MEMOIRE DE FIN D'ETUDES

MASTER D'INFORMATIQUE

Systèm de navigation dans des bases de données d'images

Réalisé par :
Le Viet Man
lvman@ifi.edu.vn

Encadré par :
Professeur Michel Verleysen
michel.verleysen@uclouvain.be
Laboratoire de Microélectronique - DICE - UCL

Hanoi, le décembre 2011

Remerciements

Je tiens d'abord à remercier mon encadrement Michel Verleysen, professeur à l'Université Catholique de Louvain, pour m'avoir encadrée pendant ces six mois. Je le remercie de son contact chaleureux, ses conseils et encouragements, son soutien permanent et la liberté de recherche qu'il a bien voulu me laisser.

Un grand merci à professeur Yves Deville et son épouse, Isabelle, pour leur accueil, leur aide et leur sympathie qu'ils m'ont donné.

Je tiens à remercier l'administration publique Wallonie-Bruxelles International (WBI) qui m'a consciencieusement accueilli et donné de la bourse pour ce stage.

Mes plus sincères remerciements vont également à tous les professeurs et les personnels de l'Institut de la Francophonie pour l'Informatique (IFI) pour m'avoir donné des cours de très grande qualité et pour leur soutien tout au long de mes études à l'IFI.

Je remercie chaleureusement mes camarades de la promotion 15 de l'IFI pour leur soutien et leur amitié sans faille.

Je voudrais aussi remercier les confrères du Collège de l'économie - Université de Hue où je suis en train de travailler, qui m'ont donné les meilleures conditions pour que je puisse bien passer ma scolarité à l'IFI.

Finalement, j'adresse un grand merci à toute ma famille et plus particulièrement à mon épouse pour leur soutien et leur énorme encouragement tout au long de mes études à l'IFI.

Hanoi, le 14 décembre 2011

Le Viet Man

Résumé

La disponibilité de grandes bases d'images sous forme numériques nécessite des outils efficaces qui permettent aux utilisateurs de naviguer, rechercher et interagir avec ces bases de données. Récemment, certaines approches ont été introduites pour fournir une interface plus intuitive pour naviguer et explorer des bases d'images en organisant les images sur base de leurs similarités. Ces approches utilisent des algorithmes d'apprentissage pour créer la visualisation sur base de similarités où les positions des images qui sont similaires, sont aussi proches les unes des autres dans l'espace de visualisation. Ces algorithmes sont les méthodes dites de projection (linéaire ou non-linéaire), appelées également « réduction de dimension » ou « manifold learning ». Mais ces approches existent quelques limitations, telles que la complexité de calcul, le chevauchement et la flexibilité.

Dans ce mémoire, nous présentons un modèle de navigation dans des bases d'images qui utilise l'algorithme des cartes auto-organisatrices de Kohonen (SOM) comme une méthode de visualisation des images sur une grille en deux dimensions où les images similaires sont positionnées de façon proches. Cet algorithme a été adapté pour supporter des caractéristiques multiples, la combinaison arbitraire des caractéristiques et la détermination des rôles de chaque caractéristique dans la mesure de la similarité. Ce modèle fournit également un outil de visualisation pour naviguer facilement dans des bases d'images en utilisant une nouvelle méthode de ré-arrangement qui résout le problème de chevauchement dans la grille des images de résultat de l'algorithme SOM.

Mots clés : navigateur d'images, apprentissage, cartes auto-organisatrices, recherche d'images par le contenu, navigateur d'images par le contenu, vision par ordinateur

Abstract

The availability of large databases of images requires effective and efficient tools that allow users to browse, search and interact with databases. Recently, some approaches that provide a more intuitive interface for navigating and exploring image databases by organizing images based on their similarity, have been introduced. These approaches use machine learning algorithms to create a visualization based on similarities in which the positions of the images that are similar, are close to each other in the visualization space. These algorithms are called projection methods (linear or nonlinear), also called "dimension reduction" or "manifold learning". However these approaches have some limitations, such as computational complexity, overlap and flexibility.

In this thesis, we present an image browser using Self-Organizing Maps (SOM) as a method for visualizing images on a 2D grid where similar images are placed close together. This algorithm have been adapted to support many features, the arbitrary combination of features and for determining the roles of each feature in the measure of similarity. This image browser also provides a visualization tool for navigating in image databases using a new re-arrangement method that solves the problem of overlapping images in the grid results of the SOM algorithm.

Keywords : image browser, machine learning, self-organizing maps, content-based image retrieval, content-based image browser, computer vision

Table des matières

Remerciements **i**

Résumé **ii**

Abstract **iii**

Table des matières **4**

Table des figures **6**

Liste des tableaux **8**

1 INTRODUCTION **9**

2 ETAT DE L'ART **13**

2.1 Outils de gestion et de navigation d'images 13

2.2 Navigation d'images par le contenu 16

2.2.1 Recherche d'images par le contenu 16

2.2.2 Visualisation de bases d'images 18

2.2.3 Navigation dans la base d'images 24

3 CARACTERISATION DES IMAGES **25**

3.1 Caractéristique de couleur 26

3.2 Caractéristique de texture 27

3.3 Caractéristique de forme 28
3.4 Caractéristiques en régions 30
3.5 Architecture du module d'extensions 31

4 VISUALISATION DE BASES D'IMAGES 33
4.1 Algorithme de SOM adapté 33
4.1.1 Algorithme de SOM 33
4.1.2 Algorithme de SOM adapté 36
4.1.3 Calcul de la similarité 38
4.2 Ré-arrangement de la visualisation SOM 39
4.2.1 Idée principale 40
4.2.2 Défitions 40
4.2.3 Algorithme de réarrangement 40

5 MODELE DE NAVIGATION DANS DES BASES D'IMAGES 44
5.1 Architecture du modèle 44
5.2 Mises en page 46

6 RESULTATS EXPERIMENTAUX 49
6.1 Bases d'images utilisées 49
6.2 Mesure d'évaluation 52
6.3 Analyse des résultats 53
6.3.1 Expérimentation avec la base Wang 53
6.3.2 Expérimentation avec la base Att-faces 54
6.3.3 Expérimentation avec la base COIL-100 56
6.3.4 Expérimentation avec la base Brodatz32 59

7 CONCLUSION ET PERSPECTIVES 61

Table des figures

2.1 Quatre logiciels marquants offrent l'organisation automatique 14
2.2 Trois exemples notables qui utilisent les méthodes de réduction de dimension 23

3.1 Exemple de calcul de la caractéristique de couleur 26
3.2 Exemple de calcul d'une matrice de cooccurrences 27
3.3 Exemple de division d'une image en cinq parties 30
3.4 Exemple de l'interface de deux extracteurs 32

4.1 Exemple de visualisation en utilisant l'algorithme SOM 36
4.2 L'interface de paramètres de l'algorithme SOM 37
4.3 Un exemple de l'appariement bipartie entre deux vecteurs de texture de type 1 39
4.4 Exemple de la résultat de l'algorithme de ré-arrangement 42

5.1 Architecture générale du système y compris trois modules principaux . . . 44
5.2 Le processus principal du système 45
5.3 Trois types de la visualisation de la base d'images 46
5.4 La fonctionnalité « View Image » 47

6.1 Exemple des quatre bases d'images utilisées 50
6.2 Capture d'écran de l'expérience avec la base Wang 53
6.3 Capture d'écran de l'expérience avec la base Att-faces 55

6.4 Exemple d'images de visages qui donnent de mauvais résultats 55

6.5 Capture d'écran de l'expérience avec la base COIL-100 57

6.6 Exemple des objets sont groupés ensemble 58

6.7 Capture d'écran de l'expérience avec la base Brodatz32 59

Liste des tableaux

6.1 Le taux de recherche dans la base de Wang 54
6.2 Le taux de recherche dans la base de Att-faces 56
6.3 Le taux de recherche dans la base de COIL-100 58

Chapitre 1

INTRODUCTION

Le contenu visuel sous forme numérique joue un rôle de plus en plus important dans la télécommunication actuelle et dans notre vie quotidienne à la société informatique moderne. Grâce au développement des technologies multimédias et à la disponibilité des appareils de capture photos numériques à un prix raisonnable, tels que l'appareil photo, le scanner, le téléphone portable multimédia et le puissant ordinateur personnel, la taille des collections d'images s'accroît extraordinairement. Ces collections s'étendent des images privés, comme les photos de vacances, les photos familiales, aux images professionnelles, telles que les images de produits fabriqués par un groupe industriel, les stocks de photos d'agences de presse, les archives du musée et les photos scientifiques en médecine, en astronomie ou en biologie. Ainsi, les grandes collections d'images devenaient courantes dans de nombreaux domaines. Pour accéder à ces collections, il est donc nécessaire à développer des outils efficaces qui permettent aux utilisateurs de gérer, naviguer et rechercher des collections d'images.

Dans les logiciels de navigation d'images traditionnels, le moyen le plus commun pour présenter un ensemble d'images est l'utilisation d'une grille en deux dimensions des miniatures (*thumbnail* en anglais) [1]. La recherche des images pertinentes est basée sur les informations originales comme le nom d'image, la date, etc. et également sur les informations assignées manuellement, telles que les mot-clés et les descriptions [2]. Évidemment, la recherche exhaustive d'images constitue un gaspillage de temps, surtout dans les grandes bases d'images. Par ailleurs, l'ordre dans lequel les images sont affichées est basé sur les attributs déjà cités ci-dessus qui ne reflètent souvent pas le contenu de l'image réelle.

Récemment, certaines approches ont été introduites pour fournir une interface plus intuitive pour naviguer et explorer des bases d'images [3, 4, 5] en organisant les images sur base de leurs similarités mutuelles. Ces approches utilisent des algorithmes d'apprentissage pour créer la visualisation sur base de similarités. Les positions des images qui sont similaires, selon le calcul des métriques de similarité d'images basées sur des caractéristiques provenant du contenu [6], sont aussi proches les unes des autres dans l'espace de visualisation. Ce principe diminue le temps nécessaire pour localiser des images [1]. Ces algorithmes sont les méthodes dites de projection (linéaire ou non-linéaire), appelées également « réduction de dimension » ou « manifold learning ». En particulier, trois techniques utilisées couramment pour l'établissement de cette projection sont l'Analyse en Composantes Principales (PCA – *Principal Component Analysis* en anglais) [7, 8], la mise à l'échelle multi-dimensionnelle (MDS – *Multi-Dimensional Scaling* en anglais) [9, 10, 11] et les cartes auto-organisatrices de Kohonen (SOM – *Self-Organizing Map* en anglais) [12, 13, 14].

Dans ces approches, toutes les images dans la base sont affichées simultanément. Leurs positions sont dépendantes de leurs similarités par rapport à toutes les autres images dans la base. L'utilisateur peut parcourir facilement la base d'images selon une approche de haut en bas d'une manière intuitive. Il peut obtenir l'aperçu et la structure de la base, c'est-à-dire les relations entre les images.

Pourtant, ces approches possèdent quelques inconvénients. La PCA ne peut pas toujours préserver correctement les relations originales dans l'espace de grande dimension. Un inconvénient de l'approche MDS est sa complexité de calcul [3, 4, 5]. Un autre inconvénient de toutes les approches, est que beaucoup d'images sont occluses tandis que d'autres se chevauchent les unes les autres. Cela conduit à une expérience de navigation moins intuitive [3]. La plupart des systèmes existants ne prennent pas en compte ce problème à l'exception de [11, 15], mais leur traitement du problème est plutôt ad-hoc, c'est-à-dire conçu pour régler un problème particulier, ou ne garantit pas la similarité entre les images. En outre, le dernier inconvénient, c'est que ces systèmes sont construits pour traiter seulement quelques caractéristiques. L'utilisateur ne peut pas choisir la combinaison de ces caractéristiques et ne peut pas préciser le rôle de chaque caractéristique dans la mesure de la similarité.

Les limitations ci-dessus sont les challenges que nous souhaitons résoudre. Le but du travail est de développer un outil de visualisation de bases d'images et de navigation interactive dans ces bases. Il s'agit d'un travail d'implémentation informatique avec des aspects d'intégration de méthodes d'apprentissage et d'expérimentation sur des bases d'images.

Les contributions principales de ce stage sont :

- Une bibliographie sur les navigateurs dans les images, la recherche d'images par le contenu, les navigateurs d'images par le contenu et les techniques de projection développées et utilisées dans les navigateurs d'images par le contenu.
- Le développement et la représentation de notre outil glMiSOM, y compris la description en détail de l'utilisation des cartes auto-organisatrices de Kohonen (SOM) comme une méthode de visualisation des images sur une grille en deux dimensions où les images similaires sont positionnées de façons proche et une nouvelle méthode d'arrangement qui permet de ré-arranger la grille des images de résultats d'algorithme SOM pour résoudre le problème de chevauchement. Le modèle proposé fournit également les avantages suivants :
  - Il supporte l'utilisation des caractéristiques multiples, tant visuelles que non visuelles.
  - Il supporte la combinaison arbitraire des caractéristiques et la détermination des rôles de chaque caractéristique dans la mesure de la similarité.
  - Il possède l'architecture des modules d'extension (*plug-in* en anglais), permettant l'ajout ou la suppression facile des caractéristiques.
  - Il contient un outil de visualisation fourni par l'algorithme SOM en utilisant la librairie graphique QOpenGL [16], ce qui facilite la navigation sur la base d'images.
- Un ensemble de résultats expérimentaux avec trois caractéristiques au bas niveau de la sémantique de l'image (l'histogramme de couleur, des caractéristiques de texture et la forme), quatre bases d'images et des paramètres pour démontrer la validité de notre approche.

Hormis le chapitre 1 d'introduction et le chapitre 7 de conclusion, ce rapport est organisé en cinq parties majeures. La première (chapitre 2) est consacrée à l'état de l'art,

au niveau des méthodes de réduction de dimensions, au niveau de la recherche d'images par le contenu et au niveau des navigateurs d'images par le contenu. La seconde partie (chapitre 3) est orientée vers le domaine « Vision par ordinateur » pour traiter l'extraction préalable des caractéristiques. La troisième partie (chapitre 4) présente l'utilisation de l'algorithme des cartes auto-organisatrices de Kohonen pour créer la visualisation de la base d'images et notre nouvelle méthode de ré-arrangement. Dans la quatrième partie (chapitre 5), on y retrouve le détail de notre modèle de navigation dans des bases d'images. Et enfin, la cinquième et dernière partie (chapitre 6) présente les résultats obtenus avec le modèle lorsqu'il est soumis à de vraies bases d'images.

Chapitre 2

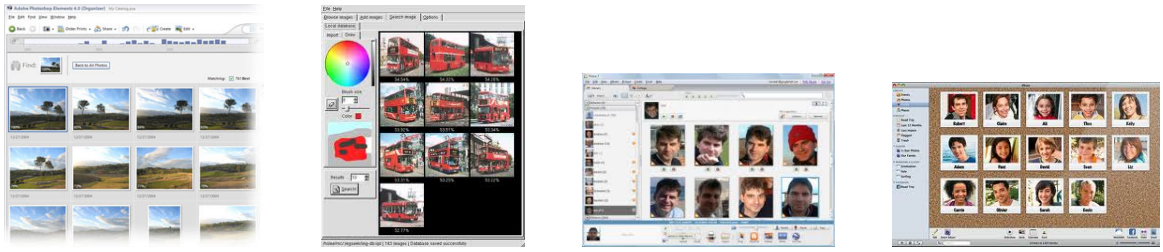
ETAT DE L'ART

Dans ce chapitre, nous verrons d'abord une synthèse des logiciels de gestion et de navigation d'images dans le monde réel. Ensuite, nous allons étudier les travaux qui ont été publiés dans le domaine des navigateurs d'images par le contenu. Dans cette partie, nous allons également étudier quelques méthodes de réduction de dimension qui ont été utilisées comme les méthodes de visualisation de bases d'images par similarités. En ce qui concerne le domaine « Vision par ordinateur », nous utilisons au sein de ce stage exclusivement des algorithmes classiques d'extraction de caractéristiques sur les images. Nous n'aborderons donc pas cet aspect dans cet état de l'art.

2.1 Outils de gestion et de navigation d'images

Depuis des années 1980, lorsque le premier appareil photo numérique a été introduit, des logiciels de gestion et de navigation d'images ont été disponibles. Actuellement, il est difficile de compter le nombre de ces logiciels. On peut les trouver depuis des versisons traditionnelles comme Windows Explorer sur le système d'exploitation Windows ou Preview sur Mac OS X, jusqu'à des versions professionnelles telles que ACDSee, Adobe Photoshop Lightroom, Adobe Photoshop Elements, Aperture, iPhoto, F-Spot, imgSeek, Picasa, etc. [17]. Ces logiciels sont performants pour la gestion, l'organisation et le traitement d'images, mais ils possèdent encore des limitations, spécialement dans la navigation d'images.

En organisation d'images, ces logiciels offrent couramment soit l'une, soit l'autre, soit les deux options suivantes :



(a) (b) (c) (d)

FIGURE 2.1 – Quatre logiciels marquants offrent l'organisation automatique à base des informations visuelles. (a) Adobe Photoshop Elements, (b) imgSeek, (c) Picasa et (d) iPhoto

- *Organisation manuelle d'images* : Ce type d'organisation fournit une vue directe des dossiers d'images sur le disque dur. Elle permet seulement à l'utilisateur de voir les photos et ne fournit aucune caractéristique d'organisation automatique. Elle donne une flexibilité maximale pour l'utilisateur et affiche exactement ce que l'utilisateur a créé sur son disque dur. Dès lors l'efficacité de l'organisation manuelle dépend de la propre méthode de l'utilisateur pour organiser ses photos. Actuellement, il existe deux méthodes principales pour organiser manuellement des images : la méthode d'annotation (ou étiquette – *tag* en anglais) et la méthode sur base de dossiers.
- *Organisation automatique d'images* : Le logiciel analyse les données dans les images numériques et utilise le résultat pour créer automatiquement une structure d'organisation. Il y a deux niveaux d'information extraits : les informations non-visuelles et visuelles.
  - Les informations non-visuelles sont le temps, l'emplacement et l'événement où les photos ont été prises. Ces informations peuvent être recueillies grâce à l'annotation manuelle ou des techniques d'annotation semi-automatique, telles que les systèmes de positionnement géographique (GPS – *Geographic Positioning Systems* en anglais) et les enregistreurs de date intégrés dans les appareils photo. La plupart des logiciels de gestion d'images actuels soutiennent ce type d'information.
  - Les informations visuelles sont les caractéristiques extraites du contenu visuel des images, comme cela est étudiées dans le domaine de la recherche d'images

par le contenu (CBIR – *Content-based Image Retrieval* en anglais). Actuellement, il y a seulement certains logiciels qui soutiennent ce type d'information, tels que Adobe Photoshop Elements, imgSeek, iPhoto, Picasa. Parmi eux, Adobe Photoshop Elements et imgSeek extraient les caractéristiques visuelles des images (couleur, forme, texture, luminosité, objets et régions) pour chercher plus exactement des images similaires visuelles, mais dans ces deux logiciels, les images pertinents sont organisées par la similarité sous forme d'une grille en deux dimensions. iPhoto et Picasa permettent de reconnaître les visages pour organiser les photos selon les personnes, mais les techniques sont encore semi-automatiques.

Dans notre système, nous utiliserons ces deux catégories. Avec l'organisation manuelle, malgré son faible comme l'exigence de l'effort des utilisateurs dans l'organisation des images, nous souhaitons fournir une comparaison entre l'organisation manuelle et l'organisation sur base de similarités. Du côté de l'organisation automatique, nous ajouterons une flexibilité donnée à l'utilisateur dans le choix des caractéristiques visuelles, pour résulter en une organisation semi-automatique.

En visualisation de bases d'images, une façon répandue utilisée par tous les logiciels actuels est d'offrir une interface sous forme d'une grille en deux dimensions des miniatures des images. La forme d'une grille est familière pour l'utilisateur des ordinateurs depuis que le système d'exploitation avec l'interface graphique est né. Selon Judith Gelernet [18], les gens donnent du sens à l'affichage d'informations en appliquant des connaissances existantes à de nouvelles informations. Des visualisations devraient apparaître familières à l'utilisateur ou des visualisations devraient rappeler un objet déjà vu ou une interface utilisée auparavant. Donc, avec une telle organisation et selon une telle structure, une visualisation d'une grille en deux dimensions est familière pour l'utilisateur. C'est pourquoi dans la partie suivante, nous étudierons seulement des méthodes de visualisation qui peuvent fournir facilement une grille en deux dimensions d'images.

En ce qui concerne la navigation dans une base d'images, tous les logiciels actuels fournissent également les techniques traditionnelles, telles que le zoom, le panoramique, la visualisation en plein écran et en taille originale. Avec les grandes bases d'images, ces logiciels fournissent une interface déroulante permettant à l'utilisateur de voir un nombre

limité des miniatures d'images sur l'écran à tout moment. L'utilisateur doit naviguer à travers des pages de miniatures pour voir toutes les images. Dans notre système, nous soutiendrons ces techniques de navigation.

2.2 Navigation d'images par le contenu

La navigation d'images par le contenu (SBIB - *Similarity-based image browsing* en anglais) est un nouveau aspect de recherche lorsqu'il existe encore des limitations et le décalage sémantique dans les techniques de recherche d'images par le contenu [3, 4], mais les techniques de ce nouveau concept héritent essentiellement du domaine de de la recherche d'images par le contenu. Dans cette partie, nous allons donc monter les problèmes clés dans la recherche d'images par le contenu, puis nous étudierons les techniques dans la navigation d'images par le contenu.

2.2.1 Recherche d'images par le contenu

La recherche d'images par le contenu (CBIR - *Content-based image retrieval* en anglais) est maintenant un domaine de recherche très actif [6]. Un certain nombre de systèmes de recherche d'images très connus ont été développés au cours des dernières années, notamment Query by Image Content (QBIC) de la société IBM, Photobook de MIT et WebSeek développé à l'université de Columbia. Dans ces systèmes, les images sont généralement caractérisées par des attributs connus comme des caractéristiques, de simple bas niveau, telles que la couleur et la texture, au plus complexe plus haut niveau, tel que la forme.

Selon [12], on peut classifier les tâches de recherche de CBIR en trois catégories :

- *La recherche par le but* (*target search* en anglais) dans laquelle l'utilisateur essaie de trouver une image spécifique qui peut ou ne peut pas être réellement présente dans la base d'images et qui est la seule image pertinente pour cette requête. Par exemple, l'utilisateur recherche une image de peinture déjà vue auparavant, mais ne sait pas le nom de l'artiste ou celui de peinture. Généralement, ce type de recherche est essentiellement utilisée pour la recherche d'une image connue spécifique dans les catalogues, les bases de marques ou logos, etc.

- *La recherche par la catégorie* (*category search* en anglais) dans laquelle l'utilisateur recherche des images appartenant à une certaine catégorie ou classe d'images. Toutes les images remplissant les critères de la catégorie sont considérées comme pertinentes. Le reste est alors non pertinent par rapport à la requête.
  Les premier et deuxième types utilisent également les requêtes à base d'exemples (*query by example* en anglais) où l'utilisateur doit présenter une image d'exemple (ou une esquisse dessinées à la main) comme une requête visuelle.
- *La recherche ouverte* (*open-ended search* en anglais) ou *navigation* (*browsing* en anglais) dans laquelle l'utilisateur possède d'un objectif de recherche imprécis ou inexact à l'esprit et parcourt la base d'images pour trouver quelque chose d'intéressant. Le but de la recherche peut abruptement changer pendant la session lorsque le système renvoie des images intéressantes mais inattendues. Ce type de recherche est très interactif et constitue souvent une séquence linéaire d'actions, ce qui nécessite une interface flexible.

La plupart des systèmes de recherche d'images par le contenu concentrent seulement sur les deux premiers types de recherche, alors que la navigation dans les images n'a pas reçu autant d'attention de la communauté de recherche. Cependant, bien que de nombreux efforts aient été faits, aucun système est capable d'effectuer la recherche d'images de manière aussi efficace que manuellement à cause du décalage sémantique entre les caractéristiques au bas niveau des images et leur contenu sémantique.

Une nouvelle approche Retour de pertinence (RF - *Relevence Feedback* en anglais) a été proposée pour résoudre ce problème. Dans cette technique, l'utilisateur peut guider le processus de recherche en indiquant via l'interface les images qui sont intéressantes ou non. Les opinions des utilisateurs sont comptées et reflétées dans l'étape suivante du processus de recherche et les résultats sont de plus en plus pertinents. [12] introduit un algorithme TS-SOM (*Tree Structured SOM* en anglais) pour l'implémentation de retour de pertinence. Les techniques sur base de retour de pertinence peuvent généralement améliorer les résultats de recherche d'images en incorporant davantage d'entrées d'utilisateurs, mais exige plus d'attention de l'utilisateur et une certaine compréhension de l'utilisateur sur la façon de choisir les bonnes entrées.

De plus, la plupart des systèmes utilisés pour les requêtes à base d'exemples ren-

contrent un problème. Le problème est que les utilisateurs ne peuvent généralement pas présenter de bons exemples de ce qu'ils recherchent sans navigation efficace sur la base d'images. Ainsi, un outil de navigation efficace est indispensable pour introduire l'utilisateur sur le contenu de la base de données. Après avoir identifié une image d'intérêt dans la base de navigation, l'utilisateur peut effectuer une recherche par le contenu pour trouver les images similaires à partir de la base de données. De telle façon, les systèmes de navigation ont été développés comme une alternative par rapport aux approches à base de la requête.

2.2.2 Visualisation de bases d'images

Contrairement aux systèmes CBIR, qui visent à fournir aux utilisateurs les images souhaitées en se basant sur un ensemble d'images d'entrée, les systèmes SBIB étudient comment organiser les images en se basant sur leurs similarités visuelles [5]. Le défi de la SBIB est d'arranger les images de façon à soutenir la navigation rapide, à fournir un voisinage significatif pour choisir un chemin de navigation et permettre aux utilisateurs de se positionner dans une région d'intérêt.

Récemment, plusieurs approches ont été introduites pour fournir une interface plus intuitive pour naviguer à travers des collections d'images. En général, nous pouvons les diviser en trois catégories [3, 5] :

- *La visualisation sur base de la projection* (*Mapping-based visualizations* en anglais) a pour but de représenter les relations entre les images dans l'espace de caractéristiques de grande dimension, dans un espace de dimension réduite normalement limité à deux dimensions. Alors, l'esprit humain peut plus facilement percevoir les relations entre les images dans la base d'images. Les méthodes couramment utilisées pour cette visualisation sont l'Analyse en Composantes Principales (PCA) [7, 8], la mise à l'échelle multi-dimensionnelle (MDS) [9, 10, 11] et les cartes auto-organisatrices de Kohonen (SOM) [12, 13, 14].
- *La visualisation en clusters* (*Clustered visualizations* en anglais) tente de réduire le nombre d'images qui sont nécessaires pour être affichées à tout moment en regroupant les images similaires. Cette similarité peut être définie par le contenu (c.-à-d. la similarité visuelle) [19], les méta-données (tels que les mot-clés), ou le

temps. L'utilisateur peut naviguer à travers une telle visualisation (souvent organisée hiérarchiquement) en sélectionnant une image représentative d'un groupe afin d'exploiter toutes les images dans ce groupe particulier.

- *La visualisation à base de graphes* (*Graph-based visualizations* en anglais) utilise les liens entre les images pour construire un graphe où les noeuds du graphe sont les images tandis que les côtés relient des images similaires. Les liens peuvent être établis soit par la similarité visuelle [20], soit par les annotations.

Clairement, il est difficile d'identifier les relations entre les images de différentes classes dans les visualisations en clusters et à base de graphes. Ces deux visualisations ne remplissent donc pas notre objectif de stage. Dans la partie suivante, nous étudierons trois méthodes principales utilisées pour les visualisations à base de projections.

A. **Analyse en Composantes Principales**

L'analyse en composantes principales (PCA) est l'approche la plus simple pour la réduction de dimension. C'est une méthode linéaire. Elle calcule d'abord les vecteurs propres de la matrice de covariance symétrique des données de caractéristiques et puis les classe par l'ordre décroissant de leurs valeurs propres respectives. Les composantes principales sont choisies comme étant les vecteurs propres qui correspondent au nombre de dimensions requises (c.-à-d. pour deux dimensions, les deux premiers vecteurs propres sont sélectionnés). Ces vecteurs propres sont ensuite utilisés pour tracer les données originales où les miniatures sont illustrées aux coordonnées dans l'espace de basse dimension dérivé de la projection des données de caractéristiques originales.

Les avantages de la PCA sont l'algorithme relativement simple et l'efficacité de calcul. Cependant, la PCA est une méthode linéaire ; c'est pourquoi seules les corrélations linéaires sont détectées et exploitables. Calculer les valeurs et vecteurs propres avec précision est techniquement relativement lourd, notamment si le nombre d'attributs est très élevé. De plus, le plus grand inconvénient de la PCA est qu'elle maximise la variance des données capturées, mais ne préserve pas nécessairement au mieux les relations mutuelles entre les différents éléments de données.

Le projet Personal Digital Historian (PDH) [8] utilisent des surfaces PCA afin de visualiser des images. Hu *et al.* utilise aussi la visualisation PCA pour présenter des images dans une interface virtuelle en deux dimensions basée sur diverses caractéristiques de

forme [7].

B. **Mise à l'échelle Multi-Dimensionnelle**

La mise à l'échelle multi-dimensionnelle (MDS) est une méthode de transformation non linéaire très classique pour projeter un ensemble de données dans un espace plus petit. Elle a été introduite en psychologie dans les années 1950.

Comme la PCA, elle tente de préserver les relations originales (c.-à-d. les distances) de l'espace de grande dimension le mieux possible dans la projection en basse dimension. Le MDS commence avec une matrice de similarités qui donne la distance entre tout couple de données dans l'espace original de grande dimension. L'objectif du MDS est de projeter ces distances dans un espace planaire en le déformant le moins possible. Pour cela, on définit une fonction dite « fonction de stress » qui a une forme du genre [21] :

$$STRESS = \frac{\sum_{i,j} \left(\hat{\delta}_{ij} - \delta_{ij}\right)^2}{\sum_{i,j} \delta_{ij}^2} \tag{2.1}$$

dans laquelle δ_{ij} est la distance initiale entre les objets i et j, et $\hat{\delta}_{ij}$ est la distance dans l'espace de basse dimension. Le démarrage est effectué soit sous forme d'une configuration initiale aléatoire, soit à partir des coordonnées obtenues par la PCA. L'algorithme continue à repositionner les images afin de réduire la fonction de stress jusqu'à ce qu'une des conditions de terminaison ait été atteint, par exemple un nombre maximum d'itérations ou un seuil de stress.

Dans [10], Rubner propose une technique d'affichage en deux dimensions basée sur le MDS. En se basant sur des caractéristiques de couleur et la distance Earth Mover's Distance (EMD), les auteurs ont réussi à créer une représentation de l'espace de caractéristiques de grande dimension en utilisant le MDS, en plaçant des miniatures à des coordonnées. Rodden *et al.* ont étudié l'utilisation du MDS pour la visualisation de bases d'images en se basant sur l'évaluation d'études d'utilisateurs [1, 2, 11]. Dans [11] ils comparent deux approches, une approche basée sur la disposition aléatoire d'images et une autre approche utilisant une interface MDS sur base de similarités. Les résultats montrent que le système sur base de MDS est plus rapide pour localiser des images spécifiques.

Le MDS fournit une représentation plus précise des relations entre les images dans

l'espace des caractéristiques par rapport à la PCA. Cependant, le MDS a un coût de calcul très important en raison de sa dépendance quadratique. Tout d'abord, une matrice de distances complète pour toutes les données dans la base doit être calculée. MDS est lui-même alors un processus itératif qui réarrange en permanence les positions de chaque image en minimisant les distances (euclidiennes) entre les images à l'écran et leurs distances réelles (sur base des caractéristiques) dans la base de données. La visualisation interactive d'un grand nombre d'images est donc difficile à atteindre. Des difficultés supplémentaires surgissent lorsqu'on ajoute des images à la base visualisée par MDS, car cela nécessite généralement le re-calcul de toutes les données et la re-localisation des miniatures dans la visualisation. Un autre inconvénient que MDS partage avec les visualisations à base de PCA, est que beaucoup d'images sont occluses tandis que d'autres se chevauchent les unes les autres. Cela conduit à une expérience de navigation moins intuitive [3]. Nous croyons que ces inconvénients font du MDS une option peu attrayante pour la navigation en temps réel et la visualisation des données de grande dimension, telles que des images.

C. **Cartes auto-organisatrice de Kohonen**

Les cartes auto-organisatrices de Kohonen (SOM) [22] sont un réseau de neurones bi-couche entièrement connecté utilisant l'apprentissage avec réglage en avant (*feedforward learning* en anglais). Les SOMs sont principalement destinées pour le groupement des données de grande dimension. La couche d'entrée est interprétée comme un vecteur de données. La couche de sortie est interprétée comme une carte bi-dimensionnelle des clusters. Les clusters de la carte de sortie peuvent avoir une forme rectangulaire ou hexagonale. Chaque cluster de la carte de sortie est décrit par un vecteur de poids pointant vers son centre. Dans l'apprentissage et l'application, les vecteurs de données d'entrée sont mappés au vecteur de poids avec la distance euclidienne minimale (l'unité gagnante - *best matching unit* en anglais). L'apprentissage du SOM est basé sur une taille de carte prédéfinie et les vecteurs d'entrée choisis au hasard. La règle d'apprentissage est généralement définie comme suit :

$$w_j(n+1) = w_j(n) + \eta(n)\, h_{ji(x)}(n)\,(x(n) - w_j(n)) \tag{2.2}$$

dans laquelle $w_j(n)$ est le vecteur de poids du noeud j, $\eta(n)$ est le taux d'apprentissage et $h_{ji(x)}(n)$ est la fonction de voisinage. Cette règle est utilisée pour mettre à jour les

vecteurs de poids autour de l'unité gagnante.

Une innovation majeure des SOMs par rapport aux autres méthodes de groupement est l'introduction de la fonction de voisinage. Ces fonctions bi-dimensionnelles font que non seulement l'unité gagnante est adaptée, mais également que les vecteurs de poids voisins soient adaptés. Ainsi, l'apprentissage SOM signifie l'apprentissage du groupement des voisinages. Un noyau de voisinage typique est la fonction gaussienne bi-dimensionnelle.

La première utilisation de SOM pour la visualisation de bases d'images est le système PicSOM [12]. PicSOM utilise les couches de SOM en parallèle pour former une hiérarchie, en particulier une carte auto-organisatrice structurée en arborescence (TS-SOM – *tree structured self-organizing map* en anglais). Avec cette structure, la complexité de recherche de l'unité gagnante est réduite de $O(n)$ à $O(log(n))$. Après l'apprentissage complété et mis en oeuvre sur chacun des niveaux de TS-SOM, chaque noeud est assigné à l'image la plus similaire dans la base de données. Cela conduit à ce que des images similaires se retrouvent rapprochées sur la carte en deux dimensions. Ces images représentatives peuvent ensuite être explorées de manière hiérarchique.

Deng et al. [14] a également fait l'apprentissage en utilisant l'algorithme HSOM (*Hierarchical SOM* en anglais) pour visualiser une collection de 3000 images. Cet algorithme est une variation de l'algorithme SOM utilisant des cartes hiérarchiques sous forme d'une structure pyramidale pour la recherche rapide des unités gagnantes. De même, Eidenberger [13] décrit un système dans lequel le HSOM des photos de vidéo sont créés en fonction d'une variété des descripteurs MPEG-7.

L'avantage le plus significatif des SOMs est sa capacité d'effectuer la quantification vectorielle et la mise de l'échelle multidimensionnelle simultanément. Utiliser les noyaux de voisinage rend idoine les structures de cluster légèrement « naturelles » par rapport à la perception de similarité des humains. De plus, les SOMs possèdent les caractéristiques suivantes :

- *Capacité de bonne visualisation* – les noeuds du réseau sont situés sur un treillis de basse dimension qui rend faciles la visualisation et l'interprétation. Cela rend également traçable la navigation de l'utilisateur dans l'espace de caractéristiques de grande dimension sur une carte de basse dimension.

- *Préservation de la topologie* – les entrées similaires sont mappées sur le même noeud ou des noeuds voisins sur la carte. Cela signifie que des images similaires peuvent être mappées de façon proches sur la carte, rendant également la navigation plus facile et robuste.

Ces avantages sont les raisons majeures pour laquelle nous utilisons les SOMs pour la visualisation de bases d'images.

(a) (b) (c)

FIGURE 2.2 – Trois exemples notables qui utilisent trois méthodes de réduction de dimension, (a) une carte PCA pour 18 images de voiture et 19 images de fleur (extrait de [8]), (b) une carte MDS en deux dimensions de 500 images (extrait de [10]) et (c) Un TS-SOM de trois niveaux (extrait de [12]).

Pour terminer cette partie, nous aborderons le problème commun aux visualisations sur base de la projection, à savoir le chevauchement. Dans l'espace des caractéristiques de grande dimension, les images peuvent se chevaucher ou même être totalement occluses, ce qui rend certaines images cachées à l'utilisateur. Pour résoudre ce problème, [11] tente d'ajuster la visualisation MDS à une grille régulière, [15] vise à conserver davantage la structure locale en déplaçant de matière minimale les images de leurs emplacements donnés, basé sur une fonction de coût décrivant la quantité de chevauchement par rapport à la préservation de la structure. Nous partageons l'idée de [15] pour conserver la structure locale en déplaçant des images chevauchées, mais notre méthode proposée est compatible avec la carte SOM et part du principe de la répulsion entre les images.

2.2.3 Navigation dans la base d'images

La visualisation d'un ensemble de données est utile pour obtenir un aperçu de la collection d'images. Cependant, le véritable potentiel de la navigation dans des bases d'images est seulement atteint grâce à l'interaction réelle de l'utilisateur avec le système. Pour cela, l'utilisateur doit avoir la possibilité de naviguer activement dans la base d'images. [4] a présenté deux types de navigation suivants :

- *Navigation horizontale* (*horizontal browsing* en anglais) – c'est la navigation au sein d'un seul surface plane des images visualisées. Ce type de navigation est souvent utile quand une base d'images a été visualisée grâce à un schéma à base de la projection ou grâce à un graphe, ou lorsque la visualisation concerne un seul cluster d'images. Certains outils ont été développés afin de soutenir cette expérience de navigation, tels que panoramique (*panning* en anglais), zoom, mise à l'échelle (*scaling* en anglais) et agrandissement (*magnification* en anglais).
- *Navigation verticale* (*vertical browsing* en anglais) – c'est la navigation sur plusieurs surfaces planes d'images visualisées. Dans les approches de visualisation qui sont basées sur une structure hiérarchique, les images contenues peuvent également être naviguées en utilisant des méthodes de navigation verticale. Les clusters d'images sont généralement visualisés grâce à l'utilisation des images représentatives. Ces images sont cruciales pour la navigation verticale.

Notre visualisation proposée de la base d'images est une visualisation sur base de projection. Donc, nous soutiendrons la navigation horizontale avec trois outils de navigation : panoramique, mise à l'échelle et agrandissement.

Chapitre 3

CARACTERISATION DES IMAGES

L'indexation d'images et l'extraction des caractéristiques sont généralement effectuées hors ligne dans la plupart des systèmes de navigation d'image par le contenu. Au contraire, dans notre application, ces étapes sont effectuées en ligne selon les besoins des utilisateurs.

Dans cette étape, la caractérisation des images, nous allons tout d'abord représenter les images par des vecteurs de caractéristiques ou des vecteurs de descripteurs (*Feature Vectors* en anglais). Nous utilisons des descripteurs simples et discriminants à base de valeurs continues ou discrètes. Nous allons voir comment calculer les trois caractéristiques visuelles communes. Nous choisissons d'utiliser trois caractéristiques, la couleur, la texture et la forme, pour leur bon compromis entre l'efficacité et la facilité d'implémentation. Les données sont alors stockées dans des fichiers et sont chargées au besoin lors des indexations.

L'un des objectifs de ce travail est de soutenir la flexibilité dans le choix des caractéristiques hétérogènes, la combinaison des caractéristiques, la suppression et l'ajout facile des caractéristiques. En réponse à cette demande, nous soutenons l'architecture des modules d'extension (*plug-in* en anglais) pour que l'utilisateur puisse développer et ajouter des nouvelles caractéristiques.

3.1 Caractéristique de couleur

La couleur est la caractéristique simple pour tous les types d'images en couleurs. L'oeil humain est beaucoup plus sensible aux nuances de couleur qu'aux niveaux de gris des intensités dans l'image. Pour représenter cette caractéristique de couleur, nous utilisons l'histogramme de couleur sur l'espace de couleur RGB. L'histogramme de couleur décrit la distribution des différentes couleurs dans une image en manière simple et efficace à calculer. Le principe de base est de compter pour chaque canal individuellement le nombre de pixels appartenant à des intervalles d'intensités définis. Le schéma ci-dessous montre un exemple simple de calcul de la caractéristique de couleur sur une image.

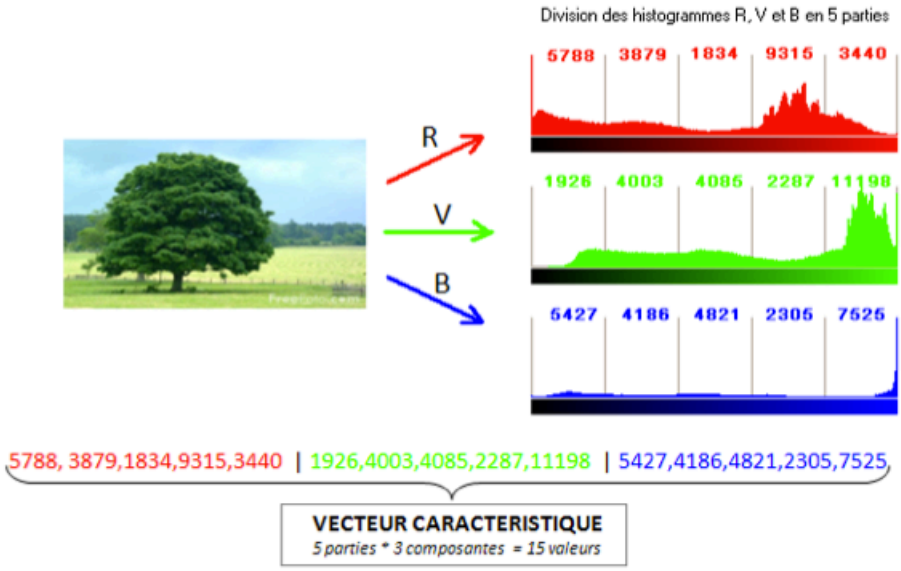


FIGURE 3.1 – Exemple de calcul de la caractéristique de couleur (5 parties) [23]

Avant de calculer l'histogramme, il est nécessaire de réduire le nombre de valeurs pour chaque canal de couleur de l'image. Pour chaque pixel, nous allons réduire la valeur de l'intensité selon la formule suivante :

$$I'_p = \left[\frac{I_p * M}{256} \right] \tag{3.1}$$

dans laquelle [ ] est l'opérateur « partie entière », I_p est l'intensité du pixel p et M est le nombre de valeurs (par couleur) à conserver. M est également un paramètre du logiciel.

Ensuite, nous allons calculer l'histogramme de couleur pour chaque image. Nous obtenons trois histogrammes pour les trois canaux de couleurs rouge, verte et bleue. En combinant les trois histogrammes, nous allons obtenir un seul histogramme de taille $3*M$.

3.2 Caractéristique de texture

La texture est une propriété innée de toutes les surfaces se référant aux motifs visuels qui ne résultent pas de la présence d'une seule couleur ou d'une intensité. Les humains distinguent souvent les textures avec des caractéristiques telles que la périodicité, la directivité, la granularité et le hasard. Il existe plusieurs méthodes statistiques pour calculer les caractéristiques de texture. Nous choisissons d'utiliser les matrices de cooccurrences qui est l'une des plus simples à mettre en oeuvre.

Les matrices de cooccurrences se calculent à partir d'images en niveau de gris. Une matrice représente de façon quantitative la texture de l'image pour une direction donnée et un pas de distance donné. Une matrice ne suffit pas à caractériser totalement la texture, nous calculons au moins huit matrices de cooccurrences correspondantes à deux distances 1 et 2 avec quatre directions 0°, 45°, 90° et 135°. Le schéma ci-dessous montre via un exemple comment calculer ces matrices.

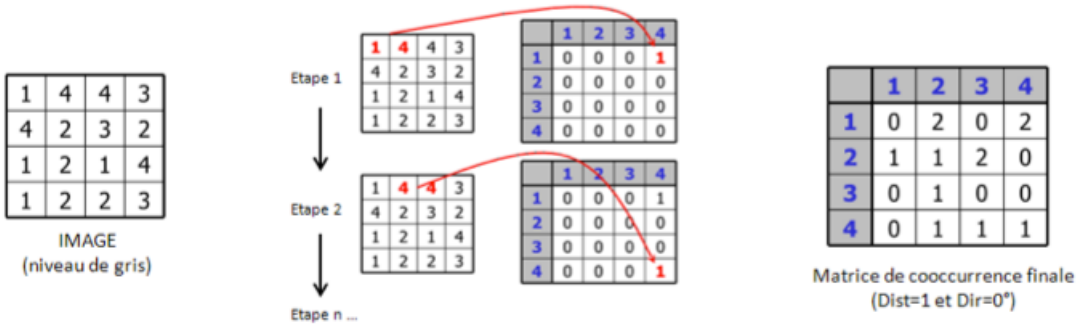


FIGURE 3.2 – Exemple de calcul d'une matrice de cooccurrences (dist=1 et dir=0°) [24]

Une fois les matrices de cooccurrences calculées, on cherche à extraire des valeurs permettant d'évaluer leurs similarités. Il existe 14 valeurs statistiques remarquables pour caractériser ces matrices [25]. Nous utiliserons ces 14 valeurs dans notre système, mais dans l'étape d'expérimentale, nous utiliserons seulement les quatre valeurs les plus répandues : l'énergie, l'entropie, le contraste et le moment :

$$ENERGY = \sqrt{\sum_{i,j=0}^{N-1} P(i,j)^2} \tag{3.2}$$

$$ENTROPIE = \sum_{i,j=0}^{N-1} P(i,j)\,(-lnP(i,j)) \tag{3.3}$$

$$CONTRASTE = \sum_{i,j=0}^{N-1} P(i,j)\,(i-j)^2 \tag{3.4}$$

$$MOMENT\ DIFFERENTIEL\ INVERSE = \sum_{i,j=0}^{N-1} \frac{P(i,j)}{1+(i-j)^2} \tag{3.5}$$

dans laquelle i, j sont les coordonnées dans la matrice et $P(i,j)$ est la valeur normalisée par le nombre d'éléments (paires de pixels) de la matrice.

Supposons que l'utilisateur choisisse n valeurs statistiques dans les 14 valeurs ($1 \leqslant n \leqslant 14$) pour chaque matrice de cooccurrence, le vecteur de caractéristiques de texture peut être construit selon deux types :

- *Type 1* : Nous combinons 8 sous-vecteurs de n caractéristiques pour 8 matrices de cooccurrences. Alors, nous allons avoir un vecteur de $n * 8$ caractéristiques.
- *Type 2* : Nous faisons la moyenne de chaque valeur pour toutes les matrices. Alors, nous allons avoir un vecteur de n caractéristiques seulement.

Nous donnons le type 2 car la mesure de similarité du type 1 (voir la partie 4.1.3) nécessite beaucoup de temps de calcul. Ceci n'est pas compatible avec le système dont tous les calculs sont effectués en ligne. De plus, le type 2 a été utilisé dans [23] et a obtenu les bons résultats expérimentaux.

3.3 Caractéristique de forme

La forme d'un objet situé dans l'espace est une description géométrique de la partie de cet espace occupé par l'objet en faisant abstraction de l'emplacement et l'orientation dans l'espace, de la taille et des autres propriétés comme la couleur, le contenu et la composition des matériaux. Les formes simples peuvent être décrites par des objets élémentaires géométriques, tels qu'un ensemble de deux ou plusieurs points, une ligne, une courbe, un plan, une figure plane (carré ou cercle), ou une figure solide (cube, sphère).

Les moments invariants ont d'abord été présentés à la communauté de reconnaissance des formes en 1962 par Hu [26], qui a employé les résultats de la théorie des invariants algébriques et a proposé ses sept invariants célèbres pour la rotation des objets en deux dimensions. Nous les adoptons pour représenter les caractéristiques de forme.

Avant de calculer les moments de Hu pour chaque image, l'image est transformée en niveau de gris. Ensuite, nous allons trouver des contours de chaque objet dans les images en utilisant le détecteur de contours Canny. C'est l'opérateur de détection de contours qui utilise un algorithme en plusieurs étapes. Tout d'abord, l'image est lissée par une convolution gaussienne. Puis un opérateur de la première dérivée en deux dimensions est appliqué à l'image lissée pour mettre en évidence les régions de l'image ayant les hautes premières dérivées spatiales. Les contours deviendrons les crêtes dans l'image de magnitude du gradient. L'algorithme suit le long du sommet des crêtes et fixe à zéro tous les pixels qui ne sont pas réellement sur les sommets de la crête afin d'obtenir une fine ligne. C'est un processus connu sous le nom la suppression non-maximale (*non-maximum suppression* en anglais). Dans la dernière étape, l'algorithme utilise la hystérésis contrôlée par deux seuils : $T1$ et $T2$ avec $T1 > T2$. Le suivi commence à un point sur la crête supérieure à $T1$. Puis le suivi continue dans les deux directions à partir de ce point jusqu'à la hauteur de la crête tombe en dessous de $T2$. Cette hystérésis permet de s'assurer que les contours bruyants ne sont pas brisés avec les crêtes ayant des multiples de fragments. Nous avons utilisé le détecteur Canny de la librairie OpenCV dans laquelle $T1$ est égal à 160, $T2$ est égal à 80 et la taille du filtre de la première dérivé est égal à 3.

Après avoir eu l'image de contours, 7 moments de Hu ci-dessous sont calculés pour l'image représentant des contours de l'objet. Nous calculons d'abord les moments centraux en ordre deuxième et troisième utilisant la formule :

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - x_c)^p (y - y_c)^q f(x, y) \, dxdy \tag{3.6}$$

dans laquelle $f(x, y)$ est la fonction d'image (ou l'image) et (x_c, y_c) dénote le centre de

gravité de $f(x, y)$. Ensuite, on calcule sept moments de Hu utilisant les formules :

$$\phi_1 = \mu_{20} + \mu_{02}, \tag{3.7}$$

$$\phi_2 = (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2, \tag{3.8}$$

$$\phi_3 = (\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2, \tag{3.9}$$

$$\phi_4 = (\mu_{30} + \mu_{12})^2 + (\mu_{21} + \mu_{03})^2, \tag{3.10}$$

$$\phi_5 = (\mu_{30} - 3\mu_{12})(\mu_{30} + \mu_{12})\left[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2\right] \tag{3.11}$$

$$+ (3\mu_{21} - \mu_{03})(\mu_{21} + \mu_{03})\left[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2\right], \tag{3.12}$$

$$\phi_6 = (\mu_{20} - \mu_{02})\left[(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2 + 4\mu_{11}(\mu_{30} + \mu_{12})(\mu_{21} + \mu_{03})\right], \tag{3.13}$$

$$\phi_7 = (3\mu_{21} - \mu_{03})(\mu_{30} + \mu_{12})\left[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2\right] \tag{3.14}$$

$$+ (\mu_{30} - 3\mu_{12})(\mu_{21} + \mu_{03})\left[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2\right] \tag{3.15}$$

Finalement, pour chaque image, nous allons avoir un vecteur de 7 caractéristiques.

3.4 Caractéristiques en régions

Dans les collections d'images diverses, le contenu important d'une image se trouve au milieu de cette image. Nous divisons donc une image en cinq parties et calculons les caractéristiques pour ces parties-là. L'utilisateur peut appliquer des poids sur chaque partie pour focaliser sur les régions principales de ces images.

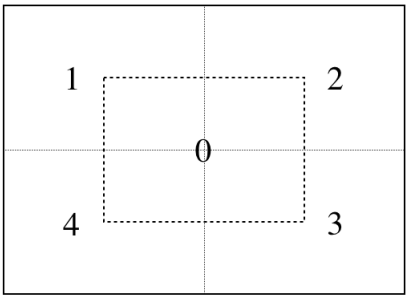


FIGURE 3.3 – Exemple de division d'une image en cinq parties

Cinq parties sont formées comme sur le schéma ci-dessus. Tout d'abord, nous extrayons au centre de l'image une région rectangulaire dont la taille est environ un quart de la superficie de l'image. Puis toute la surface de l'image est divisée en quatre régions.

Quand nous utilisons cette division dans le calcul de trois caractéristiques, la couleur, la texture et la forme déjà cité ci-dessus, le vecteur de caractéristiques comprendra cinq parties : la première partie est le vecteur de caractéristiques (de couleur ou de texture ou de forme) extrait de la zone 0 (voir la figure 3.3), la deuxième partie est le vecteur de caractéristiques extrait de la zone 1, la troisième partie est le vecteur de caractéristiques extrait de la zone 2, la quatrième partie est le vecteur de caractéristiques extrait de la zone 3, et enfin la cinquième partie est le vecteur de caractéristiques extrait de la zone 4. Alors, un vecteur de caractéristiques de couleur aura $3 * M * 5$ valeurs. Semblablement, $n * 8 * 5$ valeurs pour un vecteur de caractéristiques de texture de type 1, $n * 5$ valeurs pour un vecteur de caractéristiques de texture de type 2 et $7 * 5$ valeurs pour un vecteur de caractéristiques de forme.

3.5 Architecture du module d'extensions

Pour soutenir la flexibilité du système, nous avons construit une architecture du module d'extension. En particulier, nous avons défini deux classes d'interface : *ExtractorInterface* et *ExtractorWidgetInterface* où sont définies abstraitement les fonctions principales que les modules d'extension doivent hériter et implémenter. L'utilisateur ayant besoin d'installer un nouveau module d'extension au système, doit hériter les deux classes ci-dessus.

Les classes héritant de la classe *ExtractorWidgetInterface*, définiront l'interface d'utilisateur (*user interface* en anglais) dans lequel l'utilisateur peut identifier des paramètres pour l'extracteur de caractéristiques. *ExtractorWidgetInterface* hérite lui-même d'un objet GroupBox. Donc, l'interface de l'extracteur sera emballée dans un groupe. La figure 3.4 présente deux interfaces de deux extracteurs Color Histogram (présenté dans la section 3.1) et Hu Moments (présenté dans la section 3.3).

Les classes héritant de la classe *ExtractorInterface*, définiront les fonctions principales de l'extracteur. Elles doivent toutes définir 14 fonctions différentes, mais les plus importantes sont les trois fonctions suivantes :

- *extract* : Cette fonction sert à extraire des caractéristiques. Elle reçoit une image que l'on veut extraire et une chaîne des paramètres qui est la chaîne retournée par

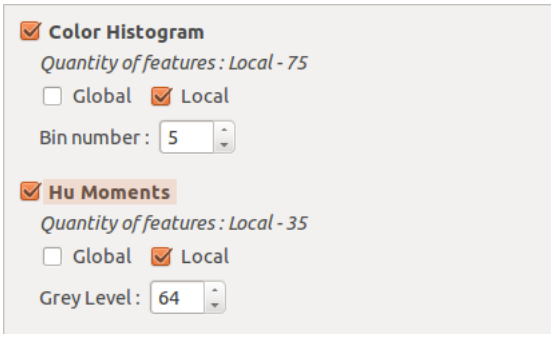


FIGURE 3.4 – Exemple de l'interface de deux extracteurs de caractéristiques de couleur et de forme

la fonction *getParams* de la classe qui hérite la classe *ExtractorWidgetInterface* ci-dessus.

- *calDistanceEuc* : Cette fonction sert à calculer la distance entre deux vecteurs de caractéristiques. Même si son nom possède le suffixe Euc (c.-à-d. euclidienne), l'utilisateur n'est pas obligé d'utiliser la mesure de la distance euclidienne ; il peut utiliser d'autres mesures de similarité.
- *adapt* : Cette fonction très importante est utilisé dans la phase de l'adaptation pour les vecteurs de poids dans l'algorithme SOM (qui sera présenté dans le chapitre 4). Nous permettons à l'utilisateur d'implémenter cette fonction afin que l'algorithme SOM soit adapté aux caractéristiques différentes et aux mesures de similarité différentes. C'est l'élément qui rend réellement flexible ce système.

Alors, selon le type de caractéristiques et l'organisation du vecteur de caractéristiques spécifiques, l'utilisateur peut définir personnellement la chaîne des paramètres, la fonction d'extraction des caractéristiques, la mesure de similarité et la méthode d'adaptation de l'algorithme SOM.

Chapitre 4

VISUALISATION DE BASES D'IMAGES

Dans ce chapitre, nous présenterons l'algorithme SOM comme une méthode de visualisation de la base d'images. En se basent sur l'algorithme SOM original proposé par Kohonen en 1982 [22], nous avons modifié certaines parties de cet algorithme afin de l'adapter avec les objectifs du système (dont un a été introduit dans la section 3.5 du chapitre 3). Notre algorithme SOM adapté permet à l'utilisateur de faire la combinaison arbitraire des caractéristiques et la détermination des rôles de chaque caractéristique dans la mesure de la similarité en utilisant des poids. Enfin, dans la dernière partie de ce chapitre, nous présenterons notre méthode de ré-arrangement pour résoudre le problème de chevauchement de la visualisation de l'algorithme SOM.

4.1 Algorithme de SOM adapté

4.1.1 Algorithme de SOM

Les cartes auto-organisatrices sont constituées d'un treillis (généralement en deux dimensions) ou d'une grille régulière de noeuds. Le type de grille SOM le plus commun est probablement la grille hexagonale, mais un choix plus naturel pour les images est d'utiliser une grille rectangulaire. Donc, nous utilisons la topologie rectangulaire afin d'organiser les images à la taille par défaut de $M * M$ où la valeur M sera calculée automatiquement en fonction du nombre d'images dans la base d'images en utilisant la formule suivante :

$$M = \sqrt{K} \tag{4.1}$$

où K est le nombre d'images dans la base. Ensuite, la valeur M sera augementée pour satisfaire la condition 4.18. Pour comprendre la raison de l'apparition de cette condition, vous voyez la partie 4.2.3.

Un vecteur de poids $w_j \in \mathbb{R}^n$ est associé à chaque noeud j de la carte. La carte vise à représenter toutes les observations disponibles $x \in \mathbb{R}^n$ avec une précision optimale en utilisant les noeuds de la carte comme un ensemble restreint des vecteurs. Pendant la phase d'apprentissage, l'ensemble des vecteurs de caractéristiques est présenté à la carte plusieurs fois en ordre aléatoire et les vecteurs de poids sont modifiés pour correspondre à la distribution et l'ordre typologique de l'espace des vecteurs des caractéristiques. Les SOMs peuvent donc être utilisés pour visualiser des données de grande dimension, généralement sur une grille en deux dimensions.

L'algorithme SOM est basé sur un processus de régression séquentielle, où $t = 0, 1, 2, \ldots t_{max} - 1$ et comprend les cinq étapes suivantes :

1. *Initialisation* : choisir des valeurs aléatoires pour initialiser les vecteurs de poids $w_j(0)$
2. *Échantillonnage* : le vecteur $x(t)$ est choisi au hasard parmi l'ensemble des données d'apprentissage et présenté à la carte.
3. *Appariement de similarité* : chercher l'unité gagnante $i(x)$ à l'itération t en utilisant la distance euclidienne :

$$i(x) = \arg\min_j \parallel x(t) - w_j \parallel,\ j = 1, 2, \ldots l \tag{4.2}$$

   où l est le nombre des neurones.

4. *Mise à jour* : un sous-ensemble des vecteurs de poids constituant un voisinage centré autour de l'unité gagnante $i(x)$ est mis à jour selon la formule suivante :

$$w_j(t+1) = w_j(t) + \eta(t)\, h_{ji(x)}(t)\,(x(t) - w_j(t)) \tag{4.3}$$

5. *Continuation* : répéter à partir de l'étape 2 pour t_{max} itérations.

Dans l'étape 1 de l'algorithme SOM, nous utilisons les composantes principales de l'espace d'entrée pour initialiser linéairement les vecteurs de poids parce que cela réduit également le risque d'avoir des cartes mal formées avec des pliages ; l'apprentissage peut donc être plus rapide.

Dans la formule (4.3), $h_{ji(x)}(t)$ est la fonction de voisinage, une fonction décroissante de la distance entre les noeuds j et $i(x)$ dans la carte. La valeur de $h_{ji(x)}(t)$ diminue en fonction du temps pour garantir la convergence des vecteurs de poids w_j. Les grandes valeurs de la fonction $h_{ji(x)}(t)$ au début de l'apprentissage initialisent la carte et les petites valeurs à la fin de l'apprentissage sont nécessaires à l'ajustement fin (*fine-tuning* en anglais). Nous soutiendrons les deux types suivants pour cette fonction :

- *Bulle* (*bubble* en anglais) : c'est une simple fonction seuil. Soit $N_{i(x)}$ l'ensemble des voisins autour de noeud $i(x)$. Alors, $h_{ji(x)}(t) = 1$ si $j \in N_{i(x)}$ et $h_{ji(x)}(t) = 0$ si $j \notin N_{i(x)}$.
- *Fonction gaussienne :*

$$h_{ji(x)}(t) = \exp\left(-\frac{d_{ji}^2}{2\sigma^2(t)}\right) \tag{4.4}$$

  dans laquelle t est le nombre actuel d'iterations et le paramètre $\sigma(t)$ définit la largeur du noyau gaussienne. C'est une fonction monotone décroissante de la distance.

Dans la formule (4.3), $\eta(t)$ est le taux d'apprentissage. Nous soutiendrons également deux possibilités pour ce paramètre :

- *Taux d'apprentissage linéaire* (comme dans l'algorithme SOM original) :

$$\eta(t) = \eta_0 \frac{t_{max} - t}{t_{max}} \tag{4.5}$$

  dans laquelle η_0 est le taux d'apprentissage initial.

- *Taux d'apprentissage en fonction inverse du temps* (*inverse-time rate* en anglais) :

$$\eta(t) = \eta_0 \frac{C}{C + t} \tag{4.6}$$

  dans laquelle $C = \frac{t_{max}}{A}$, A est une constante ($A = 100.0$ dans notre système). La fonction inverse du temps est justifiée théoriquement par la théorie d'approximation stochastique. Il est conseillé d'utiliser la fonction inverse du temps avec de grandes cartes et de longs apprentissages, pour permettre un ajustement fin des vecteurs de poids.

Après la phase d'apprentissage, toutes les images sont projectées sur la carte SOM, chacune sur son unité gagnante. La figure 4.1 montre un exemple de visualisation en utilisant l'algorithme SOM, extraite de notre système. La propriété de conservation topo-

logique de SOM assure visuellement et conceptuellement que des images similaires sont mappées sur des noeuds qui sont proches les unes des autres et vice versa.

FIGURE 4.1 – Exemple de visualisation en utilisant l'algorithme SOM, extrait de notre système. C'est la visualisation de 100 images de 10 classes de la base Wang sur une carte de $11 * 11$. Les chiffres en bas à gauche de quelques cases représentent le nombre d'images qui sont associées à ces cases.

4.1.2 Algorithme de SOM adapté

Un de trois objectifs de notre travail est la flexibilité du système. Donc, dans le chapitre 3, nous avons premièrement construit une architecture du module d'extension pour permettre à l'utilisateur de développer et ajouter de nouveaux extracteurs de caractéristiques. Deuxièmement, avec l'algorithme SOM, nous permettons à l'utilisateur de combiner arbitrairement des caractéristiques et d'identifier les rôles de chaque caractéristique dans la mesure de similarité en utilisant les poids (p). Les valeurs de poids par

défaut sont automatiquement calculés selon la contribution de chaque caractéristique dans le vecteur de caractéristiques en utilisant la formule suivante :

$$p_m = \frac{N_m}{N} \tag{4.7}$$

dans laquelle p_m est le poids de caractéristique m, N_m est le nombre de valeurs dans le vecteur de caractéristique m et N est le total de valeurs dans le vecteur de caractéristiques. La figure 4.2 montre l'interface de notre système sur laquelle l'utilisateur peut combiner des caractéristiques et ensuite identifier les poids pour chaque caractéristique.

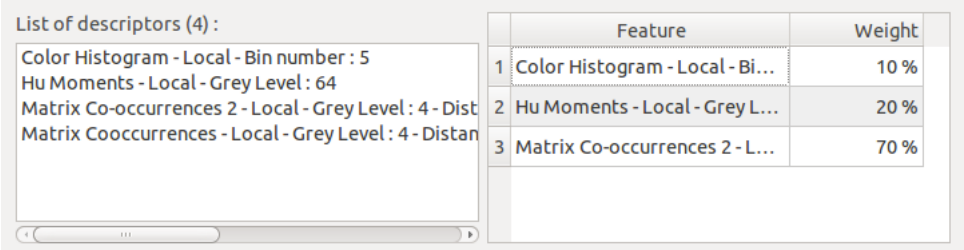


FIGURE 4.2 – L'interface de paramètres de l'algorithme SOM sur laquelle l'utilisateur peut combiner des caractéristiques et ensuite identifier les poids pour chaque caractéristique. Dans cet interface, l'utilisateur peut facilement effectuer des actions de glisser-déposer pour choisir les caractéristiques.

Troisièmement, deux changements sont effectués dans l'algorithme SOM :

- Dans la formule (4.2), la mesure de similarité doit s'adapter avec la combinaison des caractéristiques et les poids de chaque caractéristiques. Donc, nous avons utilisé la formule suivante pour calculer la similarité :

$$D\left(I_i, I_j\right) = \sum_{m=1}^{M} p_m d_m \left(f_i^m, f_j^m\right) \tag{4.8}$$

  dans laquelle I_i, I_j sont deux images i et j, M est le nombre de caractéristiques choisies dans le vecteur de caractéristiques, p_m est le poids pour la caractéristique m, f_i^m est le vecteur de caractéristique m de l'image I_i et f_j^m est le vecteur de caractéristique m de l'image I_j, d_m est la mesure de similarité de la caractéristique m (voir la partie 4.1.3 pour savoir les calculs de similarité utilisés dans ce stage).
- Dans la formule (4.3), nous avons également utilisé l'autre formule suivante :

$$w_j^m\left(t+1\right) = w_j^m\left(t\right) + \eta\left(t\right) h_{ji(x)}\left(t\right)\left(x^m\left(t\right) - w_j^m\left(t\right)\right) \text{ avec } m = 1, \ldots M \tag{4.9}$$

dans laquelle, w^m et x^m sont les vecteurs de caractéristiques m de deux vecteurs w_j et $x(t)$.

Selon notre expérience, aucune librairie disponible de cartes auto-organisatrices ne contient ces changements. Donc, nous avons ré-implémenté l'algorithme SOM avec ces changements en se basant la librairie très connue SOMpak [27].

4.1.3 Calcul de la similarité

Outre l'extraction d'un ensemble approprié de caractéristiques décrivant le contenu des images, nous avons besoin d'une mesure de similarité appropriée entre les images dans la base d'images. Les chercheurs dans la recherche d'images par le contenu ont proposé plusieurs mesures de similarité [12]. Parmi de ces mesures, la distance euclidienne est la plus simple et ne nécessite pas beaucoup de calculs. Par conséquent, elle est compatible avec un algorithme SOM et un système dont tous les calculs sont effectués en ligne. Donc, nous utilisons la distance euclidienne pour les trois caractéristiques couleur, texture de type 2 et forme.

Selon notre expérience, la distance euclidienne ne convient pas pour les vecteurs de caractéristiques de texture parce qu'elle n'est pas invariante à la rotation et à la mise à l'échelle. En effet, la même texture avec des orientations différentes donnera des résultats différents. Donc, l'ordre des matrices de cooccurrences de cette même texture avec des orientations différentes sera différent. La méthode de l'appariement bipartie est une méthode invariante à la rotation et à la mise à l'échelle. Nous avons utilisé cette méthode pour les caractéristiques de texture de type 1.

Pour appliquer l'appariement bipartie à la mesure de similarité entre deux vecteurs de texture de type 1, nous considérons le vecteur de texture de type 1 comme un graphe dont chaque noeud est une matrice de co-occurrence et les coordonnées du noeud sont des valeurs statistiques extraites de cette matrice de co-occurrence. Ainsi la distance entre deux vecteurs de texture de type 1 est égale à la somme de distances euclidiennes des appariements bipartie entre ces deux graphes des matrices de co-occurrence.

La figure 4.3 montre un exemple de l'appariement bipartie entre deux vecteurs de texture de type 1. La texture 2 est une version de rotation de la texture 1. Chaque vecteur de texture est formé par quatre matrices de co-occurrence pour une distance 1

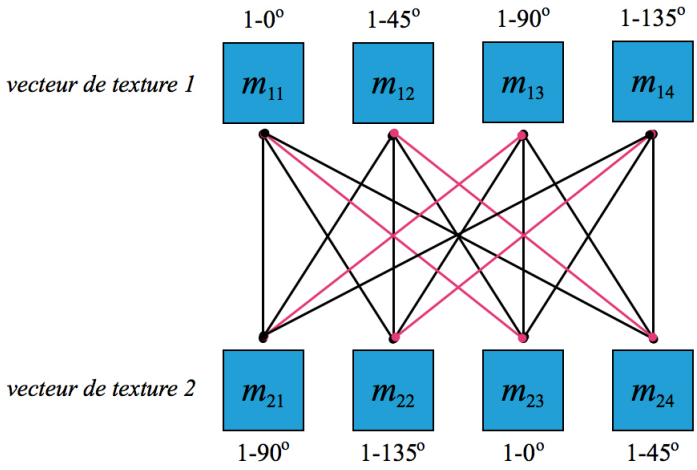


FIGURE 4.3 – Un exemple de l'appariement bipartie entre deux vecteurs de texture de type 1

et quatre directions 0°, 45°, 90° et 135°. Chaque matrice correspond à un noeud dans cette figure. Les lignes représentent les possibilités de l'appariement entre les matrices de deux vecteurs de texture. Les lignes rouges représentent l'appariement bipartie de ces deux vecteurs. Avec le résultat de l'appariement bipartie comme dans la figure (les lignes rouges), la distance entre ces deux textures sera la somme suivante :

$$D = d(m_{11}, m_{23}) + d(m_{12}, m_{24}) + d(m_{13}, m_{21}) + d(m_{14}, m_{22}) \tag{4.10}$$

Enfin, pour trouver l'appariement bipartie, nous utilisons une permutation pour obtenir les paires ayant les plus petites distances. Mais ceci demande beaucoup de temps d'exécution.

4.2 Ré-arrangement de la visualisation SOM

Une fois la visualisation effectuée, on a une carte des images. Puisque l'algorithme SOM est une méthode de clustering, plusieurs images peuvent être associées à une même case, comme dans la figure 4.1. Mais les utilisateurs n'aiment pas le chevauchement et trouvent difficile de parcourir une structure irrégulière [11]. Donc, on a besoin d'un algorithme qui permette de réarranger les images dans la visualisation SOM. Ce réarrangement doit satisfaire aux deux conditions suivantes :

1. chaque case doit contenir une seule image

2. la similarité entre deux images voisines doit être conservée

4.2.1 Idée principale

L'idée de notre algorithme de ré-arrangement est dérivé des bulles d'air. Supposons que nous avons des bulles d'air qui sont côte à côte dans un petit espace. Quand on pompe plus d'air dans une bulle étant au centre, cette bulle va gonfler et empiéter l'espace des autres bulles autour de lui, ou l'on peut dire qu'elle pousse les bulles proches. En ce qui concerne les autres bulles qui sont poussées, elles pousseront à leur tour les autres bulles autour d'elles. Semblablement, dans le cas de la grille SOM avec des cases qui sont associées à plusieurs images, des images supplémentaires pousseront les positions des autres images dans les cases voisines vers les autres cases. En poursuivant de cette manière, toutes les images pouront être réorganisés pour obtenir une seule image associée à chaque case.

4.2.2 Défitions

Dans notre algorithme, nous utilisons les deux nouvelles définitions suivantes :

Définition 1 (**La case qui peut devenir stable**). *C'est la case qui est associée au plus grand nombre d'images. Dans cet algorithme, cette case sera examinée en premier et deviendra stable après l'algorithme de réarrangement des images superflues dans cette case. Après le réarrangement, les cases stables ne recevront plus d'images.*

Définition 2 (**Image de priorité**). *C'est l'image dont la case associée est modifiée par l'algorithme. Quand l'algorithme fait le réarrangement dans la nouvelle case de cette image, cette image sera laissée la priorité de devenir l'image à conserver.*

4.2.3 Algorithme de réarrangement

Entrée : une grille de neurones où chaque neurone est rempli par des images

Sortie : une grille de neurones réarrangée où chaque neurone est rempli par une seule image

Soit $N * M$ la taille de la grille de neurones.

Soit K le nombre d'images.

Soit S une matrice de même dimension que la grille de neurones qui sert à gérer l'état de stabilisation des cases de la grille. Donc, si $C_{i,j}$ est une case stable, $S_{i,j} = \textbf{true}$.

Soit p une variable qui représente l'état de priorité d'une image. Si l'image k possède de la priorité, on a $p_k = \textbf{true}$.

L'algorithme possède les étapes suivantes :

1. *Initialisation* :

$$S_{i,j} = \textbf{false}, \qquad \forall i \in [1..N], \forall j \in [1..M] \tag{4.11}$$

$$p_k = \textbf{false}, \qquad \forall k \in [1..K] \tag{4.12}$$

2. *Chercher une case qui peut devenir stable* :

   Soit C_s est la case qui peut devenir stable, elle doit satisfaire la condition suivante :

$$C_s = \arg\max_{i,j} \mid C_{i,j} \mid, \qquad \forall i \in [1..N], \forall j \in [1..M] \text{ et } \mid C_{i,j} \mid > 1 \tag{4.13}$$

   où $\mid C_{i,j} \mid$ est le nombre d'images qui sont associées à la case de ligne i et colonne j. S'il existe plusieurs case répondant à la condition, C_s sera choisi aléatoirement parmi ces cases.

   Si C_s est vide, l'algorithme s'arrêtera.

3. *Choisir l'image à conserver* :

   Soit L est le nombre d'images dans la case C_s , I_i est la i$^{\text{e}}$ image dans C_s , P est l'ensemble des images prioritaires et R est l'ensemble des images sans priorité.

   Si P est vide, l'image à conserver est :

$$I_c = \arg\min_i \parallel x_i - w_s \parallel, \qquad \forall i \in [1..L] \tag{4.14}$$

   où x_i et w_s sont les vecteurs de caractéristiques respectivement de l'image I_i et de la case C_s .

   Si non, l'image à conserver est :

$$I_c = \arg\min_i \parallel x_i - w_s \parallel, \qquad \forall i \in P \tag{4.15}$$

   Les images restantes sont mis dans l'ensemble R :

$$R = R \cap (P \setminus I_c) \tag{4.16}$$

   Mettre la valeur **true** pour la case C_s : $S_{C_s} = \textbf{true}$

4. *Réarranger les images restantes* :

   Pour chaque image dans l'ensemble R , chercher la case la plus proche de cette image. Soit $C_{p,i}$ la case la plus proche de l'image i dans R :

$$C_{p,i} = \arg\min_{j} \parallel x_i - w_j \parallel, \tag{4.17}$$

   $\forall w_j$ des cases voisines non-stable de C_s.

   Si $C_{p,i}$ est vide, l'image i est conservée pour la case C_s.

   Mettre la valeur **true** pour la priorité de l'image i : $p_i =$ **true**

5. *Continuation* : Répéter l'étape 2.

La figure 4.4 montre le résultat du ré-arrangement de la visualisation de la figure 4.1 en utilisant l'algorithme ci-dessus.

FIGURE 4.4 – Exemple de la résultat de l'algorithme de ré-arrangement de la visualisation pour la figure 4.1

Cette méthode est simple, mais elle conserve la plupart de la similarité entre les images les plus proches après la visualisation donnée par l'algorithme SOM. Cependant, cet algorithme possède encore un problème. Dans l'étape 4, quand $C_{p,i}$ est vide, on ne peut pas réarranger l'image i. Dès lors on ne peut pas réarranger complètement la visualisation SOM. Selon nos expériences, nous trouvons quatre moyens pour résoudre ce problème :

1. le rapport entre le nombre d'images et le nombre de noeuds dans la carte SOM doit être plus grand ou égal à $\frac{3}{4}$, comme suivant la condition :

$$\frac{K}{M^2} \geqslant \frac{3}{4} \tag{4.18}$$

   Dans le système, nous avons appliqué ce rapport pour préciser la taille de la carte SOM par défault.

2. effectuer plusieurs fois l'algorithme ré-arrangement
3. accroître le nombre d'itérations d'apprentissage pour atteindre la carte SOM convergente.
4. augmenter la taille de la carte de SOM

Selon nos expériences, augmenter la taille de la carte de SOM est la méthode la plus simple qui peut résoudre rapidement le problème, mais il faut être prudent avec une grande carte SOM parce que l'on doit changer les paramètres d'apprentissage, de plus, la navigation devient difficile.

Chapitre 5

MODELE DE NAVIGATION DANS DES BASES D'IMAGES

Nous avons implémenté un outil de gestion d'images dans lequel l'utilisateur peut naviguer facilement dans la base d'images sur une grille en deux dimensions. En se basant sur la visualisation de la base d'images où les images similaires sont positionnées de façon proches, notre outil est également un outil de recherche d'images par le contenu. Dans ce chapitre, nous présentons notre modèle de navigation dans des bases d'images, y compris la description en détail du modèle, les modules du modèle, le processus principal et les fonctionnalités principales du modèle.

5.1 Architecture du modèle

Le modèle de navigation dans les bases d'images est construit sur trois modules principaux comme dans la figure 5.1. Dans ce modèle, toutes les opérations, tous les

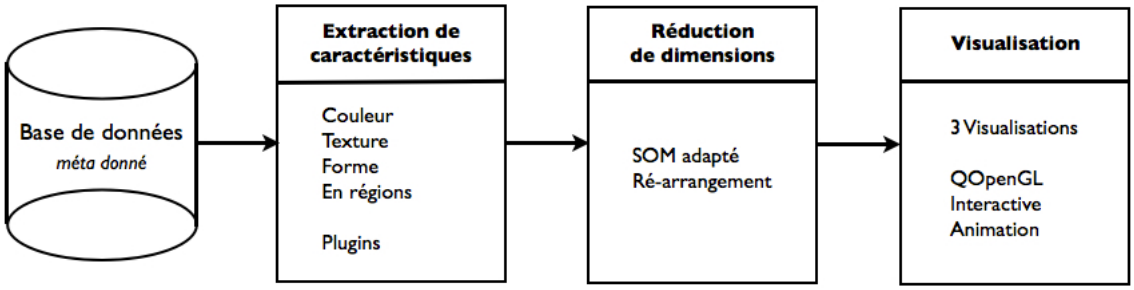


FIGURE 5.1 – Architecture générale du système y compris trois modules principaux

calculs sont effectués en ligne. Tout d'abord, le système gère toutes les bases d'images sur ses répertoires dans le disque dur et stocke les données calculées sous forme des fichiers de méta données. Ensuite, le module Extraction de caractéristiques fournit une architecture des modules d'extension, de même que trois extracteurs de caractéristiques fondamentales. Des fonctionnalités de ce module sont présentées en détail dans le chapitre 3. En se basant sur les caractéristiques extraites du module Extraction de caractéristiques, le deuxième module est la Réduction de dimensions où sont fournis deux algorithmes, un algorithme SOM adapté et un algorithme de réarrangement, afin de créer deux types de visualisation importantes de cet outil. L'utilisation de l'algorithme SOM et le détail de l'algorithme de réarrangement sont présentés dans le chapitre 4. Le dernier module est Visualisation qui fournit trois mises en page (*layout* en anglais) pour trois types de visualisation de la base d'images en utilisant la librairie QOpenGL [16]. En même temps, ce module soutient les interactions élémentaires et également l'animation pour faciliter la navigation dans la base d'images. La figure 5.2 montre le processus principal que l'utilisateur effectuera pour interagir avec le système.

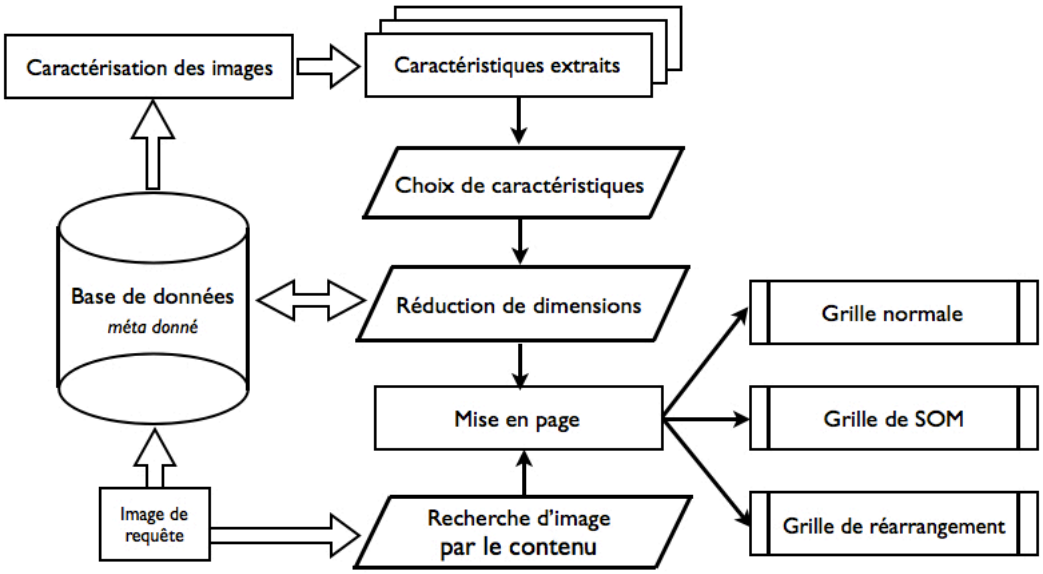


FIGURE 5.2 – Le processus principal du système

Finalement, le système fournit les fonctionnalités suivantes :

- *Outil de gestion d'images traditionnel* – fournit des fonctionnalités telles que la visualisation de la base d'images dans l'ordre des images sur le disque dur (l'ordre des noms de fichiers), voir les informations différentes de l'image, voir en plein

écran, voir l'image dans sa taille originale, importer les nouvelles images, enregistrer et charger les visualisations déjà créées.

- *Outil de gestion d'images semi-automatique* – possède des fonctionnalités telles que extraire des caractéristiques, faire la réduction de dimension, obtenir deux visualisations des images similaires sur une grille en deux dimensions, gérer la navigation (l'animation, le zoom, le paranomique, la mise à l'échelle et l'agrandissement), gérer les modules d'extension.
- *Outil de recherche d'image par le contenu* – l'utilisateur donne une image d'exemple et le système recherchera l'unité gagnante pour cette image sur la grille SOM selon les caractéristiques déjà choisies (utilisées pour calculer la grille SOM). Ensuite, la visualisation changera le foyer (*focus* en anglais) vers cette unité gagnante.

Deux modules Extraction de caractéristiques et Réduction de dimensions ont déjà été présentés dans deux chapitres précédents. Dans la suite, nous présenterons le module Visualisation.

5.2 Mises en page

(a) (b) (c)

FIGURE 5.3 – Trois types de la visualisation de la base d'images, (a) visualisation normale (b) visualisation SOM et (c) visualisation après réarrangement

Le modèle fournit trois types de visualisation de la base d'images :

- *Visualisation normale* (la figure 5.3 (a)) – créée en se basant sur l'ordre des noms des images sur le disque dur
- *Visualisation SOM* (la figure 5.3 (b)) – créée en se basant sur la grille SOM.

Pour les cases où sont associées plusieurs images, l'utilisateur peut voir toutes les images dans ces cases en utilisant la fenêtre « View Image » (la figure 5.4). Il s'agit de la même fonctionnalité pour voir l'image en taille originale, mais l'utilisateur l'utilise également pour voir ces cases.

- *Visualisation après réarrangement* (la figure 5.3 (c)) – créée en se basant sur le réarrangement de la visualisation SOM.

Avec ces trois types, l'utilisateur peut naviguer facilement, faire des comparaisons et effectuer rapidement des recherches.

FIGURE 5.4 – La fonctionnalité « View Image ». On a l'indice de l'image courante et le nombre d'images dans cette case sur le titre de la fenêtre. L'utilisateur peut changer l'image courante en cliquant, tenant le bouton de la souris et bougant la souris vers la gauche ou vers le droite.

Tous les types de visualisation utilisent une même mise en page (*layout* en anglais), une grille en deux dimensions dont la taille est calculée par la formule (4.1) et la condition (4.18). Les images dans chaque case dans la grille sont les miniatures.

Au début du travail, nous avons utilisé la librairie graphique de deux dimensions pour visualiser les grilles en deux dimensions. Nous avons implémenté un processus léger pour charger rapidement des images et créer des miniatures. Le résultat de la visualisation des miniatures, l'animation, le zoom et le panoramique sont très bons avec les petites bases d'images. Cependant, avec les grandes bases d'images, le chargement des images et l'animation sont trop lents et le zoom et le panoramique sont trop difficiles. Après un mois d'expérimentation sur la librairie graphique de deux dimensions, il ressort que cette librairie ne correspond pas aux besoins d'un outil de navigation d'images, notamment

en termes de performance et en termes d'interaction avec l'utilisateur. Cette période d'expérimentation a été très bénéfique pour le projet ; elle a permis de valider la faisabilité du système et d'en apercevoir son efficacité. Finalement, nous avons décidé d'utiliser la librairie OpenGL, en particulier la librairie QOpenGL de la plateforme Qt. Cette librairie vise à créer des graphiques de trois dimensions. Notre outil en a bénéficié. Le zoom, le panoramique et la navigation atteignent la meilleure qualité, le chargement des images est plus rapide et l'animation bouge de façon plus souple.

Chapitre 6

RESULTATS EXPERIMENTAUX

Dans les chapitres précédents, nous avons étudié le modèle de façon théorique. Nous allons maintenant mettre en pratique le modèle présenté avec de vraies images afin de vérifier l'intérêt de la navigation par similarités.

Tout d'abord, nous présenterons quatre bases d'images utilisées et notre mesure d'évaluation des résultats. Ensuite, nous allons étudier la performance du modèle de façon quantitative en interprétant visuellement les résultats et de même effectuerons une analyse quantitative via des statistiques.

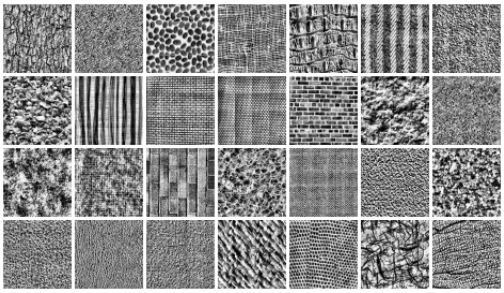
6.1 Bases d'images utilisées

Actuellement, notre système soutient trois types de caractéristiques : la couleur, la texture et la forme. Nous avons choisi quatre bases d'images qui sont créées pour tester ces trois caractéristiques. Ces quatre bases d'images qui serviront par la suite aux expérimentations sont les suivantes :

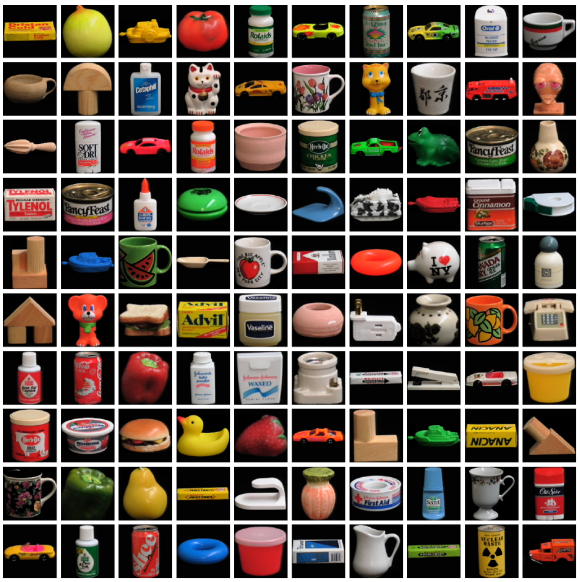
- *La base WANG* : 10 classes de 100 images, soit 1000 images. Certaines classes sont composées d'images visuellement très proches, que se soit au niveau de la couleur ou de la texture (ex : dinosaures ou fleurs), d'autres ont un lien plus sémantique (ex : nourriture). La figure 6.1(a) montre une image de chaque classe de la base Wang.
- *La base Att-faces* : un ensemble d'images de visages du laboratoire AT&T Laboratories Cambridge, qui contient dix images de 40 personnes. Pour chaque personne, les images ont été prises à des moments différents et il y a des variations de l'éclairage, d'expressions faciales (les yeux ouverts/fermés, souriant/pas souriant) et des

(a)dix classes de la base Wang

(b)100 images de visage de 10 personnes de la base Att-faces



(c)28 exemples de texture de la base Brodatz32



(d)100 objets de la base COIL-100

FIGURE 6.1 – Exemple des quatre bases d'images utilisées.

détails du visage (lunettes/ sans lunettes). Toutes les images ont été prises sur un fond sombre homogène avec les visages dans une posture frontale debout (avec une tolérance pour certains mouvements vers les deux côtés). La figure 6.1(b) montre un exemple de dix personnes de la base Att-faces.

- *La base COIL* : une librairie des images de l'Université de Columbia. C'est une base de 7200 images en couleur de 100 objets (72 images par objet). Les objets ont une grande variété de caractéristiques géométriques et réfléchissantes complexes. Nous avons seulement testé 2000 images de 100 objets. Pour chaque objet, nous avons choisi 20 poses, correspondant aux 20 degrés suivants : 0°, 15°, 20°, 35°, 40°, 60°, 80°, 100°, 120°, 140°, 160°, 180°, 200°, 220°, 240°, 260°, 280°, 300°, 320°, 340°. La figure 6.1(d) montre un exemple de 100 objets de la base COIL-100.
- *La base Brodatz32* : une base d'image pour la classification de textures composée de 32 textures de l'album Brodatz. C'est une sous-base d'images de la base d'images de textures provenant de l'Université de Oulu en Finlande. Cette base comprend 64 images per classe (donc 2048 images au total), qui comprennent les sous-ensembles d'images suivants : 16 images d'origine, 16 versions après rotation des images d'origine, 16 versions réduites des images d'origine et 16 versions après rotation et réduction des images d'origine. En effet, nous avons choisi 4 images par classe (128 images au total), y compris une image d'origine, une version après rotation, une version réduite et une version après rotation et réduction. La figure 6.1(c) montre un exemple de 28 textures de la base Brodatz32.

6.2 Mesure d'évaluation

Jusqu'à présent il n'existe aucune bonne mesure pour évaluer la performance des navigateurs d'images par le contenu. Des mesures qualitatives et quantitatives sont nécessaires. L'analyse qualitative est purement visuelle ce qui la rend moins précise et subjective. En revanche, dans notre cas, elle est importante car le modèle mélange l'interface, la navigation et les traitements en un tout.

En évaluation quantitative, la précision et le rappel qui sont les mesures les plus connues et les plus largement utilisées dans la recherche d'images par le contenu, ne sont pas adaptées à une visualisation des images par similarités sur une grille en deux dimensions. Dans la recherche d'images par le contenu, on utilise la similarité (c.-à-d. la distance) pour donner des images pertinentes lors que dans la navigation d'images par le contenu, on utilise la position des images pour donner des images pertinentes. Donc, nous proposons une nouvelle mesure pour évaluer de façon quantitative le système.

Notre système crée une visualisation sur base de similarités, où les positions des images similaires sont proches les unes des autres dans l'espace de visualisation. Dans ce système, le point qu'il est nécessaire d'évaluer est le nombre d'images de même classe que l'utilisateur peut « chercher » à partir d'une image quelconque. Nous définissons le concept « chercher » comme suit : l'utilisateur commence avec une image quelconque de départ sur la grille des images et cherche des images de même classe dans les huit cases voisines autour de l'image de départ. Ensuite, à partir des images de même classe recherchées, l'utilisateur continue à chercher selon la méthode ci-dessus. Le processus de rechercher s'arrête lorsque l'utilisateur ne trouve aucune nouvelle image de même classe. Le nombre d'images de même classe trouvées est le résultat montrant l'efficacité du système. Le rapport entre ce nombre et le total d'images de cette classe est le taux de recherche. En effet, les images peuvent être réparties en plusieurs zones, nous choisissons la zone la plus grande pour calculer le taux de recherche.

6.3 Analyse des résultats

6.3.1 Expérimentation avec la base Wang

La figure 6.2 montre la capture d'écran de l'expérience avec la base Wang. C'est le résultat de la combinaison entre trois caractéristiques : couleur, texture de type 2 et forme avec respectivement les poids 10%, 70% et 20%. La visualisation est créée sur une grille de $35 * 35$.

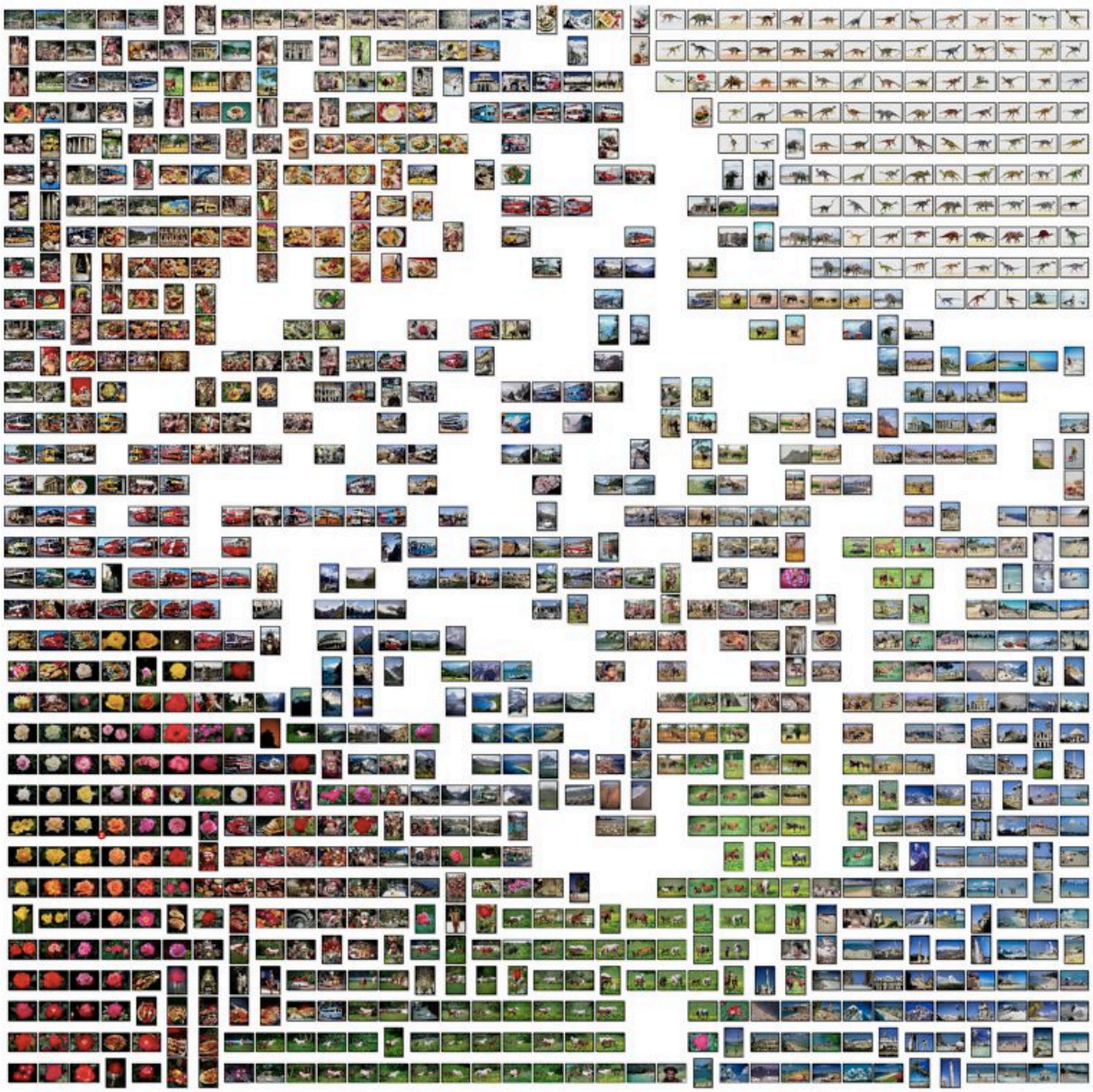


FIGURE 6.2 – Capture d'écran de l'expérience avec la base Wang.

Dans la visualisation, d'une part, les images de même classe sont positionnées dans

des régions spécifiques. Les dinosaures sont en haut à droite. Les fleurs sont en bas à gauche. Les plages et les monuments sont en bas à droite. Les buses, les nourritures et les photos d'afriques sont en haut à gauche. Les chevaux sont en bas au centre. Les éléphants au centre à droite. Les montagnes sont au centre. D'autre part, les positions des classes sont très remarquables. Les fleurs qui sont visuellement très différentes des dinosaures sont aux antipodes de la zone de répartition des images comme le résultat dans [23]. Entre eux, tour à tour, on a les chevaux, les montagnes et les éléphants. De façon similaire, nous possédons deux zones opposées : une zone des plages, des monuments, et une autre zone des nourritures, des afriques Entre les deux zones, on a respectivement les chevaux, les éléphants, les montagnes et les buses. Cette répartition est légèrement dépendante de la couleur, mais notre résultat (voir le tableau 6.1) est meilleur pour certaines classes particulières, telles que les buses, les plages, les montagnes, les nourritures parce que nous avons appliqué des poids importants pour la forme et la texture.

L'efficacité du système pour la navigation par similarités est représentée dans le tableau 6.1. En comparant avec le résultat dans [28], nous trouvons que notre résultat est similaire la méthode de classification et meilleur que la méthode de CBIR traditionnelle. Cela montre qu'un système de navigation par similarités permet à l'utilisateur de rechercher plus efficacement, plus précisément que les systèmes de CBIR traditionnels.

Classe	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Moy.**
SBIB par SOM	46	56	16	74	100	86	89	95	58	54	64,5

TABLE 6.1 – Le taux de recherche dans la base de Wang

6.3.2 Expérimentation avec la base Att-faces

L'expérimentation avec la base Att-faces vise à évaluer l'efficacité dans la reconnaissance de visages, une application qui est maintenant apparue dans quelques outils de gestion d'images. Nous avons combiné deux caractéristiques : la texture de type 2 et la forme avec les poids respectivement 70% et 30%. La visualisation est créée sur une grille de $22 * 22$. La figure 6.3 montre la capture de l'écran de cette visualisation.

Tout d'abord, nous regardons l'influence de la forme sur le résultat de la visualisation. Les images avec des lunettes sont à droite et les autres (sans lunettes) sont à

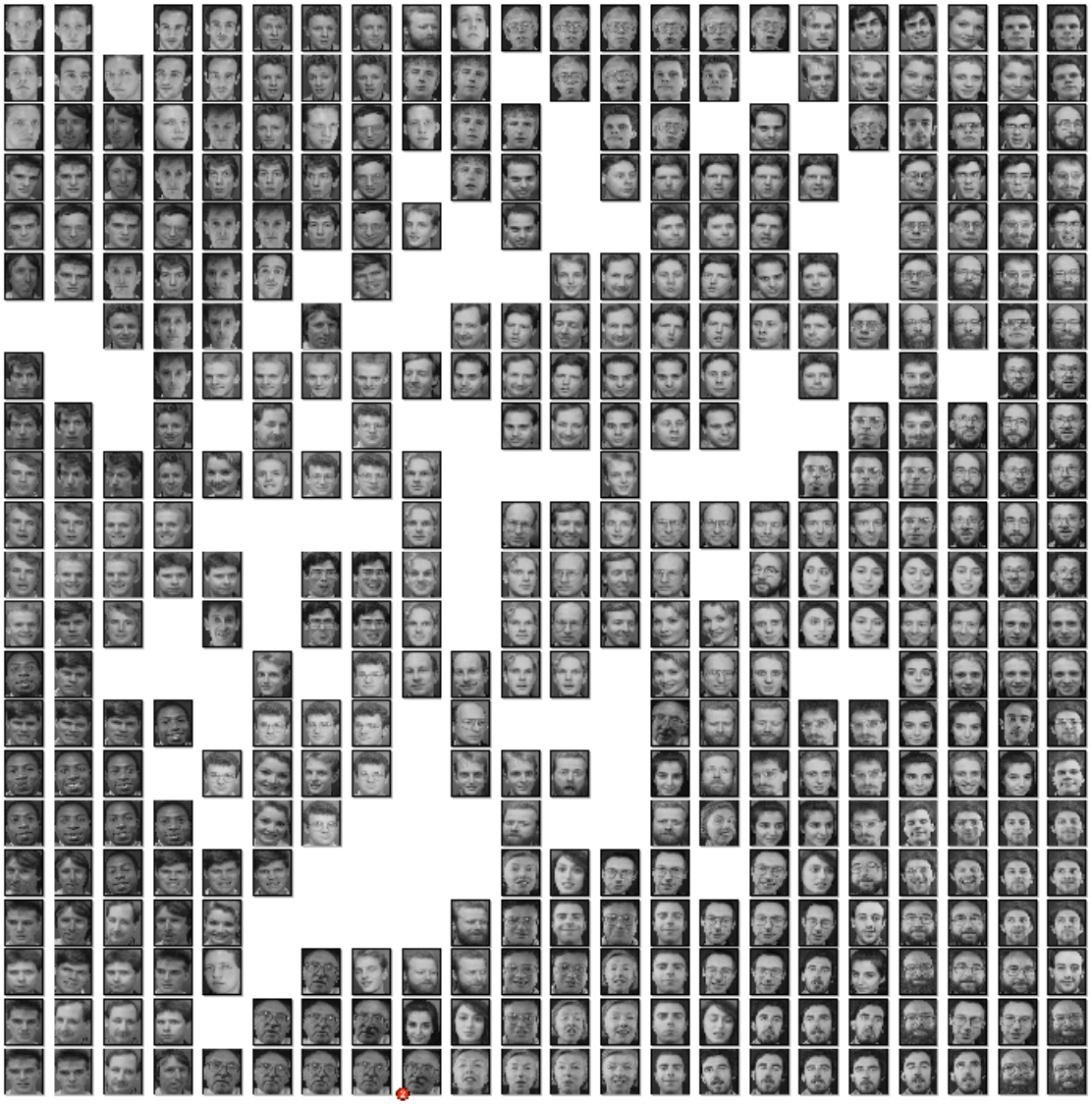


FIGURE 6.3 – Capture d'écran de l'expérience avec la base Att-faces.

(a)s23 (b)s35 (c)s36 (d)s37

FIGURE 6.4 – Exemple d'images de visages qui donnent de mauvais résultats.

gauche. Les images longes sont en haut et les autres (rondes) sont en bas. Ensuite, quantitativement, le taux de recherche est à peine acceptable (voir le tableau 6.2). Les images atteignant de bons résultats sont les images avec peu de variations et les images (voir la figure 6.4) donnent de mauvais résultats. La raison est que dans ces images, il y a des

mouvements spécifiques (la tête baissée ou consternée), porter des lunettes ou non, la tête déformée. C'est également la raison avec laquelle quelques classes de visages sont réparties en deux zones différentes. Mais selon nous, ce résultat reste satisfaisant parce que notre système n'est pas conçu pour la reconnaissance de visages.

Classe	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
SBIB par SOM	50	90	40	50	100	70	90	60	60	50	
	11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	
	100	50	60	100	50	50	50	80	50	40	
	21	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	
	70	90	30	90	50	90	80	50	50	50	
	31	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**Moy.**
	70	90	30	90	30	90	80	50	50	50	60,5

TABLE 6.2 – Le taux de recherche dans la base de Att-faces

6.3.3 Expérimentation avec la base COIL-100

La base COIL-100 est une librairie d'images d'objets destinées à évaluer les caractéristiques de forme. Nous avons donc combiné trois caractéristiques : la forme, la couleur et la texture de type 2 avec des poids dégressifs de 70%, 20% et 10%. La visualisation est créée sur une grille de $47 * 47$. La figure 6.5 montre la capture de l'écran de cette visualisation.

D'une part, bien que la contribution de la couleur soit trop petite (20%), nous trouvons que dans le résultat de cette visualisation, les régions correspondant aux différentes couleurs sont discriminées. La région du blanc est à droite, celle du rouge est à gauche, celle de l'orange est en bas, celle du bleu est en haut à gauche. La région du vert et du jaune entourent la région du rouge. D'autre part, avec les contributions de la forme et de la texture, les classes d'objets se sont groupées correctement. Des objets plats, tels que la voiture, le paquet de chewing-gum, sont à gauche. Des objets ronds ou ayant la tête ronde, tels que les fruits, les boîtes, les tasses, sont en bas. Des objets dont le corps est long et effilé, tels que les boîtes de cola, les boîtes de médicament, sont en haut. De plus, des objets dont la forme est similaire, mais dont la couleur ou les détails sont différents

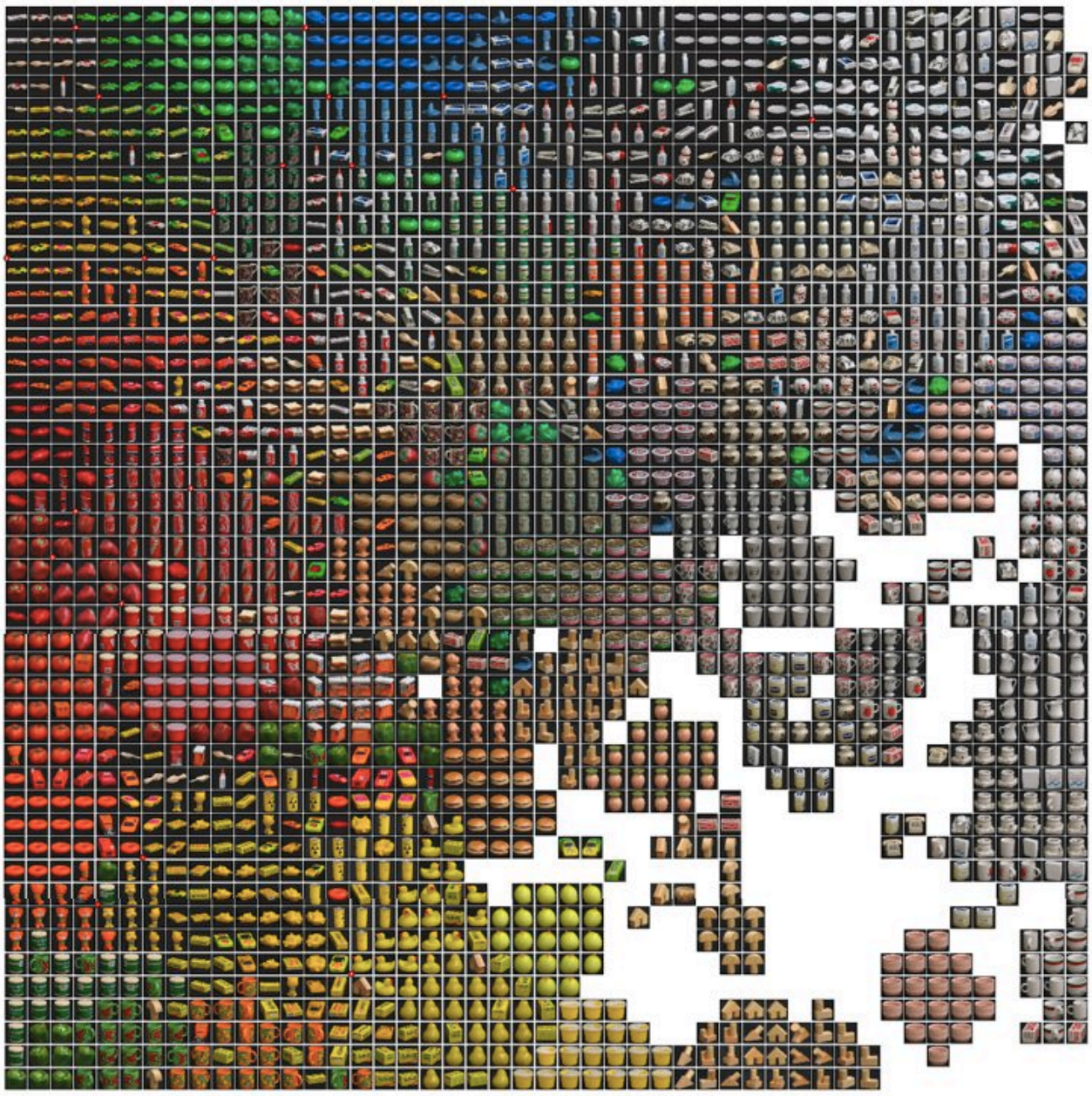


FIGURE 6.5 – Capture d'écran de l'expérience avec la base COIL-100.

(ex : les voitures, les bateaux, les paquets de chewing-gum, les boîtes de cola, le piment), sont groupés ensemble (voir la figure 6.6).

Selon le tableau 6.3 du taux de recherche, nous remarquons que le résultat est assez bon. Les objets qui possèdent de mauvais résultats, sont souvent des objets plats ou complexes, tels que les voitures, les paquets de chewing-gum, les téléphones. Les objets ayant une forme simple obtiennent toujours de bons résultats. Selon nous, le résultat moyen de cette visualisation $(65,4\%)$ est assez bon, mais en continuant à diminuer le

FIGURE 6.6 – Exemple des objets dont la forme est similaire, mais dont la couleur ou les détails sont différents.

C-T	C-T	C-T	C-T	C-T	C-T	C-T	C-T	C-T	C-T
1–35	**11**–85	**21**–40	**31**–25	**41**–85	**51**–55	**61**–45	**71**–60	**81**–55	**91**–50
2–100	**12**–60	**22**–40	**32**–100	**42**–40	**52**–55	**62**–95	**72**–100	**82**–40	**92**–70
3–80	**13**–30	**23**–35	**33**–40	**43**–70	**53**–70	**63**–55	**73**–100	**83**–100	**93**–75
4–95	**14**–35	**24**–95	**34**–20	**44**–5	**54**–70	**64**–75	**74**–100	**84**–30	**94**–90
5–90	**15**–50	**25**–100	**35**–75	**45**–90	**55**–100	**65**–30	**75**–65	**85**–70	**95**–100
6–25	**16**–100	**26**–95	**36**–30	**46**–40	**56**–100	**66**–100	**76**–35	**86**–100	**96**–50
7–100	**17**–50	**27**–40	**37**–40	**47**–85	**57**–50	**67**–25	**77**–55	**87**–100	**97**–90
8–45	**18**–100	**28**–50	**38**–70	**48**–60	**58**–65	**68**–35	**78**–50	**88**–85	**98**–40
9–45	**19**–65	**29**–90	**39**–80	**49**–95	**59**–95	**69**–20	**79**–45	**89**–75	**99**–80
10–40	**20**–100	**30**–95	**40**–55	**50**–100	**60**–20	**70**–100	**80**–35	**90**–85	**100**–35

TABLE 6.3 – Le taux de recherche dans la base de COIL-100. **C** – Classe, **T** – Taux de recherche.

poids des caractéristiques de couleur et en augmentant celui de caractéristiques de forme, le résultat pour les objets complexes sera meilleur.

6.3.4 Expérimentation avec la base Brodatz32

Nous avons fait des expériences avec la base Brodatz32 afin d'évaluer l'algorithme SOM avec les caractéristiques de texture de type 1 qui utilise la méthode de l'appariement bipartie pour calculer la similarité entre deux textures. Nous avons seulement utilisé la caractéristique de texture de type 1 pour le vecteur de caractéristiques. La visualisation est créée sur une grille de $15 * 15$. Le résultat de la visualisation est représenté dans la figure 6.7.

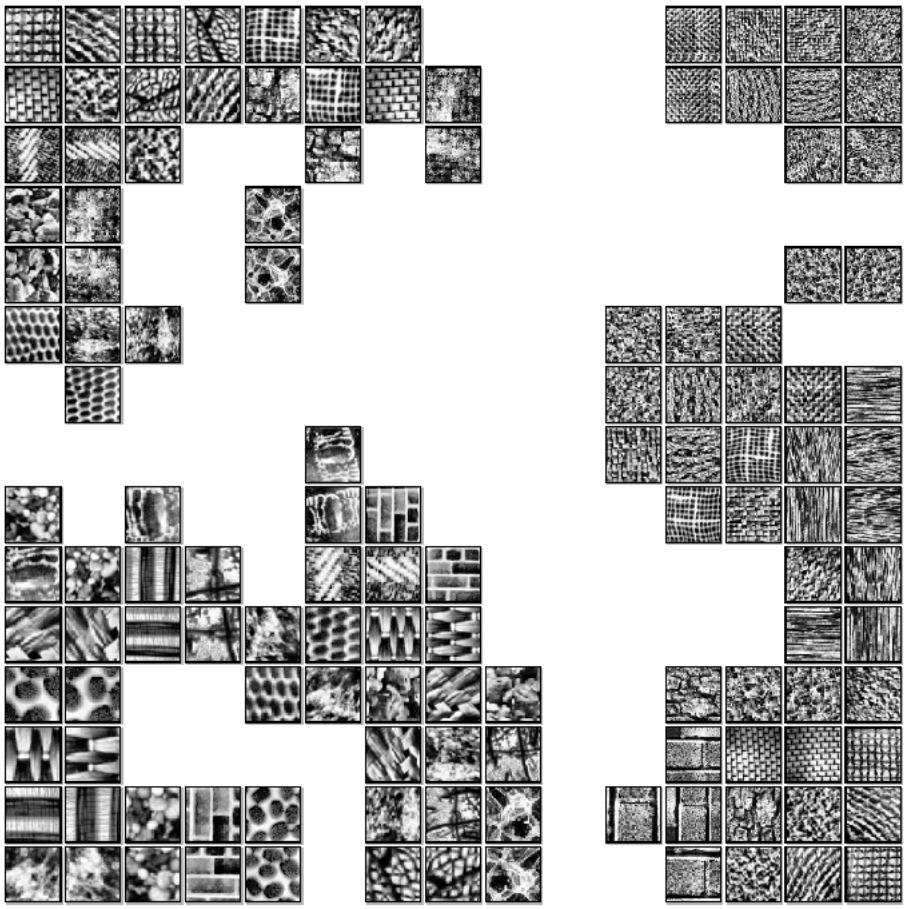


FIGURE 6.7 – Capture d'écran de l'expérience avec la base Brodatz32.

La plupart des textures atteignent le taux de recherche de 50%, à l'exception de la texture de image15 qui est de 100% et de la texture de $d52$ qui est de 25%. Nous obtenons ce résultat parce que la plupart des classes de texture se sont divisées en deux groupes. Un

groupe comprend la texture d'origine et la texture après rotation de la texture d'origine. Un autre groupe comprend la version réduite de la texture d'origine et la version après rotation et réduction de la texture d'origine. On peut dire que la méthode de l'appariement bipartie pour calculer la similarité entre deux textures convient à la rotation et ne convient pas à la mise à l'échelle.

Un autre point intéressant dans le résultat de cette visualisation, c'est que les textures sont réparties en quatre coins. Le coin en haut à droite comprend les textures avec de petits détails, de petits traits, de petits points, telles que l'herbe, le raphia, le sable de la plage, la peau de cochon. À l'opposite, le coin en bas à gauche comprend les textures avec de grands détails, telles que les haricots, le reptile. Le coin en haut à gauche comprend le mélange de textures de petits traits et de détails moyens, telles que les arbres, la toile à sac. En dernier lieu, le coin en bas à droite, c'est le mélange entre des textures de petits traits et des textures de moyens traits, telles que le grain de bois, l'eau.

Chapitre 7

CONCLUSION ET PERSPECTIVES

Ce mémoire présente un système de navigation dans des bases d'images en se basant sur la similarité entre des images. Notre système utilise l'algorithme des cartes auto-organisatrices de Kohonen comme une méthode de visualisation des images sur une grille en deux dimensions où les images similaires sont positionnées de façon proches. Cet algorithme est simple mais efficace, il a fournit à notre système de bonnes caractéristiques, telles que la capacité de bonne visualisation et la préservation de la topologie. Nous avons implémenté cet algorithme avec quelques modifications pour l'adapter aux objectifs de notre système comme la possibilité de combinaison arbitraire des caractéristiques et la détermination des rôles de chaque caractéristique dans la mesure de la similarité. L'architecture des modules d'extension que nous avons proposée a également pour but de soutenir cette flexibilité du système. Cela permet à ce système de devenir un outil pour tester la combinaison des différentes caractéristiques. Même si, jusqu'à maintenant, nous fournissons et testons seulement trois caractéristiques simples et discriminantes, la couleur, la texture et la forme, les résultats expérimentaux démontrent l'efficacité de ces trois caractéristiques, de même notre visualisation. Par ailleurs, nous avons proposé une nouvelle méthode de ré-arrangement pour résoudre le problème de chevauchement dans la grille des images de résultat de l'algorithme SOM. Cet algorithme vient de l'idée simple de répulsion des bulles d'air et réussit à conserver correctement la similarité entre deux images voisines sur les cartes auto-organisatrices. À partir de ces résultats, la visualisation de notre système a satisfait trois exigences générales pour un système de navigation des images : l'aperçu, la visibilité et la préservation de la structure.

L'organisation et la visualisation des images sur un treillis en deux dimensions sont deux problématiques récurrentes au niveau des systèmes de navigation dans des images. Notre système fusionne l'organisation manuelle et l'organisation semi-automatique. Dans ce système, l'utilisateur possède donc trois visualisations de la base d'images : la visualisation normale, la visualisation de la grille SOM et la visualisation après ré-arrangement. Ceci rend faciles la navigation, la comparaison et la recherche d'images. Notre visualisation a profité de la puissance de la librairie OpenGL afin de créer des visualisations avec une bonne qualité d'images, une navigation facile et un zoom, une animation de façon plus souple.

Les résultats obtenus ont montré qu'il était possible d'appliquer la similarité entre des images pour faciliter la navigation et la recherche des images dans les outils de gestion ou de navigation des images. Les résultats expérimentaux ont également montré qu'un système de navigation d'images peut surmonter les faiblesses d'un système de CBIR traditionnel.

Pourtant, le système comporte encore des limites. Premièrement, le système utilise l'algorithme SOM. Le temps de calcul de cet algorithme dépend du nombre d'itérations. Pour atteindre une carte convergente, le nombre d'itérations est très grand avec les grandes cartes. Deuxièmement, la limitation du système apparaît lorqu'il faut ajouter ou supprimer des images dans la base d'images. Le système doit recalculer la carte SOM. Troisièmement, la liberté dans la combinaison des caractéristiques rend le système flexible, mais le système demande une connaissance de l'utilisateur sur ces caractéristiques. Ceci peut réduire les possibilités d'utilisation de ce système dans certaines applications.

Nos perspectives visent à résoudre ces limites. Premièrement, dans l'algorithme SOM, la recherche de l'unité gagnante est effectuée de façon séquentielle à travers toutes les unités de la carte SOM. Avec une carte SOM de grande dimension, ceci fait accroître le temps de calcul du système. On peut alors utiliser des cartes hiérarchiques [14]. La recherche commencera à la racine, ensuite descend au niveau suivant pour localiser l'unité gagnante, et ainsi de suite, jusqu'à ce que l'unité gagnante soit trouvée au niveau de la feuille.

Deuxièmement, on a besoin d'une mesure généique afin tester la convergence de la carte SOM après un petit nombre d'itérations.

Troisièmement, pour augmenter la performance du système dans le groupement des images similaires, il faut extraire des meilleures caractéristiques, telles que MPEG-7 ou les caractéristiques extrayant des régions d'intérêts comme les régions invariantes (ou les sacs de mots - *bag of word* en anglais).

Quatrièmement, il est nécessaire d'effectuer un choix intelligent des caractéristiques. Cela signifie que le système doit trouver automatiquement les caractéristiques appropriées. L'utilisateur donne seulement une certaine description pour la base d'images, comme le type d'images, le type du fond, la position du contenu important, la gamme principale de couleurs, etc. Le système choisit automatiquement les caractéristiques pertinentes pour l'extraction et l'indexation dans les étapes suivantes.

Pour terminer, à partir des résultats obtenus, un système semblable pour des bases des vidéos est possible. Dans ces systèmes, l'utilisateur pourra chercher des vidéos similaires en utilisant deux ou plusieurs cartes SOM.

Références

[1] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood, "Does organisation by similarity assist image browsing ?" *Proceedings of the SIGCHI conference on Human factors in computing systems CHI 01*, no. 3, pp. 190–197, 2001.

[2] K. Rodden and K. R. Wood, "How do people manage their digital photographs ?" *Proceedings of the conference on Human factors in computing systems CHI 03*, pp. 409–416, 2003.

[3] W. Plant and G. Schaefer, "Visualising Image Database," *IEEE International Workshop on Multimedia Signal Processing*, pp. 1–6, 2009.

[4] ——, "Navigation and Browsing of Image Databases," *International Conference of Soft Computing and Pattern Recognition*, pp. 750–755, 2009.

[5] D. Heesch, "A survey of browsing models for content based image retrieval," *Multimedia Tools and Applications*, vol. 40, no. 2, pp. 261–284, 2008.

[6] R. Datta, D. Joshi, J. Li, and J. Z. Wang, "Image retrieval : Ideas, influences, and trends of the new age," *Computing*, vol. 40, no. 2, 2008.

[7] G. Hu and Q. Gao, "An Interactive Image Feature Visualization System for Supporting CBIR Study," *Image Rochester NY*, pp. 239–247, 2009.

[8] B. Moghaddam, Q. Tian, N. Lesh, C. Shen, and T. S. Huang, "Visualization and User-Modeling for Browsing Personal Photo Libraries," *International Journal of Computer Vision*, vol. 56, no. 1/2, pp. 109–130, 2004.

[9] G. Schaefer, "A next generation browsing environment for large image repositories," *Multimedia Tools and Applications*, vol. 47, no. 1, pp. 105–120, 2009.

[10] Y. Rubner, L. J. Guibas, and C. Tomasi, "The Earth Movers Distance, Multi-Dimensional Scaling, and Color-Based Image Retrieval," *Proceedings of the ARPA Image Understanding Workshop*, pp. 661–668, 1997.

[11] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood, "Evaluating a visualisation of image similarity," *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval SIGIR 99*, pp. 275–276, 1999.

[12] M. Koskela, "Interactive Image Retrieval Using Self-Organizing Maps," Ph.D. dissertation, Helsinki University of Technology, november 2003.

[13] H. Eidenberger, "A video browsing application based on visual MPEG-7 descriptors and self-organising maps," *International Journal Of Fuzzy Systems*, vol. 6, no. 3, pp. 125–138, 2004.

[14] D. Deng, J. Zhang, and M. Purvis, "Visualisation and Comparison of Image Collections based on Self-organised Maps," *Science*, vol. 32, pp. 97–102, 2004.

[15] G. Nguyen and M. Worring, "Interactive access to large image collections using similarity-based visualization," *Journal of Visual Languages Computing*, vol. 19, no. 2, pp. 203–224, 2008.

[16] (2011, novembre) QtOpenGL Module. [Online]. Available : http ://doc.qt.nokia.com/latest/qtopengl.html

[17] (2011, novembre) Comparison of image viewers. [Online]. Available : http ://en.wikipedia.org/wiki/Comparison_of_image_viewers

[18] J. Gelernter, "Visual classification with information visualization (infoviz) for digital library collections," *Knowledge Organization*, vol. 34, no. 3, pp. 128–143, 2007.

[19] Z. Pecenovic, M. N. Do, M. Vetterli, and P. Pu, "Integrated browsing and searching of large image collections," *Proc of 4th International Conf on Visual Information Systems*, no. 21, pp. 279–289, 2000.

[20] C. Chen, G. Gagaudakis, and P. Rosin, "Content-Based Image Visualization," *Information Visualization*, vol. 00, pp. 13–18, 2000.

[21] J. Kruskal and M. Wish, "Multidimensional Scaling," *Sage*, 1978.

[22] S. Haykin, *Neural Networks : A Comprehensive Foundation*, M. Horton, Ed. Prentice Hall, 1999.

[23] G. Chiron, "Modèle multi-agent d'attraction/répulsion pour la recherche d'images par le contenu," Master's thesis, Institut de la Francophonie pour l'Informatique, juillet 2009.

[24] A. Boucher, "Cours de vision par ordinateur."

[25] Les formules de 14 valeurs statistiques pour les matrices de cooccurrences. [Online]. Available : http ://fp.ucalgary.ca/mhallbey/equations.htm

[26] J. Flusser, "Moment Invariants in Image Analysis," *Engineering and Technology*, vol. 11, no. 102, p. 196–201, 2006.

[27] T. Kohonen, J. Hynninen, J. Kangas, and J. Laaksonen, "SOM PAK : The Self-Organizing Map Program Package SOM PAK : The Self-Organizing Map Program Package," *Changes*, 1996.

[28] A. Boucher, T.-H. Dang, and T. L. Le, "Classification vs recherche d'information : vers une caracterisation des bases d'images," *12emes Rencontres de la Societe Francophone de Classification (SFC)*, 2005.