future, we envisage to adapt our model for the structuring phase of a real CBIR system in order to study the variation of the retrieval results when the user is involved in the structuring phase for improving the indexed structure.

Another direction of our work, a technological perspective, is to improve the visualization interface. The efficient visualization of the clustering results of huge image databases is still an open challenge. At the moment, only up to 30 clusters can be visualized on a 2D PCA plane summarizing, for instance, roughly 65% of the data information of the Wang and PascalVoc2006 images databases. This percentage may be reduced when the size of the database is higher. However, the visualization of the clustering results strongly influences the performance of the interactive process. A more adapted representation could help the user better observe the clustering results and give more relevant feedback which could lead to a better performance of the re-clustering phase, and therefore the further retrieval.

Our final direction, a technological/theoretical perspective, lies on improving the user interaction. At the moment, the user can provide cluster-level feedback via some simple interactions (clicks for specifying positive and/or negative images for each cluster, drags and drops images from a cluster to another). In fact, the user may sometimes want to merge different clusters into only one cluster, he may also want to split a cluster into multiple clusters. These kinds of interactions are

realistic, and do not require any prior knowledge about the ground truth, but can be performed by only basing on the visualizing of the clustering results. In our near future, we will adapt our interactive interface for providing these kinds of interactions to the user. For instance, the user can drag a cluster, by clicking on the circle corresponding to this cluster, and drop it to another cluster for demanding to merge these two clusters. For splitting cluster, for instance, the new interactive interface can also provide the user the possibility to create a new "empty" cluster on the screen, and the user then can drag and drop some images from another cluster to this empty cluster. Our semi-supervised clustering also has to be adapted according to these new kinds of user interactions. For instance, we can create must-links between images of the clusters which should be merged, or cannot-link between images of the cluster which should be split and thus perform the re-clustering with the corresponding number of clusters. Another hierarchical clustering can also be envisaged for clustering the CF entries of the BIRCH tree instead of the k-means for more adapting to the merge and split interactions.

Our publications

International Journals with Selection Committee

Published papers:

1. Hien Phuong LAI, Muriel VISANI, Alain BOUCHER and Jean-Marc OGIER. A new interactive semi-supervised clustering model for large image database indexing. *Pattern Recognition Letter* (2013),

   http://dx.doi.org/10.1016/j.patrec.2013.06.014.

2. Hien Phuong LAI, Muriel VISANI, Alain BOUCHER and Jean-Marc OGIER. An experimental comparison of clustering methods for content-based indexing of large image databases. *International Journal on Pattern Analysis and Applications*, volume 15, issue 4 (2012), pp. 345-356.

Submitted papers (pending for results):

1. Hien Phuong LAI, Muriel VISANI, Alain BOUCHER and Jean-Marc OGIER. Unsupervised and interactive semi-supervised clustering for large image database indexing and retrieval. *Fundamenta Informaticae Internaltional Journal* (submitted in december 2012).

Proceedings of International Conferences With Selection Committee

1. Hien Phuong LAI, Muriel VISANI, Alain BOUCHER and Jean-Marc OGIER. Unsupervised and Semi-Supervised Clustering for Large Image Database Indexing and Retrieval. 2012 IEEE RIVF *International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future* (RIVF), pp. 1-6, 2012, doi: 10.1109/rivf.2012.6169869.

Bibliography

[1] A.E. Abdel-Hakim and A.A. Farag. Csift: A sift descriptor with color invariant characteristics. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1978–1983, Newyork, NY, USA, 2006.

[2] Hervé Abdi and Lynne J. Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, 2010.

[3] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, SIGMOD '98, pages 94–105, New York, NY, USA, 1998. ACM.

[4] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, SIGMOD '99, pages 49–60, New York, NY, USA, 1999. ACM.

[5] P. Antonopoulos, N. Nikolaidis, and I. Pitas. Hierarchical face clustering using sift image features. In *CIISP 2007. IEEE Symposium on Computational Intelligence in Image and Signal Processing*, pages 325–329, april 2007.

[6] P. Antonopoulos, N. Nikolaidis, and I. Pitas. Hierarchical face clustering using sift image features. In *IEEE Symposium on Computational Intelligence in Image and Signal Processing, CIISP 2007.*, pages 325–329, april 2007.

[7] Sandra Avila, Nicolas Thome, Matthieu Cord, Eduardo Valle, and Arnaldo de A Araújo. Pooling in image representation: The visual codeword point of view. *Computer Vision and Image Understanding*, 117(5):453–465, 2013.

[8] G. H. Ball and D. J. Hall. Isodata: A novel method of data analysis and pattern classification. Technical report, Menlo Park: Stanford Research Institute, 1965.

[9] Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. Semi-supervised clustering by seeding. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 27–34, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[10] Sugato Basu, Mikhail Bilenko, and Raymond J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the tenth ACM*

SIGKDD international conference on Knowledge discovery and data mining, KDD '04, pages 59–68, New York, NY, USA, 2004. ACM.

[11] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, June 2008.

[12] R. Bayer and E.M. McCreight. Organization and maintenance of large ordered indexes. *Acta Informatica*, 1(3):173–189, 1972.

[13] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: an efficient and robust access method for points and rectangles. *SIGMOD Rec.*, 19(2):322–331, May 1990.

[14] J.L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, September 1975.

[15] Jon Louis Bentley and Jerome H. Friedman. Data structures for range searching. *ACM Comput. Surv.*, 11(4):397–409, December 1979.

[16] Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel. The x-tree: An index structure for high-dimensional data. In *Proceedings of the 22th International Conference on Very Large Data Bases*, VLDB '96, pages 28–39, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

[17] A. Bosch, A. Zisserman, and X. Muñoz. Scene classification using a hybrid generative/discriminative approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(4):712–727, April 2008.

[18] Y-Lan Boureau, Francis Bach, Yann LeCun, and Jean Ponce. Learning mid-level features for recognition. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pages 2559–2566. IEEE, 2010.

[19] S.K. Chang, E. Jungert, and Y. Li. Representation and retrieval of symbolic pictures using generalized 2d strings. Technical report, University of Pittsburgh, 1988.

[20] S.K. Chang, Q.Y. Shi, and C.W. Yan. Iconic indexing by 2-d strings. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 9(3):413–428, May 1987.

[21] Jian Cheng and Kongqiao Wang. Active learning for image retrieval with co-svm. *Pattern Recognition*, 40(1):330–334, 2007.

[22] M. Cord and P. Cunningham. *Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval*. Cognitive Technologies. Springer, 2008.

[23] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2):1–60, 2008.

[24] Ian Davidson and Sugato Basu. A survey of clustering with instance level constraints, 2010.

[25] Ian Davidson and S. S. Ravi. Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. In *Lecture notes in computer science*, pages 59–70. Springer, 2005.

[26] Avinava Dubey, Indrajit Bhattacharya, and Shantanu Godbole. A cluster-level semi-supervision model for interactive clustering. In *Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases: Part I*, ECML PKDD'10, pages 409–424, Berlin, Heidelberg, 2010. Springer-Verlag.

[27] Martin Ester, Hans P. Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231, Portland, Oregon, 1996. AAAI Press.

[28] R. A. Finkel and J. L. Bentley. Quad trees a data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, 1974.

[29] Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. In *Machine Learning*, pages 139–172, 1987.

[30] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Qian Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The qbic system. *Computer*, 28(9):23–32, September 1995.

[31] H. Fonga. Pattern recognition in gray-level images by fourier analysis. *Journal of Pattern Recognition Letters*, 17(14):1477–1489, dec 1996.

[32] E. B. Fowlkes and C. L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983.

[33] Benjamin C.M. Fung, Ke Wang, and Martin Ester. Hierarchical document clustering using frequent itemsets. In *Proceedings of the SIAM INTERNATIONAL CONFERENCE ON DATA MINING 2003 (SDM 2003)*, 2003.

[34] V. Gaede and O. Günther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, June 1998.

[35] R. G. Gallager, P. A. Humblet, and P. M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Trans. Program. Lang. Syst.*, 5(1):66–77, January 1983.

[36] D. Geman, S. Geman, C. Graffigne, and P. Dong. Boundary detection by constrained optimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(7):609–628, July 1990.

[37] Mark A. Gluck and James E. Corter. Information, uncertainty, and the utility of categories. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, pages 283–287, Hillsdale, NJ, 1985. Lawrence Earlbaum.

[38] David Gorisse, Matthieu Cord, and Frédéric Precioso. Salsas: Sub-linear active learning strategy with approximate k-nn search. *Pattern Recognition*, 44(10):2343–2357, 2011.

[39] P.-H. Gosselin and M. Cord. Active learning methods for interactive image retrieval. *Image Processing, IEEE Transactions on*, 17(7):1200–1211, 2008.

[40] Nizar Grira, Michel Crucianu, and Nozha Boujemaa. Active semi-supervised fuzzy clustering. *Pattern Recognition*, 41(5):1834–1844, 2008.

[41] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Cure: an efficient clustering algorithm for large databases. *SIGMOD Rec.*, 27(2):73–84, June 1998.

[42] Antonin Guttman. R-trees: a dynamic index structure for spatial searching. *SIGMOD Rec.*, 14(2):47–57, June 1984.

[43] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. Clustering validity checking methods: part ii. *SIGMOD Rec.*, 31(3):19–27, September 2002.

[44] R.M. Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5):786–804, may 1979.

[45] R.M. Haralick, I. Dinstein, and K. Shanmugam. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–621, nov 1973.

[46] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of Fourth Alvey Vision Conference*, pages 147–151, Manchester, UK, 1988.

[47] Ji He, Ah-Hwee Tan, Chew-Lim Tan, and Sam-Yuan Sung. On quantitative evaluation of clustering systems, 2002.

[48] Andreas Henrich, Hans werner Six, and Peter Widmayer. The lsd tree: spatial access to multidimensional point and non-point objects. In *Proceedings of the 15th International Conference on Very large data bases*, pages 45–53, 1989.

[49] Alexander Hinneburg and Daniel A. Keim. A general approach to clustering in large databases with noise. *Knowledge and Information Systems*, 5:387–415, 2003.

[50] Steven C.H. Hoi, Rong Jin, Jianke Zhu, and Michael R. Lyu. Semi-supervised svm batch mode active learning for image retrieval. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR2008)*, pages 1–7, Alaska, US, 2008.

[51] Pengyu Hong, Qi Tian, and Thomas S. Huang. Incorporate support vector machines to content-based image retrieval with relevant feedback. In *ICIP*, pages 750–753, 2000.

[52] P. Howarth and S. Rüger. Evaluation of texture features for content-based image retrieval. In Peter Enser, Yiannis Kompatsiaris, NoelE. O'Connor, AlanF. Smeaton, and ArnoldW.M. Smeulders, editors, *Image and Video Retrieval*, volume 3115 of *Lecture Notes in Computer Science*, pages 326–334. Springer Berlin Heidelberg, 2004.

[53] M.-K. Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8(2):179–187, feb 1962.

[54] J. Huang, S.R. Kumar, M. Mitra, W.-J. Zhu, and R. Zabih. Image indexing using color correlograms. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 762–768, Puerto Rico, jun 1997.

[55] P.W. Huang and Y.R. Jean. Using 2d c+-strings as spatial knowledge representation for image database systems. *Pattern Recognition*, 27(9):1249–1257, 1994.

[56] T.S. Huang, C.K. Dagli, S. Rajaram, E.Y. Chang, M.I. Mandel, Graham E. Poliner, and D.P.W. Ellis. Active learning for interactive multimedia retrieval. *Proceedings of the IEEE*, 96(4):648–667, 2008.

[57] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Survey*, 31(3):264–323, September 1999.

[58] Sangoh Jeong. Histogram-based color image retrieval. Technical report, Psych221/EE362 Project Report, Stanford University, 2001.

[59] E. Jungert. Extended symbolic projection as a knowledge structure for image database systems. In *Fourth BPRA Conference on Pattern Recognition*, pages 343–351. Springer-Verlag, mar 1988.

[60] Norio Katayama and Shin'ichi Satoh. The sr-tree: an index structure for high-dimensional nearest neighbor queries. *SIGMOD Rec.*, 26(2):369–380, June 1997.

[61] T. Kato, T. Kurita, N. Otsu, and K. Hirata. A sketch retrieval method for full color image database-query by visual example. In *Proceedings of the 11th IAPR International Conference on Pattern Recognition, Vol.I. Conference A: Computer Vision and Applications*, pages 530–533, 1992.

[62] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, New York, 1990.

[63] L. Kaufman and P.J. Rousseeuw. *Clustering by means of Medoids*, pages 405–416. Y. Dodge, North-Holland, 1987.

[64] Y. Kinoshita, N. Nitta, and N. Babaguchi. Interactive clustering of video segments for media structuring. In *IEEE International Conference on Multimedia and Expo, 2005. ICME 2005.*, pages 630–633, 2005.

[65] Dan Klein, Sepandar D. Kamvar, and Christopher D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 307–314, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[66] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.

[67] P. Kruizinga, N. Petkov, and S. E. Grigorescu. Comparison of texture features based on gabor filters. In *Proceedings of the 10th International Conference on Image Analysis and Processing*, ICIAP '99, pages 142–147, Washington, DC, USA, 1999. IEEE Computer Society.

[68] Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, February 1956.

[69] B. Kulis, S. Basu, I. Dhillon, and Raymond J. Mooney. Semi-supervised graph clustering: A kernel approach. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 457–464, Bonn, Germany, August 2005.

[70] G. N. Lance and W. T. Williams. A general theory of classificatory sorting strategies: 1. hierarchical systems. *The Computer Journal*, 9(4):373–380, February 1967.

[71] G.N. Lance and W.T. Williams. Mixed-data classificatory programs, i.) agglomerative systems. *Australian Computer Journal*, 1:15–20, 1967.

[72] C. Lau, D. Tjondronegoro, J. Zhang, S. Geva, and Y. Liu. Fusing visual and textual retrieval techniques to effectively search large collections of wikipedia images. In *Comparative Evaluation of XML Information Retrieval Systems*, volume 4518 of *Lecture Notes in Computer Science*, pages 345–357. Springer Berlin Heidelberg, 2007.

[73] S.Y. Lee and F.J. Hsu. 2d c-string: a new spatial knowledge representation for image database systems. *Pattern Recognition*, 23(10):1077–1087, October 1990.

[74] T.S. Lee. Image representation using 2d gabor wavelets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):959–971, oct 1996.

[75] F. Leymarie and B. Kimia. From the infinitely large to the infinitely small: Applications of medial symmetry representations of shape. In Kaleem Siddiqi and StephenM. Pizer, editors, *Medial Representations: Mathematics, Algorithms and Applications*, volume 37 of *Computational Imaging and Vision*, pages 327–351. Springer Netherlands, 2008.

[76] Aristidis Likas, Nikos Vlassis, and Jakob J. Verbeek. The global k-means clustering algorithm. *Pattern Recognition*, 36(2):451–461, 2003.

[77] Wen-Cheng Lin, Yih-Chen Chang, and Hsin-Hsi Chen. Integrating textual and visual information for cross-language image retrieval: A trans-media dictionary approach. *Information Processing & Management*, 43(2):488–502, 2007.

[78] T. Lindeberg. *Scale-Space Theory in Computer Vision (The Springer International Series in Engineering and Computer Science)*. Springer, 1 edition, December 1993.

[79] Haiming Liu, Dawei Song, Stefan Rüger, Rui Hu, and Victoria Uren. Comparing dissimilarity measures for content-based image retrieval. In *Proceedings of the 4th Asia information retrieval conference on Information retrieval technology*, AIRS'08, pages 44–50, Berlin, Heidelberg, 2008. Springer-Verlag.

[80] Hong Liu and Shang teng Huang. Evolutionary semi-supervised fuzzy clustering. *Pattern Recognition Letters*, 24(16):3105–3113, 2003.

[81] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[82] Ke Lu, Jidong Zhao, Mengqin Xia, and Jiazhi Zeng. Semi-supervised learning for image retrieval using support vector machines. In *Advances in Neural Networks – ISNN 2005*, volume 3496 of *Lecture Notes in Computer Science*, pages 677–681. Springer Berlin Heidelberg, 2005.

[83] W.-Y. Ma and B.S. Manjunath. Netra: a toolbox for navigating large image databases. *Multimedia System*, 7(3):184–198, May 1999.

[84] V. Macario and F.A.T. de Carvalho. A new approach for semi-supervised clustering based on fuzzy c-means. In *2010 IEEE International Conference on Fuzzy Systems (FUZZ)*, pages 1–8, 2010.

[85] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. Fifth Berkeley Symp. on Math. Statist. and Prob.*, volume 1, pages 281–297. Univ. of Calif. Press, 1967.

[86] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *J. Mach. Learn. Res.*, 11:19–60, 2010.

[87] A. Materka and M. Strzelecki. Texture analysis methods - a review. Technical report, Institute of Electronics, Technical University of Lodz, 1998.

[88] T. Matsuyama, K. Saburi, and M. Nagao. A structural analyser for regularly arranged textures. *Computer Vision Graphics Image Process*, 18:259–278, 1982.

[89] Geoffrey J. McLachlan and Thriyambakam Krishnan. *The EM Algorithm and Extensions (Wiley Series in Probability and Statistics)*. Wiley-Interscience, 2 edition, March 2008.

[90] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Proceedings of the Eighth IEEE International Conference on Computer Vision*, volume 1, pages 525–531, Vancouver, Canada, 2001. IEEE Computer society.

[91] Glenn W. Milligan, S. C. Soon, and Lisa M. Sokol. The effect of cluster size, dimensionality, and the number of clusters on recovery of true cluster structure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(1):40–47, jan. 1983.

[92] F. Mindru, T. Tuytelaars, L. Van Gool, and T. Moons. Moment invariants for recognition under changing viewpoint and illumination. *Computer Vision and Image Understanding*, 94(1-3):3–27, 2004.

[93] Y. Mingqiang, K. Kpalma, and R. Joseph. A survey of shape feature extraction techniques. *Pattern Recognition Techniques, Technology and Applications*, pages 43–90, November 2008.

[94] B. Mirkin. *Mathematical classification and clustering.* Kluwer Academic Press, 1996.

[95] P. Murugavel and M. Punithavalli. Improved hybrid clustering and distance-based technique for outlier removal. *International Journal on Computer Science and Engineering (IJCSE)*, 3(1):333–339, 2011.

[96] Raymond T. Ng and Jiawei Han. Clarans: A method for clustering objects for spatial data mining. *IEEE Trans. on Knowl. and Data Eng.*, 14(5):1003–1016, September 2002.

[97] Nhu Van Nguyen. *Représentations visualles de concepts textuels pour la recherche et l'annotation interactives d'images.* PhD thesis, Université de La Rochelle, 2011.

[98] J. Nievergelt, Hans Hinterberger, and Kenneth C. Sevcik. The grid file: An adaptable, symmetric multikey file structure. *ACM Trans. Database Syst.*, 9(1):38–71, March 1984.

[99] T. Ojala, M. Pietikainen, and D. Harwood. Comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, 29(1):51–59, 1996.

[100] Michael Ortega-Binderberger and Sharad Mehrotra. Relevance feedback techniques in the mars image retrieval system. *Multimedia Systems*, 9:535–547, 2004.

[101] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.

[102] A. Pentland, R.W. Picard, and S. Sclaroff. Photobook: Content-based manipulation of image databases. *International Journal of Computer Vision*, 18(3):233–254, June 1996.

[103] E.G.M. Petrakis. Image representation, indexing and retrieval based on spatial relationships and properties of objects, 1993.

[104] David Picard, Matthieu Cord, and Arnaud Revel. Image retrieval over networks: active learning using ant algorithm. *Multimedia, IEEE Transactions on*, 10(7):1356–1365, 2008.

[105] Konstantinos N. Plataniotis and Anastasios N. Venetsanopoulos. *Color image processing and applications*. Springer-Verlag New York, Inc., New York, NY, USA, 2000.

[106] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technology Journal*, 36:1389–1401, 1957.

[107] J. Puzicha, J.M. Buhmann, Y. Rubner, and C. Tomasi. Empirical evaluation of dissimilarity measures for color and texture. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1165–1173, 1999.

[108] J. Puzicha, T. Hofmann, and J.M. Buhmann. Non-parametric similarity measures for unsupervised texture segmentation and image retrieval. In *Proceedings of the 1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 267–272, jun 1997.

[109] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[110] Lecourtier Y Ribert A, Ennaji A. An incremental hierarchical clustering. In *Proceedings of the 1999 vision interface (VI) conference*, page 586–591, 1999.

[111] C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979.

[112] John T. Robinson. The k-d-b-tree: a search structure for large multidimensional dynamic indexes. In *Proceedings of the 1981 ACM SIGMOD international conference on Management of data*, SIGMOD '81, pages 10–18, New York, NY, USA, 1981. ACM.

[113] J. Rocchio. *Relevance Feedback in Information Retrieval*, pages 313–323. Prentice Hall, Englewood, Cliffs, New Jersey, 1971.

[114] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, 2007.

[115] Peter Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(1):53–65, November 1987.

[116] Hanan Samet. The quadtree and related hierarchical data structures. *ACM Comput. Surv.*, 16(2):187–260, June 1984.

[117] Timos K. Sellis, Nick Roussopoulos, and Christos Faloutsos. The r+-tree: A dynamic index for multi-dimensional objects. In *Proceedings of the 13th International Conference on Very Large Data Bases*, VLDB '87, pages 507–518, San Francisco, CA, USA, 1987. Morgan Kaufmann Publishers Inc.

[118] Lokesh Setia. *Machine learning strategies for content based image retrieval.* PhD thesis, University of Freiburg, 2008.

[119] Ron Shamir and Roded Sharan. Algorithmic approaches to clustering gene expression data. In *Current Topics in Computational Biology*, pages 269–300. MIT Press, 2001.

[120] Gholamhosein Sheikholeslami, Surojit Chatterjee, and Aidong Zhang. Wavecluster: a wavelet-based clustering approach for spatial data in very large databases. *The VLDB Journal*, 8(3-4):289–304, February 2000.

[121] J. Sivic and A. Zisserman. Video google: a text retrieval approach to object matching in videos. In *Proceedings of the ninth IEEE International Conference on Computer Vision*, volume 2, pages 1470–1477, Nice, France, oct 2003.

[122] J.R. Smith and S.-F. Chang. Visualseek: a fully automated content-based image query system. In *Proceedings of the fourth ACM international conference on Multimedia*, MULTIMEDIA '96, pages 87–98, New York, NY, USA, 1996. ACM.

[123] Pascal Soucy and Guy W. Mineau. Beyond tfidf weighting for text categorization in the vector space model. In *Proceedings of the 19th international joint conference on Artificial intelligence*, IJCAI'05, pages 1130–1135, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[124] M.A. Stricker and M. Orengo. Similarity of color images. In *SPIE Conference on Storage and Retrieval for Image and Video Databases III*, volume 2420, pages 381–392, San Diego/La Jolla, CA, USA, 1995.

[125] M.R. Teague. Image analysis via the general theory of moments. *Journal of the Optical Society of America*, 70(8):920–930, aug 1980.

[126] Sabrina Tollari, Philippe Mulhem, Marin Ferecatu, Hervé Glotin, Marcin Detyniecki, Patrick Gallinari, Hichem Sahbi, and Zhong-Qiu Zhao. A comparative study of diversity methods for hybrid text and image retrieval approaches. In *Proceedings of the 9th Cross-language evaluation forum conference on Evaluating systems for multilingual and multimodal information access*, CLEF'08, pages 585–592, Berlin, Heidelberg, 2008. Springer-Verlag.

[127] Simon Tong and Edward Chang. Support vector machine active learning for image retrieval. In *Proceedings of the ninth ACM international conference on Multimedia*, MULTIMEDIA '01, pages 107–118. ACM, 2001.

[128] S. Tsuji and F. Tomita. A structural analyzer for a class of textures. *Computer Vision Graphics Image Process*, 2(3-4):216–231, 1973.

[129] M. Tuceryan and A.K. Jain. Texture analysis. In C. H. Chen, L. F. Pau, and P. S. P. Wang, editors, *Handbook of pattern recognition & computer vision*, pages 235–276. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1993.

[130] K.E.A. Van de Sande, T. Gevers, and C.G.M. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582–1596, 2010.

[131] J.C. Van Gemert, C.J. Veenman, A.W.M. Smeulders, and J.-M. Geusebroek. Visual word ambiguity. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(7):1271–1283, 2010.

[132] R.C. Veltkamp and M. Tanase. Content-based image retrieval systems: A survey. Technical report, Department of Computing Science, Utrecht University, 2000.

[133] R.C. Veltkamp, M. Tanase, and D. Sent. Features in content-based image retrieval systems: a survey. *State-of-the-Art in Content-Based Image and Video Retrieval*, 22:97–124, 2001.

[134] F.M. Vilnrotter, R. Nevatia, and K.E. Price. Structural analysis of natural textures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):76–89, 1986.

[135] Kiri Wagstaff and Claire Cardie. Clustering with instance-level constraints. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1103–1110, 2000.

[136] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 577–584, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[137] Wei Wang, Jiong Yang, and Richard R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, VLDB '97, pages 186–195, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[138] Joe H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, March 1963.

[139] David A. White and Ramesh Jain. Similarity indexing with the ss-tree. In *Proceedings of the Twelfth International Conference on Data Engineering*, ICDE '96, pages 516–523, Washington, DC, USA, 1996. IEEE Computer Society.

[140] Rui Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, may 2005.

[141] Charles T. Zahn and Ralph Z. Roskies. Fourier descriptors for plane closed curves. *IEEE Trans. Comput.*, 21(3):269–281, March 1972.

[142] Bin Zhang, Meichun Hsu, and Umeshwar Dayal. K-harmonic means – a data clustering algorithm, 1999.

[143] D. Zhang and G. Lu. A comparative study on shape retrieval using fourier descriptors with different shape signatures. *Journal of Visual Communication and Image Representation*, 1(14):41–60, 2003.

[144] D. Zhang, A. Wong, M. Indrawan, and G. Lu. Content-based image retrieval using gabor texture features. In *Pacific-Rim Conference on Multimedia*, pages 392–395, Sydney, Australia, dec 2000.

[145] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. *SIGMOD Rec.*, 25(2):103–114, June 1996.

[146] Ying Zhao and George Karypis. Criterion functions for document clustering: Experiments and analysis. Technical report, Department of Computer Science, University of Minnesota, 2002.