Interactive, incremental and multi-level exploration of large collections of images

Frédéric Rayar

UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

ÉCOLE DOCTORALE MIPTIS

Laboratoire d'Informatique (EA 6300)

THÈSE présentée par :

Frédéric RAYAR

soutenue le : 22 Novembre 2016

pour obtenir le grade de : Docteur de l'Université François-Rabelais de Tours

Discipline/ Spécialité : INFORMATIQUE

Exploration interactive, incrémentale et multi-niveau de larges collections d'images

Interactive, incremental and multi-level exploration of large collections of images

THÈSE DIRIGÉE PAR :

VENTURINI Gilles — Professeur, Université François Rabelais de Tours

RAPPORTEURS :

UCHIDA Seiichi — Professeur, Université de Kyushu, Japon

VISANI Muriel — Maitre de Conférences (HDR), Université de La Rochelle

JURY :

BARRAT Sabine — Maitre de Conférences, Université François Rabelais de Tours

BOUALI Fatma — Professeur, Université de Lille 2

MELANÇON Guy — Professeur, Université de Bordeaux

UCHIDA Seiichi — Professeur, Université de Kyushu, Japon

VENTURINI Gilles — Professeur, Université François Rabelais de Tours

VIENNET Emmanuel — Professeur, Université de Paris 13

VISANI Muriel — Maitre de Conférences (HDR), Université de La Rochelle

The most exciting phrase to hear in science, the one that heralds new discoveries, is not "*Eureka*!" but "*That's funny.*"

credited to Isaac Asimov

Remerciements

Les remerciements ... touche finale de ce manuscrit de thèse. Ce n'est pas la partie la plus dure à écrire, mais bizarrement, ce n'est pas la plus simple non plus !

Je vais commencer tout d'abord par m'excuser. Milles excuses aux personnes que je vais oublier de mentionner ici, car je vais en oublier, c'est certain !

Je remercie chaleureusement mes encadrants de thèse : Sabine Barrat, Fatma Bouali et Gilles Venturini. Tout d'abord, merci d'avoir eu la patience de m'attendre, car j'ai pris du temps à l'accepter cette thèse ! Mais surtout, merci pour la liberté que vous avez pu me donner durant ces trois années. Et ce n'est pas une critique, bien au contraire ! Sachez que j'ai l'ai souhaité et fortement apprécié ! De la liberté pour mener mes recherches, proposer des idées, argumenter mes choix et confronter mes pensées. Je ne peux que citer Francois I^{er}, qui accueilli Leonard de Vinci avec ces paroles : "Ici Léonard, tu seras libre de rêver, de penser et de travailler ".

Merci à Mme Visani, Mr. Viennet et Mr. Melançon d'avoir accepté de faire parti de mon jury. J'ose espérer que la lecture de mon manuscrit ne vous a pas ennuyé, et que les images disséminées ont su apporter de la gaieté dans vos lectures.

A special thanks to you, Uchida Sensei. It is a real honor for me that you have accepted to review my PhD dissertation, seven years after I got an internship in your laboratory. Know that, not only I enjoyed my experience in Kuydai, but also that it was this internship that made me want to continue in the research field, especially in image processing and computer vision! Thank you for your efforts, making time to visit me in Tours and staying up late to attend my defense via videoconference. Many thanks to you!

Merci à vous, membres de l'équipe RFAI : Mathieu, Nicolas, Hubert, Romain, Donatello. C'est un réel plaisir d'évoluer professionnellement à vos cotés ! Grosse dédicace à Jean-Yves Ramel, qui m'a accueilli au sein de cette équipe en 2011, et qui doit encore se demander pourquoi ! Merci patron ! Merci à toi de m'avoir laissé de la liberté pour enseigner et faire de la recherche durant ces années pré-thèse. Et merci de m'avoir fait voyager aux quatre coins de l'Europe.

Merci aux collègues du laboratoire avec qui j'ai pu collaborer dans l'enseignement et l'encadrement de projets étudiants : Carl, Alexis, Mohand. Merci aussi à ces collègues "de l'ombre", au secrétariat, service info et bibliothèque, que l'on ne remercie que peu et qui pourtant font partie intégrante dans la réussite d'un doctorant : Christelle, Béatrice, Annie, Karine (la rousse !), Karine (la blonde !), Julie, Sylvie, Betty, Gérald, Sébastien,

Florian, Mike, Pascal.

Merci à Mathieu Daudignon et Kevin Renault, vos projets de fin d'année font partie intégrantes de mes travaux de thèse ! Sachez que vous n'avez pas bossé pour rien ! Merci aussi à tous ces étudiants que j'ai eu le plaisir d'encadrer sur des projets "cool" comme ART-Chess, le Quarto, le Dobble, l'ORBox avec des "jouets" sympas comme la PixelSense, la Sphero ou autres. Ces parenthèses pédagogiques, mais ludiques, ont été grandement appréciées durant mes travaux de thèse.

Bon, on arrive à ceux que moi, j'ai du supporter tous les jours : les autres doctorants et cie. En cinq ans, j'en aurais vu des gens dis donc ! Vous le savez peut-être, à mon sens la recherche seule dans son coin n'a que peu d'intérêt. C'est en discutant, en partageant, en étant curieux de ce que l'autre fait que l'on peut arriver à ouvrir son esprit, continuer d'apprendre et prendre du recul sur son propre travail. Merci à vous tous donc avec qui j'ai pu discuter en français ou en anglais : Partha, MML, Fareed, The Anh, Alireza, Nathalie, Alina, Barthélémy, Cyrille, Alexandre, Tanmoy, Khoi, Quang-Cheu, Ut, Khaoutar, Aymen, Azeddine, Mohamed, Ismaila, Lei, Maxime, Mostapha, Flavie, Damien, etc. Merci pour toutes ces pauses café, parties de baby-foot, repas partagés au RU, pique-niques, ... Merci !

Des remerciements un peu particuliers à 3 groupes de personnes :

- Les "vieux" : Faiza, Octavio et Julien : c'est grâce à votre présence au labo que je me suis plu à Tours, et donc que j'y suis resté ! Je sais que vous continuez vos aventures ailleurs, et je ne peux que vous souhaitez bonne route, en attendant que nos chemins se recroisent !

- Les "jeunes" : Gaetan et Anu : pinaize, qu'est ce que j'ai été content de vous voir arriver après le départ des vieux ! Merci pour toutes ces pintes partagées ! Merci pour toutes ces pauses "mercredicales" ! N'abusez pas trop du Picon !

- Les "aoutiennes" : Zeina, Mona et Eun-Jin : un grand merci à vous mesdemoiselles ! Merci de m'avoir accompagné pendant ce mois d'août, sans vous je n'aurais jamais eu le courage de continuer la rédaction seul. Sans vous, l'année 2016 n'aurait jamais pu apparaitre sur la page de garde de ce manuscrit. Merci ! :)

Pour finir, j'ai une pensée pour mes amis toulousains : Thibault, Andra, Gouf, FH, Vivian, Totof, Benoit, Gtunt, Baptiste. Votre amitié m'est précieuse, et j'ose espérer que même si nous ne communiquons pas au quotidien, cette amitié perdura à travers nos vies.

Une pensée pour Sarah, qui m'a supporté pendant presque la totalité de ces trois années de thèse.

Et enfin, last but not least, une pensée pour ma famille qui a façonné une majeure partie de ce que je suis aujourd'hui : Papa, Maman, Raoul et Sabrina.

Merci.

Résumé

Notre société vit actuellement dans un monde submergé par les données numériques. Parmi cette masse de données, nous nous intéressons particulièrement aux images dans ces travaux de thèse. En effet, la dernière décennie a vu la réduction du coût des appareils photos, des webcams, des scanners, mais aussi celui des supports de stockage. De fait, la quantité d'images capturées par tout un chacun a explosé, qu'elles soient générées dans le cadre privé, commercial ou dans celui de projets de numérisation. De plus, l'avènement d'Internet a accentué le fait que le nombre d'images mises en ligne croît de manière exponentielle, touchant différents domaines tels que les réseaux sociaux, les humanités numériques, l'astronomie ou encore la médecine.

Au même titre que le stockage de ces images, un défi majeur est la possibilité d'explorer ces grandes collections d'images, éventuellement dans le but d'en extraire de l'information et de la connaissance. Durant la dernière décennie, de nombreux travaux se sont penchés sur la gestion et l'analyse de grandes collections d'images. Cependant, la majeure partie de ces efforts a été orientée vers la recherche d'image par le contenu. Ces travaux ont donné lieu à des systèmes de recherche très performants. Néanmoins, ces derniers offrent peu de support pour une exploration interactive, où l'utilisateur pourrait *flâner* dans la collection d'image, permettant une certaine sérendipité dans l'exploration.

Dans ces travaux de thèse, nous abordons cette problématique du point de vue de *l'analyse et l'exploration interactive des données*. Pour permettre à un utilisateur d'explorer interactivement une grande collection d'images, nous tirons profit du paradigme de navigation par similarité. Nous visons à respecter simultanément les trois contraintes suivantes : *(i)* traiter de grandes collections d'images, *(ii)* traiter des collections dont le nombre d'images ne cesse de croître au cours du temps et *(iii)* donner des moyens d'explorer interactivement des collections d'images. Pour ce faire, nous proposons d'effectuer une étude conjointe de l'indexation et de la visualisation de grandes collections d'images qui s'agrandissent au cours du temps.

Notre contribution est triple. Dans un premier temps, nous nous concentrons sur la construction incrémentale d'un graphe de voisinage, à savoir le Graphe des Voisins Relatifs (GVR), en vue de structurer une collection d'images. Le choix de ce graphe est justifié, puis un travail connexe est présenté et discuté. Nous proposons alors une méthode exacte et une méthode approchée pour construire incrémentalement un tel graphe, dans le but de traiter de grandes collections d'images. Ces méthodes sont étudiées et discutées, et des expérimentations, traitant jusqu'à un million d'images, sont présentées.

Dans un second temps, nous présentons une structure hybride, combinant hiérarchie

et graphe, et qui est visualisable pour permettre une exploration interactive de grandes collections d'images. Un algorithme de partitionnement de données, à savoir l'algorithme BIRCH, est exploité. Ce choix est justifié et les améliorations apportées pour satisfaire notre objectif sont présentées. La structure d'arbre générée par BIRCH est ainsi améliorée par : *(i)* l'assignation d'images pertinentes aux nœuds intermédiaires de l'arbre, permettant ainsi une meilleure interprétation des nœuds durant le processus d'exploration et *(ii)* la structuration de la partie intérieure de l'arbre grâce à des GVR. Ces améliorations sont apportées de manière à obtenir un algorithme incrémental, gérant ainsi les collections croissantes. Des expérimentations, traitant jusqu'à 7 millions d'images, sont présentées.

Enfin, nous présentons notre plateforme d'exploration interactive de grandes collections d'images. Cette plateforme repose en partie sur la structure hybride mentionnée ci-dessus. Le choix du tracé d'un graphe dans un plan 2D n'est pas choisi aléatoirement, mais justifié par rapport à l'état de l'art. Pour évaluer notre plateforme, nous présentons une évaluation utilisateur, aux cotés de différents cas d'utilisation, tels qu'un système de recommandations ou encore un outil d'analyse visuelle pour l'analyse d'images de documents. L'ensemble de ces cas d'utilisation implique des collections d'images réelles.

Mots clés : Larges collections d'images, Indexation incrémentale, Visualisation, Exploration interactive, Graphe des voisins relatifs, Clustering, Algorithme BIRCH, Structure hybride, Sérendipité, Analyse visuelle, Analyse d'images de documents.

Abstract

At the present time, our society is overwhelmed with digital data. Among this deluge of data, this thesis work shed light on images. Indeed, there is an exponential growth of online image data. This phenomenon can be explained by the cost reduction of capturing and storage devices and the advent of the Internet. Hence, massive online image upload occurs nowadays in several fields such as social networks, humanities, astronomy or healthcare.

One challenge, along with the storage of this huge volume of images, is the exploration of these image collections, in order to eventually extract information from them. To address this issue, the last decades have witnessed plenty of research efforts on large-scale multimedia indexing. However, most of those efforts are dedicated to image retrieval systems. Such image retrieval systems perform well in search tasks. However, they lack support for exploratory data mining in which the user can wander around the images and do not provide means for serendipity in the exploration process.

This thesis addresses this issue from the perspective of Interactive Data Exploration and Analytics. In order to allow the user to interactively explore an image collection, we take advantage of the similarity-based image collection browsing paradigm, where one has the possibility to navigate within an image collection that has been structured beforehand. We aim at meeting simultaneously the three following constraints: *(i)* handling large image collections, up to millions of images, *(ii)* handling ever-growing image collections, and *(iii)* providing interactive means to explore image collections. To do so, we jointly study the indexing and the interactive visualisation of large and ever-growing image collections.

Our contribution is three-fold. First, we focus on the incremental construction of a proximity graph, namely the *Relative Neighbourhood Graph* (RNG), to structure the image collection. The choice of this graph is justified and an existing work is presented and discussed. We propose an exact approach and an approximative approach to yield incrementally such a graph to handle large image collection. The approaches and their complexities are discussed, and experimentations that have been performed up to a million images are presented.

Second, we propose a hierarchical and graph-based hybrid *viewable* structure to allow an interactive exploration of large image collection. A data partitioning algorithm, namely *BIRCH*, is used. The choice of the algorithm is justified and the improvements brought to fit our objectives of visualisation are presented. Thus, the tree structure yield by BIRCH is enhanced by *(i)* computing representative images for each internal node, to allow a better interpretation during the tree exploration and *(ii)* structuring intermediate levels and internal nodes of the tree with a RNG. The incremental construction property is

preserved to build this structure in order to fulfil the dynamic aspect of image collection. Experimentations performed up to several millions of images are presented.

Last, we present our interactive visualisation platform for large image collections. This platform relies on the hybrid structure mentioned above. The way to draw graphs in a plane is not be randomly chosen but is justified with regard to the state of the art. To evaluate the platform, along with an user evaluation, several different use cases are discussed, such as clustering results visualisation, recommendation system for ever-growing image collections and visual analysis system for Document Images Analysis. These use cases are based on real world large image collections.

Keywords: Large Image Collection, Ever-growing Image Collection, Incremental Structuring, Visualisation, Interactive Exploration, Relative Neighbourhood Graph (RNG), Clustering, BIRCH Algorithm, Hybrid Structure, Serendipity, Visual Analytics, Document Image Analysis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Abstract

In this chapter, we present the context and the questions that have motivated the present work. The contributions brought by this thesis are listed, and the organisation of the manuscript is detailed.

Contents

1.1 Context

At the present time, our society is overwhelmed with digital data. Thanks to recent advances in technology, one can collect data from almost everywhere, everything and everyone in a continuous way. This permanent data flow can be nearly infinite and occurs in sundry and various fields, such as sciences, healthcare, education or sport. In the last decade, the advent of the so-called *big data* area brought to researchers a whole new set of challenges to be addressed. These challenges are related to every step of the data life cycle and may include, for instance, the acquisition, the storage, the analysis and the visualisation of the information.

In this deluge of data, we shed light on images. Indeed, the exponential growth of image data is a perfect illustration of the big data phenomenon. Two reasons may explain this growth: first, the last decade has witnessed the cost reduction of capturing devices such as cameras and scanners, but also storage media. This has resulted in a sharp increase of the amount of captured images. These images can be generated in a private setting, a commercial framework or in the context of digitisation projects. Second, the advent of the Internet has emphasised the fact that the number of images that are uploaded online grows exponentially. The main reason for this is the diversity of social networks, that gave birth to new online behaviours. For instance, thousands of photos are added each minute on social networks such as Snapchat, Instagram or Facebook [Domo, 2015] (see infographic in Appendix A).

However, massive online image upload is not the prerogative of social networks. Actually, several fields are affected by this phenomenon, such as humanities, astronomy, healthcare, *etc.* One example is the OpenGLAM initiative [DM2E, 2015]. Many institutions such as galleries, libraries, archives and museums promote "free and open access to digital cultural heritage" by making digital scans or photos of their collections available to the public. Others well-known institutions, such as the British Library of London, or the Rijkmuseum of Amsterdam, are making efforts towards the open access of their collections. Figure 1.1 illustrates this trend in the last decade. Another example, where privacy is a main concern, is digitised documents. Either in public institutions (*e.g.* tax office), or private companies (*e.g.* banks, insurance), many customers send digital copies of their documents. Handling this huge ever-growing amount of document images in a efficient way becomes mandatory for these institutions and companies.

1.2 Motivations

One challenge, along with the storage of this huge volume of images, is the exploration of these image collections. In order to extract information from these images, one needs to have a relevant representation to observe their global topology and search local information. To address this issue, the last decade have witnessed plenty of research efforts on large-scale multimedia indexing. However, most of those efforts are dedicated to image retrieval systems. These systems are designed for searching and retrieving images from a large image

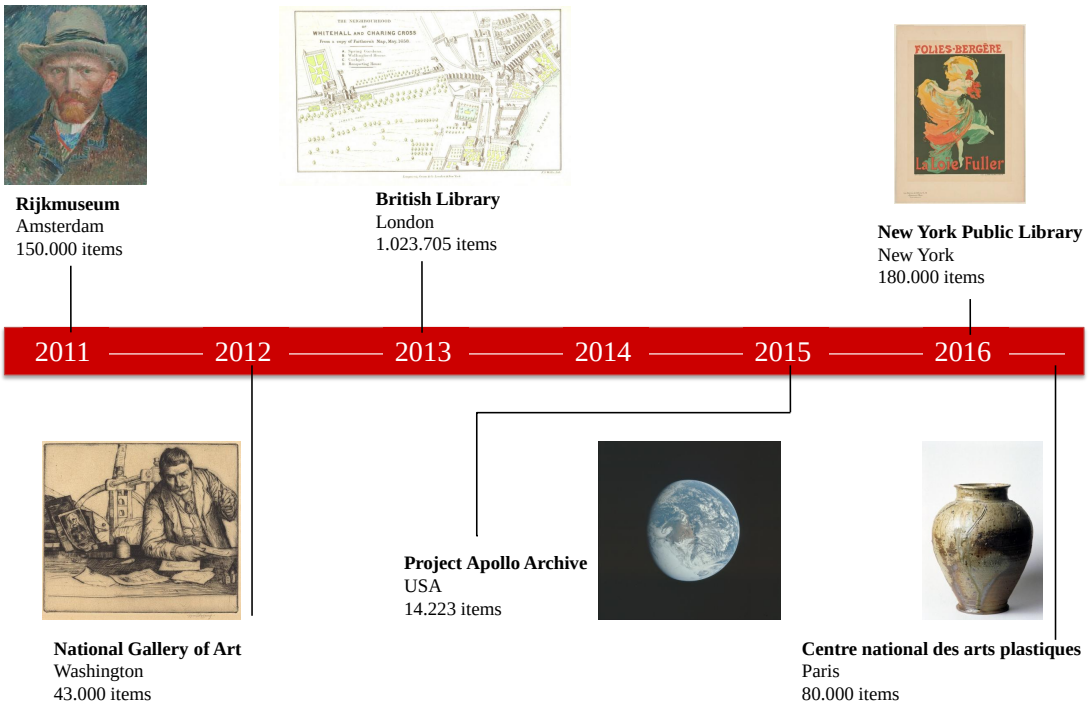


Figure 1.1: Timeline of some important efforts towards open cultural heritage access in the current decade.

collections, given a text query or an image query. One of the cutting-edge methods for this purpose, that still attracts extensive research interest, is the locality-sensitive hashing [Gionis et al., 1999]. This method solves the approximate (or exact) nearest neighbour search in high dimensional spaces, thus providing the most similar images for a given query.

Such image retrieval systems perform well in search tasks. However, they lack support for exploratory data mining in which the user can wander around the images and do not provide means for serendipity in the exploration process. Our work falls into the scope of *Interactive Data Exploration and Analytics*. In a workshop of Knowledge Discovery and Data Mining (KDD) 2013 [Chau et al., 2013], it has been defined as:

> *The Interactive Data Exploration and Analytics (IDEA) workshop addresses the development of data mining techniques that allow users to interactively explore their data. We focus and emphasize on interactivity and effective integration of techniques from data mining, visualization and human-computer interaction (HCI). In other words, we explore how the best of these different but related domains can be combined such that the sum is greater than the parts.*

In order to allow the user to interactively explore an image collection, another approach can be used, namely the image collection browsing. Here, one has the possibility to navigate within an image collection that has been structured beforehand. A common

and useful way of browsing image collections is using *image similarity*. Some paradigms have been proposed to visualise image collections, such as projection in a 2D space or clustering. However, the proposed techniques do not meet simultaneously the three following constraints:

(i) *handling large image collections*: several studies experiment their contribution with small or rather medium data sets, up to a few thousands of images. However, the challenge is to handle real world larger data sets like medical image collections or archive collections;

(ii) *handling ever-growing image collections*: most of the existing works study static data sets, in which a static structure is precomputed and then used for the visualisation. However, as previously mentioned, as images keep flooding in a continuous way, it becomes mandatory to build structure in a dynamic way: images shall be either added or edited if it is needed. Thus, it becomes essential to have incremental algorithms to build and update viewable indexing structures, in order to keep organizing images in a dynamic, yet consistent way;

(iii) *providing interactive means to explore image collections*: first, the visual interface must provide relevant visualisations with symbolic objects that are easy to interpret by the user. Second, the visualisation itself is not enough: useful interactions must be provided to allow the user to wander around the collection and make discoveries by following his intuition.

1.3 Objective and contributions

The underlying questions that arise from the aforementioned constraints, and that are addressed throughout this thesis, are:

1. Can we incrementally build a relevant structure for large and ever-growing image collections?
2. Can we build such a structure in a way that it can be interactively visualised?
3. Can we provide the user an interactive interface for exploring large image collection?

Thus, we aim in this thesis to jointly index and interactively visualise dynamic and large image collections.

Contributions

Our contribution is three-fold.

First, we focus on the incremental construction of a *Relative Neighbourhood Graph* (RNG) to structure the image collection. The choice of this graph is justified and an existing work is presented and discussed. Then, we propose an exact approach and an approximative approach to yield such a graph to handle large image collections. The

approaches and their complexities are discussed, and experimentations that have been performed up to a million images are presented.

Second, we propose a hierarchical and graph-based hybrid structure to allow an interactive visualisation of large image collections. A data partitioning algorithm, namely *BIRCH*, is used. The choice of the algorithm is justified and the improvements brought to fit our objectives of visualisation are presented. The incremental construction property is preserved for this structure. Experimentations performed up to several millions of images are presented.

Last, we present a custom platform that allows interactive explorations of large image collections. This platform relies on the hybrid structure mentioned above. The way to draw graphs in a plane is not be randomly chosen but is justified with respect to the state of the art. Furthermore, current state-of-the-art visualisation design guidelines [Munzner, 2014] have been considered during the implementation of our platform. To evaluate the platform, along with this thesis work, three different use cases are discussed.

1.4 Organisation of the manuscript

The rest of the thesis is organised as follows:

- Part I presents the literature survey on several points.
  - Chapter 2 introduces the main concepts of interactive data exploration and its usefulness;
  - Chapter 3 reviews how an image can be described to have a relevant representation and presents the state of the art of image collection visualisation;
  - Chapter 4 sums up how we have worked to fulfil our objectives regarding the existing works.
- Part II presents the contributions of this thesis.
  - In Chapter 5, we focus on the incremental construction of the RNG to structure large image collections;
  - In Chapter 6, we detail the hierarchical and graph-based hybrid structure, built using the BIRCH algorithm;
  - In Chapter 7, we briefly discuss graph drawing algorithms and present the interactive visualisation platform that have been developed during this thesis.
- Part III presents several use cases that have been done during the thesis to illustrate and evaluate our work.
  - ImageNet eXplorer: in Chapter 8, we present the interactive visualisation of an image collection of 7 millions of images in the implemented platform;
  - APOD eXplorer: in Chapter 9, we propose a use case where our work is used to build a recommendation system on an ever-growing image collection;

  - DINet Viewer: in Chapter 10, we illustrate how our platform could be used as a visual analytics tool to assist researchers in the evaluation of image features and distances.
  - In Chapter 11, we present a user evaluation that has been conducted to evaluate the proposed platform.

- Finally, this thesis ends up with Chapter 12, where we sum up and discuss all of our work and propose some perspectives that will be interesting to look into, following this work.

Part I

State of the Art

Chapter 2

Interactive Data Exploration

Abstract

In this chapter, we step back from the interactive visualisation of image collection, and describe the larger discipline of Interactive Data Exploration (IDE). We define the concepts and the goals behind IDE. We especially discuss how Information Visualisation is a key component and a relevant mean to support IDE. We then argue for the necessity of integrating interactivity in the visualisations, to address the scalability challenge. Finally, we briefly introduce the discipline that integrates analytics and reasoning in visualisations.

Contents

2.1 From data to wisdom

2.1.1 Data

As mentioned in the Introduction, our world is overwhelmed by data. Let us first try to define this object of study. There is no universal definition, however early works in the management field have given the definition below. According to Russell L. Ackoff [Ackoff, 1989]:

Definition 2.1.1 (Data)**.** Data are raw. They are defined as symbols or facts that represent properties of objects, events and their environment. They are the products of observation. They simply exist and have no significance beyond their existence (in and of themselves). They can exist in any form, usable or not. They do not have meaning of themselves.

Following this broad definition, one can easily imagine the considerable variety of types that data can have: geospatial, temporal, multivariate, text documents, images, videos, etc.

2.1.2 Data, Information, Knowledge, Wisdom

Having data at our disposal, what can one do with it, and for what purposes? A common answer is to analyse the data, in order to make observations, have insights, extract information and discover knowledge.

Early works from management [Ackoff, 1989] have also studied these questions and have proposed the fruits of their labour in a hierarchy: the Data, Information, Knowledge and Wisdom (DIKW) pyramid (see Figure 2.1). We define below the three last terms, expanding [Ackoff, 1989], [Rowley, 2007] and [Keller and Tergan, 2005] definitions.

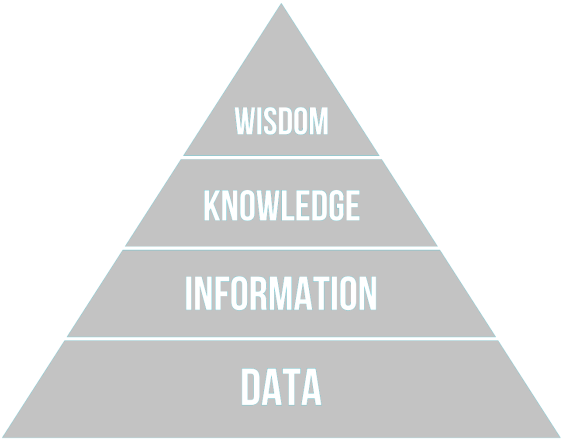


Figure 2.1: DIKW pyramid. Structured representation of relationships between data, information, knowledge, and wisdom.

Definition 2.1.2 (Information)**.** Information is data that has been given meaning through interpretation by way of relational connection. This "meaning" can be useful, but does not

have to be. Information provides answers to "who", "what", "where", "why", or "when" questions. From there, data that has been given meaning by somebody and, hence, has become information, may still be data for others who do not comprehend its meaning. The difference between data and information is functional, not especially structural.

Definition 2.1.3 (Knowledge)**.** Knowledge is an appropriate collection of information, that is supposed to be useful. The information has been cognitively processed and might be integrated into an existing human knowledge structure. Knowledge can be obtained either by transmission from another who has it, by instruction, or by extracting it from experience.

Definition 2.1.4 (Wisdom)**.** Wisdom is the most difficult concept to explain. Wisdom relies on the understanding of existing knowledge. It is understanding that is evaluated. Wisdom is the essence of philosophical probing. Unlike the previous levels, it asks questions to which there is no (easily-achievable) answer, and in some cases, to which there can be no humanly-known answer.

This hierarchy can also be represented in a chart (see Figure 2.2). Here, understanding connections between data leads to information. Understanding patterns in information leads to knowledge. All these steps build an experience on which one relies on to apprehend future understanding: wisdom.

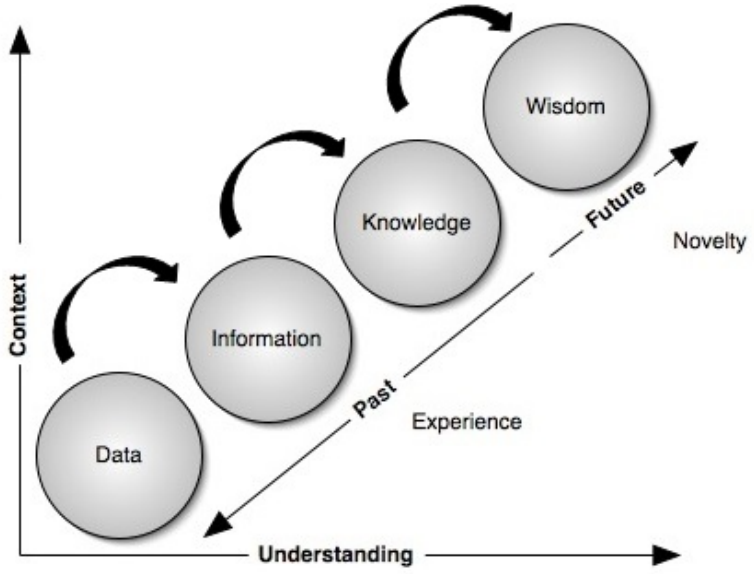


Figure 2.2: DIKW diagram. Straightforward scheme of the conversion of data into wisdom.

2.1.3 Challenges

Several challenges, either old or new, have to be faced in this journey toward wisdom, among others:

- *heterogeneous data*: considerable variety of data types,

- *distributed data*: data are stored in physically different storage media,
- *"big data"*: large amount of data,
- *streaming data*: continuous flow of data.

2.2 Information Visualisation

2.2.1 Definition

To assist the human in the process of having insights from data, a possible medium is visualisation, as defined in [Card et al., 1999],

Definition 2.2.1 (Visualisation)**.** Visualisation is the use of computer-supported, interactive, visual representations of data to amplify cognition.

Visualisation is a field that has emerged from the Computer Graphic community. It has been applied to scientific data, resulting in *Scientific Visualisation* (SciVis) field, to allow scientists to see the phenomenon in their data, often physical data (*e.g.* human body, molecules, earth).

Visualisation techniques have also been applied on abstract data (*e.g.* financial data, business information). This gave birth to a new field, namely *Information Visualisation* (InfoVis). According to [Card et al., 1999], we define:

Definition 2.2.2 (Information Visualisation)**.** Information Visualisation is the use of computer-supported, interactive, visual representations of **abstract** data to amplify cognition.

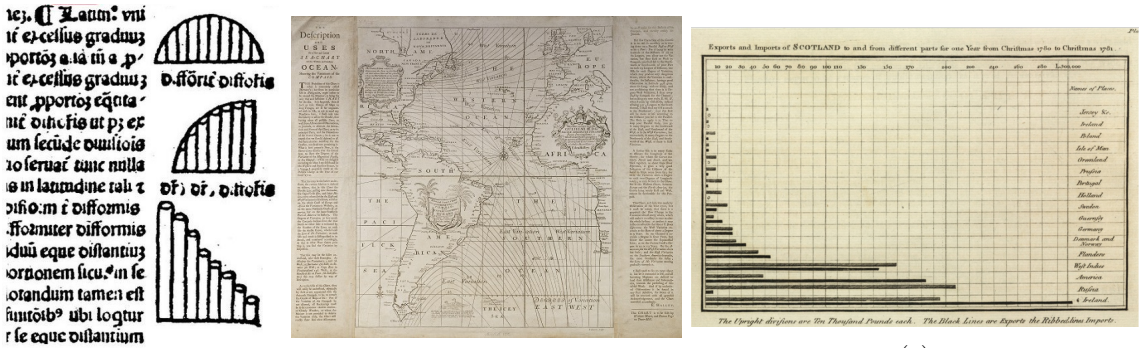
The latter field is the focus of this chapter. For more details on Scientific Visualisation, one can refer to [McCormick et al., 1987]. From now on, we will use interchangeably *visualisation* and *information visualisation* in the rest of the manuscript.

2.2.1.1 History

Earlier works in graphic representation are attributed to Nicolas Oresme (circa 1323-1382) with the use of bar graph (see Figure 2.3a). He drew a graph of velocity versus time for an object moving with constant acceleration, like a falling ball. In the end of the 17th century, Edmond Halley (1656-1742) used maps to show geographical observations (see Figure 2.3b).

William Plaifair (1759-1823) is credited for being the first one to use graphic representations to display economic data. He introduced during his work three well-know diagrams, namely time series, histograms (see Figure 2.3c) and pie charts.

In the 20th century, classic approaches to plot data were developed following the theories of Jacques Bertin [Bertin, 1967] and Edward Tufte [Tufte, 1983]. The former proposed a framework to design diagrams while the latter gave guidelines to maximise the density of useful information in data graphics. One can find a thorough study of the visualisation history in [Friendly, 2008].



(a) (b) (c)

Figure 2.3: Earlier graphic representations. (a) Nicolas Oresme bar graph, (b) Edmund Halley map and (c) Playfair bar chart.

2.2.2 Why visualisation?

One can wonder about the relevance of using visualisation to extract knowledge from data. In what extent visualisation is useful for getting insight from data? As mentioned in Card's book [Card et al., 1999],

> *The purpose of visualisation is insight, not pictures. The main goal of this insight are discovery, decision making and explanation.*

Thus, by definition, Information Visualisation is a useful support in the process of having insight from data. Other studies, in the psychological field, back up this statement. We describe below the two most important psychological arguments, namely visual perception and cognitive benefits.

2.2.2.1 Visual perception

According to Ware [Ware, 2004], two theories can explain how vision is a relevant canal to perceive features and shapes.

First, at low level, the pre-attentive processing theory [Treisman, 1985] studies visual properties that can be processed rapidly ($< 200ms$) and accurately by the human visual system. Hence, such processing does not require for focusing attention. For instance, in Figure 2.4, finding the circle among the shapes takes no time and no effort. In this example, shape is the feature that is used to pre-attentively find the circle. Several others visual properties can be used depending on the tasks and the data, among others, color, shape, size. One can find a thorough study of these visual properties in "Perception in Visualization" [Healey and Enns, 2012].

Second, at higher level, the gestalt theory [Koffka, 1935] argues that the mind understand better the whole rather than the sum of its part, under the assumption that the parts are organised or structured. The author gives several principles, among others:

1. *proximity*: objects that are close together are perceptually grouped together

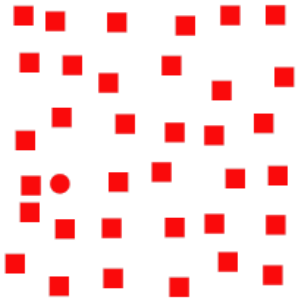


Figure 2.4: Pre-attentive processing. Finding the circle among the shapes takes no time and no effort.

2. *similarity*: objects that share visual features are perceptually grouped together
3. *closure*: objects that are not complete can be seen as a whole, our perception filling the visual gap
4. *symmetry*: two objects that are symmetrically disposed are perceived as a whole

Figure 2.5 illustrates the four principles listed above.

These two theories have a deep influence on the Information Visualisation field. Indeed, the researchers of this field rely on them to:

- choose the visual properties of the objects to represent the data, and try to maximise the pre-attentive perception,
- follow the principles of the gestalt theory to highlight groups of objects in their visualisations.

2.2.2.2 Cognition benefits

According to its definition, the goal of Information Visualisation is to "amplify cognition". How visualisation can amplify cognition? We can cite two studies that have addressed this question.

In "Why a Diagram is (Sometimes) Worth Ten Thousand Words" [Larkin and Simon, 1987], the authors compare the difference of solving physic problems with or without diagram representations. They conclude that visual representations can support the user and improve the fulfilment of his tasks in terms of effectiveness and rapidity. However, they note that for some tasks, visualisations can conversely bother the user.

In [Card et al., 1999], the authors propose a list of six ways where visualisation can amplify cognition:

1. increasing the memory and processing resources available to the users,
2. reducing the search for information,

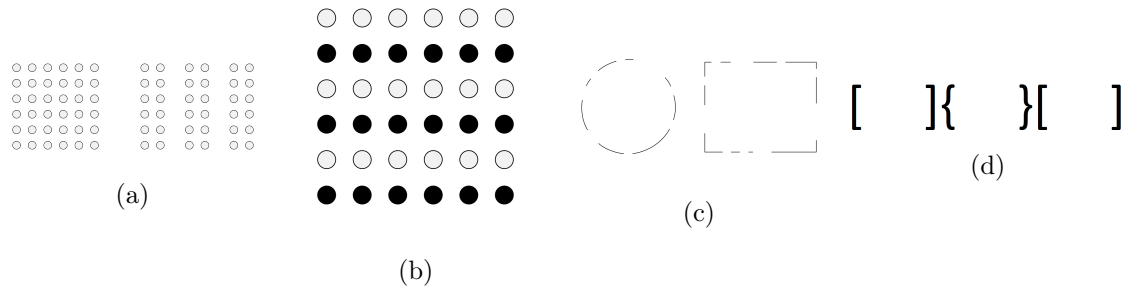


Figure 2.5: Gestalt theory. Illustration of some principles: (a) proximity, (b) similarity, (c) closure and (d) symmetry.

3. using visual representation to enhance the detection of patterns,
4. enabling perceptual inference operations,
5. using perceptual attention mechanisms for monitoring,
6. encoding information in manipulable medium.

2.2.3 Useful visualisations

To argue for the usefulness of Information Visualisation, we list below a few relevant use cases.

2.2.3.1 From the past ...

The map of Minard (1781-1870) is, according to Tufte [Tufte, 1983], the *"best statistical graphic ever drawn"*. The map depicts the number of soldiers of Napoleon army during the 1812 invasion of Russia. The map is presented in Figure 2.6. It displays several types of data in the same visualisation:

- *geography*: rivers, battles, cities
- *number of soldiers*: width of the path
- *army direction*: golden path leading Into Russia and black one leading out of Russia
- *temperature*: plot of negative temperatures over time
- *time*: span of the war

The map of John Snow (1813-1858) helped to discover the source of a cholera epidemic in London in 1855. The map is displayed in Figure 2.7. He plotted in the map the location of the deaths and the water pumps ones. Thanks to this, he discovered that a certain water pump was most certainly the source of the cholera outbreak.

Other examples can be found in "A Brief History of Data Visualization" [Friendly, 2008].

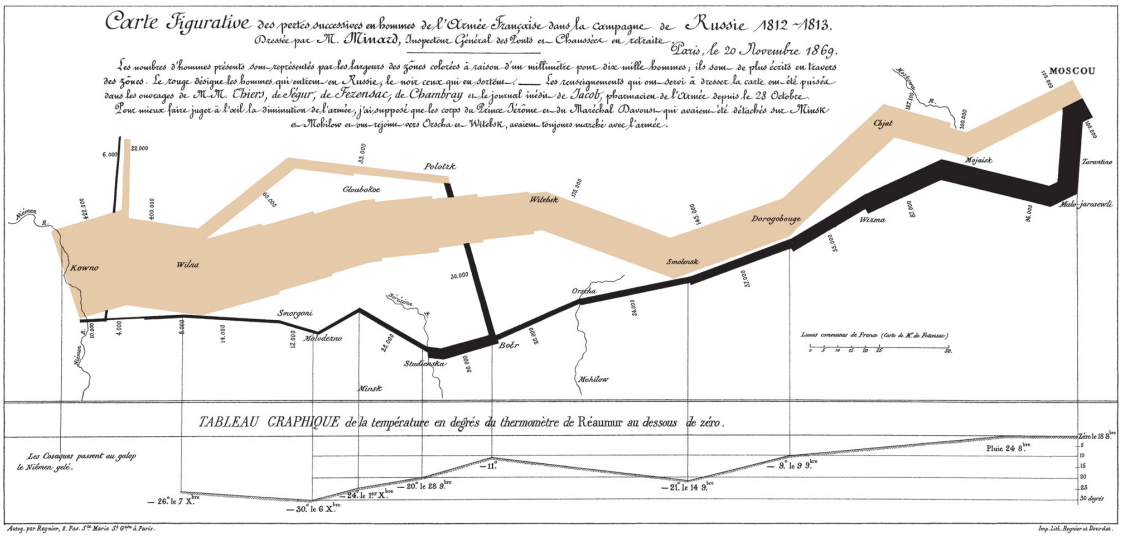


Figure 2.6: Map of Minard. Evolution of the number of soldiers of Napoleon army and the temperature during the the invasion of Russia in 1812.

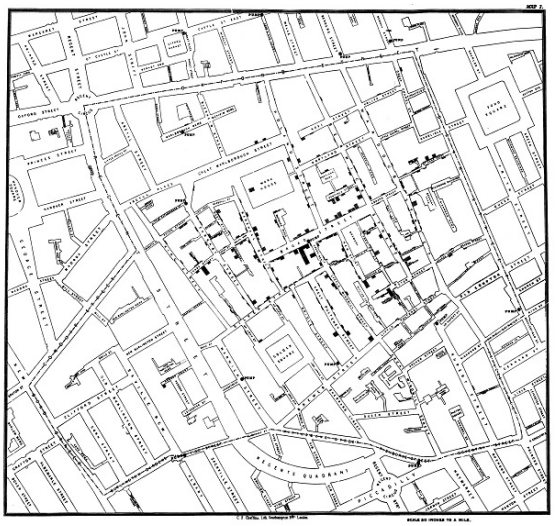


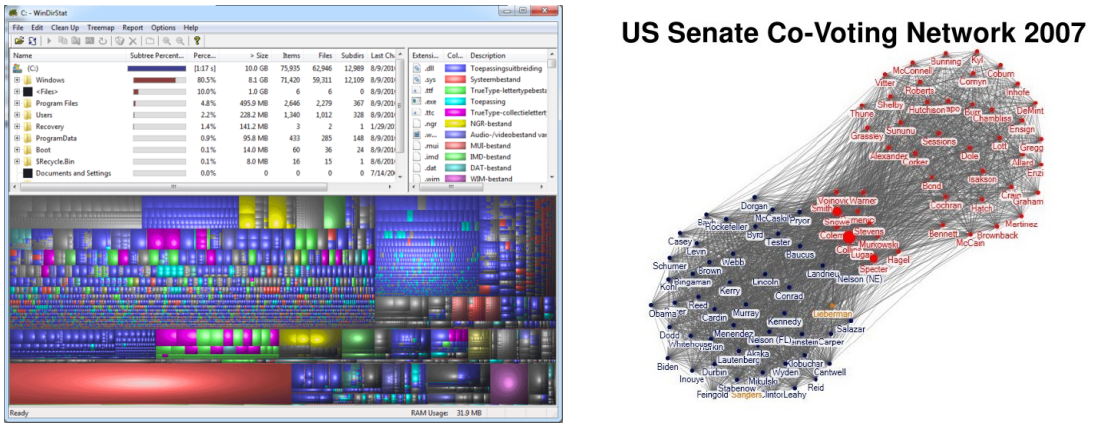
Figure 2.7: Map of John Snow. The map overlays location of deaths and the water pumps ones.

2.2.3.2 ... to the present

In [Shneiderman, 1992], Ben Schneiderman uses space filling nested rectangles to represent hierarchical data. An application to the proposed *treemaps* is presented to display hard disk space usage. Figure 2.8a shows the screenshot of a software that uses treemap for hard disk usage.

In Figure 2.8b, one can see the US senators voting relationships that are represented in a network. Each node represents a US senator, and a connection exists between two

senators if they have voted together 65% or more in the same session.



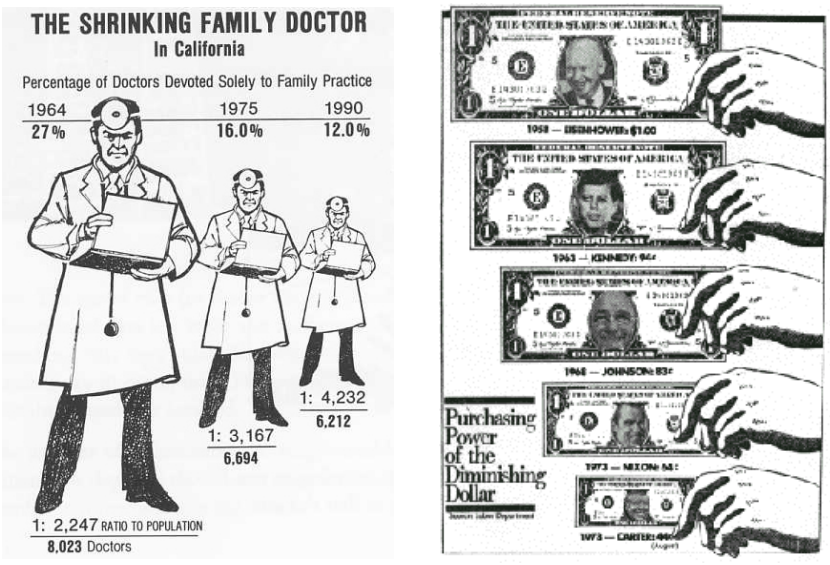
(a) (b)

Figure 2.8: Recent striking visualisations. (a) Screenshot of WinDirStat. Treemaps are used to display the hard disk usage. (b) 2007 US senators co-voting network.

2.2.4 Shortcomings

In view of these studies and success stories, it seems that Information Visualisation is a relevant and powerful medium to extract knowledge from data. However, it is not an ideal solution. Indeed, we can find a few shortcomings of such media, among others:

1. user confusion, mainly due to bad designs,
   - Tufte [Tufte, 1983] qualifies such visualisations of *chartjunk*.
   - Stephen Few discusses several bad visualisation in his personal website [Few, 2004]. For each example, he presents an analysis of its problems and the proposed solutions.
2. miscommunication, where wrong messages are conveyed,
   - Tufte [Tufte, 1983] gives examples of such visualisation. Figures 2.9 illustrates two classic examples.
   - Tyler Vigen illustrates fun correlations between data that have nothing in common in his Spurious Correlations project [Vigen, 2014].
3. user disappointment and frustration: most of the presented visualisations are static, thus, the user cannot really play its data. To address this issue, studies have been done to integrate interaction in visualisation.



(a) (b)

Figure 2.9: Miscommunicated information. (a) Shrinking doctor and (b) shrinking dollar famous visualisations, where areas are misused to show the evolution of one-dimensional data.

2.3 Interactive visualisation

Interactivity is a natural continuation of visualisation. Indeed, fully automatic data analysis faces its limitations when dealing with challenges such as scalability or heterogeneity. Thus, the need to make the user part of the data analysis process appeared. Therefore, it is relevant to take into account the user skills and knowledge. This led the

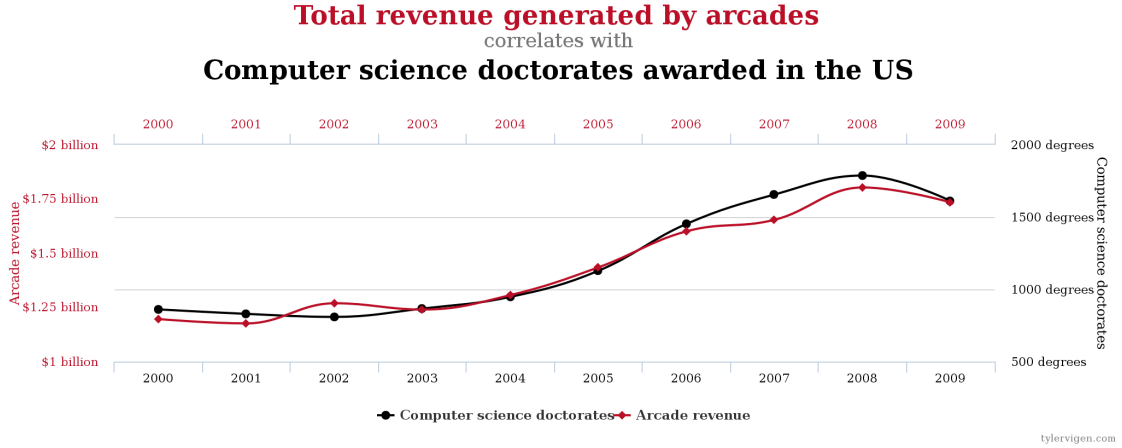


Figure 2.10: Spurious Correlation. Odd correlation between the total revenue generated by arcades and the computer science doctorates awarded in the United States between 2000 and 2009.

visualisation community to propose user-centric or human-in-the-loop approaches.

Hence, the human helps the computer to analyse challenging data sets. Such paradigm has also benefits for the human. Indeed, being part of the process allows him to understand more the pipeline, hence the observed visualisation. It leads to less frustration toward the system when the user is facing bad or unexpected results.

Furthermore, it allows the user to really wander around the data set and explore it, giving a chance to serendipity. Thanks to the interactivity property, one can address the case when the user has no predefined objective with regard to his data.

The most well-known mantra in Information Visualisation, that fits interactive exploration, has been proposed by Ben Schneiderman [Shneiderman, 1996]:

> *Overview first, zoom and filter, then details-on-demand.*

It highlights the need to assist the user in his data exploration throught relevant interactions. Common interactions are: sorting, filtering, browsing, zooming and comparing. A more complete list of interactions can be found in [Card et al., 1999].

2.4 Visual Analytics

Visualisation focus was first on scientists (SciVis), and then on general users (InfoVis). A recent trend, namely Visual Analytics (VA), aims at creating visualisation for analysts (*e.g.* in bank, in insurance). Figure 2.11 illustrates the different groups involved.

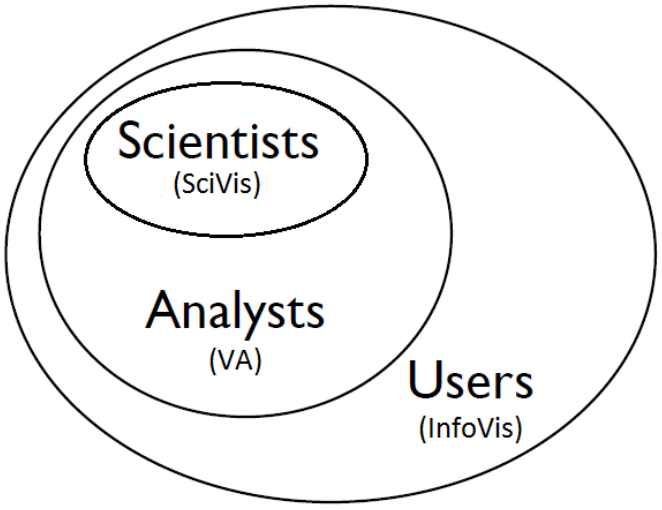


Figure 2.11: Visual Analytics. Visualisation of nested groups of final users.

Visual Analytics has been first defined by Thomas and Cook [Thomas and Cook, 2005] as "*the science of analytical reasoning facilitated by interactive visual interfaces*". However, considering nowadays practises, the following definition is a better fit:

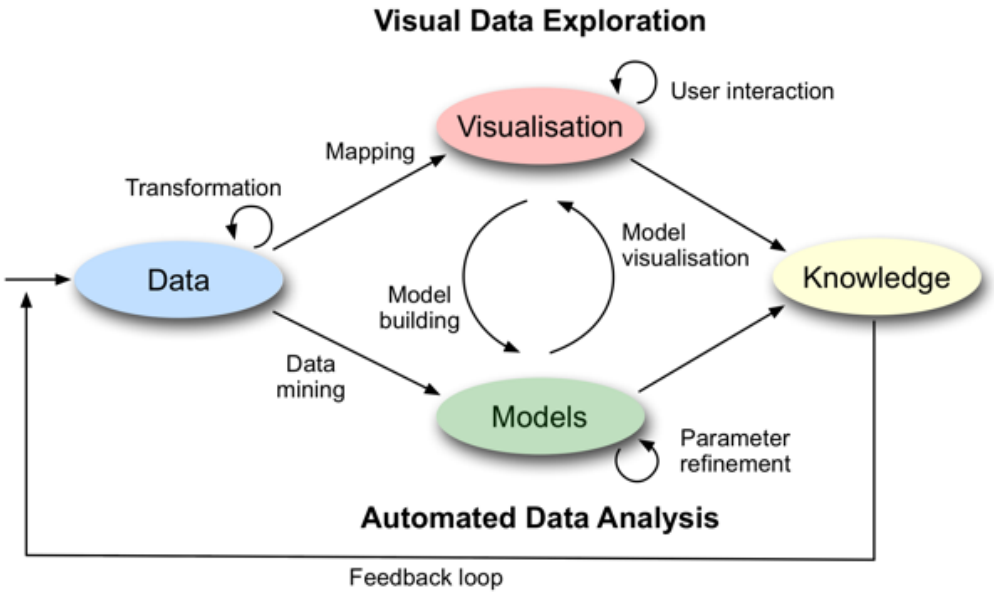


Figure 2.12: Visual Analytics. Workflow of VA: a combination of visual data exploration and automated data analysis.

Definition 2.4.1 (Visual Analytics)**.** Visual analytics combines automated analysis techniques with interactive visualisations for an effective understanding, reasoning and decision making on the basis of very large and complex data sets.

Hence, Visual Analytics aims at facilitating the creation, iterative refinement and validation of hypotheses regarding the structure or knowledge that may lie within a data set. To do so, humans and machines cooperate using their respective capabilities in order to achieve the most effective results. Figure 2.12 illustrates the workflow of Visual Analytics. More details on this discipline can be found on "Visual Analytics - Definitions, Process and challenges" [Keim et al., 2008].

Chapter 3

Image collection visualisation

Abstract

In this chapter, we start with an overview of the different ways in which images can be described and compared. Then, we review the literature on the visualisation and exploration of image collections.

Contents

3.1 Image description and comparison

In order to have a relevant representation of an image collection, one first needs to describe and compare images. Describing and comparing images are challenging and important steps in several fields such as image analysis, computer vision, pattern recognition and machine learning. To address these challenges, several solutions can be found in the literature.

3.1.1 Image description

Depending on the source of the images, one can consider various types of information:

1. *Temporal and geographic descriptors.* Usually stored in the metadata, temporal information (date and time) and location information (geotags) can be used to characterise images. In the early 2000s, the EXIF format (Exchangeable image file format) includes these information in the generated file description of digital images. Thus, most recent digital images have such information available in their metadata.

2. *Concepts-based descriptors.* Here, textual information is used. The text can be obtained from images keywords, descriptions or annotations. The most well-known example is Google Images, where images are indexed using keywords. Another example is the museum online galleries one, where a description of the artwork is often available for each image. Recently, thanks to the usage of *deep learning*, a major breakthrough has been achieved in the automatic description generation of natural images [Bernardi et al., 2016].

3. *Content-based descriptors.* In this case, one takes advantage of information that can be extracted from the images themselves. These information can then be embedded in either statistical, structural or transformation-based descriptors. These include but are not limited to:

   - colour, texture or shape based,
   - histogram of gradients based: SIFT [Lowe, 1999], SURF [Bay et al., 2008] and GLOH [Mikolajczyk and Schmid, 2005] ,
   - binary based: BRIEF [Calonder et al., 2010], ORB [Rublee et al., 2011], BRISK [Leutenegger et al., 2011] and FREAK [Alahi et al., 2012].

   We will not go into details on all the aforementioned descriptors in this manuscript. Only the ones that have been used in our work will be described later. Readers can refer to the original papers describing the others descriptors.

3.1.2 Image comparison

The choice of the image (dis)similarity metric to compare images depends a lot on the the descriptors used to characterise images. For instance:

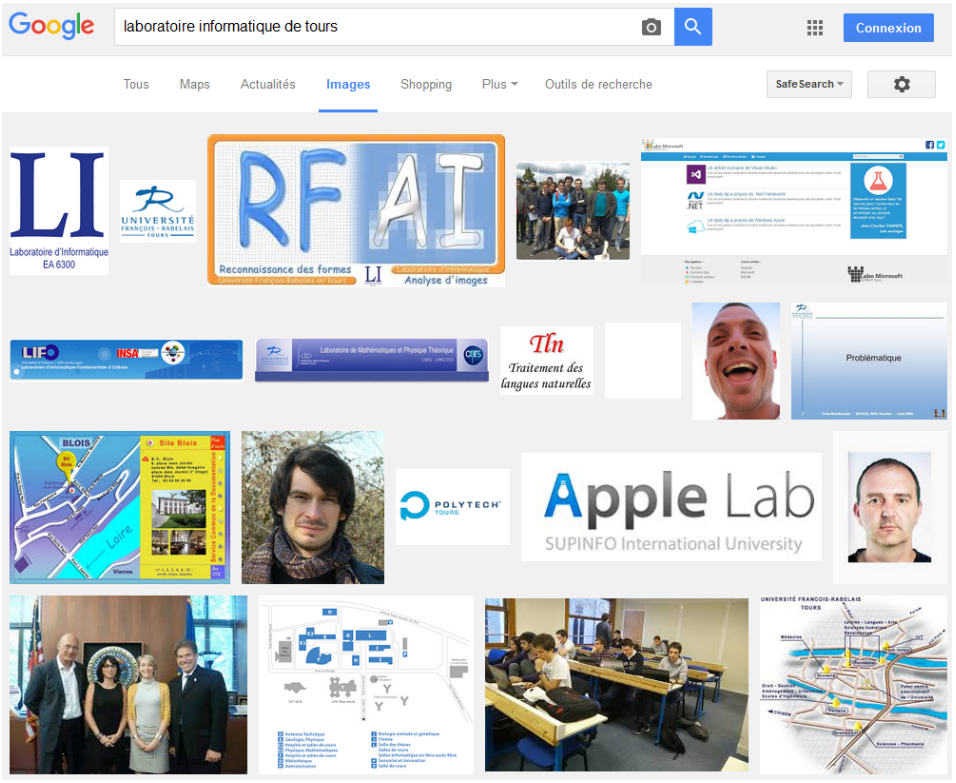


Figure 3.1: Plain grid visualisation. Images results of Google Images for the query "Laboratoire Informatique de Tours".

- Manhattan or Euclidean distances can be used to compare numeric descriptors such as timestamps, numeric vectors or more complex features.
- Cosine similarity [Singhal, 2001] can be used for textual descriptors.
- Earth mover's distance [Rubner et al., 1998] performs well to compare histograms.

3.2 Image collection visualisation

The most common visualisation of image collections, popularised by text-based query image search systems, is a plain grid. Images are simply laid out in a grid, and the user can scroll down to browse the collection. Figure 3.1 shows such a visualisation.

Let us assume that temporal or location information are available for each image of a collection, then one can take advantage of familiar visualisations that users are familiar with, such as timelines and geographic maps [Nollenburg, 2007],[Jaffe et al., 2006]. Figure 3.2 illustrates a map-based visualisation.

Now, let us assume that only content-based descriptors are available. Then, an image can be described by a set of d descriptors: it can be considered as a point in the descriptors space. Therefore, the visualisation of image collections boils down to the visualisation

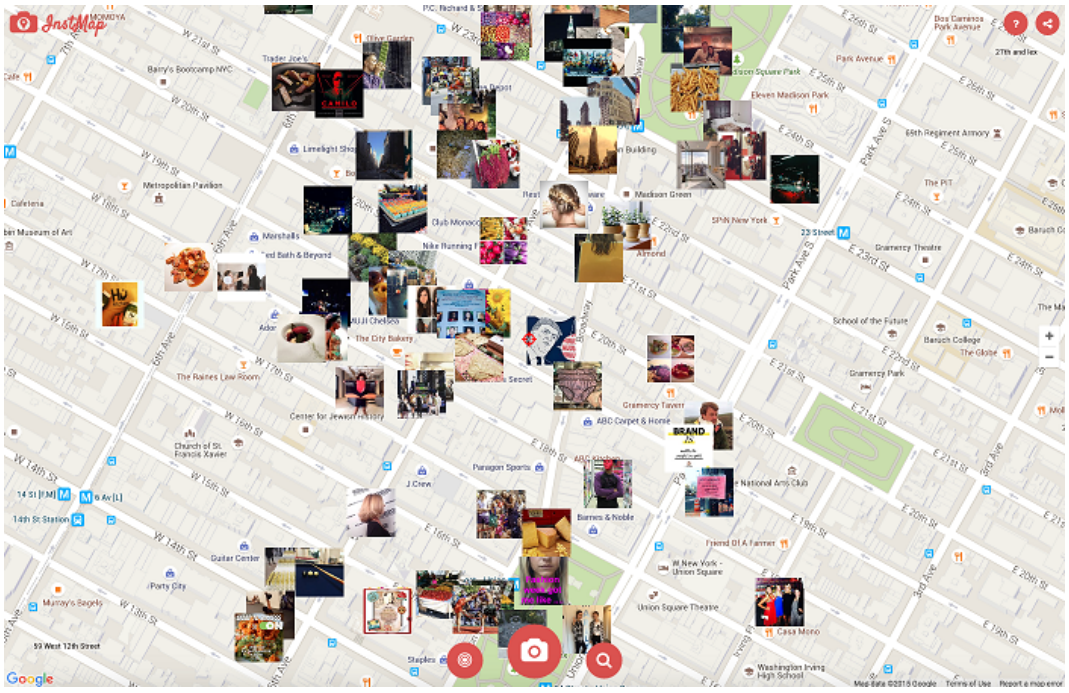


Figure 3.2: Map-based visualisation. Public photographies posted in Instagram in a district of New York.

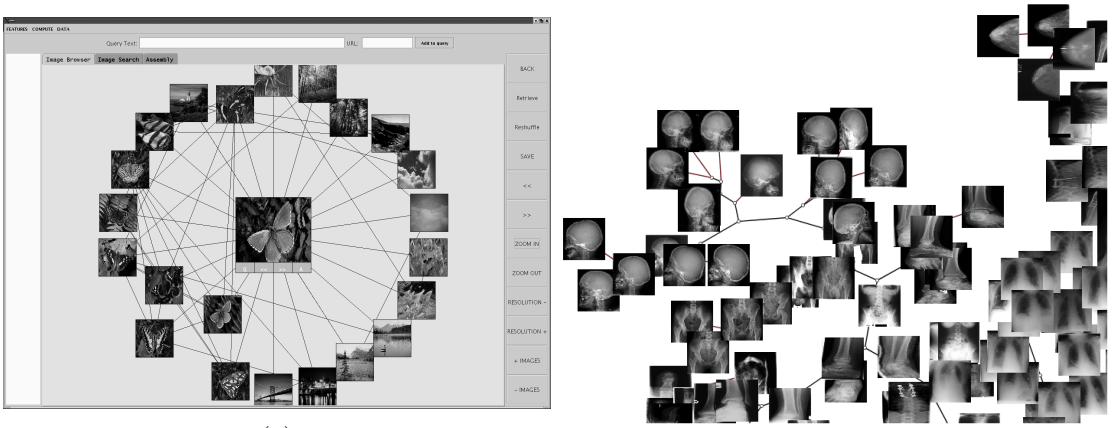
of a d-dimension point cloud. Thus, images will be arranged according to their mutual similarity: images that are similar are placed together. This organisation has been proved to be useful for the image browsing [Rodden et al., 2001]. Following this observation, three major paradigms have been proposed in the literature to visualise image collections:

1. *graph-based* approaches,
2. *clustering-based* approaches,
3. *projection-based* approaches.

We describe below the key ideas behind each paradigm. A more thorough survey on this topic can be found in [Camargo and González, 2009] and [Plant and Schaefer, 2011].

3.2.1 Graph-based

In these approaches, an image graph is built. Each image is assigned to a node and an edge between two nodes corresponds to the (dis)similarity of the related images. Thus, it boils down to wonder about which edges should be created and which graph drawing algorithm should be used. Most of the proposed solutions rely on k-nearest neighbour graphs ([Ding et al., 2015] and [Heesch and Rüger, 2004]). Figure 3.3a presents an image graph from [Heesch and Rüger, 2004]. In [Chen et al., 2000], the authors propose to use PathFinder networks, that have been originally developed for the analysis of proximity data in psychology. Only small collections (<500 images) are used in these works.



(a) (b)

Figure 3.3: Graph-based visualisations. (a) In [Heesch and Rüger, 2004], a NN^k graph is visualised and (b) part of the visualisation of 537 X-ray images using Pex-Image.

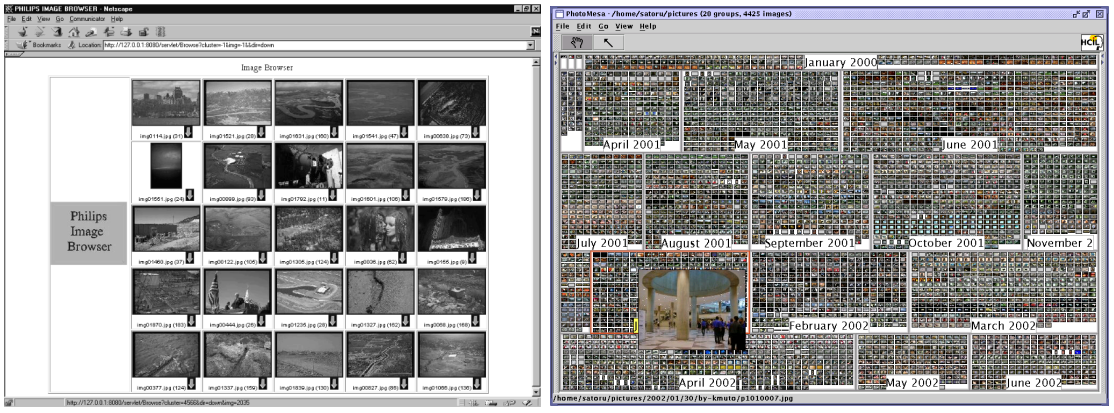
In [Eler et al., 2009], the authors present PEx-Image, where a tree that has been originally developed for phylogenetic, namely the neighbour joining (NJ) tree, is built. This method gives good visual results. However, it requires that the similarity between each pair of images must be pre-computed and stored in a distance matrix. This requirement implies a huge memory availability for collections of tens of thousands of images, and is intractable for million image collections. Moreover, only static visualisation are given for graphs of thousands nodes. Figure 3.3b displays the visualisation of 537 X-ray image using Pex-Image.

The main limitation with graph-based visualisations is the difficulty to draw a useful and interactive graph when either the number of nodes or edges is large. With the growing interest in the visualisation of large graphs, many researchers from the Graph Drawing field proposed original techniques to improve node-link graph representations. The two most used paradigms are:

1. *multilevel graph coarsening*, by the mean of node clustering, in order to reduce the amount of nodes to display in the screen, as in [Abello et al., 2006]. The clustering can be based on the structure of the graph itself or depending on the data and their content.

2. *edge bundling*, by the mean of edge clustering. As visualisation of thousand of edges is often a visual clutter (hairball effect [Kosara, 2012]), the edge bundling method spatially groups edges to improve the visibility ([Holten, 2006], [Gansner et al., 2011]).

Thus, on the one hand, current systems focus on nicely visualizing small or medium-sized graphs. On the other hand, most of the recent academic techniques in large graph visualization focuses on providing static drawings of the graphs.



(a) (b)

Figure 3.4: Clustering-based visualisations. (a) In [Krishnamachari and Abdel-Mottaleb, 1999] software, the first level is displayed, and the user can click on a representative image to view the set of images assigned to its cluster. (b) PhotoMesa software displaying 4425 images placed in 20 folders.

3.2.2 Clustering-based

The key idea of clustering-based approaches is to reduce the amount of images to display. Thus, similar images are brought together to form homogeneous *cluster*. Clusters' representatives are then selected to be displayed.

In [Krishnamachari and Abdel-Mottaleb, 1999] and [Chen et al., 1998], a hierarchical agglomerative clustering [Jain and Dubes, 1988] is used to yield a hierarchical structure. This structure is then used to produce a multilevel visualisation (see Figure 3.4a). However, the distance matrix is pre-computed and stored before performing the clustering. As mentioned previously, it is intractable for collections of million images.

In [Bederson, 2001], PhotoMesa presents an interface to navigate in a hierarchical structure of the collection on a 2d plane using the zoom interaction. In [Gomi et al., 2008], the authors extend the work of PhotoMesa to image collections using both image keywords and visual descriptors. They experimented only on a collection of 2360 images. Figure 3.4b presents the view of PhotoMesa.

In [Wang et al., 2015], the authors provide an interesting interactive visualisation using spiral layout (see Figure 3.5). However, they have tested their system only on 4560 images. Furthermore, the clustering needs to be done each time the collection is modified (*e.g.* the addition of an image), thus it does not handle well dynamic collections.

3.2.3 Projection-based

These types of visualisations fall into *dimensionality reduction* techniques. The idea is to project all the images that lie in the original d-dimension descriptor space in a two-dimensional (or three-dimensional) space. Such projections shall preserve the topology of the points in the original space. This is motivated by the fact that the human visual system

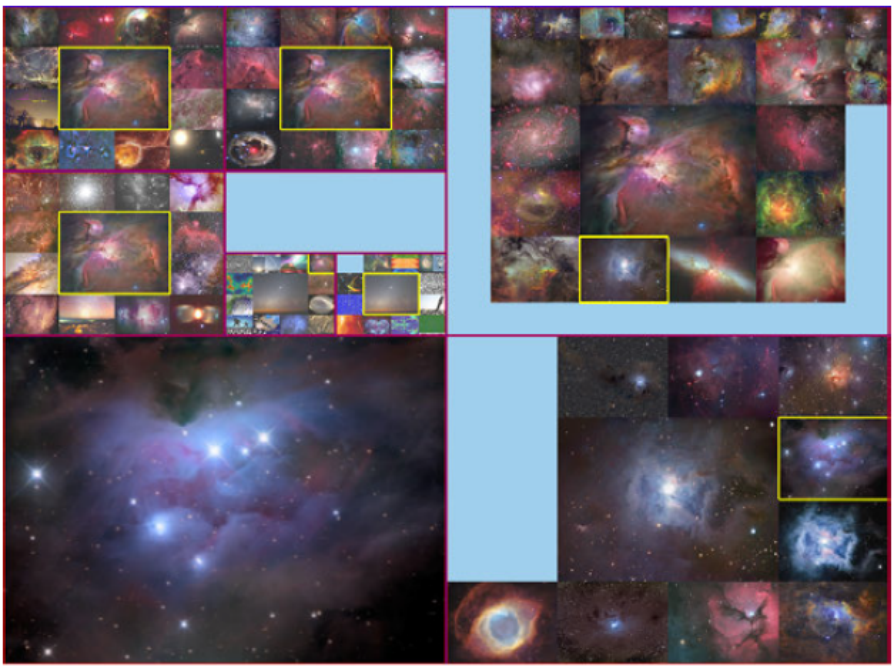


Figure 3.5: Clustering-based visualisation. iMap multilevel visualisation using a spiral layout.

can handle well low dimensional spaces and grasps the structure of the projection.

Notable projections are:

- the principal components analysis (PCA) [Jolliffe, 1986],
- the multidimensional scaling (MDS) [Cox and Cox, 2000],
- the locally linear embedding (LLE) [Roweis and Saul, 2000]
- the t-distributed stochastic neighbour embedding (t-SNE) [Van der Maaten and Hinton, 2008].

A global 2D map is built, in which image thumbnails can be displayed (see Figure 3.6 (left)). A drawback of these methods are overlapping thumbnails, that damage visibility. However, this is addressed in [Liu et al., 2004]: thumbnails are rearranged to reduce the overlapping as much as possible. A common method is to make the thumbnails fit a 2D grid. Figure 3.6 (right) illustrates such a rearrangement. Self-organising maps (SOM) [Kohonen et al., 2001] is another technique that produces directly a layout map that has few overlaps for image collection visualisation [Deng, 2007] (see Figure 3.7).

The main limitation of these methods is that they perform very well for small image collections (< 1000). However, they do not always scale well in terms of computation, hence they do not handle large image collections. t-SNE, prize-winning technique for dimensionality reduction, has been used to visualise 50,000 images [Karpathy, 2015]. However, the

visualisation obtained is a static grid. Moreover, one can wonder about the relevance of visualising simultaneously ten thousands of images in the 2D plane, especially in term of interaction.

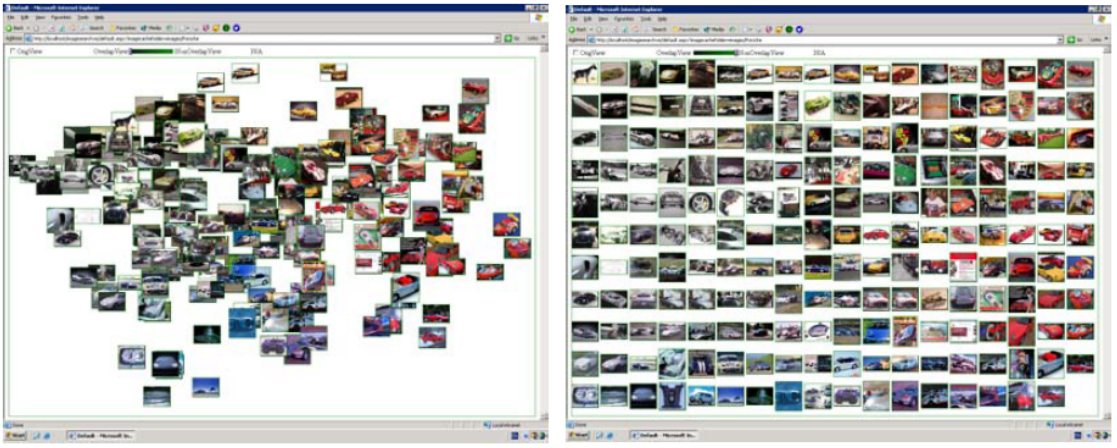


Figure 3.6: Projection-based visualisation. Rearrangement from MDS visualisation (left) to 2D grid visualisation (right) to avoid image overlaps. Figure from [Liu et al., 2004].

Figure 3.7: Projection-based visualisation. Self-organising maps image collection visualisation.

3.2.4 Others

A few other methods, that do not fall on the previous categories, can also be found in the literature.

Coordinated and multiple views visualisation It is based on classical data analysis visualisations, such as histograms, scatter plots, or parallel coordinates. The added value lies in the fact that they are shown simultaneously and that an interaction on one of them impacts the others, as in [Matković et al., 2009] (see Figure 3.8).

Unconventional visualisations In [Porta, 2006], the authors present unconventional experiments for images browsing, mainly using animations (*e.g.* fade-in, fade-out). However, they have tested the proposed visualisations on two collections of only 200 and 1,000 images (see Figure 3.9).

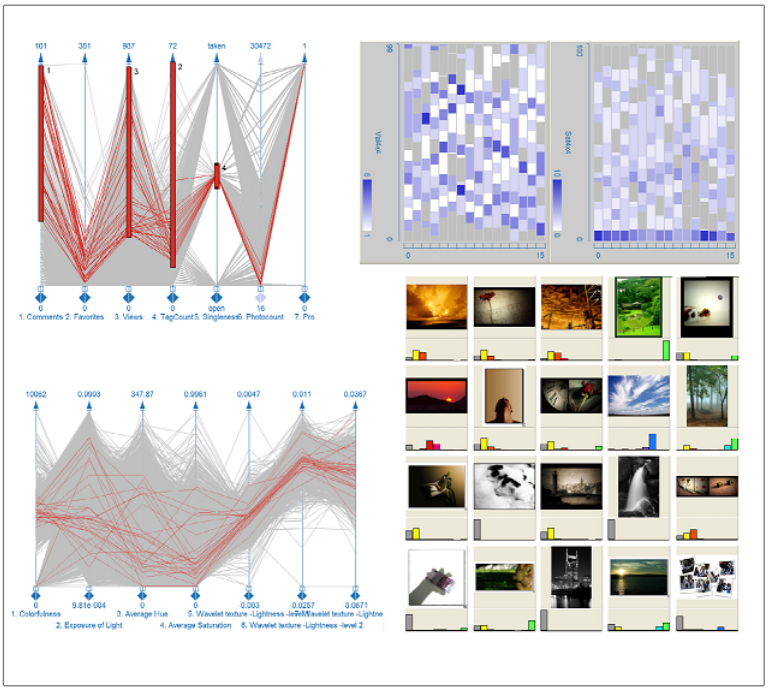


Figure 3.8: Visualisation using coordinated and multiple views approach.

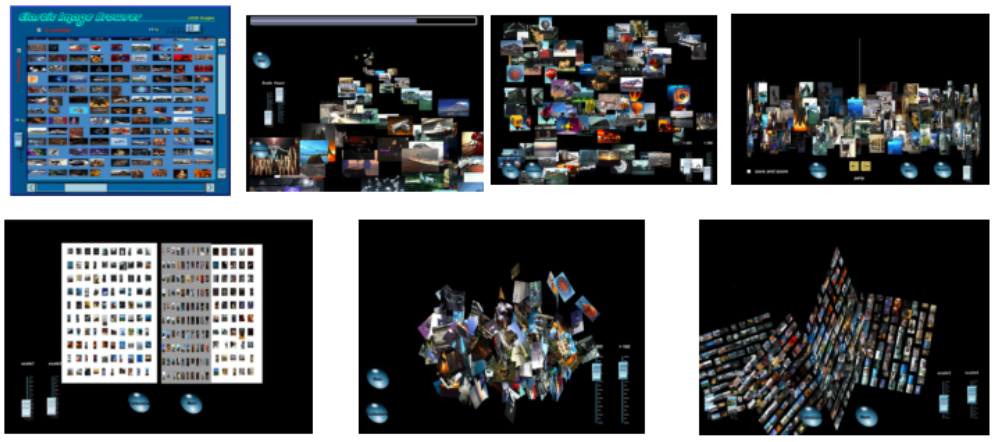


Figure 3.9: Unconventional visualisations. From top left to bottom right: elastic image browser (grid display), shot display (image go through the browser from top to bottom), spot display (images appear at random positions), cylinder display, rotor display, tornado display and planes of tornado display.

Chapter 4

Synthesis

In this chapter, we contextualise our motivations and our work with respect to the literature. In what extent do we contribute to the state of the art? In what extent do we take advantage of existing works?

This thesis endeavours to allow one to explore large and ever-growing image collections. By explore, we mean to have insights from the collections or their underlying structures, and possibly to extract knowledge.

We address two general use cases where:

1. one has a peculiar goal and knows (or not) about his image collection
   $\Rightarrow$ *objective-oriented* exploration
2. one does not have any objective and just aims to wander around his known (or not) image collection
   $\Rightarrow$ *serendipity-oriented* exploration

Several challenges to transform data into knowledge are addressed, especially:

(i) handling large image collections,

(ii) handling ever-growing image collections,

(iii) providing interactive means to explore image collections.

To do so, we take advantage of Information Visualisation techniques, that have been proved useful for having relevant insights from data. We pay attention to the possible shortcomings and pitfalls of such techniques. To address the third constraint aforementioned, we will focus on the interactivity property of the proposed visualisations, to involve the user in the exploration process. Hence, we will conduct a joint study on the indexing and the visualisation of the image collections.

In order to manage available online image collections, the present work relies on the images themselves. We use content-based descriptors and fall into step with MPEG-7 [Pereira and Koenen, 2001] standard by using Colour Layout Descriptor [Manjunath et al., 2001] (CLD) or Edge Histogram Descriptor [MPEG, 2000] (EHD), respectively colour and texture descriptors. These descriptors are detailed in Appendix C. The euclidean distance will be used to compare the numeric vectors describing two images. Note that we will not discuss the choice of the descriptors and distances in this work because it is not the main point of the thesis. However, the presented work can be used along with others image descriptors and others metrics for image comparison.

As seen in the previous chapter, related works to visually explore image collection exist in the literature. However, as shown in the Table 4.1, they do not satisfy the three aforementioned constraints simultaneously. First, most of the reviewed works used small collections ($<$5000 images). Only a few works have been experimented on medium data sets ($\approx$ 50,000 images), but they only provide static visualisations. Second, none of the works addresses the dynamic aspects of image collections. A similarity matrix is almost always built from all the images in the collection, which is intractable for larger collections of millions images. Third, to a certain extent, most of the works embed interactivity in the proposed visualisations, but only for the small data sets experimented on.

Therefore, we propose a solution that takes advantage and combines the best of the three paradigms to fulfil our constraints:

- *Graph-based.* A proximity graph, namely the Relative Neighbourhood Graph, will be used to index images in a relevant structure (Chapter 5).
- *Clustering-based.* A data partitioning algorithm, namely BIRCH, will be used to yield a viewable hierarchical and graph-based hybrid structure from an image collection (Chapter 6).
- *Projection-based.* Projection-based graph drawing algorithms will be favoured in the visualisations embedded in our custom image collection exploration platform (Chapter 7).

Table 4.1: Comparison of the existing works to visually explore image collections with regard to our constraints. In the *'Large collection'* column, by large, we mean at least 1,000,000 images. The number in brackets indicates the maximum number of images used in the experiments.

		Large collection	Ever-growing collection	Interactive exploration
Graph	[Ding et al., 2015] k-NN graph	No (400)	No	Yes
	[Heesch and Rüger, 2004] NNk network	No (32,000)	No	Yes
	[Chen et al., 2000] PathFinder network	No (279)	No	Yes
	[Eler et al., 2009] NJ tree	No (9000)	No	Yes
Clustering	[Krishnamachari, 1999] Agglomerative clustering	No (3856)	No	Yes
	[Chen et al., 1998] Agglomerative clustering	No (10,000)	No	Yes
	[Bederson, 2001] PhotoMesa	No (4425)	No	Yes
	[Gomi et al., 2008] Agglomerative clustering	No (2360)	No	Yes
	[Wang et al., 2015] Agglomerative clustering	No (4560)	No	Yes
Projection	[Liu et al., 2004] MDS	No (3400)	No	Yes
	[Deng, 2007] SOM	No (1380)	No	No
	[Karpathy, 2015] t-SNE	No (50,000)	No	No

Part II

Contributions

Chapter 5

Incremental construction of the Relative Neighbourhood Graph

Abstract

In this chapter, we aim at structuring large and ever-growing image collections in a graph. To this end, we use a proximity graph, namely the Relative Neighbourhood Graph (RNG). This graph is defined and its choice is justified. To address the large and ever-growing constraint, we propose to yield this graph incrementally. An existing work is presented and discussed. Then, we propose approaches to build incrementally an exact RNG and an approximate RNG. These contributions are experimented on real-world data sets, up to a million entries, and the results are discussed to draw lessons from this work.

Contents

5.1 Proximity graphs

5.1.1 Graphs: definitions and notations

Let us first introduce some common graph notations and definitions, that will be used throughout the manuscript.

A *graph* $G = (V, E)$ is defined by the set $V = V(G)$ of *vertices* (or *nodes*), and $E = E(G)$ the set of *edges* between vertices of V.

An *undirected graph* is a graph in which each edge in E has no *direction*. Thus, the edge $\overline{vw}$ is the same as the edge $\overline{wv}$.

A *loop* is an edge that connects a vertex v to itself.

A *simple graph* is an undirected graph that has no loop, and where each pair of vertices $(v, w) \in V \times V$ can have *at most* one edge.

A *weighted graph* is a graph where a value, the *weight*, is assigned to each edge $e \in E$. The expression *cost*, *length* or *capacity* can also be found in the literature.

A *path* between two vertices v and $w \in V$ is a sequence of edges that connect v to w.

A graph is *connected* if a path exists between each pair of vertices $(v, w) \in V \times V$.

The degree of a vertex $v \in V$, noted $deg(v)$, is the number of *incident* edges, *i.e.* edges that have v as an endpoint.

5.1.2 Proximity graphs: definition

Proximity graphs [Toussaint, 1991] are simple weighted graphs. They aim at extracting the structure of a data point set $D \subset \mathbb{R}^d$, where each point is represented by a node. They associate an edge between two points of D if they are close enough to be considered as neighbours. Depending on how one defines the notion of neighbourhood, different proximity graphs can be built from the same data point set. In [Lankford, 1969], some definitions of neighbourhood are studied. The notable proximity graphs include:

- the k-nearest neighbour graph (k-NNG),
- the relative neighbourhood graph [Toussaint, 1980] (RNG),
- the Gabriel graph [Gabriel and Sokal, 1969] (GG),
- the Delaunay graph [Delaunay, 1934] (DG).

An elegant and well-known property [Urquhart, 1983] links these proximity graphs. For a given data set D, we have:

$$1\text{-}NNG(D) \subset RNG(D) \subset GG(D) \subset DG(D)$$

In the present work, we focus our attention on the RNG. We first define it thoroughly and then justify this choice in the next section.

5.2 Relative neighbourhood graph

5.2.1 Definition

The relative neighbourhood graph has been introduced in the work of G. Toussaint [Toussaint, 1980]. The construction of this graph is based on the notion of *relatively close* neighbours, that defines two points as relative neighbours if "*they are at least as close to each other as they are to any other points*". From this definition, we can define $RNG = (V, E)$ as the graph built from the points of D where distinct points p and q of D are connected by an edge $\overline{pq}$ if and only if they are relative neighbours. Thus,

$$E(RNG) = \{\overline{pq} \mid p, q \in D, p \neq q, \delta(p, q) \leq max(\delta(p, r), \delta(q, r)), \forall r \in D \backslash \{p, q\}\}.$$

where $\delta : D \times D \rightarrow \mathbb{R}$ is a distance function. An illustration of the *relative neighbourhood* of two point $p, q \in \mathbb{R}^2$ is given in Figure 5.1.

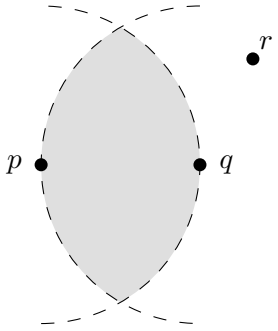


Figure 5.1: Relative neighbourhood (grey area) of two points $p, q \in \mathbb{R}^2$. If no other point of D lays in this neighbourhood, then p and q are relative neighbours.

One can use a straightforward and brute-force construction algorithm to build a RNG (see Algorithm 1). This algorithm has a complexity of $O(n^3)$, where $n = |D|$ is the number of data points.

5.2.2 RNG choice explanation

The RNG has been used in several works in various fields such as computer vision, geographic analysis or pattern classification [Toussaint, 2014]. Its main benefit is that it is the smallest *connected* proximity graph that highlights the topology of the data and embeds local information about vertex neighbourhood. We give in Appendix B, the proof that the RNG is a connected graph. This connectivity property guarantees that each image can be reachable during a content-based exploration of the image collections. Moreover, it is a desirable property for graph drawing algorithms.

The choice of the RNG over others proximity graphs is justified as follows. First, it has been selected over the $k - NNG$ to ensure the connectivity property. Indeed, the

Algorithm 1 Classical RNG construction algorithm (in $O(n^3)$).

Input: D
Output: $RNG = (V, E)$

```
1: V = D
2: E = ∅
3: isEdge = True
4: for each p ∈ V do
5:    for each q ∈ V do
6:       for each r ∈ V do
7:          if ( δ(p,r) < δ(p,q) ) && ( δ(q,r) < δ(p,q) ) then
8:             isEdge = False
9:             break
10:         end if
11:      end for
12:      if isEdge then
13:         E = E ∪ {pq}
14:      end if
15:   end for
16: end for
17: return RNG = (V, E)
```

$k - NNG$ is not always connected. Second, it has been selected over the DG because the later construction complexity is almost exponential for high dimensional data points. Therefore, it is not possible to build DG for such data. Third, it has been selected over the GG because the former is sparser. This allows to have a less complex graph in terms of edges when processing large data sets.

The main drawback of the RNG is its construction complexity. As mentioned above, the classic algorithm to yield a RNG has a cubic complexity. Hence, it is the main obstacle to use it for large data sets. Few works in the literature address this complexity for 2D and 3D points. Their key idea is to build a supergraph of the RNG (*e.g.* the Delaunay graph), and adopt a strategy to eliminate some edges to yield the RNG. Thus, one can find in the literature [Jaromczyk and Toussaint, 1992] algorithms for 2D and 3D points, whose complexity are $O(nlog(n))$ and $O(n^{\frac{23}{12}}log(n))$, respectively. Other works tried to address the RNG construction for higher dimensional data sets. In [Liu et al., 2004], the authors propose to use GPU computing power to build the RNG. They present experimentations on 293,769 entries, described by 793 features, but state that their algorithm could not be used for some higher dimensions because of the matrix storage memory requirement. Goto et al. [Goto et al., 2015] propose a quadratic algorithm to yield the RNG. However, the produced RNG is not evaluated in terms of accuracy and they do not deal with dynamic collections.

Hence, the RNG construction seems to be the main obstacle to use it for large image collections. Moreover, all the images are not necessarily available at first: they can come in a continuous way. These observations lead us to think about an incremental algorithm to create a RNG.

5.3 Related work

To the best of our knowledge, only one series of works can be found in the literature on the incremental construction of the RNG: the one of Djamel A. Zighed, Hakim Hacid and al.. Their main works are described below.

Scuturici and al. [Scuturici et al., 2003] state the first experimentations on incremental RNG (iRNG) for queries in image databases. In this paper, no *real updates* are performed. The relative neighbours of the query are computed and associated edges are added in the graph. However, the other edges which should have been updated stayed unchanged. In [Scuturici et al., 2004] from the same authors, the graph is still not really updated. A simple heuristic is proposed to assign neighbours to the newly inserted point: if we consider the new point q to add, and its nearest neighbour $nn(q)$, then the neighbours of $nn(q)$ in the graph are assigned to q.

In [Hacid and Zighed, 2005], the authors propose *"an algorithm to perform local update"* of an RNG following the insertion of a new point. This algorithm is used to incrementally build the RNG. It is detailed and discussed in this section.

5.3.1 Hacid et al. work

In [Hacid and Yoshida, 2007], the authors propose an algorithm to perform local update of a RNG following the insertion of a new vertex. This algorithm is used to incrementally build the RNG. Given a set of n vertices V, the incremental construction of the RNG that is proposed by Hacid et al. consists in *(i)* randomly selecting 2 vertices of V and creating an edge between them and *(ii)* iteratively inserting the other vertices by locally updating the RNG. The insertion algorithm (Algorithm 2) is detailed below.

Let RNG be the relative neighbourhood graph that is built from the vertices of V, q be a new vertex to insert, and $\epsilon \in [0;\ 1]$. First, the nearest vertex nn of q is sought in V (line 1). The farthest relative neighbour fn of nn is retrieved in the graph RNG (line 2). A hypersphere SR that is centred around q is then computed as its neighbourhood. All vertices that lay in that hypersphere are retrieved (lines 6-11). The radius of this hypersphere corresponds to the sum of the distance between q and nn, and the one between nn and fn. Note that this hypersphere radius can be magnified thanks to the parameter ϵ (line 3). The neighbourhood relationships of the hypersphere SR are updated (line 12) with the classic brute-force algorithm in $O(n'^3)$, with $n' = |SR|$.

The complexity of this insertion algorithm is $O(2n+n'^3)$, where $n = |V|$ and $n' = |SR|$. The $2n$ term corresponds to the search of the nearest neighbour and the search of vertices that lay in the hypersphere. The second term is the time for updating the neighbourhood relations between the points within the hypersphere with the classic RNG algorithm.

5.3.2 Discussion

The authors state that the incrementally built RNG corresponds exactly to the RNG that is built with a brute-force algorithm, using graph correspondence.

Algorithm 2 Hacid et al.'s insertion algorithm

Input: $RNG = (V, E)$, q, ϵ
Output: $RNG' = (V', E')$

```
nn = nearest_vertex(q, V)
fn = farthest_relative_neighbour(nn, RNG)
sr = (δ(q, nn) + δ(nn, fn)) · (1 + ε)
V' = V ∪ {q}
E' = E
SR = ∅
for each p ∈ V' do
    if δ(p, q) ≤ sr then
        SR = SR ∪ {p}
    end if
end for
E' = Update(SR)
return RNG' = (V', E')
```

However, we have noticed several drawbacks regarding this insertion algorithm. First, the choice of the parameter ϵ, that is used by the authors to expand the neighbourhood of q. It is empirically set at $\epsilon = 0.1$ in [Hacid and Yoshida, 2007]. However, no proof that relative neighbours of the newly inserted point must lay in this magnified hypersphere is given. This could be the cause of losing relative neighbours as illustrated in Figure 5.2.

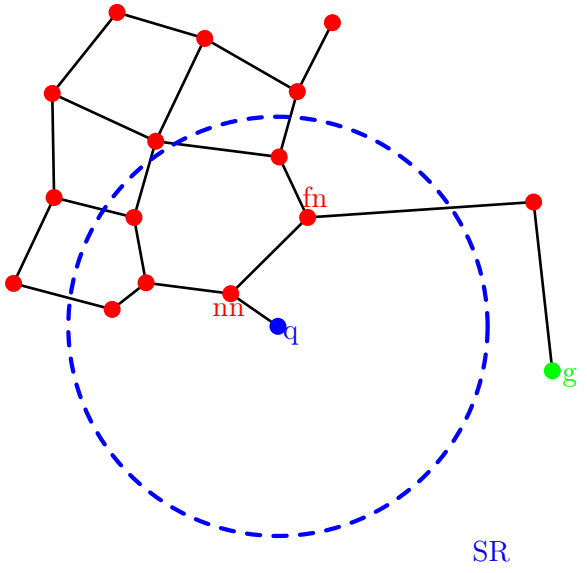


Figure 5.2: Vertex g is a relative neighbour of vertex q. The insertion algorithm of Hacid et al. fails to retrieve this relationship because g does not lay in the (dashed blue) neighbourhood SR of q, due to ϵ value (0.1 in this toy example).

Second, due to the spherical definition of the neighbourhood SR, the update step may create false edges. Indeed, the classical RNG algorithm is performed only considering the vertices laying in SR. Figure 5.3 illustrates such a erroneous edge creation. Thus, the insertion algorithm that is described above might not incrementally yield the exact RNG, as stated by the authors, due to the loss of edges or the inclusion of bad ones. This has been observed and reported in the experimentation section. Third, an assumption is done stating that $n' << n$, *i.e.* the number of vertices in the hypersphere is less than the number of previously added points. For instance, it may not be the case if a set of dense points laying in the same part of the space is considered. This has been experimentally observed for one data set. Thus, the term n'^3 in the complexity of the algorithm that is proposed by Hacid et al. might be an issue.

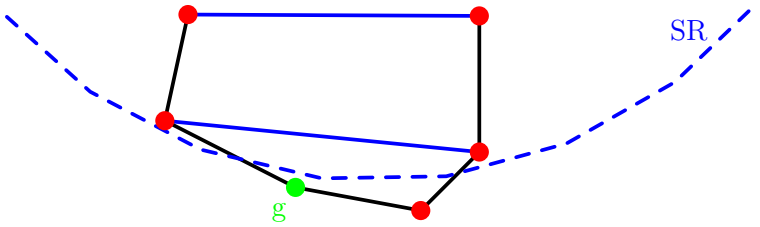


Figure 5.3: False (blue) edges are created. Indeed, as the green point g does not lay in the (dashed blue) neighbourhood SR, it was not considered during the creation of the blue edges.

Hence, these observations lead us to think that the Algorithm 2 might not yield incrementally the exact RNG. In the next sections, we present our contributions to address these drawbacks:

1. First, we propose a formal proof to gather the edges that might be impacted by the insertion of a new data. Based on this proof, we propose an exact insertion algorithm and the related algorithm in order to incrementally build an exact RNG (Section 5.4).

2. Second, we define an edge-based neighbourhood and propose a strategy that takes advantage of it to locally update an RNG, following the insertion of a new data. This strategy is then used to incrementally yield an approximate RNG (Section 5.5).

5.4 Exact approach

5.4.1 Concept

We try to characterise an edge of an existing RNG that might be impacted by the insertion of a new point in the graph. Let D be a set of points in $\mathbb{R}^d$ at an instant t, we

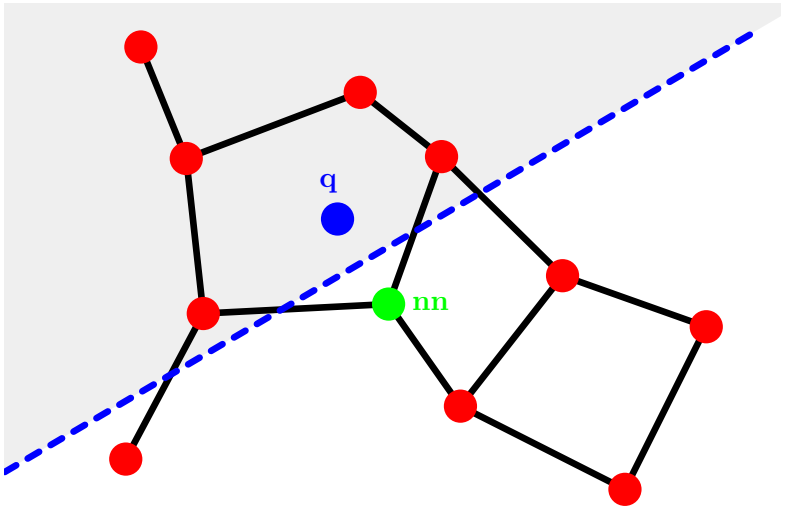


Figure 5.4: The equation (E2) $\delta(\alpha, q) \leq \delta(\alpha, nn)$ boils down to consider vertices α that lay above the dashed blue line, in the grey area, *i.e. scan a half-space.*

note:

$$\begin{aligned} &RNG^t = RNG(D), \\ &V^t = V(RNG^t) = D, \\ &E^t = E(RNG^t). \end{aligned} \tag{5.1}$$

At $t+1$, we consider the insertion of point $q \notin V^t$ (*i.e.*, we do not consider duplicates). Let us consider $\alpha, \beta \in V^t$, $\overline{\alpha\beta} \in E^t$, such as $\overline{\alpha\beta}$ should be removed at $t+1$ by the insertion of q.
$\overline{\alpha\beta} \in E^t$, so:

$$\forall r \in V^t \backslash \{\alpha, \beta\}, \delta(\alpha, \beta) \leq \max(\delta(\alpha, r), \delta(\beta, r)). \tag{5.2}$$

$\overline{\alpha\beta}$ should be removed at $t+1$ because of q, so:

$$\begin{cases} \delta(\alpha, q) < \delta(\alpha, \beta) \\ \delta(\beta, q) < \delta(\alpha, \beta). \end{cases} \tag{5.3}$$

Let nn, be the nearest neighbor of q in V^t, if, $\alpha \neq nn$, and $\beta \neq nn$, then we have from (5.2):

$$\delta(\alpha, \beta) \leq \max(\delta(\alpha, nn), \delta(\beta, nn)). \tag{5.4}$$

Using (5.4) in (5.3), we have the equations:

$$\begin{cases} \delta(\alpha, q) \leq \max(\delta(\alpha, nn), \delta(\beta, nn)) \\ \delta(\beta, q) \leq \max(\delta(\alpha, nn), \delta(\beta, nn)). \end{cases} \tag{5.5}$$

Let $\delta(\alpha, nn) \geq \delta(\beta, nn)$ (E1), the two equations above boil down to:

$$\begin{cases} \delta(\alpha, q) \leq \delta(\alpha, nn) & (E2) \\ \delta(\beta, q) \leq \delta(\alpha, nn)). & (E3) \end{cases}$$

Algorithm 3 Local Update Strategy to build incrementally the exact RGN

INPUT: $G = (V, E)$ // Existing RNG graph
INPUT: q // Point to insert

1: $nn \Leftarrow nearest\_neighbors(q, V)$
2: **for** all vertex p in V **do**
3: $compute\_distance(nn, p)$
4: **end for**
5: $candidates \Leftarrow \{p \in V, \delta(p, q) \leq \delta(p, nn)\}$
6: $candidates \Leftarrow candidates \cup \{nn\}$
7: STEP_1: Update_incident_edges(candidates,G)
8: STEP_2: Create_relative_neighbors(q,G)

Algorithm 4 Update_incident_edges

INPUT: *candidates* // candidates
INPUT: $G = (V, E)$ // Existing RNG graph

1: **for** all vertex c in candidates **do**
2: $N \Leftarrow$ neighbours(c,G)
3: **for** all vertex v in N **do**
4: **if** $\delta(c, nn) \geq \delta(v, nn)$ && $\delta(c, q) \leq \delta(c, nn)$ **then**
5: **if** $\delta(q, c) < \delta(c, v)$ && $\delta(q, v) < \delta(c, v)$ **then**
6: remove_edge($\overline{cv}$,E)
7: **end if**
8: **end if**
9: **end for**
10: **end for**

Thus, any edge $\overline{\alpha\beta}$ that may be potentially invalidated by the insertion of new point q must verify the equations (E1), (E2) and (E3).

Figure 5.4 illustrates equation (E2).

5.4.2 Algorithm

From the above considerations, we can deduce a local update algorithm (Algorithm 3) that incrementally produces the exact RNG.

Line 5 of the Algorithm 3 first takes advantage of the equation (E2) to gather potential candidates, in terms of vertex, whose incident edges may be invalidated. Then, equations (E1) and (E3) are used in Algorithm 4 (line 4) to reduce the number of edges to process in the update operation.

The complexity of Algorithm 4 is $O(n_c \cdot deg)$ where $n_c = |candidates|$, and deg is the average degree of the graph. This leads to a local update Algorithm 3 in $O(2n + n_c \cdot deg + n^2)$.

The above local update algorithm successfully allows to incrementally yield the exact RNG, however its complexity is not so low any more. First, the equation (E2) $\delta(\alpha, q) \leq \delta(\alpha, nn)$ boils down to scanning an $\mathbb{R}^d$ half-space for potential candidates α, thus n_c can end up being quite large. Second, the computation of the relative neighbours of q, in $O(n^2)$ is quite problematic. Overall, although the algorithm can be used for medium-scale data sets, its complexity does not fit well with large data sets. Hence, we propose in the next section, an incremental approach to yield a refined approximation of the exact RNG, with a lower complexity than the existing work.

5.5 Approximative approach

5.5.1 Edge-based neighbourhood for local update

Let us consider a simple weighted graph $G = (V, E)$, and a vertex $q \in V$. We introduce the notion of *neighbourhood order*, and recursively define the l^{th}*-order vertex neighbours* (Eq. 5.6) and the l^{th}*-order edge neighbours* (Eq. 5.7) of a vertex q. Such sets are illustrated in Figure 5.5.

$$\begin{cases} N^1(q) = \{p \in V \mid \overline{pq} \in E\} \\ N^l(q) = \{p \in V \mid \overline{pr} \in E, r \in N^{l-1}(q)\}, \text{ for } l > 1 \end{cases} \tag{5.6}$$

$$\begin{cases} N_e^1(q) = \{\overline{pq} \in E \mid p \in N^1(q)\} \\ N_e^l(q) = \{\overline{pr} \in E \mid p \in N^l(q) \text{ and } r \in N^{l-1}(q)\} \end{cases} \tag{5.7}$$

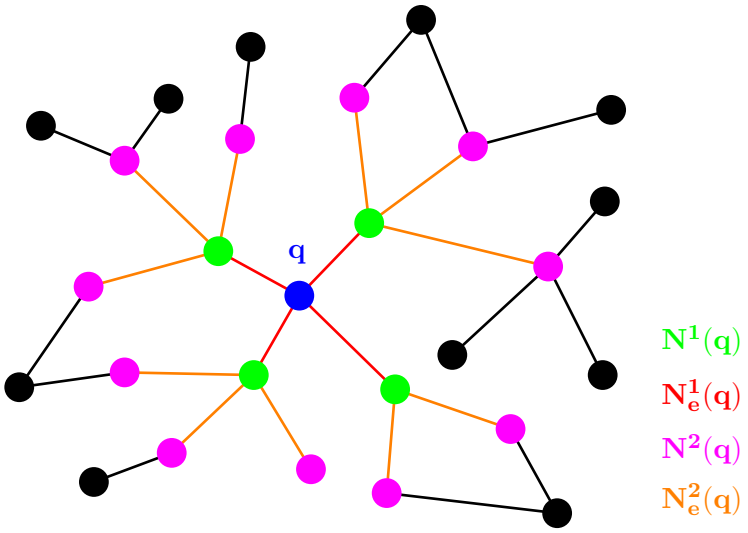


Figure 5.5: First and second order *vertex neighbours* (respectively in green and magenta) and *edge neighbours* (respectively in red and orange) of the vertex q (in blue)

Thus, for a given order L, one can define an edge-based neighbourhood of a vertex q given by:

$$N_e^L(q) = \bigcup_{i=1}^{L} N_e^i(q)$$

5.5.2 Algorithm

We propose an insertion algorithm (Algorithm 5) that relies on our edge-based neighbourhood N_e^L and a local update strategy. The main steps of this algorithm are as follows:

1. The first steps (lines 1-9) are the same as in Algorithm 2: as explained in the previous section, the hypersphere SR that is centred around q and its content are computed.

2. Then, we retrieve the relative neighbours of q in SR (lines 12-16). For each vertex p in SR, the pair (p, q) is considered. For each vertex r in SR, we check if r lays in the relative neighbourhood of the pair (p, q). If no vertices lay in this relative neighbourhood, then p and q are relative neighbours, and the edge $\overline{pq}$ is created. This step is carried out in $O(n'^2)$, where $n' = |SR|$.

3. STEP_1 gathers all the edges that belong to the edge-based neighbourhood $N_e^L(q)$ of q, given an order L. This is performed with a recursive algorithm. First, we initialise an empty set of edges A. For each relative neighbour q' of q in RNG, we recursively compute the $(L-1)^{th}$-order edge neighbours of q' and store them in A. At the end of this step, the set A contains the list of edges that belong to $N_e^L(q) \backslash N_e^1(q)$.

4. Finally, the effective update is made in STEP_2: for each edge e in A, we check if e has to be removed due to the apparition of q, *i.e.,* if q lays in the relative neighbourhood of the two endpoints of the considered edge.

The overall complexity of the proposed insertion algorithm is $O(2n + n'^2 + deg^L)$, where deg is the average degree of the graph RNG. Thus, we propose here an algorithm that has a lower complexity than the existing work. Indeed, when dealing with large data sets, $deg << n'$, and experimentations have shown that small value of L can be considered to have good results. Moreover, the edge-based neighbourhood allows to check more edges that may be concerned by the apparition of a new vertex, and this improve the accuracy of the computed graph. The trade-off between computation time and accuracy that can be achieved will be studied in the experimentations.

5.6 Experimentations

5.6.1 Evaluation

Following our study, we aim at answering the following questions:

1. How much time do the proposed algorithms need to yield incrementally the whole graph? More specifically, what is the cost, in terms of time, of the insertion of a

Algorithm 5 Edge-based neighbourhood local update strategy

Input: $RNG = (V, E)$, q, ϵ, L
Output: $RNG' = (V', E')$

$nn = nearest\_vertex(q, V)$
$fn = farthest\_relative\_neighbour(nn, RNG)$
$sr = (\delta(q, nn) + \delta(nn, fn)) \cdot (1 + \epsilon)$
$SR = \emptyset$
for each $p \in V$ **do**
  if $\delta(p, q) \leq sr$ **then**
    $SR = SR \cup \{p\}$
  end if
end for
$V' = V \cup \{q\}$
$E' = E$
for each $p \in SR$ **do**
  if $are\_relative\_neighbours(p, q)$ **then**
    $E' = E' \cup \overline{pq}$
  end if
end for
Step_1: $A = compute\_edge\_neighbourhood(q, L, RNG)$
Step_2: $update\_edges\_in\_neighbourhood(q, A, E')$
return $RNG' = (V', E')$

new data in the existing graph? Indeed, this is a key information of our study. It highlights the benefit of an incremental approach to locally update the graph instead of building the whole new graph from scratch.

2. How accurate are the graphs built incrementally by our algorithms? As some heuristics are proposed to cut the complexity of the local update operation, to what extent one can rely on the yielded approximated graph?

To answer these questions, we settle on two metrics:

1. The *computation time* of the insertion of a single data, and the whole graph construction. Note that as the insertion time of a data depends on how much data have already been inserted previously but mostly how dense/sparse is the subspace where the data is inserted, it is not so easy to give a relevant and succinct insertion time metric (*e.g. the average insertion time*), without a chance of introducing a bias in the interpretation. Indeed considering an existing graph of $50,000$ data, it is possible to insert the $50,001^{st}$ entry in 3 seconds and the $50,002^{nd}$ one in 3 milliseconds. Thus we provide the insertion time distribution.

2. The *confusion matrix*, where the percentage of wrongly added edges is extracted and, more precisely, the exact number of added and removed edges. Indeed, in our case, the graph comparison operation boils down to comparing the graphs edges, as the vertices and the edges length are the same.

Table 5.1: Data sets used for our experimentations. The number of vertices, their dimension and the number of edges in the exact RNG are given. "*n.t.*" means that the RNG computation was not tractable.

D	Type	| V |	d	| E(RNG) |
Iris	real world	150	4	195
WDBC	real world	569	30	712
Breiman	artificial	5000	40	17,837
Corel68k	real world	68,040	57	190,410
MIRFLICKR-1M	real world	1,000,000	150	*n.t.*

5.6.2 Experimental setup

The algorithms presented in this chapter have all been implemented in C++. In order to speed up some operations (*e.g.* the nearest neighbour search), they have been parallelised with OpenMP [OpenMP, 1998].

For a fair comparison, the algorithm of Hakim Hacid and al. (Algorithm 2) has been implemented under the same environment and assumptions than our algorithms (Algorithms 3 and 5), *i.e.* the same data structures and the same parallelised operations. Only the local update strategies differ.

The classic $O(n^3)$ RNG algorithm (Algorithm 1) has also been implemented for reference, to assess the graph accuracy. As this implementation stores the distances matrix, it was not used on large data sets.

In the present experimentations, the whole data set was loaded in memory, and then each piece of data was inserted one by one. The graph was stored as an adjacency list. ϵ value was set to 0.1 as in [Hacid and Yoshida, 2007]. We used an Intel Xeon CPU W3520 (quadcore) at 2.66Ghz, with 8GB of RAM.

5.6.3 Data sets

To test and stress our algorithms, five data sets have been selected (available online [Bache and Lichman, 2013]). They are either synthetic or real-world multidimensional data sets from various fields such as medical measures, images, *etc.* Table 5.1 summarises the specifications of the data sets. The three first, that can be considered as *small* data sets, have been used mainly to assess the validity of our algorithms and the quality of their results. The two last are larger data sets, more precisely medium and large image collections, up to a million images. Note that for the last data set, the exact computation of the RNG is intractable in reasonable time, thus the number of edges of the RNG does not appear in the table.

All the five data sets share one common property: their attributes are numerical, hence the euclidean distance has been considered to compare data between them. Note that this work can be applied to data described by categorical features with an appropriate distance function.

Table 5.2: Number of wrongly added edges and removed edges in the RNGs computed by Algorithm 2 (Hacid et al.) and Algorithm 5 (proposed approximative algorithm). The symbol == means that the approximate graph corresponds exactly to the exact graph.

	| E(RNG) |	Algorithm 2 (Hacid et al.)	Algorithm 5 (Approximate)		
			$L = 2$	$L = 3$	$L = 4$
Iris	195	+10/-2	+8/-1	==	==
WDBC	712	+2/-1	+10/-0	+3/-0	==
Breiman	17,837	==	+1161/-0	+299/-0	+26/-0
Corel68k	190,410	+20,363/-11	+9,089/-356	+2,165/-388	+637/-397

We describe below each data set in a few words:

- *Iris*: well-known data set in pattern recognition field. Data correspond to flowers' attributes of 3 types of iris plants.
- *WDBC (Wisconsin Diagnostic Breast Cancer)*: data correspond to fine-needle aspiration image features of a breast mass, taken for a group of both healthy and sick patients.
- *Breiman*: synthetic noisy waves generated from a combination of "base" waves.
- *Corel68k*: large collection of image from various categories (food, landscapes, *etc*). Data correspond to image features such as color histogram or color moments.
- *MIRFLICKR-1M* [1]: 1 million Flickr images under the Creative Commons license. Edge Histogram Descriptors (MPEG-7 Visual Standard) are used to describe the images.

5.6.4 Accuracy evaluation

First, we evaluate the accuracy of the proposed algorithms. The exact RNG was computed for the four first data sets. Since exact RNG could not be produced in reasonable time, this experimentation is not reported for the *MIRFLICKR-1M* image collection. The number of edges of the exact graphs are used as ground truth and the graph correspondence is computed to evaluate the computed graphs. Table 5.2 gives the number of erroneously added edges and removed edges. Algorithm 3 yields as expected the exact RNG, thus has a 100% accuracy. We do not mention it in the table for a better readability. The sensitivity of Algorithm 5 with regard to the parameter L is also presented.

One interesting observation in this experimentation is that the main difference between the graph produced incrementally and the exact graph is often the addition of wrong edges. Actually, it is not the addition of wrong edges, but rather the fact that some edges are not invalidated after an insertion due to the proposed edge-based neighbourhood. Thus,

[1] http://press.liacs.nl/mirflickr/

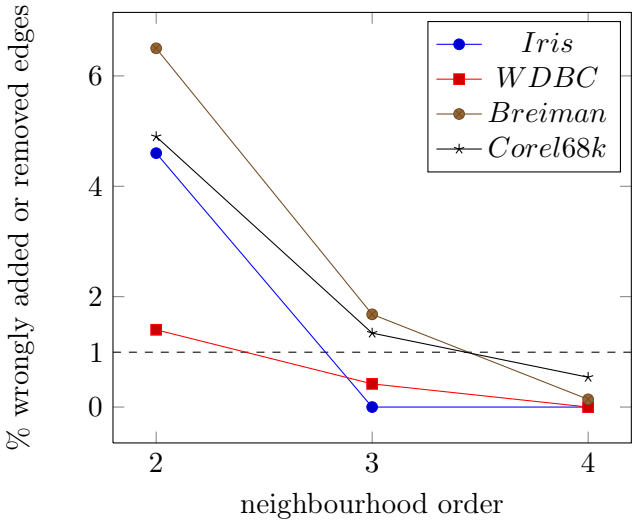


Figure 5.6: Algorithm 5 accuracy. Percentage of wrongly added or removed edges over the neighbourhood order L.

our algorithm leads to create a few number of false similarities between data, which may not be critical in some applications (*e.g.* similar images retrieval or user recommendation systems).

We notice that Hacid et al.'s algorithm (Algorithm 2) does not always incrementally yield the exact RNG as stated in their paper. Furthermore, our proposed algorithm performs at least as well as, if not better, than Hacid et al.'s algorithm in terms of accuracy, considering low edge-based neighbourhood order ($L = 4$). For the *Corel*68*k* data set, our approximate algorithm has computed almost 20 times less wrong edges than the algorithm of Hacid et al.

One can also see that for the small *Breiman* data set, our approximate algorithm performs less than the existing algorithm of Hacid et al. Actually, this specific data set illustrates the incremental worst case scenario. Due to the topology of this peculiar data set, at every iteration, the hypersphere SR contains all the vertices previously added. Thus, at the last insertion, SR contains all the vertices of the data set. Since the exact $O(n^3)$ algorithm is used during the update operation, the final RNG corresponds exactly to the exact RNG.

Figure 5.6 shows the percentage of the wrongly added or removed edges in the obtained approximate graphs. As expected, this percentage decreases as the edge-based neighbourhood order increases. Indeed, as more edges are checked, less erroneous edges are left, thus improving the accuracy of the approximate RNG. It is possible to build such as graph with less than 1% of wrongly added or removed edges considering low order of edge-based neighbourhood (such as $L = 4$).

5.6.5 Computation time evaluation

Table 5.3 presents the computation time of Algorithms 2, 3 and 5.

Table 5.3: Comparison of the computation times of Algorithms 2 (Hacid et al.), 3 (proposed exact algorithm) and 5 (proposed approximate algorithm). Computation times are given in seconds. "*n.t.*" means that the RNG computation was not tractable.

	Algorithm 2 (Hacid et al.)	Algorithm 3 (Exact)	Algorithm 5 (Approximate)		
			$L = 2$	$L = 3$	$L = 4$
Breiman	7,692	234	16	25	178
Corel68k	439,200	121,744	889	1371	1604
MIRFLICKR-1M	*n.t.*	*n.t.*	522,000	543,600	651,600

We present the computation times for the small *Breiman* data set in order to highlight the fact that the computation is high for Algorithm 2. As mentioned previously, the topology of this peculiar data set illustrates the incremental worst case scenario. It is a perfect counterexample of the assumption $n' << n$, made by Hacid et al., as mentioned in 5.3.2.

For the proposed exact algorithm, on the medium-scale data set *Corel68k*, the RNG can be computed. However, the computation time is higher than the approximate approaches. We have a slow down ratio of 76 compared to our approximate algorithm with $L = 4$. The average insertion time one can expect is around 2 seconds. For the larger data set, the exact algorithm did not succeed to process a third of the data set in more than two weeks.

For the *Corel68k* data set, the computation time of Algorithm 2 is five days while our approximate algorithm yields the graph in less than half an hour with $L = 4$. The achieved speed-up ratio is slightly less than 273. For the million images collection, the proposed approximate algorithm succeeds in computing an approximate RNG while Hacid et al.'s algorithm is not tractable in reasonable time. Figure 5.7 presents the insertion times of Algorithm 5 over the neighbourhood order L for the two large image collections. These experimentations have shown that one can expect an average insertion time of less than 500 ms. This is promising for almost real time updates.

5.7 Conclusion

In this chapter, we have proposed two incremental approaches to build the Relative Neighbourhood Graph:

- one based on a formal proof, that builds incrementally the exact RNG,
- one to incrementally build a refined approximation of the exact RNG.

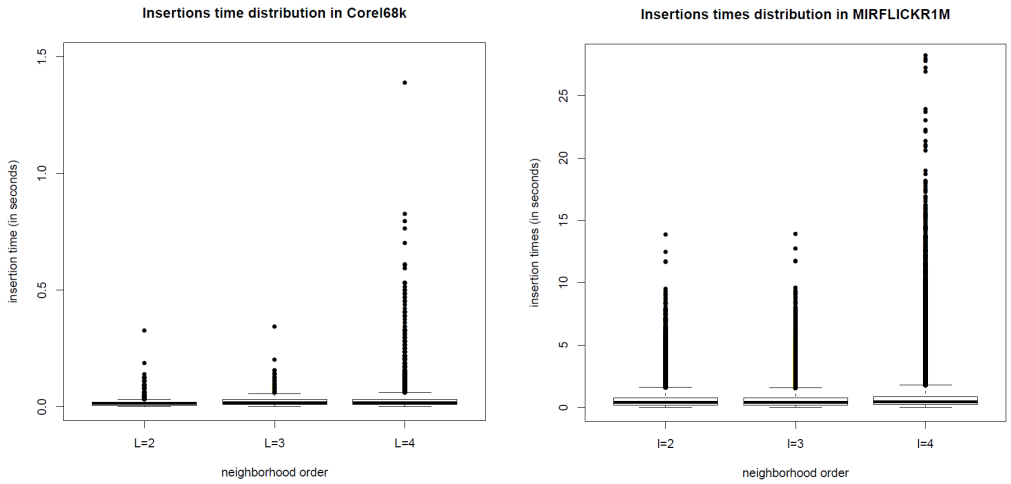


Figure 5.7: Algorithm 5 insertion time distribution over neighbourhood order L for the Corel68k and the MIRFLICKR-1M image collections (in seconds).

Table 5.4: Similarity distance storage size (in bits) and RAM requirement for a data set with n entries.

n	$\mid U_{ij}\mid$	RAM
10,000	399,960,000	$\approx$ 400Mo
50,000	9,999,800,000	$\approx$ 10Go
100,000	39,999,600,000	$\approx$ 40Go
500,000	999,998,000,000	$\approx$ 1To

We discuss now the lessons drawn from this work:

1. the straightforward and brute-force $O(n^3)$ algorithm is an *embarrassingly parallel* algorithm. Hence, for small or even medium data sets, one should use this algorithm to build the RNG. On the one hand, if one stores the distance matrix, the RAM is the main constraint. Table 5.4 illustrates the RAM requirement for medium-sized data sets. We assume that the distances are float values, that the size of a float is 8, and that U_{ij} is only the upper triangular matrix of the distance matrix. Hence, the storage size of U_{ij} is given by $|U_{ij}| = 8 \cdot \frac{n(n-1)}{2}$. Note that these values do not take into account the data memory requirement, nor the graph storage memory requirement. On the other hand, if one uses a technique where the distance matrix is not stored, the distance calculations will considerably increase the graph computation time.

2. If one seeks to yield an exact RNG for larger and/or ever-growing data sets, he can use the proposed Algorithm 3 while it is not tractable using the brute-force one. However, its complexity is also cubic and the insertion time is not negligible (from one second to several minutes).

3. Hence for large and/or ever-growing data sets, we advise to use the proposed approx-

imate algorithm. Indeed, it has a good trade-off between insertion time and accuracy, and beats the existing work.

4. Finally, one can argue that the proposed approximate algorithm computation time still seems quite high. These times can be reduced by considering an hybrid approach. First, one can compute the exact RNG of a subset of the data with a parallel implementation of the $O(n^3)$ algorithm. This subset may contain as much data as possible, only if the memory can handle its distance matrix and other memory requirements. Second, one can process the rest of the images using our incremental approach, either exact or approximate depending on the user objective, the RAM availability and the allowed time to process the data set.

In this chapter, we have addressed the large and the ever-growing constraints of our endeavour work. However, the interactive visualisation is not necessarily met. Indeed, an early experimentation, that is summarised in Appendix D, highlighted some limitations of existing software.

Hence, we propose in the next chapters to study jointly the indexing and the visualisation of large and ever-growing image collections. To do so, we present our viewable hierarchical and graph-based hybrid structure (Chapter 6) and an image collection exploration platform that has been implemented in the scope of this thesis work (Chapter 7).

Chapter 6

A viewable hierarchical and graph-based hybrid structure

Abstract

In this chapter, we conduct a joint study on the indexing and the visualisation of image collections. We propose a viewable hierarchical and graph-based hybrid structure to meet our constraints. To this end, the BIRCH algorithm is used. We present this algorithm and justify its choice with regard to our constraints. Then, we detail two of the improvements brought to the BIRCH algorithm. First, representative images are assigned to internal nodes of the tree that is yielded by BIRCH. Second, we organise the inner hierarchical structure with RNGs. The overall complexity of our modified insertion algorithm is discussed, and experimentations on large image collections, up to several millions of images, are presented.

Contents

6.1 Introduction

Following the structuring of large and ever-growing image collections in RNGs, we have conducted some experimentations regarding the visualisation of the yielded graphs. During an early experimentation, that is summarised in Appendix D, an image collection of 43,000 images from the National Gallery of Art of Washington has been built. It has then been structured in a RNG using the approximate algorithm proposed in the previous chapter. Using existing software, namely Tulip [Auber et al., 2012] and Gephi [Bastian et al., 2009], on a common computer (Intel Core i5 CPU M460 at 2.53GHz), we have faced some limitations regarding the interactions with the graphs. Indeed, displaying the images that are associated with the nodes, was not necessarily feasible, nor easy. Moreover, the interactions to wander around the graph were not always smooth.

Hence, we propose in this chapter to jointly study the indexing and the visualisation of large and ever-growing image collections. To do so, we present a viewable hierarchical and graph-based hybrid structure. More precisely, we take advantage of the tree yielded by the BIRCH algorithm, and enhance it to overcome some limitations with regard to our visualisation objective.

In the rest of this chapter, the BIRCH algorithm is presented and its choice justified. Then, two improvements are detailed. First, representative images are assigned to internal nodes of the tree in order to assist the user in the hierarchy navigation. Second, we organise the inner hierarchical structure with RNGs. Experimentations to evaluate the proposed algorithm on large image collections, up to several millions of images, are presented.

6.2 BIRCH definition

BIRCH [Zhang et al., 1996] is a clustering algorithm that aims at partitioning large data set that cannot fit entirely in the memory. The key idea is to go through the data set only once and organise the data points in a *Cluster Feature* tree (CF-tree).

Definition 6.2.1 (Cluster Feature (CF))**.** A Cluster Feature is a numeric vector that sums up a set of n data points $\{x_1, ..., x_n\}$. It is defined by a triple $CF = (n, LS, SS)$ where LS and SS are the linear sum and the square sum, respectively:

$$LS = \sum_{i=1}^{n} x_i \quad \text{et} \quad SS = \sum_{i=1}^{n} x_i^2$$

For a given cluster C and its CF, one can easily deduce the centroid x_0, the radius R (average distance from member points to the centroid), and the diameter D (average pairwise distance between member points) of C. Thus, a cluster is well described by its CF. Furthermore, a nice property of this description is its additivity. Let us consider two clusters C_1 and C_2 defined by $CF_1 = (n_1, LS_1, SS_1)$ and $CF_2 = (n_2, LS_2, SS_2)$ respectively, then the cluster $C = C_1 \cup C_2$ CF entry is defined by:

$$CF = CF_1 + CF_2 = (n_1 + n_2, LS_1 + LS_2, SS_1 + SS_2)$$

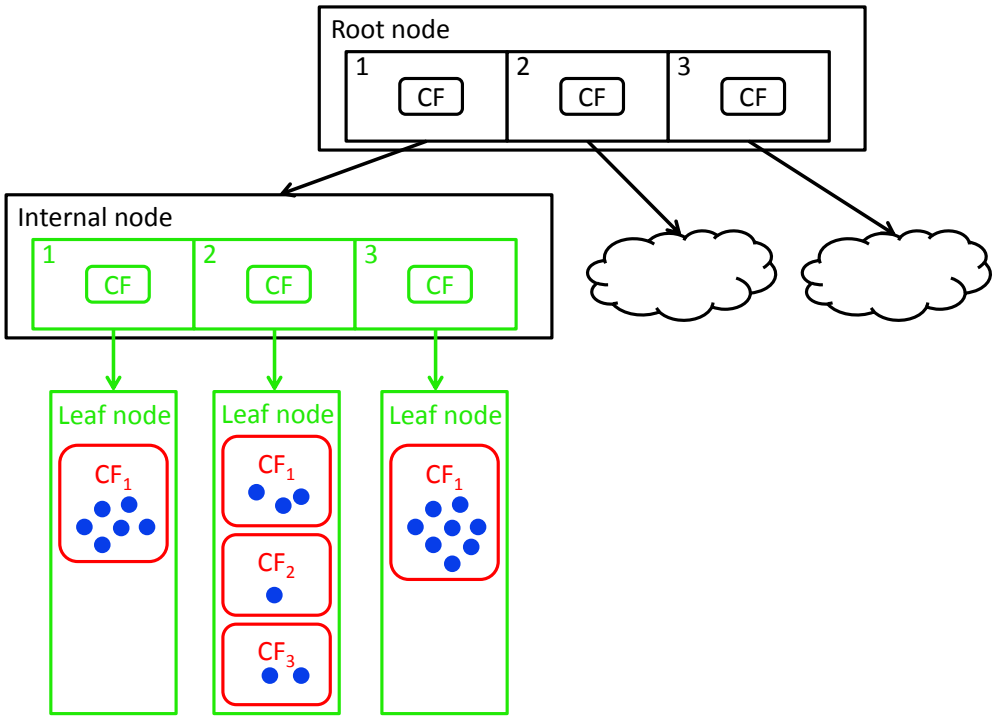


Figure 6.1: Illustration of a CF-tree output by the BIRCH algorithm (with $B = L = 3$). The points that appear within the leaves' CF are actual data points.

Definition 6.2.2 (CF-tree)**.** The CF-tree is a height-balanced tree with three parameters B, L and T:

- each internal node contains at most B entries $[CF_i, child_i]$, where CF_i describes the subtree pointed by $child_i$,
- each leaf contains at most L entries $[CF_i]$,
- each leaf entry CF_i must have a radius smaller than the threshold T.

Figure 6.1 presents an illustration of a CF-tree.

Algorithm 6 presents the insertion of a new data point p in an existing CF-tree t. Given a distance function, the insertion algorithm consists in the following steps:

1. *identifying the appropriate leaf*: starting from the root of t, p recursively descends t and chooses the closest child node according to a distance metric.
2. *updating the leaf*: when p reaches a leaf node, the closest leaf entry l_p is found. If l_p can absorb p without violating the threshold constraint, then p is added to l_p, and the CF vector of l_p is updated. Else, a new entry containing only p is created. If the

Algorithm 6 CF-tree insertion algorithm $Insert(Data, Node)$

Input: T, B, L, global parameters.
Input: $Data$, new data point to be inserted.
Input: $Node$, current node which correspond to the root of a sub-tree of the CF-tree.
Output: $Bsplit$, flag that notify if a split operation need to be done.

```
1: Entries = GetEntries(Node)
2: ClosestEntry = GetClosestEntry(Data, Entries)
3: if Node is not a leaf then
4:    ClosestEntryNode = GetChild(ClosestEntry)
5:    Bsplit = Insert( Data, ClosestEntryNode)
6:    if Bsplit then
7:       Split(Node)
8:    end if
9: else
10:   Dist = GetDistance(Data, ClosestEntry)
11:   if Dist ≤ T then
12:      AddDataPoint(Data, ClosestEntry)
13:      UpdateCFVector(ClosestEntry)
14:      Bsplit = FALSE
15:   else
16:      NewEntry = CreateNewEntry()
17:      AddDataPoint(Data, NewEntry)
18:      UpdateCFVector(NewEntry)
19:      if GetChildNumber(Node) ≤ L then
20:         AddEntry(Node, NewEntry)
21:         Bsplit = FALSE
22:      else
23:         Data = NewEntry
24:         Bsplit = TRUE
25:      end if
26:   end if
27: end if
```

leaf node does not have enough space to include the new entry without violating the maximum children constraint, then it must be split.

3. *updating the path*: after inserting p in a leaf node, internal nodes that lie on the path must be updated. This update operation could be either: *(i)* the update of the CF vectors or *(ii)* the split of the internal nodes.

Thanks to this insertion algorithm, one can process large data sets, that do no fit entirely in the memory, in a *single-pass* scheme. Indeed, one can simply keep a reference to object in the leaves, after these objects have been processed. Algorithm 6 has a complexity of $O(d \cdot B \cdot (log_B(N_{max}) + 1))$, with d, the dimension of the data and N_{max} the maximum number of CF nodes that can fit in the memory. This last value depends on the available memory and the CF node structure size. The complexity expression is explained as follows:

- $log_B(N_{max}) + 1$ corresponds to the number of nodes from the root to a leaf,
- B corresponds to the number of entries to consider in each node,
- d corresponds to the comparison complexity between the new data and a CF entry.

It follows that the BIRCH algorithm has a complexity in $O(n)$. More details on the insertion algorithm and the BIRCH algorithm, such as refinements and performance evaluation, can be found in the original article [Zhang et al., 1996].

6.3 BIRCH discussion

The choice of the BIRCH algorithm is motivated by several reasons. First, it produces a tree structure which is desirable for a multilevel visualisation. Indeed, using a multilevel scheme, a large image collection will be gradually displayed. Second, as one can see in the Figure 6.1 on page 90, the internal nodes contain only the CF entries and not the data points. This reduces drastically the memory usage and allows to process large data sets that cannot fit in the memory. Last, the algorithm processes the data points in a *single pass* scheme, thus it could be implemented in an incremental way, allowing to handle growing data sets or data streams.

However, the BIRCH algorithm has a few limitations. First, the algorithm relies on three parameters, namely T, B and L, which may affect the quality of the generated tree. Second, the leaves that appear in the tree do not always correspond to natural clusters. Last, the resulting tree depends on the data points order. However, these limitations can be addressed with refinement steps and additional passes over the data.

There are also limitations with regard to our objective of image collection visualisation. Indeed, the algorithm has been designed to focus on the leaves content and the tree structure is not exploited except for the insertion. Moreover, as mentioned previously, the internal nodes contain only CF entries, which are difficult to interpret by the user during a visual exploration of the tree.

In order to address the latter limitations, we propose to enhance the CF-tree structure yielded by the BIRCH algorithm, and modify the algorithm accordingly. Among the most important ones, we detail in the next section: *(i)* the assignment of representative images to CF entries and *(ii)* the structuring of CF entries at each level using the RNG described in Chapter 5.

6.4 Improvements

6.4.1 CF entries representatives assignment

In order to allow the user to easily interpret a CF entry, a set of k relevant images are assigned to each CF entry as representatives. An incremental and bottom-up approach

Algorithm 7 CF entry representative update algorithm GetRepresentatives(k, $Entry$)

Input: k, number of representatives to assign to a node, global parameter.
Input: $Entry$, current entry.
Output: Rep, computed list of representatives of the current entry.

```
1: if Entry is in a leaf node then
2:    p = GetPrototype(Entry)
3:    Knn = GetKnn(p, Entry, k)
4:    Add(Rep, Knn)
5: else
6:    n = GetSize(Entry)
7:    EntryNode = GetChild(Entry)
8:    ChildNumber = GetChildNumber(EntryNode)
9:    for i from 1 to ChildNumber do
10:      CF_i = GetEntry(i, EntryNode)
11:      n_i = GetSize(CF_i)
12:      k_i = max(⌊ (n_i / n) · k ⌋, 1 )
13:      Kinn = GetRepresentatives(k_i, CF_i)
14:      Add(Rep, Kinn)
15:   end for
16: end if
```

is used: relevant images are pulled up from the leaves to the root of the tree. This representatives assignment has been embedded in the BIRCH original CF-tree construction. When a new image is inserted in the tree structure, an update operation is performed. The representatives of the entries that belong to the insertion path are updated to take into account the newly inserted image.

Algorithm 7 presents the update operation at a CF entry of the insertion path. For a given CF entry E_l in a leaf node, its prototype is computed. The prototype corresponds to the nearest image to the centroid of the CF. Then, the prototype and its $k-1$ nearest neighbours are defined as the representatives of E_l. This is done in $O(n_l)$, with $n_l = |E_l|$. For a given CF entry E_{nl} in an internal node (non leaf), the k_i representatives of each child entries CF_i are pulled up. The values k_i are automatically computed with regard to the number of images contained in CF_i. In addition, we have $\sum_i k_i = k$. Thus, we assign k relevant images to E_{nl}: these images can be found in the lower levels of E_{nl} child sub-tree. This operation is done in $O(k)$.

Figure 6.2 presents an illustration of the representative selection operation. In this toy example, k is set to 5. Let us assume that for the leaf node, each CF entries has its own set of representatives. Let us consider $CF_{1.1}$, we aim at selecting its representatives. Its size n is the number of images that it contains, thus $n = n_1 + n_2 + n_3 = 6 + 6 + 1 = 13$.

Following the representative update Algorithm 7, we have :

$$k_1 = max(\lfloor \frac{n_1}{n} \cdot k \rfloor,\ 1\ ) = max(\lfloor \frac{6}{13} \times 5 \rfloor,\ 1\ ) = 2$$
$$k_2 = max(\lfloor \frac{n_2}{n} \cdot k \rfloor,\ 1\ ) = max(\lfloor \frac{6}{13} \times 5 \rfloor,\ 1\ ) = 2$$
$$k_3 = max(\lfloor \frac{n_3}{n} \cdot k \rfloor,\ 1\ ) = max(\lfloor \frac{1}{13} \times 5 \rfloor,\ 1\ ) = 1$$

Thus, the two first representatives of both $CF_{1.1.1}$ and $CF_{1.1.2}$, and the first representative of $CF_{1.1.3}$ are pulled up and assigned to $CF_{1.1}$ as its list of representatives. The same reasoning is used to pull up representative images from the leaves to the root of the tree.

Selecting cluster representatives is a challenging task, and in the aforementioned solution, we select the medoid and its nearest neighbours. These representatives are referred to as *nearest representatives*. They mainly highlight the *center* elements of the cluster. In order to give a different type of information to the user, we also have used another approach, based on the CURE algorithm. CURE (Clustering using representative) is a clustering technique that aims at addressing topological limitations of the BIRCH algorithm. Note that the the CURE algorithm has not been selected over the BIRCH algorithm because the former do not yield a hierarchical structure. The CURE algorithm uses a technique to represent a cluster by a given number of *"well scattered points within the cluster. These representative points attempt to capture the physical shape and the geometry of the cluster"*. In our algorithm, at a leaf entry level E_l, we allow the computation of k *farthest representatives* of E_l, that can be considered as outliers. To do so, we first assign the farthest point of the medoid as the first farthest representative. Then, we successively assign the farthest point of the i^{th} farthest representative as the $i + 1^{th}$ one. This can be done in $O(k \cdot n_l)$. The assignment of the k farthest representatives of an internal node is done in the same way as the nearest representatives. Such farthest representatives might be useful for one who performs an objective-oriented exploration.

6.4.2 CF entries structuring

We present here how the RNG presented in Chapter 5 has been used to yield hierarchical and graph-based hybrid structures. The tree built by the BIRCH algorithm is enhanced by structuring internal levels or internal nodes with graphs. This organises the hierarchical inner structure and will contribute to the clarity of the visualisation. The proposed structure is built in a incremental way, and updated following the insertion of a new image. Two ways of building a hybrid structure are proposed below, namely the *hierarchical* RNG (HRNG) and the *multilevel* RNG (MLRNG).

6.4.2.1 Hierarchical RNG

Here, the tree structure is exploited to allow one to navigate in the image collections by exploring the tree itself. Starting from the root CF entries, the user will be able to descend in one of the branch of the tree. The possibility to backtrack and explore another part of the tree will obviously be available.

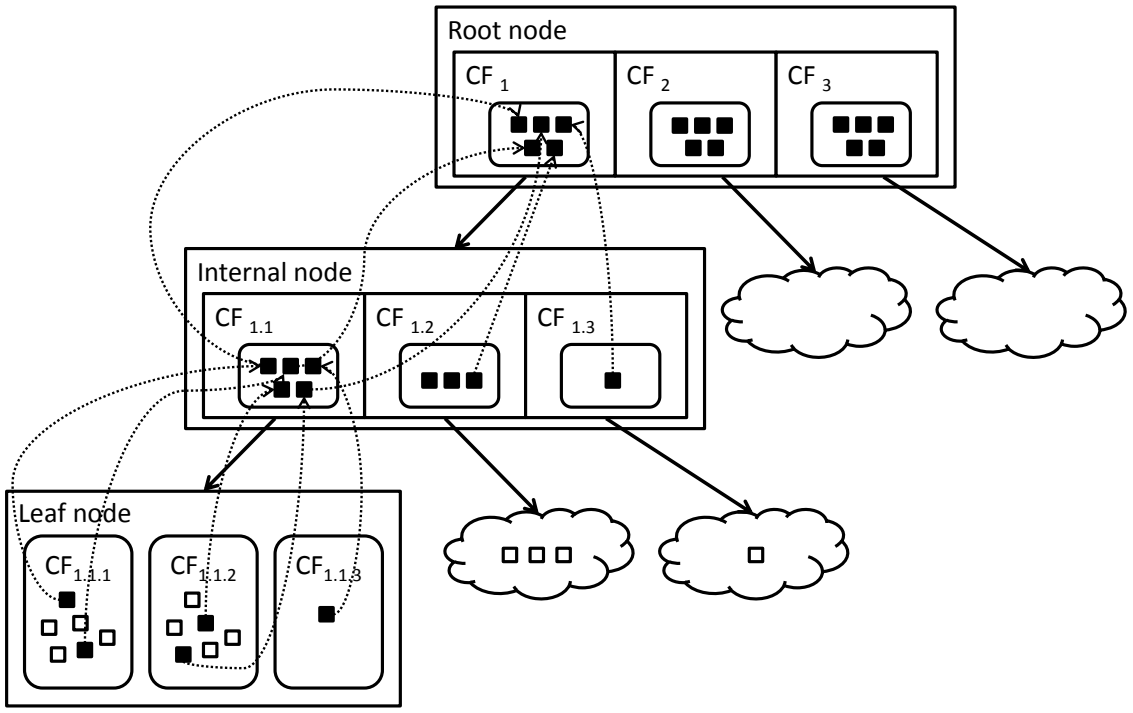


Figure 6.2: Illustration of an internal node CF entry representative update. k is set to 5. The squares correspond to actual images of the collection. The filled squares correspond to the representatives that will be pulled up to the upper level.

Thus, in the HRNG, for a given internal node $N = \{[CF_i, child_i]\}$, a RNG is built between the entries CF_i. Furthermore, for each CF entry that belongs in a leaf, a RNG is built between the images that it contains. Figure 6.3 illustrates the proposed HRNG hybrid structure.

In order to compute the RNGs, we take advantage of the lessons drawn from our previous work, described in Chapter 5. We have pointed out that the incremental construction paradigm of the approximate RNG is relevant only when the number of data points is high ($> 50,000$). Otherwise, it is more interesting to recompute the whole graph in a parallel way since the brute-force algorithm is an embarrassingly parallel problem. Hence, we adopt the following strategy:

- to structure the entries of an internal node: due to the properties of the BIRCH algorithm, the number of children of an internal node is limited by the parameter B. To allow a smooth exploration and graph visualisations, we empirically set this parameter with a small value (< 100). Therefore, for an internal node, the RNG of its entries is recomputed with a parallel version of the cubic classic algorithm.

- to structure the images contained in a leaf entry: we use our incremental approximate RNG construction algorithm if the number of images exceed a given threshold (*e.g.* 10,000 images), else the parallel version of the classic algorithm is used.

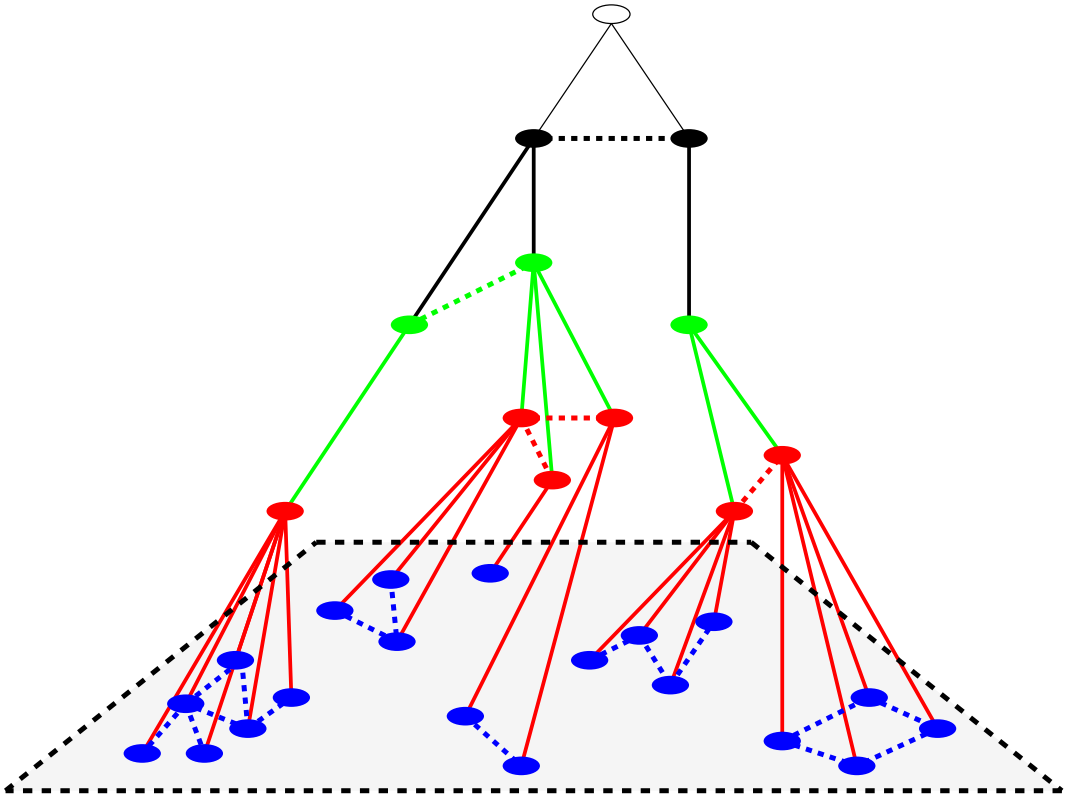


Figure 6.3: Representation of the HRNG structure. The most upper node corresponds to the root of the structure. The (blue) nodes that lay in the grey area correspond to images. Each node that does not lay in the grey area corresponds to a CF entry of the CF-tree. Coloured dotted lines correspond to an edge in a RNG. An edge between an upper level entry CF_u and a lower level entry CF_l corresponds to the fact that CF_l is a CF entry of the CF-tree node pointed by CF_u.

6.4.2.2 Multilevel RNG

In this multilevel exploration, the idea is to visualise each level of the tree. Starting from the root level, the user will be able to descend in the lower levels of the tree. Again, the possibility to backtrack and roll up to the upper levels will be available.

Figure 6.4 illustrates the proposed MLRNG enhanced tree structure. In this figure, two parts of the tree stand out. The upper levels are the levels in which the RNG or an approximate RNG are actually computed using the $O(n^3)$ or the incremental algorithm respectively. The lower levels correspond to levels in which the number of CF are too high, not for computing the approximate RNG using the incremental paradigm, but rather because the visualisation of the computed graph would not be relevant. Indeed, the objective of this structure is to be viewable and responsive in terms of interactions.

In the experimentations of this thesis, we do not present many results related to the

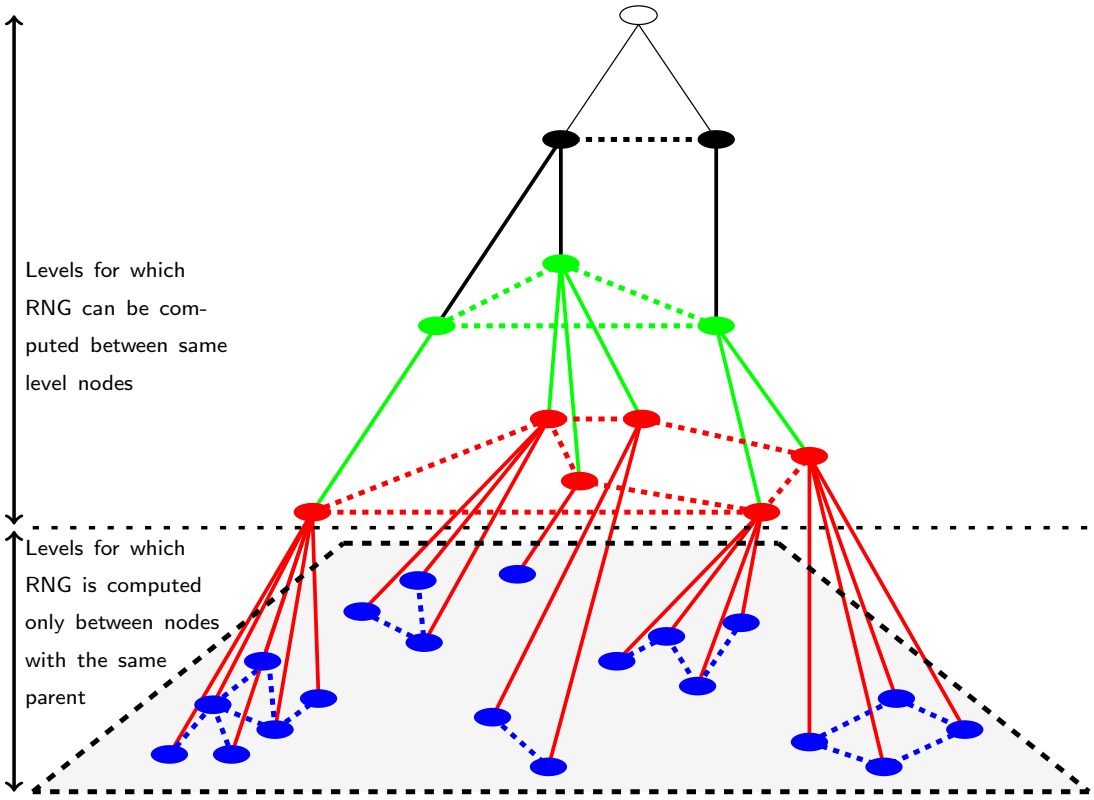


Figure 6.4: Representation of the MLRNG structure. The most upper node corresponds to the root of the structure. The (blue) nodes that lay in the grey area correspond to images. Each node that does not lay in the grey area corresponds to a CF entry of the CF-tree. Coloured dotted lines correspond to an edge in the RNG. The upper levels are the levels in which the RNG (or an approximate RNG) is actually computed using the $O(n^3)$ (or the incremental) algorithm. The lower levels correspond to levels in which the number of CF are too high, not for computing the approximate RNG using the incremental paradigm, but rather because the visualisation of the computed graph would not be relevant.

MLRNG. Indeed, at a level l of the tree, following the BIRCH definition, one can find at most B^l nodes. For instance, for $B = 20$ and at level $l = 3$, we already have 8000 nodes to display, which can lead to a visualisation of the computed graph that may not be interactive.

However, such a navigation in the hierarchy could be used either for *(i)* MLRNG yielded with a small value of B or *(ii)* for a quasi-static visualisation of the collection, up to a certain level, in order to rapidly grasp the dispersion of the image collection at different levels.

6.4.3 Overall complexity

Let us consider the HRNG hybrid structure. When a new image is inserted in the structure, our proposed modified insertion algorithm consists in:

1. *identifying the appropriate leaf*: this step is the same as the original BIRCH insertion algorithm (Algorithm 6),

2. *updating the leaf*: in addition to the operations done by the original algorithm, we update the RNG of the images contained in the leaf and the representatives of the leaf,

3. *updating the path*: in addition to the operations done by the original algorithm, for each internal node of the considered path, the RNG of its CF entries are recomputed and their representatives are updated.

Hence, the overall complexity of our modified insertion algorithm is:

$$O(\ d \cdot B \cdot (log_B(N_{max}) + 1)\ \ +\ \ (2n + n_l^2 + deg^{\mathcal{L}}) + (k+1) \cdot n_l\ \ +\ \ log_B(N_{max}) \cdot (B^3 + 2k)\ \ )$$

with:

- d: dimension of the image descriptors
- B: parameter of BIRCH algorithm
- N_{max}: the maximum number of CF nodes that can fit in the available memory
- n: the number of images already inserted in the structure before the insertion of the new image
- n_l: number of images in the leaf entry E_l where the new image has been inserted
- deg: average degree of the RNG structuring the images in E_l
- $\mathcal{L}$: order of the edge neighbourhood, as described in Chapter 5
- k: number of representative images in the inner structure's nodes

In this complexity expression, two terms can be large and thus dominant:

1. d, if one uses a really large quantity of descriptors,

2. n_l, if the leaf entry where the new image has been inserted already contains at lot of images ($> 50,000$ images).

For the former, it is related to the original BIRCH insertion algorithm itself. For the latter, our previous experimentations have shown that the insertion in the approximate RNG is done in reasonable time (around 10 ms).

6.5 Experimentations

6.5.1 Data sets

Table 6.1 presents the properties of the five image collections used in these experimentations. Most of them are public and available online. The two first data sets are small ($< 10,000$ images). The other data sets are larger image collections, up to several millions. A colour visual descriptor (CLD) has been used to describe each image in the data sets, except for the MIRFLICKR-1M, for which publicly available texture features (EHD) have been used [1]. We succinctly describe below the image collections:

- *Wang.* The *Corel Wang* [Li and Wang, 2003] image data set is a public photo database that contains 1,000 images classified in 10 classes of 100 images. It has mainly been used to evaluate content-based information retrieval systems.
- *NASA-APOD.* Astronomy Picture of the Day (APOD) [Nemiroff and Bonnell, 1995] is a project conducted by the NASA. Its website mentions that *"each day a different image or photograph of our universe is featured, along with a brief explanation written by a professional astronomer".* From June 1995 to a cut-off date (December 4, 2015), this represents a data set of almost 7,000 images.
- *MIRFLICKR-25000.* This collection [Huiskes and Lew, 2008] consists of 25,000 images downloaded from the photography hosting and social media site Flickr through its public API. Under the Creative Commons license, the data set is enhanced by user-supplied image Flickr tags, EXIF metadata and additional manual image annotations.
- *MIRFLICKR-1M.* This collection [Mark J. Huiskes and Lew, 2010] is an extension of the MIRFLICKR-25000 collection. Two enhancement have been done. First, the number of images has been extended to 1 million images. Second, texture descriptors from the MPEG-7 multimedia content description standard are given for the whole set of images.
- *ImageNet-7M.* ImageNet [Deng et al., 2009] is a large image data set of 14 millions of images, organised according to the WordNet [Miller and Fellbaum, 1985] hierarchy. Semantic concepts in WordNet are described by multiple words or word phrases, called "synonym set" or "synset". This project has been done to give researchers a real large-scale image database for various purposes. Almost half of the data set has been collected and used in our experimentations, a subset which we refer to as ImageNet-7M.

6.5.2 Performance evaluation

The algorithms presented in this paper were implemented in C++. In order to speed up some operations (*e.g.* the nearest neighbour search), they were parallelised using OpenMP

[1] `http://press.liacs.nl/mirflickr/mirdownload.html`

Table 6.1: Data set description: n is the number of images and d is the dimension of the descriptors. The computation times of the proposed hierarchical and graph-based hybrid structure (HRNG) are given in seconds.

Data set	n	d	Computation time
Wang	1,000	192	3
APOD	6,871	192	27
MIRFLICKR-25000	25,000	192	100
MIRFLICKR-1M	1,000,000	150	34,407
ImageNet-7M	6,930,901	192	392,636

[OpenMP, 1998]. In the present experimentations, for each collection, we start from an empty tree, and insert incrementally the images one by one. The whole data set is loaded in memory, except for the ImageNet-7M data set that does not fit in the memory. The graphs are stored as adjacency lists. For runtime experiments, we used an Intel Xeon 6-cores processor E5-2620 v2 at 2.10Ghz with 64Go RAM. Note that all the RAM was not used in our experimentations thanks to our incremental approach. One can exploit a less powerful machine and still use our approach. However, the processing time of the whole collection will be longer.

In order to test the scalability of the proposed incremental indexing, the three larger data sets are considered, namely the MIRFLICKR-25000, MIRFLICKR-1M and the ImageNet-7M image collections. Table 6.1 presents the computation times of the proposed incremental hierarchical indexing method, with $T = 20$ and $B = L = 50$. The computation time consists of the data loading on the RAM and the image insertion times. We have tractable computation times: less than 10 hours for the million image collection, and less than 4.5 days for the larger one of almost 7 millions images. Thus, the proposed modifications do not conflict with the scalability of the original BIRCH method. Furthermore, as a hierarchical structure is built, the visualisation will be smooth thanks to the reduction of nodes to display. Only the leaves that contain hundreds of images might take a few seconds if the user wants to display the images. However, this could be addressed by performing an additional clustering on the leaves as proposed in the original BIRCH algorithm [Zhang et al., 1996].

We also present the insertion time distribution of the proposed algorithm for the ImageNet-7M image collection (see Figure 6.5). This validates the incremental approach to handle dynamic and large image collections. Indeed, the results are promising for almost real time updates: one can expect an average insertion time of 10 ms. Note that the boxplot in Figure 6.5 (right) is cropped and does not display some 'outliers' insertion times that can go up to a few seconds. The scatter plot of these insertion times is also displayed in Figure 6.5 (left). In this figure, one can see the outliers aforementioned, that were not displayed in the boxplot. Most of them are inserted in around one second, and for a really small minority of them (around 5 out of a million), in a few seconds.

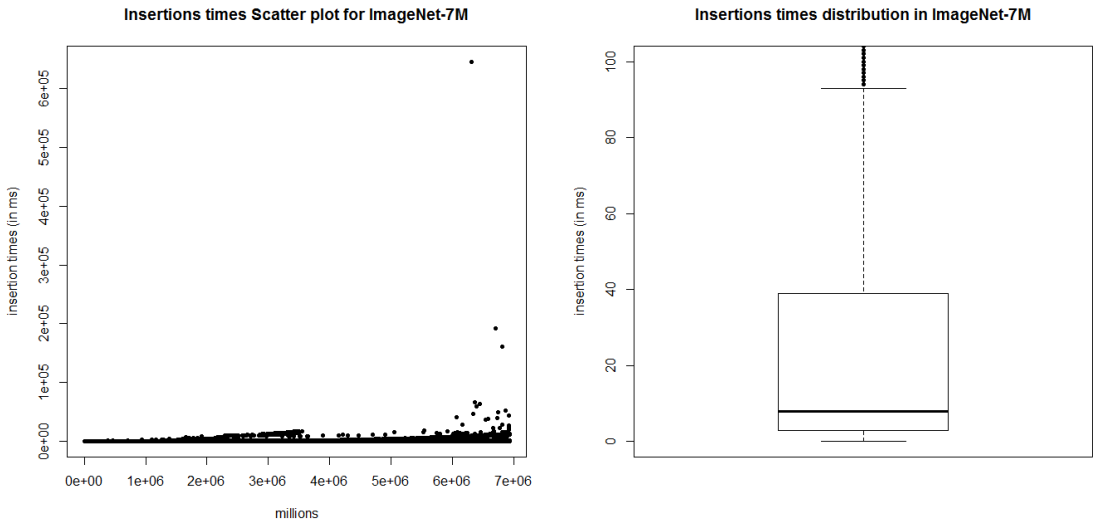


Figure 6.5: Scatter plot and boxplot of the insertion time distribution for the ImageNet-7M image collection (in milliseconds).

6.6 Conclusion

In this chapter, we have presented a viewable hierarchical and graph-based hybrid structure to visualise ever-growing and large image collections. This work endeavours to satisfy three constraints, namely large image collection handling, ever-growing collection handling and the provision of an interactive image collection exploration system.

The BIRCH data partitioning algorithm has been modified to fit an interactive visualisation objective. The tree structure yielded by BIRCH is enhanced. First, to allow a better interpretation during the tree exploration, we compute representative images for each internal node. Second, to structure the inner hierarchical structure and contribute to the clarity of the visualisation, we structure intermediate levels and internal nodes of the tree with a relative neighbourhood graph. The proposed hybrid hierarchical-RNG structure is built in a incremental way to address the second constraint.

Hence, thanks to this joint study on indexing and visualisation, we are able to process image collections that contain millions of images. We have presented an experimentation that have processed an image collection of seven millions images in a few days. Furthermore, our incremental algorithm can handle images that come in a stream, with a rough estimation of 1.5 million of images per day.

Again, we wonder about our third constraint, namely the interactive exploration. This time, thanks to the joint study, the proposed structure is viewable. To take advantage of this hybrid structure, we have implemented an image collection interactive exploration platform. This platform is presented in the next chapter and is evaluated through several use cases in the Part III of this manuscript.

Chapter 7

An interactive platform for image collections exploration

Abstract

In this chapter, we first discuss about graph drawing algorithms. We support the need of this discussion and briefly give an overview of the state of the art. Then, the chosen graph drawing algorithm is presented and this choice is justified with regard to our objectives. In a second part, we present our image collection interactive exploration platform, its characteristics and its functionalities. The latest version of the platform is available online at `http://frederic.rayar.free.fr/ice/`

Contents

7.1 Graph Drawing algorithms

The viewable structure presented in the previous chapter contains graphs. Hence, in order to visualise it, we need to draw graphs in a two-dimensional surface at some point. To do so, we use the common *node-link diagrams*.

Generally, when one needs to draw graphs, available graph drawing algorithms are used without peculiar consideration. Such graph drawing algorithms are often default ones, and sometimes not even mentioned.

In order to pursue our joint study on the indexing and visualisation of image collections, we discuss in this section which graph drawing algorithm may fit the best our objective. Hence, in the rest of this section, we first briefly give an overview of the *Graph Drawing* literature. Secondly, we present the selected algorithm and justify this choice regarding our constraints.

7.1.1 Graph Drawing

Graph Drawing (GD) is a field that addresses the issue of visual depiction of graphs in two (or three) dimensional surfaces. To do so, it takes benefit of Graph Theory and Information Visualisation fields.

Node-link diagrams are a common way to represent graphs. In such depictions, vertices (or nodes) of the graph are represented as disks, boxes, or textual labels. The edges (or links) are represented as segments or curves in the plane. Node-link diagrams can be traced back to early ages, especially for tree visualisations. The first work on these diagrams drawing algorithms is the one of William T. Tutte in 1963: "*How to draw a graph*" [Tutte, 1963]. This first work has then inspired much research in the following decades.

In this manuscript, we do not present thoroughly the state of the art of GD algorithms. Excellent surveys on this topic can be found in Di Battista et al.'s book [Battista et al., 1998] and Tamassia's one [Tamassia, 2013]. One can also refer to the *International Symposium on Graph Drawing and Network Visualisation* website [GD, 1992] for more resources on this topic.

Nonetheless, we briefly describe below two of the main categories of GD algorihms that have been proposed following the work of Tutte, namely force-directed algorithms and graph-theoretic distance approaches.

7.1.2 Force-directed algorithms

Force-directed graph drawing algorithms consider physical metaphors: nodes are considered as physical entities, and forces are assigned to pairs of nodes. These forces are either attractive in the case of pairs of nodes linked by an edge, or repulsive for pairs that are not linked by an edge in the graph. The main idea is then to place the nodes at a random position, and let the system find an equilibrium state.

Several metaphors have been considered, in particular, mechanical, electrical and magnetic systems and laws. For instance, one of the first force-directed algorithm [Eades, 1984]

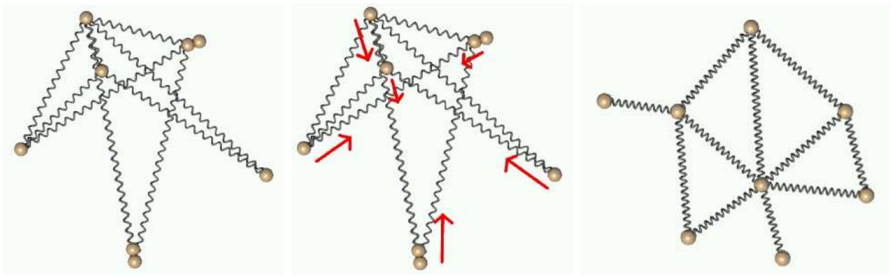


Figure 7.1: Force-directed algorithms. Illustration from [Kobourov, 2013] of a generic spring metaphor. Starting from randomly placed nodes, the graph is treated as a spring system and one looks for a stable configuration.

uses spring-like attractive forces based on Hooke's law to attract pairs of nodes that are linked by an edge in the graph, while repulsive forces like those of electrically charged particles based on Coulomb's law are used to separate the rest of the pairs. Figure 7.1 illustrates such a spring-like metaphor of a graph.

The first force-directed algorithms are Tutte's barycentric method [Tutte, 1963], the layout algorithm of Peter Eades [Eades, 1984] and the Fruchterman-Reingold's one [Fruchterman and Reingold, 1991]. As mentioned in [Kobourov, 2012], *"these algorithms tend to be aesthetically pleasing, exhibit symmetries and tend to produce crossing-free layout for planar graphs"*. However, this stands for relatively small graphs and such algorithms perform poorly for graphs with hundreds of nodes. More recent works have addressed the scalability issue with multilevel layout techniques, and nowadays, large graphs can be drawn with algorithms such as FM3 [Hachul and Jünger, 2005], Lin Log layout [Noack, 2007] and Force Atlas 2 [Jacomy et al., 2014].

Nevertheless, two drawbacks of the force-directed algorithms remain: *(i)* edges tend to have uniform length, and *(ii)* nodes that are not connected by an edge tend to be placed further apart (because of the repulsion forces).

7.1.3 Graph-theoretic distances approaches

Another approach, that addresses the limitations of force-directed algorithms, is to take into account the graph-theoretic distances between nodes. In their work, Kamada and Kawai [Kamada and Kawai, 1989] state: *"we regard the desirable geometric (euclidian) distance between two vertices in the drawing as the graph-theoretic distance between them in the graph"*.

Thus, the idea is to place the nodes in a two-dimensional surface such that the distance between two nodes in that surface is proportional to their graph-theoretic distance. For instance, one can use the *geodesic distance* between each pair of node, which is defined by the length of the shortest path between them. As such a placement could not always be achieved, the authors propose to minimise the difference between the geometric and graph distances. Hence, it boils down to an optimisation problem to solve.

In [Gansner et al., 2005], Gansner et al. propose to use a different method to solve the optimisation problem and present the Stress Majorization algorithm that outperforms

Kamada and Kawai algorithm. In a second time, Gansner et al. propose a maxent stress model [Gansner et al., 2013] to cut the complexity of computing the all-pair shortest path.

7.1.4 Discussion

As mentioned in Chapter 3, there is three main paradigms to visualise image collections, namely graph-based, clustering-based and projection-based. The proposed viewable hierarchical and graph-based hybrid structure (in Chapter 6) takes advantage of the two first paradigms. Hence, in order to combine the best of the three paradigms, we propose to use a graph drawing algorithm that shares the same properties as a projection algorithm, *i.e.* the placement of each object in lower-dimensional space such that the distances between objects in the original space are preserved as well as possible.

In [Gansner et al., 2005], the authors drew a parallel between the optimisation problem of Kamada and Kawai, and the MDS [Cox and Cox, 2000] projection algorithm. Moreover, the definition of a "good" graph layout as defined by Kamada and Kawai, fits perfectly the projection mantra. Hence, graph-theoretic distances approaches seem to fit our objective.

For these reasons, in our platform, the Stress Majorization [Gansner et al., 2005] graph drawing algorithm will be the default one. Indeed, in [Gansner et al., 2005], the authors use a different method, namely majorization, to solve the optimisation problem. Thanks to this, their algorithm outperforms Kamada and Kawai's one in terms of layout quality and running time. In order to give more significance to the weights of the edges, we consider a different graph-theoretic distance, noted $\delta_{gt}(p, q)$ of two nodes p and q, given by:

$$\delta_{gt}(p, q) = \sum_{i=1}^{s-1} w(\overline{v_i v_{i+1}}).$$

where $(v_1, v_2, \ldots, v_s)$ is the shortest path between the nodes p and q, $v_1 = p$, $v_s = q$, and $w(\overline{pq})$ is the weight of the edge $\overline{pq}$.

7.2 Platform description

In this section, we present our image collection interactive exploration platform. First, we justify the platform technical choices and give an overview of the platform. Second, we present the different functionalities implemented to explore an image graph. Last, we explain how one can use the platform to visualise the HRNG hybrid structure proposed in this thesis. The presented screenshots have been taken by exploring the *Wang* collection, that has been described with CLD. The latest version of the platform is available online at `http://frederic.rayar.free.fr/ice/`

7.2.1 General description

The proposed exploration platform has been implemented using web technologies. This choice is explained as follows:

Figure 7.2: Technologies used for the implementation of the proposed platform.

1. the platform targets either the general public or experts from various fields such as health or digital humanities. Nowadays, a majority of users that are not experts in computer science can still manage well web navigation. Thus, such users are familiar with web browsers. We think that presenting the system as a light web platform would make the users more disposed to use it and take advantage of it.

2. the choice of discarding a server can be justified by two arguments: *(i)* there is no upload of the images to a server, an operation that may cost time and *(ii)* as the images are not sent nor stored in an external server, we respect the potential confidentiality or license issues that are related to the image collections. Hence, we allow one to perform the indexing and then the exploration in an offline mode.

Hence, we have used HTML5 [W3C, 2014], CSS3 [W3C, 1999] and JavaScript [ECMAScript, 1995] web technologies to implement our platform. Regarding the javascript part, we have taken advantage of the AngularJS [Google, 2009] framework, mainly maintained by Google. Along with this framework, we have exploited dedicated libraries to visualise graphs in web browsers. We have used SigmaJS [Jacomy, 2012] in the first version of our platform, and then shifted to the Linkurious [Heymann, 2013] library, which is built on top of SigmaJS, and has more functionalities. All these libraries are open-source and have either a GPL or MIT license. Figure 7.2 presents a visual overview of the technologies used.

The platform can be packaged in an archive and be used offline, by embedding the javascript libraries. This participates to address the potential confidentiality of the image collections.

Figure 7.3 shows the platform interface. It consists in three parts: *(1)* the graph management and graph visual attribute manipulation control panel, *(2)* the graph visualisation canvas, and *(3)* the information panel.

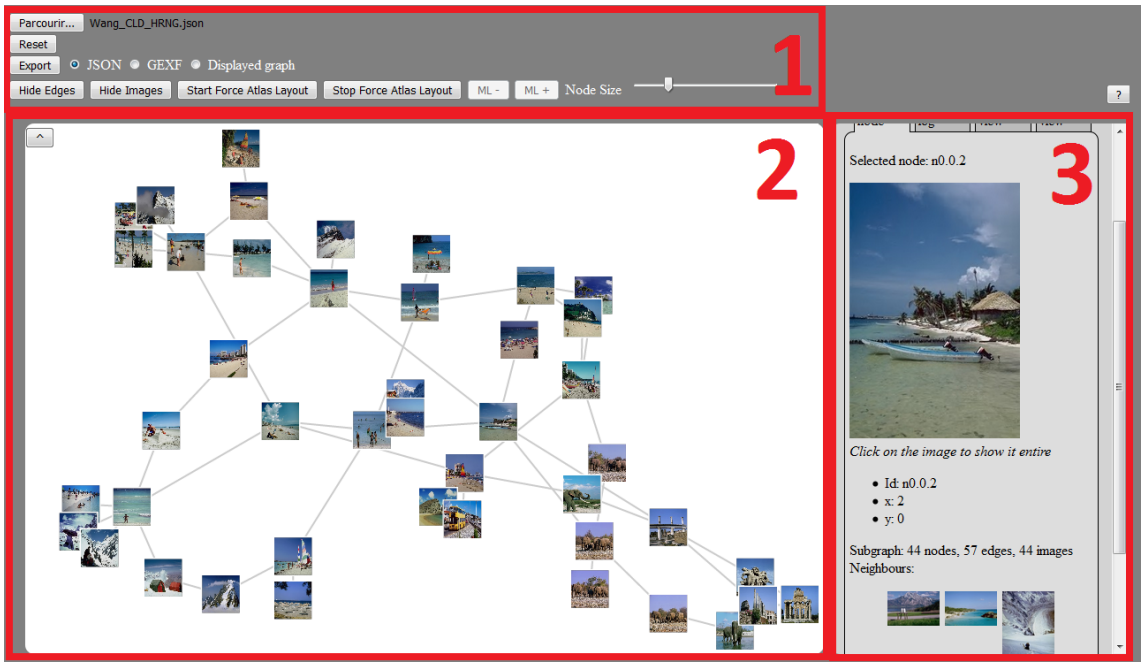


Figure 7.3: Screenshot of the proposed image collection exploration visual interface. (1) corresponds to the export/save functionalities and the graph visual attribute manipulation panel. (2) corresponds to the graph visualisation canvas. (3) corresponds to the information panel.

7.2.2 Graph visualisation

7.2.2.1 Graph format

Two graph formats are considered in the proposed platform, namely GEXF [Heymann, 2007] and JSON [Crockford, 2000]. The gexf description of a graph containing 2 nodes and 1 edge between them is given below.

```
<?xml version="1.0" encoding="UTF-8"?>
<gexf xmlns="http://www.gexf.net/1.2draft" version="1.2">
    <graph mode="static" defaultedgetype="directed">
        <nodes>
            <node id="0" label="Hello" />
            <node id="1" label="Word" />
        </nodes>
        <edges>
            <edge id="0" source="0" target="1" />
        </edges>
    </graph>
</gexf>
```

To describe hierarchical graph, the JSON format is preferred. We give a snippet of a HRNG stored in a json file below. We use nested tags to embed the hierarchy information. However, one can imagine that the file can be split in several ones to avoid a single large HRNG file, when dealing with large image collections.

```
"directed": false,
"graph": [],
"multigraph": false,
"nodes": [
  {
    "id": "n0.0",
    "label": "n0.0",
    "x": 0,
    "y": 0,
    "size": 1,
    "color": "#3366CC",
    "nb_images": 796,
    "representative": "1.jpg",
    "near_representatives": "1.jpg,2.jpg,3.jpg,4.jpg, ...",
    "far_representatives": "8.jpg,9.jpg,10.jpg,11.jpg, ...",
    "children": {
        "nodes": [
            ...
        ],
        "edges": [
            ...
        ]
    }
  }, ...
],
"edges": [
  {
    "id": "e0.0",
    "source": "n0.0",
    "target": "n0.14",
    "weight": "21.7236"
  },
  {
    "id": "e0.1",
    "source": "n0.1",
    "target": "n0.10",
    "weight": "21.2644"
  }, ...
]
```

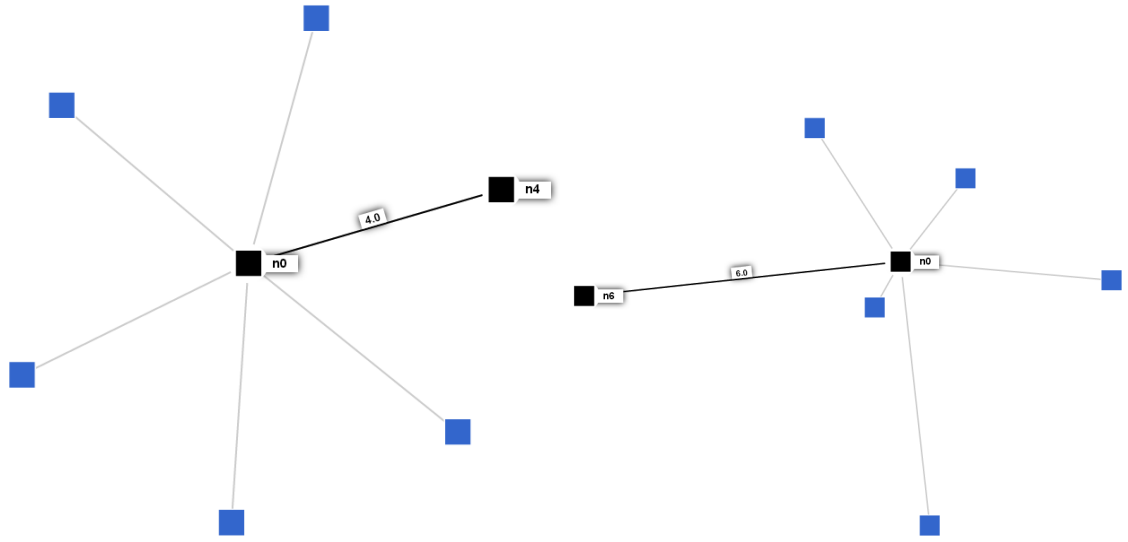


Figure 7.4: Graph layout. Layout of a toy graph using any of the three force-directed layouts (left), and the stress majorization algorithm (right).

7.2.2.2 Graph layout

As mentioned previously, the Stress Majorization drawing algorithm is favoured in our platform. However, we give to the user the possibility to use other graph drawing algorithms. In the platform prototype, the following algorithms are available:

- force Atlas 2 [Jacomy et al., 2014],
- force-Link, derived from Force Atlas 2,
- Fruchterman-Reingold [Fruchterman and Reingold, 1991],
- stress majorization [Gansner et al., 2005],
- maxent-Stress Models [Gansner et al., 2013].

The three first algorithms are force-directed algorithms already provided by the Linkurious library while the two last have been implemented by us during this thesis as plugins.

Figure 7.4 highlights the difference one could expect using either a force-directed algorithm or a graph-theoretic distance approach. We have used a toy graph, composed of seven nodes $\{n_0, n_1, n_2, n_3, n_4, n_5, n_6\}$, with six edges $\overline{n_0 n_i}, \forall i \in [\![1,\ 6]\!]$ such as $w(\overline{n_0 n_i}) = i$. In the left part of the figure, one can see the drawing obtained using any of the three first algorithms, which are force-directed algorithms. The edges have uniform length, while having different weights. In the right part of the figure, one can see the drawing obtained using the stress majorization algorithm. It gives more information than the other drawing as the edges lengths are proportional to their weights.

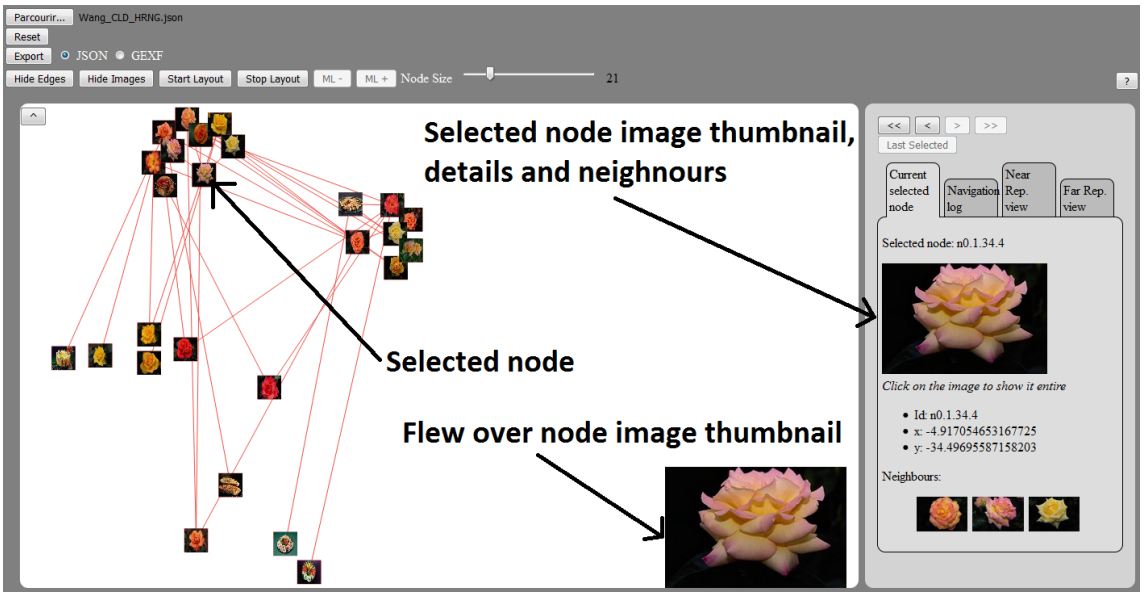


Figure 7.5: Graph visualisation. Each node can display its related image. The thumbnail of the selected node is displayed in the bottom-right of the canvas. Informations about the current selected node, such as its neighbours thumbnails, are displayed.

7.2.3 Image graph visualisation

This visualisation can be used either for a plain image graph or any level of the MLRNG structure proposed in Chapter 6. Common interactions such as zoom and pan are available to allow the user to manipulate the graph and select a region of interest. Some of the others interactions on image graphs are given below.

7.2.3.1 Viewing nodes images

One important feature of the platform is the possibility to display the images that are represented by the nodes. Thus, one can either:

- see the nodes as disks and visualise the associated image as a pop-up thumbnail on-demand,
- display each node as images and control the size of the image nodes thanks to a slider.

In Figure 7.5, one can see an image graph with the images displayed. By hovering the cursor on a node, one can display a larger thumbnail on the bottom-right of the graph canvas. It is also possible to embed information about the flew over image in a custom contextual menu that may appear as a pop-up.

The image collection can either be stored locally or on a remote server. Indeed, it is possible to assign a local path or an url to a node's image. This participates to address

the potential confidentiality of an image collection. Regarding the load of the images in the browser, a simple cache memory strategy has been used. Hence, 250 local images of 400×300 pixels could be loaded in a couple of seconds on a common computer (i5 M460 at 2.53Ghz). This is a promising result in order to visualise small image graph of a few hundreds of nodes. Larger image graphs can be viewed, but the load of the data may take a longer time. In any case, we advise the use of thumbnails to reduce the memory consumption and increase the smoothness of the platform when visualising an image graph.

7.2.3.2 Focusing on an image

The user can focus his attention on any image by clicking on it. A zoom is done by placing the selected image on the center of the graph canvas. Its adjacencies are highlighted and related information is given in the right panel. A larger thumbnail is displayed and can be clicked to display the image in its original size in a separate tab or window. Displayed information can be either metadata, description or annotations.

Furthermore, the graph neighbours of the image are displayed to allow the user to quickly visualise images similar to the selected image. This allow one to focus his attention on a region of interest in the graph, and perform a local visualisation of the graph.

7.2.3.3 Exploring the image collection

In order to explore the image collections, several means are proposed to the user:

- first, one can navigate in the graph canvas by clicking successively on the images he is interested in,
- second, one can navigate using the graph adjacencies. Once an image is selected, its neighbours are displayed in the information panel and are clickable. By clicking one of the thumbnails, the graph focus on the canvas changes to the selected thumbnail node. Thus, one can perform a step by step exploration of the image collections by using the graph adjacencies,
- last, the navigation in the image graph is stored and a log is available to allow one to rapidly go back to any of its previously visited images.

7.2.4 HRNG visualisation

In this visualisation, we take advantage of the HRNG proposed in Chapter 6. Figure 7.6 presents the visualisation in the platform, when visualising the HRNG at a given internal node level. Most features presented above for the image graph visualisation are also available for the HRNG visualisation.

At the beginning, the root CF entries are displayed as internal nodes and organised by their RNG. Then, one can navigate in the hierarchical structure by using:

1. a drill down operation: hence, one can go one level down on a internal node by double clicking it. The children, either internal nodes or images, are then displayed,

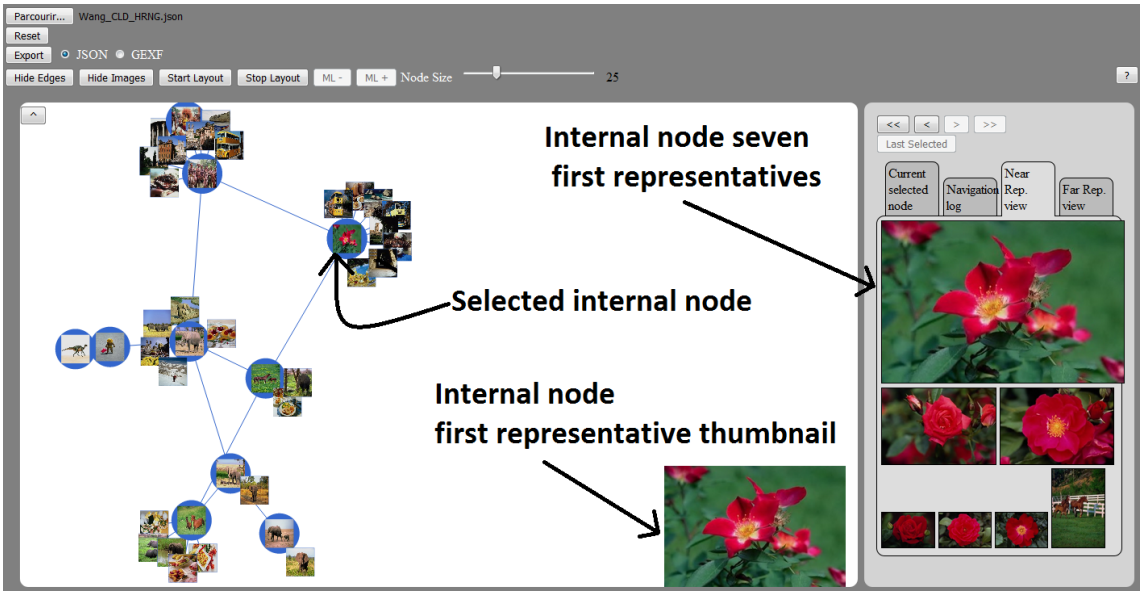


Figure 7.6: HRNG visualisation. Blue nodes are internal nodes that have a subtree and can be drilled-down. If the user flies over an internal node, its first representative thumbnail is displayed in the bottom-right of the graph canvas. In the 'Current node' tab, the neighbours of the selected internal node are displayed. In the 'Nearest representatives' ('Farthest representatives') tab, its seven first nearest (farthest) representatives are displayed.

2. a roll up operation: hence, one can backtrack on the hierarchy and go one level up, to explore other branches of the tree.

To allow one to visualise the node images during his navigation, we take advantage of the representatives computed for each internal node in our hybrid structure. Hence, when this option is triggered, internal nodes are represented by their first nearest representative. This first nearest representative is also shown as a thumbnail in the bottom-right of the graph canvas when the cursor is flew over the internal node. To give more information related to an internal node, its seven first nearest and farthest representatives are displayed in the information panel using visual tree map [Shneiderman, 1998] like representations. If there is no seven nearest or farthest representatives for a given internal node, we display available representatives. The layout of these tree maps has been implemented during this thesis and takes into account several parameters such as the ratio of the images, the orientation of the images and the rendering of the images. Figure 7.7 presents the visual tree map representation in our platform. Moreover, when the user selects a node, the number of direct children and the number of images contained in the subtree are displayed in the information panel. Thus, the user is assisted in the navigation of the hierarchical structure.

In order to allow users to keep track of their locations within the hierarchical structure, we have also implemented a visual breadcrumb during the development of this platform. Figure 7.8 presents the proposed visual breadcrumb representation. Given an internal node N in the hierarchy, the breadcrumb displays the series of internal nodes that lay in the

Figure 7.7: Tree map visualisation. Nearest and farthest representatives are displayed in visual tree maps. The more the image is important, the larger it is in the tree map. By important, we mean either it belongs in a large child for the nearest representatives, or in a small child for farthest representatives.

path from the root to N (blue rectangle). Furthermore, between each pair of levels, we place an arrow button. If one click on it, we display the visual summary of the 4 largest internal nodes of the upper level (red rectangle). To represent an internal node, we can either chose: its name, its first nearest representatives, its tree map, *etc.*

7.3 Conclusion

In this chapter, we have briefly discussed about graph drawing algorithms and gave an overview of the state of the art. This discussion supported us for selecting and justifying the choice of the drawing algorithm we favour in our work. Then we have presented the functionalities of our image collection interactive exploration platform.

This platform has been implemented as a proof-of-concept. Therefore, several improvements can be done to enhance it. To do so, many existing works in the literature could be used. Regarding the visualisation, we can implement existing node overlapping removal algorithms [Dwyer et al., 2006; Gansner and Hu, 2009]. Regarding the interaction, we can implement two interesting navigation interactions of [Moscovich et al., 2009], namely 'link sliding' and 'bring&go'. This will allow a more richer local exploration of a graph. The work of Daniel Archambault [Archambault et al., 2008], that allows to see nodes from different layers in the same view, would also be interesting in our context. Regarding the

Figure 7.8: Breadcrumb visualisation. This graphic control allows one to situate himself in the hierarchy, and navigate using level shortcuts. Internal nodes are depicted with their tree maps in the breadcrumb. The blue rectangle corresponds to the internal nodes that lay in the current path. The red rectangle corresponds to the visual shortcuts of previous level.

implementation, a good direction for future work would be the conception of strategies to enhance the image load in the memory cache. For instance, when displaying a level, we could pre-load representative images of the lower level. This will allow to bring more smoothness during the exploration.

Part III

Use cases

Chapter 8

INeX: ImageNet eXplorer

Abstract

In this chapter, we describe ImageNet eXplorer (INeX), a platform that allows an interactive exploration of the ImageNet-7M image collection. First, the WordNet database and the ImageNet image collection are briefly presented. Then, the approach that has been used to build the ImageNet-7M collection is detailed, along with a description of the INeX platform.

Contents

8.1 Introduction

In this chapter, we present more thoroughly the experimentation conducted with the ImageNet [Deng et al., 2009] image collection. We present in particular some visualisation results of the subset used in our experimentation, namely ImageNet-7M.

The ImageNet image collection has a distinctive feature: it is organised according to a semantic hierarchy, namely WordNet [Miller and Fellbaum, 1985]. However, in this use case, we present a work that does not rely on the existing hierarchy, but rather takes benefit of the proposed indexing and visualisation contributions described in Chapter 6 and Chapter 7 respectively.

Hence, we aim in this chapter to support the scalability property of the proposed joint study on indexing and visualisation for large image collections.

The rest of this chapter is organised as follows: first, we briefly introduce WordNet and ImageNet. Second, an overview of the work that is performed to obtain and process ImageNet-7M is presented. Third, our ImageNet eXplorer (INeX) platform is briefly described. Finally, we present a few observations following the use of INeX before concluding.

8.2 WordNet and ImageNet

ImageNet is an image collection organised according to the WordNet hierarchy. Hence, we present below WordNet in a few words, in order to allow readers to have a better understanding of ImageNet.

8.2.1 WordNet

As defined in the official homepage of the WordNet project[1]:

> *"WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts can be navigated with the browser. WordNet is also freely and publicly available for download. WordNet's structure makes it a useful tool for computational linguistics and natural language processing."*

Figure 8.1 illustrates a part of the WordNet hierarchy.

Hence, in the WordNet hierarchy:

- the nodes of the tree are synsets, *i.e.* groups of words brought together based on their meanings, mainly by synonymy,

[1] `https://wordnet.princeton.edu/`

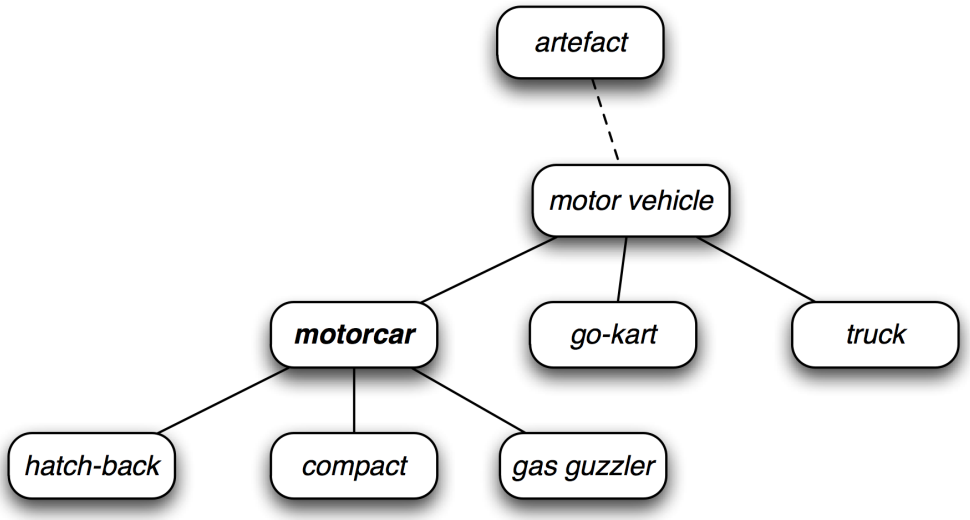


Figure 8.1: Part of the WordNet concept hierarchy.

- the synsets are linked based on super-subordinate relations, either *hyponymy* (X is a - kind of - Y) or *meronymy* (X is a part of Y).

As of today, WordNet's latest version is 3.1, under a BSD-like license.

The WordNet database contains 155,287 words organized in 117,659 synsets[2]. Compressed, this ASCII format database is about 16 megabytes in size.

8.2.2 ImageNet

As defined in the official homepage of the ImageNet project[3]:

> *"ImageNet is an image database organised according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. Currently we have an average of over five hundred images per node. We hope ImageNet will become a useful resource for researchers, educators, students and all of you who share our passion for pictures."*

First presented in 2009 [Deng et al., 2009], ImageNet contains today 14,197,122 images dispatched in 21,841 synsets. This huge data set has been built as follows:

1. *Collecting Candidate Images.* For each selected synset, candidate images have been gathered from the Internet by querying several image search engines (Google, Msn, Yahoo, Flickr). Each synset words and expressions have been used as queries, and

[2] `http://wordnet.princeton.edu/wordnet/man/wnstats.7WN.html`
[3] `http://www.image-net.org/`

in different languages (English, Chinese, Spanish, Dutch and Italian). Hence, over 10,000 images per synset on average have been collected.

2. *Cleaning Candidate Images.* To have a highly accurate dataset, the authors have relied on humans to verify each candidate image collected for a given synset. To do so, the crowdsourcing service Amazon Mechanical Turk (AMT) has been used. A labelling task has been proposed to check whether an image contains an object of the associated synset.

Hence, ImageNet's authors do not own the copyright of the images, and only the associated hyperlinks can be freely accessible . However, the original images can be provided upon request for researchers and educators who wish to use the images for non-commercial research and/or educational purposes.

The main goal of the authors was to give to the image and vision research community a large database that could be used either as:

- a training resource, essential for some machine learning methods (*e.g.* deep learning),
- a benchmark dataset, to evaluate algorithms at a large scale,
- a support for the human vision research.

As mentioned previously, we use a subset of ImageNet in our experimentations to support the scalability of the indexing and visualisation approach that is proposed in this thesis.

8.3 Overall approach

8.3.1 Getting ImageNet

Figure 8.2 presents the workflow that has been used to compute a description of the ImageNet image collection.

The first step of this pipeline is the download of an archive of 1.2Tb from the ImageNet server, namely *fall11_whole.tar*. This archive contains 21,841 tar archives, one for each synset. Secondly, we have spread those archives in 5 machines. Each machine extracted the archives and computed two MPEG-7 visual descriptors (CLD and EHD), described in Appendix C, for each image. From this, we have built one feature file per synset and per feature. Last, the 21,841 **.cld* and **.ehd* files have been compressed in two archives: *ImageNet_MPEG-7_CLD_whole.zip* and *ImageNet_MPEG-7_EHD_whole.zip*, 1Gb and 2Gb respectively.

These two archives will be soon freely available to the research community for an effort towards open research.

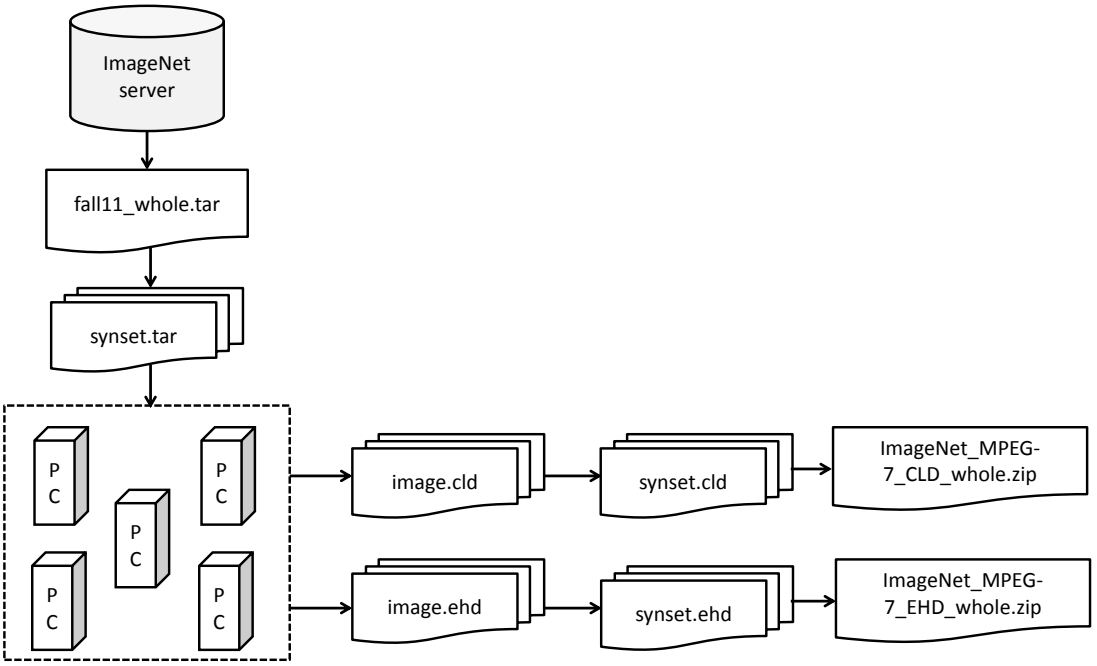


Figure 8.2: ImageNet. Pipeline that has been used to obtain and describe the ImageNet image collection.

8.3.2 ImageNet-7M

From the ImageNet image collection, we have extracted a subset of almost 7 millions images which we refer to as ImageNet-7M. This subset corresponds to the first 9,350 synsets and contains $6,930,901$ images. As mentioned in Chapter 6, it has been structured by our algorithm in 4,5 days. The hybrid HRNG structure that is generated has a height of 6. It contains $1,276,497$ nodes (with $1,232,522$ leaves) and $1,293,528$ graph edges.

During our experimentations, we have exported temporary HRNG files. Indeed, after each batch of 500 processed synsets, the current HRNG has been exported as a JSON file. Figure 8.3 presents the evolution of the HRNG file size with regard to the number of processed synsets. As one can expect, the file size increases along with the number of processed synsets. The final ImageNet-7M HRNG file has a size of almost 5Gb.

The load of this huge JSON file in a browser may be an issue. Moreover, even if a browser can handle to load such a file, the RAM usage will be considerable. To address this issue, we propose to split this huge file into several ones. Two solutions can be used:

1. generate one file for each node in the hierarchy,
2. generate one file for the inner hierarchy and one file for each leaf.

In our implementation, we have selected the fist option. Indeed, the inner hierarchy file has a size of about 900Mb, which is still a large number for both the browser memory capacity and the RAM usage. Hence, we have generated as many files as nodes in the HRNG

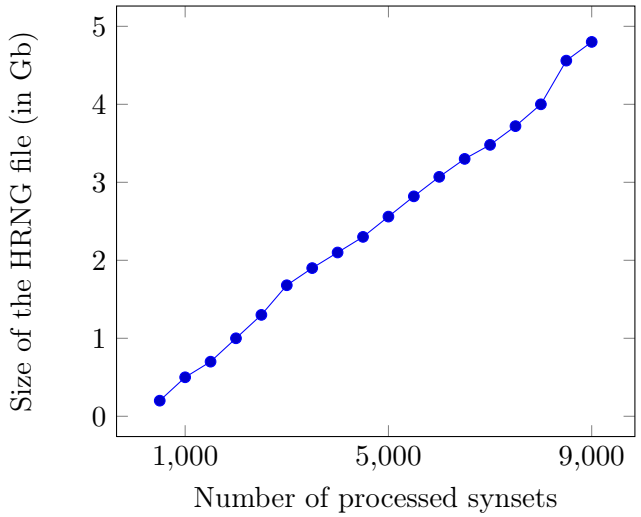


Figure 8.3: ImageNet subset HRNG file size (in Gb).

structure. Each file is a standalone graph (in json), and the parent-child relationship is handled by giving the child node id, with corresponds to its file name.

Another issue is the fact that a considerable number of the images are not available online anymore. Indeed, to deal with licence issues, INeX does not store the original images, but rather gets them by their hyperlinks. The ImageNet image collection being built in 2009, many of these links are not valid. To address this issue, we have gathered and stored the images in a remote server from our university. Unfortunately, because of permissions issues, only a local version of the platform is operational.

8.4 The INeX platform

Figure 8.4 presents the architecture of the INeX platform. The most interesting subfolder is the *data/* one. It contains:

- *images/*
  a copy of the original images, stored in a remote server from the university, to address the issue of invalid url,
- *rngs/*
  the aforementioned json files that describe the HRNG,
- *urls/*
  one file per synset, that contains the urls of the original images.
  This could be used if one wants to use the original location of the images,
- *structure_released_2016.xml*
  a file that contains the Wordnet description associated with ImageNet.

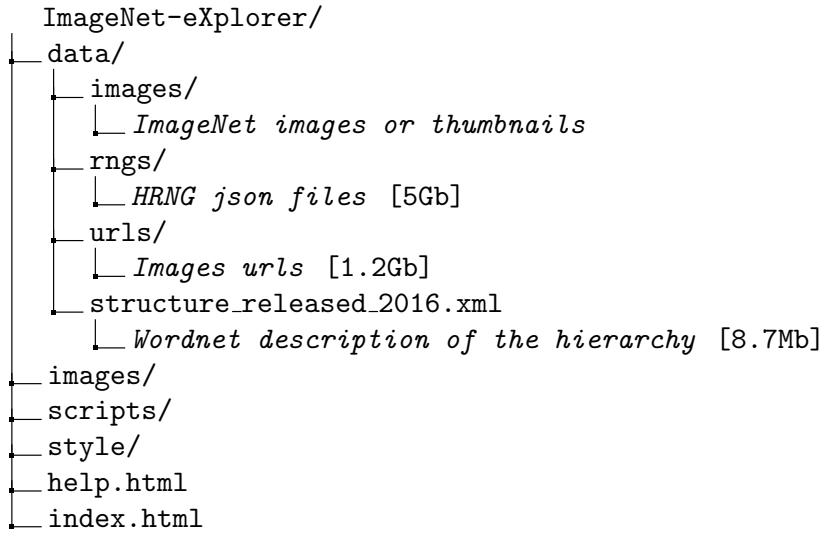


Figure 8.4: Directory tree of ImageNet eXplorer.

Figures 8.5 and 8.6 show the logo and the interface of the INeX platform interface respectively. We do not detail here the visualisation and interactions available in the INeX platform. Indeed, most of them are the ones described in the previous chapter (please refer to sections 7.2.3 and 7.2.4).

The exploration of the ImageNet-7M collection using the proposed platform got good feedbacks from early users and the people involved in the user evaluation (see Chapter 11). These results are promising for the usage of the proposed paradigm for the interactive exploration of large collection of images.

8.5 Conclusion and future work

In this chapter, we have presented INeX, an interactive platform for the ImageNet-7M image collection exploration. This use case supports the scalability property of the joint study on the indexing and visualisation of this thesis. Even if it is not presented in detail, the incremental aspect is addressed by our algorithms and INeX could easily handle this aspect with file update. Moreover, our custom platform allows one to wander around the image collection thanks to the provided interactions.

Several future directions can be considered. First, the INeX platform could be enhanced with new interactions (*e.g.* remove the node overlap; given a internal node, allow a direct access to its leftmost leaf; *etc*). Second, since ImageNet relies on an existing hierarchy, it could be interesting to take advantage of it and think about dedicated interactions (*e.g.* multi-synsets visualisation, possibility to add tags to images or synsets, *etc*). Third, we might process the whole ImageNet data set. Indeed, thanks to our incremental indexing

algorithm, it boils down to inserting the non-processed synsets in the HRNG of ImageNet-7M. Last, we may create an online version of INeX.

Figure 8.5: Logo of ImageNet eXplorer.

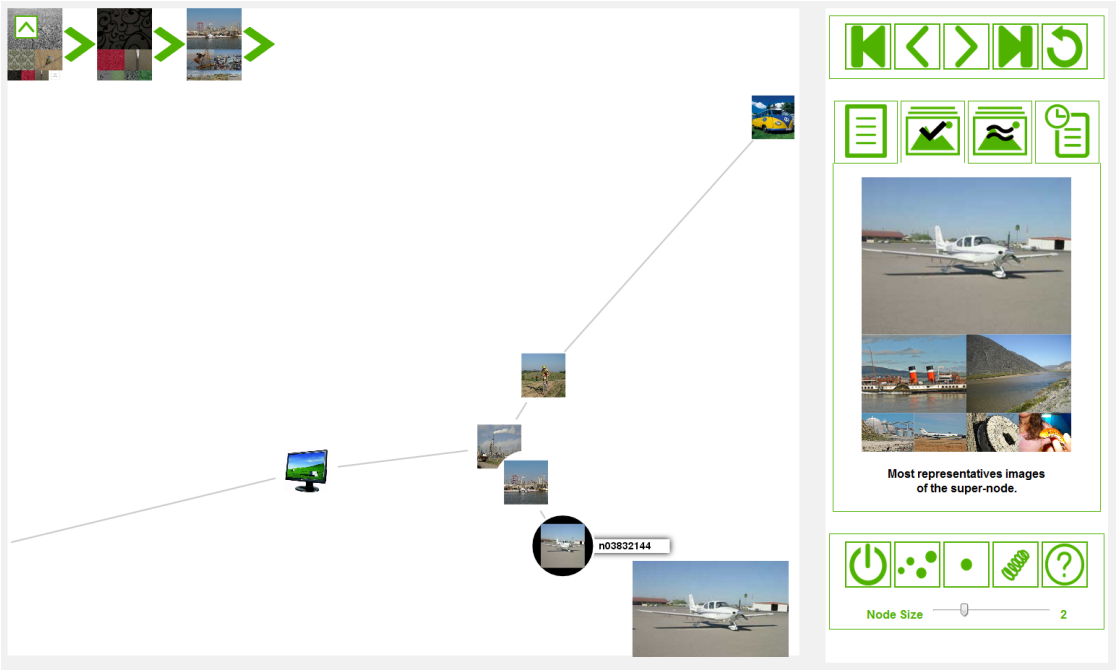


Figure 8.6: Interface of ImageNet eXplorer.

Chapter 9

APoD eXplorer

Abstract

In this chapter, we describe how our work has been used to build a recommendation system and an interactive exploration platform for an ever-growing image collection. To illustrate the relevance of such tools, we present APoD eXplorer, a platform that enhances the exploration of the NASA Astronomy Picture of the Day (APoD) image collection. The platform is available online at `http://frederic.rayar.free.fr/apod/`.

Contents

9.1 Introduction

In this use case, the NASA Astronomy Picture of the Day (APoD) [1] image collection has been used. It is a nice example of growing image collection. Every day, a new astronomical picture - sometimes a video - is put online, along with a brief description written by a professional astronomer. The project started in 1995 - almost 7600 entries up-to-date - and is still active - more than 14,000 likes and followers on Facebook and Google+ fan pages, respectively. Note that this use case deals with copyrighted images:

> *"All the images on the APoD page are credited to the owner or institution where they originated. Neither NASA nor APoD can grant permission to use copyrighted images."*

In the main page of the project website [Nemiroff and Bonnell, 1995], one can see the new image and its information. A link to the archive and a text-based search engine are also available. Figure 9.1 illustrates the homepage of the APoD official website.

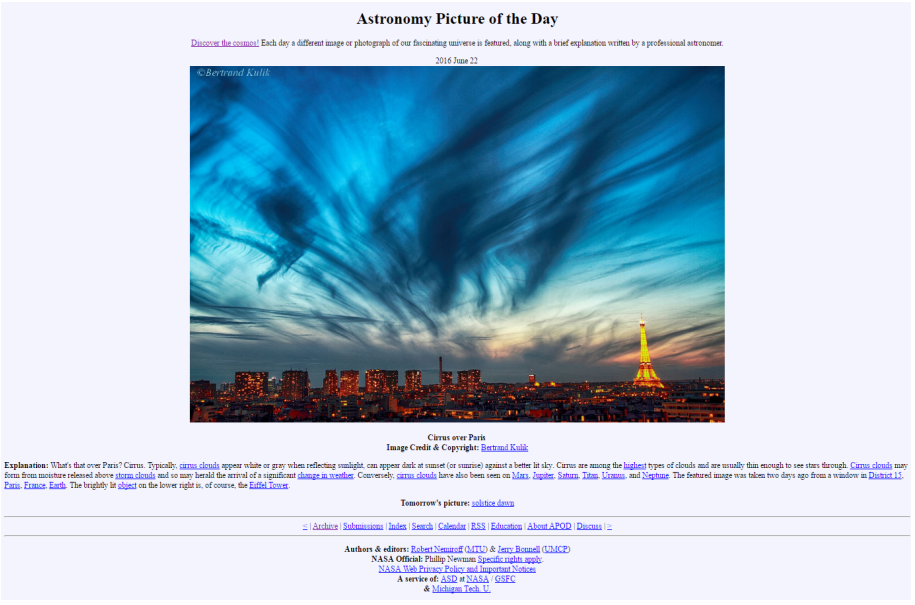


Figure 9.1: APoD project official website homepage. 2016 June 22 - *"Cirrus over Paris"*.

We think that a richer experience could be brought to users, by using our indexing and exploration platform. Indeed, as soon as a new image is posted, it could be retrieved, analysed and then inserted in our hybrid structure. The structure can then be:

- exploited to propose similar images as recommendations to users with regards to the newly posted image,
- visualised to allow an interactive exploration of the image collection.

[1] `http://apod.nasa.gov/apod/`

Thus, one could wander around the collection instead of just seeing the image of the day. This could lead to nostalgia (the APoD project exists since more than 20 years) and even serendipity.

We describe in this chapter how our structure can be used to build a content-based recommendation system, and how a daily service could be created upon our work to give the user a more richer experience than what the official homepage provide currently.

9.2 Recommendation system

We propose here to take advantage of the proposed image indexing structure to build a simple content-based recommendation system [Jannach et al., 2010]. The underlying idea is to provide to the user recommendations that correspond to similar images to the current one.

Several schemes could be considered:

- build a graph, for a given image descriptor (*e.g.* color), and consider the direct neighbours of the current image as recommendations,
- build a graph, for a given image descriptor, and consider all nodes that are within a given *graph geodesic* distance d as recommendations,
- build a graph, for a given image descriptor, and consider a fixed number k nodes that are within a given *graph geodesic* distance d as recommendations,
- build several graphs, for a set of image descriptors (*e.g.* color, texture, shape), and consider one of the three strategies given above to create the recommendations.

In the implemented proof-of-concept platform, that is described in this chapter, the first option has been selected. There is no specific reason that motivates this option, except for convenience.

9.3 Toward a daily service scenario

Thanks to the incremental property of our algorithm, our system can be deployed on a server, and daily takes into account the new image posted on the APoD website. Figure 9.2 illustrates a feasible workflow.

A daemon could run on the APoD eXplorer server. It will get the day information using the NASA API. If it is a video, we could directly update the main page of our system, and propose recent videos as recommendations. If it is an image, the daemon will download it and eventually convert it to be handled by our descriptor extraction application. As soon as the descriptors are computed, we can discard the image from the server to avoid any copyright related issues.

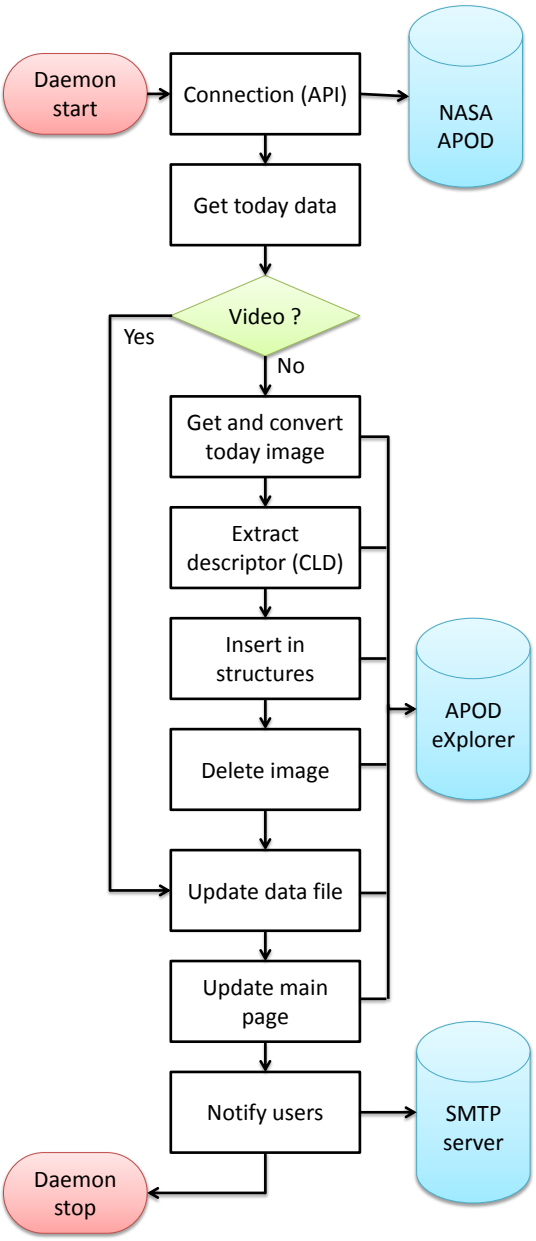


Figure 9.2: APoD eXplorer: Online scenario workflow.

The new image is now represented by descriptors, but also by some metadata given by the API (*e.g.* title, url, etc.). It is then inserted in the structure and related files are updated. Finally, the main page of our platform is updated to present today astronomy picture, its information and its recommendations. One can also imagine a scenario where users will be notified of this update with a mail or a RSS stream.

9.4 APoD eXplorer

We present here APoD eXplorer, a proof-of-concept platform that implements a recommendation system and an interactive exploration of the NASA APoD image collection. The daily service scenario presented above is not currently deployed, but rather an "offline processing" is used.

APoD eXplorer is available to test online at `http://frederic.rayar.free.fr/apod/`.

9.4.1 Data set

To yield the presented results, images and their descriptions have been retrieved from the APoD website using NASA Open API [2]. 6,871 images have been crawled, from January 1996 until a cut-off date (October 2015). Sometimes, a video is proposed instead of an image. We did not retrieve nor process the videos in this work. A few images in GIF format have also not been considered.

In this implementation, colour descriptor, namely Colour Layout Descriptor (CLD) [Manjunath et al., 2001], has been computed for each image, and used to yield our structures. The information extracted has also been stored by year to be leveraged in timelines.

9.4.2 Overall description

Figures 9.4, 9.5 and 9.6 present the views of the three main visualisations of APoD eXplorer.

Figure 9.4 presents the main view and the recommendation system. The current day image is displayed, along with its information. In the right panel, recommendations are proposed, and the user can navigate and wander in the collection by selecting successively proposed recommendations.

The user can also explore the archive using a well known visualisation: timelines. Figure 9.5 presents the timeline visualisation. Indeed, as the images have time related information, it is relevant to propose a visualisation which most users are familiar with. In order to avoid having more than 6,000 elements to display in the timeline, we have segmented the images by year. Timeline JS [Lab, 2015] - an open source JavaScript library - has been used to implement the timelines.

In Figure 9.6, the hierarchical and graph-based hybrid structure visualisation is displayed. It consists of the graph visualisation canvas (left) and controls/information panel (right). Linkurious JS [Heymann, 2013] - another open source JavaScript library - has been used to manage the graph visualisation. The first level of the proposed hybrid structure is displayed in the main frame. Some of the available interactions are briefly described below.

[2] `https://api.nasa.gov/api.html`

9.4.3 Interactive exploration

9.4.3.1 Graph manipulation

Common interactions such as zoom, pan and graph dragging are available to allow the user to manipulate the graph and select a region of interest in the graph. Furthermore, if the user does not want to have a visual notification of the adjacencies and their information, he can hide the edges, that will not be rendered. A graph drawing algorithm is also runnable to layout the graphs.

9.4.3.2 Image visualisation

One important feature of the platform is the possibility to display the images that are represented by a node. Thus, the user can either *(i)* continue to see the node as circles and visualise the related images as thumbnails by hovering the cursor over nodes or *(ii)* display each node as image and select the size of the thumbnails thanks to a slider.

9.4.3.3 Focusing on an image

The user can focus his attention on a single image by clicking on it. A zoom is done by placing the selected image on the center of the visualisation canvas. Its adjacencies are highlighted and related informations are given in the right panel. A larger thumbnail is displayed and can be clicked to display the image in its original size in a separate window. Displayed information can be either metadata, description or annotations. Furthermore, the graph neighbours of the image are displayed to allow the user to quickly visualise the neighbourhood of the selected image.

9.4.3.4 Exploring a graph

In order to explore a graph, several means are proposed to the user. First, one can navigate in the graph canvas by clicking successively on the nodes he is interested in. Secondly, one can navigate using the graph adjacencies. Once an image is selected, its graph neighbours are displayed in the information panel and are clickable. By clicking on one of the thumbnail, the graph's focus on the canvas changes to the selected thumbnail node. Thus, one can perform a step-by-step exploration of the image collections by leveraging the graph adjacencies. Finally, the navigation in the image graph is stored and a log is available to allow one to rapidly go back to any of its previously visited images.

9.4.3.5 Exploring the hierarchical structure

If the user decides to display node images, internal nodes are represented by their first representative. Regarding the navigation, when the user flies over an internal node, its first representative is shown as a thumbnail in the bottom-right of the graph canvas and seven of its nearest representatives are displayed in the information panel. Seven of the farthest representatives - that can be considered as outliers - are also displayed in the information panel. Thus, the user is assisted in his navigation of the hierarchical structure. Clicking on

an internal node will display its subtree. The number of images contained in the subtree is displayed in the information panel. Roll-up is also possible to explore different branches of the tree. A visual breadcrumb is also provided to help one to locate himself in the hierarchy exploration.

9.5 Conclusion and future work

In this chapter, we have described how our joint study of indexing and visualisation has been used to build a recommendation system and an interactive exploration platform for an ever-growing image collection. We have presented APoD eXplorer, a proof-of-concept platform that allows to enhance the exploration of the NASA Astronomy Picture of the Day image collection.

As future work, firstly, we want to finish implementing a few functionalities and interactions under development in the current proof-of-concept version. For instance, we could consider a different recommendation scheme. Secondly, we plan to implement the daily service scenario, where the whole image collection will be accessible (no cut-off date). Lastly, we would like to take advantage of the metadata. Indeed, as a title and a description are available for each image, it would be interesting to use textual descriptors to compute similarities between images.

Figure 9.3: APoD eXplorer logo.

Figure 9.4: APoD eXplorer: Recommendation visualisation. On the left, informations about the selected day are displayed. On the right, recommendations related to the selected day are proposed to the user.

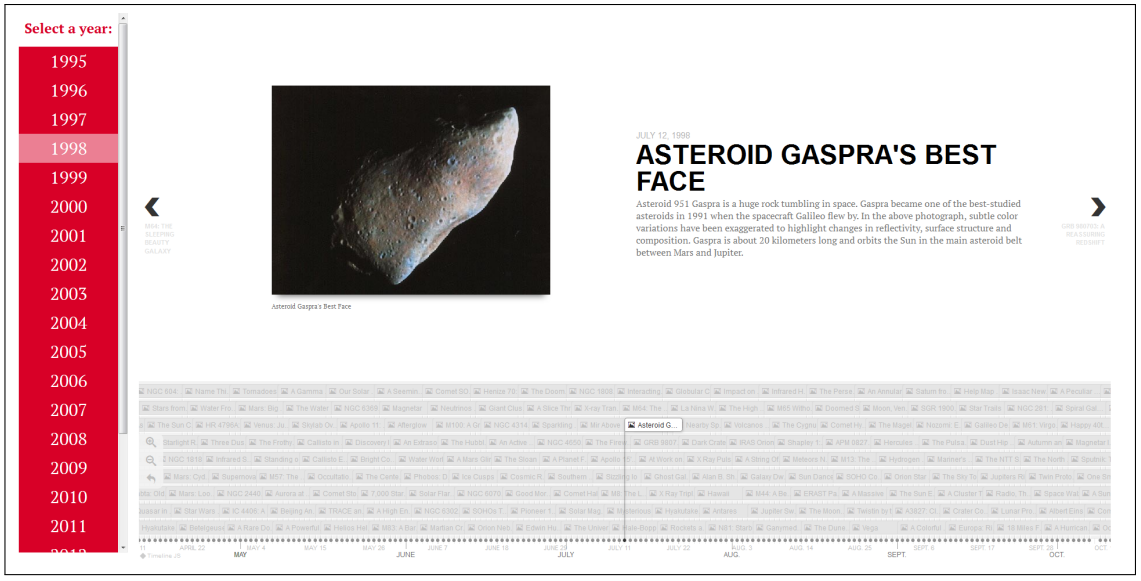


Figure 9.5: APoD eXplorer: Timeline visualisation with Timeline JS. The user can visualise one timeline per year.

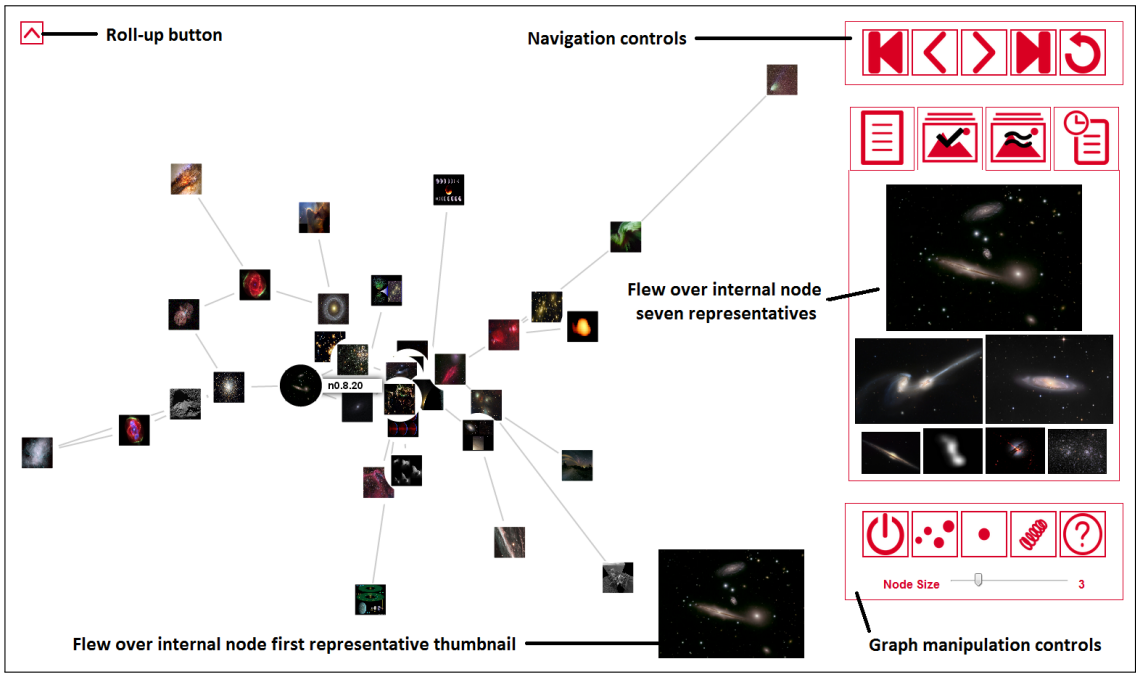


Figure 9.6: APoD eXplorer: Interactive exploration visualisation. On the left, the visualisation of the proposed hybrid hierarchical and graph-based structure. On the right, selected node informations are displayed, along with general controls.

Chapter 10

DINet Viewer

Abstract

In this chapter, we present a use case that takes advantage of our contributions to build a visual analysis system. This system allows one to qualitatively assess the features and distance functions that are used for calculating similarities between images. We present an application in the Document Image Analysis field, where word images from a historical book are considered. This collection is indexed using a RNG and visualised thanks to our platform, namely DINet Viewer. This interactive visualisation tool inherently helps users to quickly analyse and understand the relevance and robustness of selected features and corresponding distance function in an unsupervised way, i.e. without any ground truth. Experimentations are performed on a handwritten data set of segmented words and consider three types of features and four distance functions. These material are used to evaluate the relevance of the built graph and the usefulness of our visual analysis system. A demonstration of the platform is available online at `http://frederic.rayar.free.fr/dinet/`

Contents

10.1 Introduction

In many fields that deal with image data, describing and comparing images are essential yet challenging tasks. We focus in this chapter on the Document Image Analysis (DIA) field. Indeed, the choice of features and distances is a crucial step and has a deep impact on many applications like optical character recognition, word-spotting or writer identification. One can find many papers where new features or distances are proposed to address new challenges: difficult scripts, historical documents or camera-based input.

In most of these works, an evaluation of the proposed features or distances is done to assess the quality of the contributions regarding an objective and the studied data set(s). Both qualitative and quantitative experimentations can be done. In this process of evaluation, one usually performs the following steps: *(i)* use a public accessible data set or create one, *(ii)* create the ground-truth if not already available, *(iii)* extract the proposed features, *(iv)* then use a classifier and cross-validation to generate some metrics, for instance the accuracy or precision-recall. However, obtaining data with ground-truth requires quite a lot of time and effort. Furthermore, after using this kind of procedure, it is difficult to look into what went wrong and why, like misclassification or erroneous recognition. Thus, one may miss a chance to reassess the used features or distance functions and improve their relevance.

Visual analytics is a paradigm that combines automated analysis methods with interactive visual interfaces to allow one to reason and understand complex data sets. Only a few works take benefit of this paradigm to analyse documents entities. For example, [Uchida et al., 2012] and [Goto et al., 2013] use graphs to study the distribution of handwritten digits, while [Nakamoto et al., 2013] and [Uchida et al., 2015] study font distributions. In [Akao et al., 2014], multidimensional scaling is used to visualise the diversity of japanese handwritten hiragana. In these works, features and distances are chosen and not discussed afterwards.

In this chapter, we propose a visual analysis system that allows a fast and qualitative assessment of word image features and distances. The images are structured using a relative neighbourhood graph (RNG) and our platform, namely DINet Viewer, is used to interact with the data. The rest of the chapter is organised as follows: first, we present the overview of the system and we describe DINet Viewer. Then, the material used in the experimentations are detailed. Finally, we evaluate the relevance of the built graph and the usefulness of the platform before concluding.

10.2 System overview

Figure 10.1 presents the workflow of the proposed approach. First, images are described by features and a distance function is considered to compare images. From this, a distance matrix can be computed. Second, the matrix is used to structure the word images in a RNG. Third, the graph is laid out in a two-dimensional plane, thanks to graph drawing algorithms. Last, our platform allows one to visualise the graph depiction and interact with it.

We argue that the graph that is used allows to highlight the topology of the data: it

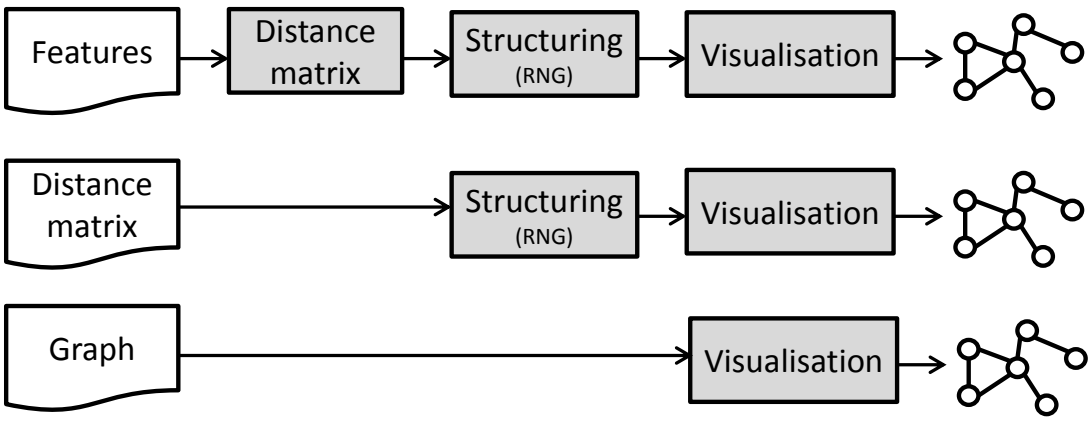


Figure 10.1: Workflows of the approach proposed in the DINet Viewer use case.

gives the global distribution and underlines the similarity between the data. Moreover, the global visualisation is a good qualitative clue to assess the quality of the features or distances. One can also zoom to study local neighbourhood of a node or explore the graph neighbour by neighbour while having a visual feedback. This interaction allows one to discuss some adjacencies that can be erroneous and put into question the choice of features or distances. Thus, one should expect the following observations: *(i)* similar word images may be linked by an edge, *(ii)* on the contrary, dissimilar word images should not be linked, or at least by a long edge. This last phenomenon occurs because of the connectivity property of the selected proximity graph.

10.3 Platform description

Figure 10.2 shows the platform interface. An online demonstration of this platform is available at `http://frederic.rayar.free.fr/dinet/`.

Some of the visualisation and interaction specifications of the platform are briefly described below:

- first, the graph is laid out and shown as a whole. The user can quickly have a glimpse on the global distribution of the word images and their topology. If the features and distances are discriminant, one may already perceive some connected components,
- the user can display the word images related to each node. Thus, the network can be interpreted more easily,
- to have a better view of the word images, the user can move the pointer above a node: a larger thumbnail is displayed,
- when a node is selected, a zoom is performed and the focus is set on it. The adjacencies of this node are highlighted. Thus, the user can study local neighbourhood of a node,

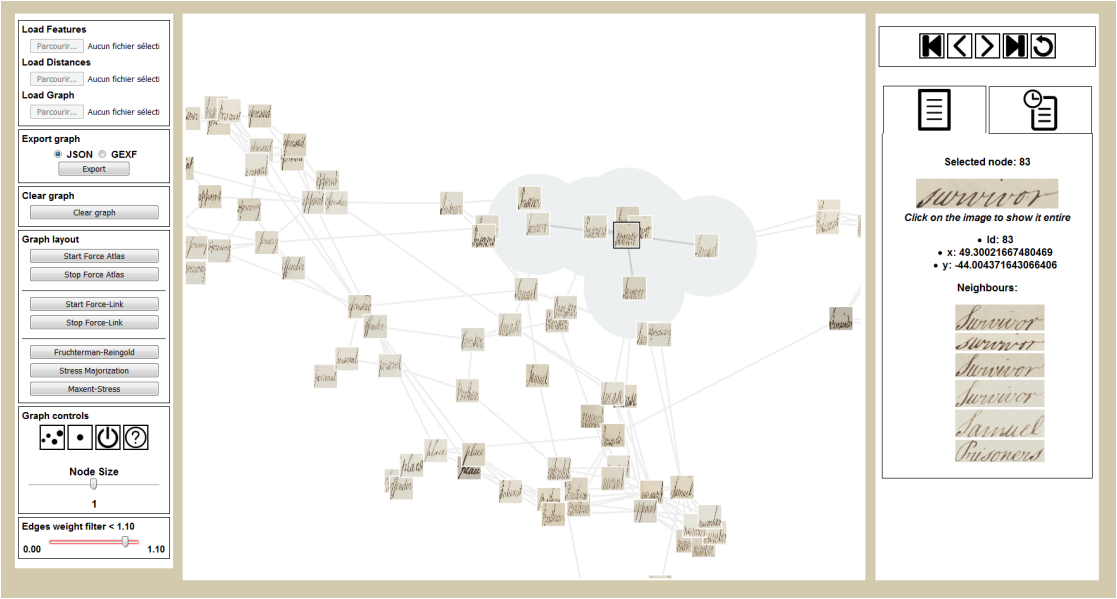


Figure 10.2: Interface of DINet Viewer.

- in addition, when a node is selected, related information is displayed on the right of the graph. Among these details, the larger thumbnail is displayed, and clicking on it allows to display the image at its original size. The relative neighbours of the selected graph are also displayed, and the user can perform a step by step exploration,

- also, a history of the user navigation is stored and displayed, allowing him to return to any previous step of his exploration.

In addition to these interactions, we have also implemented more specific functionalities following early users feedbacks. The comments were on the edge interactions, such as filtering or statistic overview. To address these comments, we have proposed to implement dedicated widget-like views. Figure 10.3 illustrates such widgets.

10.4 Material

10.4.1 Data set

The Bentham data set [Gatos et al., 2014] consists of a series of documents from the Bentham collection. It has been prepared in the tranScriptorium project [1]. This data set mainly includes manuscripts that were written by Jeremy Bentham (1748-1832) himself over a period of sixty years, as well as written copies by Bentham's secretarial staff. In our experimentations, 100 word images have been selected and ground-truthed. The ground-truth has been used only to evaluate the relevance of the used graph.

[1] http://transcriptorium.eu/ icdar15kws/data.html

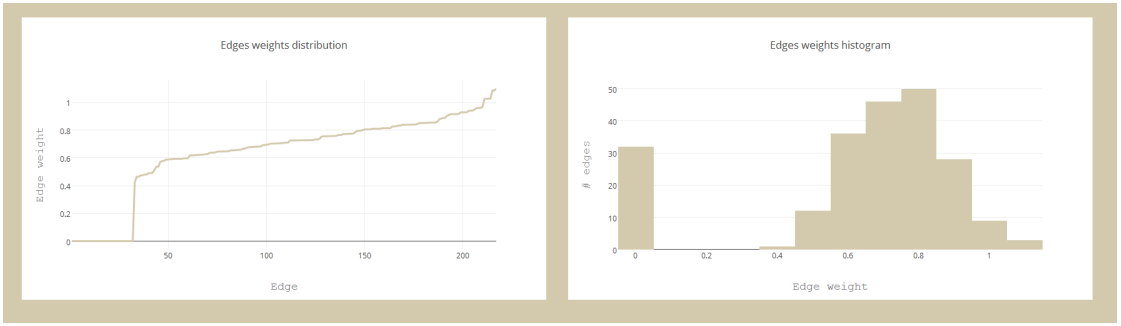


Figure 10.3: Visualisation of DINet Viewer widgets. (Left) Edge weights distribution plot. (Right) Edge weights histogram.

10.4.2 Features

In DIA applications, once the segmented words are obtained from the data sets, one need to extract features from them. In this section, we briefly describe three different categories of features that are used in the literature for word image description, and that are considered in our experimentations.

Column-based features We use a set of statistical column-based features previously used for handwriting recognition [Marti and Bunke, 2001]. Although these features can be outperformed in terms of accuracy by more complex features, they remain quite interesting due to their less computational cost and comparative accuracy. Here, we have chosen 8 features to define each pixel column. Thus, for an image with a pixel width of N, sequences of feature vectors are obtained by moving from the left to right over the word image. Please see [Mondal et al., 2014] for more details on this column-based features.

Slit style HOG based features The slit style HOG (SSHOG) [Terasawa and Tanaka, 2009] is a specially modified version of HOG [Dalal and Triggs, 2005], to make it suitable for *word spotting* applications. A fixed sized slit window is slid over the image in an horizontal direction for extracting features from each slit. Please see [Terasawa and Tanaka, 2009] for more details on SSHOG.

Block style HOG based features The classical HOG descriptor was improved by [Felzenszwalb et al., 2010]. For a given height normalised word image of $M \times N$ pixels, the image is divided into fixed size cells of size c pixels. We extract HOG descriptor, that consists of 31 individual features, for each cell. Thus, we get a $m \times n$ matrix of HOG descriptors, where $m = \frac{M}{c}$ and $n = \frac{N}{c}$. Finally, we create a $(31 * m) \times n$ matrix, where for each column, we concatenate the m HOG descriptors of the $m \times n$ matrix.

10.4.3 Distance functions

After extracting the features from each image, one needs a distance function to calculate the (dis)similarities between the images. In these experimentations, four distances have

been considered. We briefly give the ideas behind these algorithms below. More details on the algorithms and their implementations can be found in the original papers.

Dynamic Time Warping (DTW) This technique [Albrecht, 2009] is a dynamic programming (DP) based approach for calculating the optimal correspondence between two feature sequences X and Y. To align these two sequences, we construct a matrix, where each element of the matrix corresponds to the squared distance between elements of the sequences. Then, DTW computes a path cost matrix $\mathfrak{P}$ using dynamic programming.

Itakura Parallelogram To speed up DTW and to avoid pathological matching, constraints are widely imposed for the calculation of warping path. It reduces time complexity by limiting the number of cells that are evaluated in the cost matrix. Itakura band [Albrecht, 2009] is a global constraint that gives an efficient trade-off between accuracy and speed, when it is defined properly.

Pseudo Local DTW (LDTW) This approach extends the DTW algorithm to perform pseudo-local alignment using a specific DP-path [Listgarten et al., 2005]. It applies different DP paths at different location of path cost matrix ($\mathfrak{P}$) for handling stretching and compression of individual points in time series data.

CDP This technique [Oka, 1998] is able to perform subsequence matching (full query in longer target) and to locate multiple occurrences of the query in the target. Even so, this algorithm works well with properly segmented words.

10.5 Graph relevance

In order to evaluate the relevance of the graph, we have used the *Bentham* data set. The main objective is to check whether the observations that can be done with a classical information retrieval metric can also be done using the graph. Using the ground-truth, the mean average precision (mAP) of each feature and distance pair has been computed. Furthermore, the RNG has been built for each of the mentioned pairs, and a graph metric has been calculated.

First, let us define the mean average precision. Given a query, we define *Rel* as the set of relevant similar word images with regard to the query and *Ret* as the set of ranked retrieval results from the data set. The precision at k, noted $P@k$ is obtained by computing the precision by considering only the k top most results that are returned by the system. The mAP is the average of the precision at k for each relevant answer in the ranked retrieval results. Let $r(k)$ be the binary function on the relevance of the k^{th} item in the returned ranked list, the mAP is calculated as follows:

$$mAP = \frac{\sum_{k=1}^{|Ret|} P@k \times r(k)}{|Rel|}$$

Table 10.1: Mean average precision for the Bentham data set over studied features and distances.

	CDP	DTW	Itakura	LDTW
Column-based	0.23	0.22	0.26	0.23
SSHOG	0.03	0.11	0.09	0.10
BlockHOG	0.19	0.26	0.28	0.24

Table 10.2: Graph average precision for the Bentham data set over studied features and distances.

	CDP	DTW	Itakura	LDTW
Column-based	0.37	0.32	0.30	0.38
SSHOG	0.12	0.21	0.20	0.26
BlockHOG	0.32	0.39	0.41	0.37

Table 10.1 gives the average mAP values that have been computed for each presented feature and distance pair in Section 10.4. Each image has been considered as a query, and searched in the remaining 99 word images, hence $k = 99$. Then, the average mAP on the hundred queries has been computed.

Second, we define the metric that has been considered to evaluate the computed graphs. Let us consider a graph $G = (V, E)$, where V and E are the set of nodes and edges of G, respectively. For each node $n \in V$, we compute the precision $P(n)$ given by $P(n) = \frac{TP(n)}{RN(n)}$, where $TP(n)$ corresponds to the number of relative neighbours of n that have the same class as n and $RN(n)$ corresponds to the number of relative neighbours of $n \in G$. Then we compute the average precision $P(G)$ of the graph with

$$P(G) = \sum_{n \in V} \frac{P(n)}{|V|},$$

where $|V|$ is the number of nodes of G. Table 10.2 gives the average precision of the computed graphs for each feature and distance pair that are presented in Section 10.4.

As we can see in Table 10.1 and Table 10.2, the graph metric allows one to make almost the same observations than the mean average precision. For instance, regardless of the distance function, one can state that SSHOG performs less well than the column-based feature or the BlockHOG. As well, if we choose one set of features, we can rank the distance functions and decide which one is more prone to perform well with the selected features.

Thus, the generated graphs highlight the quality of the chosen features and distances as well as a classic information retrieval metric.

10.6 Visual analysis system usefulness

In this section, we illustrate the usefulness of the proposed visual analysis system with a set of use cases. These use cases are scenarios that could be helpful to DIA experts

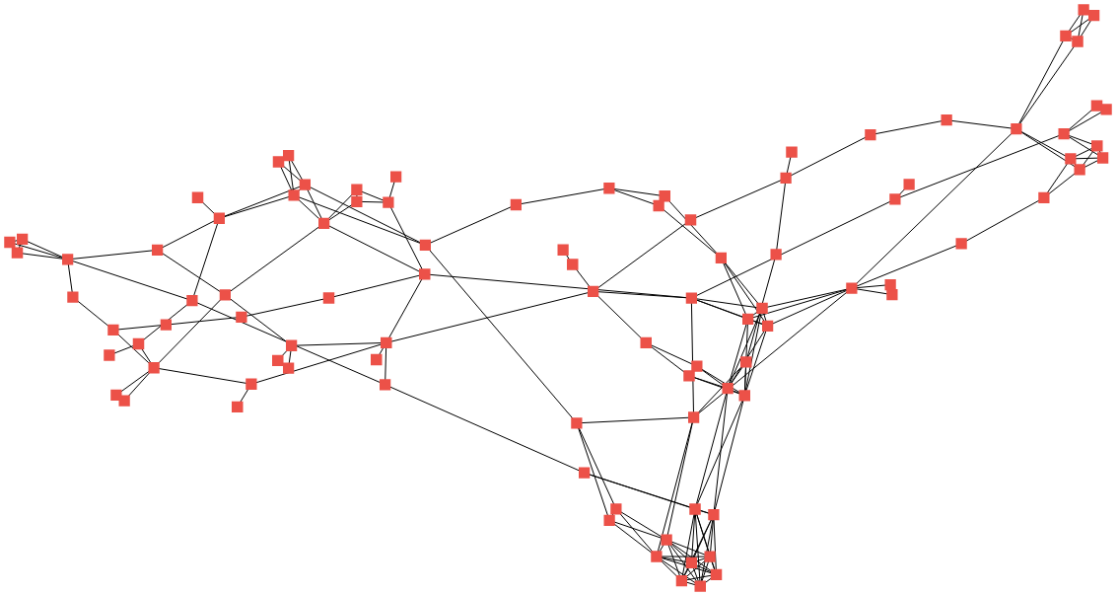


Figure 10.4: RNG drawing using BlockHOG features and Itakura distance. One can observe a community in the bottom of the graph.

when assessing the quality of the features, the distance functions or both. Here, the experimentations are done in an unsupervised way, *i.e.*, the ground-truth of the word images is not presented. A discussion about some limitations of the platform is also presented at the end of this section.

10.6.1 Global visualisation

The first visualisation that is presented to the user is the whole graph. It could be laid out thanks to drawing algorithms, to underline the topology of the graph, and hence the data set. Using this visualisation, one can quickly get a glimpse of the distribution of the images. On the one hand, *communities* (*i.e.* group of nodes that are densely connected between them) may highlight very similar word images, with regard to the chosen feature and distance pair. On the other hand, a graph with no structure may be synonymous of a non discriminative pair. In Figure 10.4, one can observe a graph that displays a community and other dense groups of nodes while in Figure 10.5, no immediate structure appears.

10.6.2 Local neighbourhood study

One of the main goals of the proposed platform is to allow the user to locally study parts of the graph. Thus, using available interactions, one can zoom on a region of interest of the graph, and select a given node. Then, one can visualise and make observations on the neighbourhood of the selected node. Figure 10.6 illustrates a case where a word image is correctly linked to its similar images. At the same time, it is linked to other images which are not really similar. This could be either due to the choice of the features and distance pair or because it is a consequence of the graph connectivity property. In Figure 10.7, we

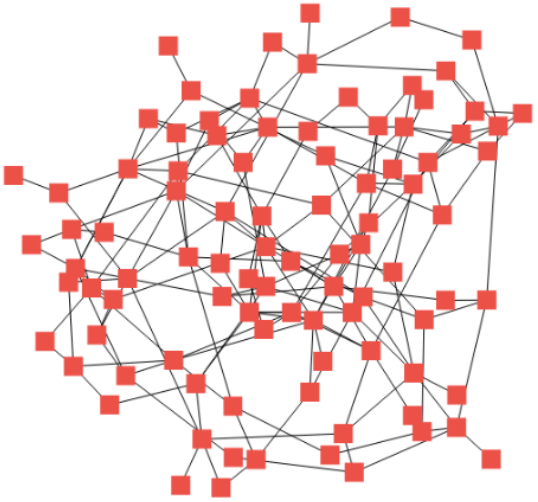


Figure 10.5: RNG drawing using SSHOG features and Itakura distance. One can observe that no specific graph structure is highlighted.

observe a situation where a node has only one relative neighbour and the related word images are not similar at all. Here the bad segmentation of the bottom word image could be an explanation.

Figure 10.6: RNG drawing using BlockHOG features and LDTW distance. The center word image is linked to possible similar word images.

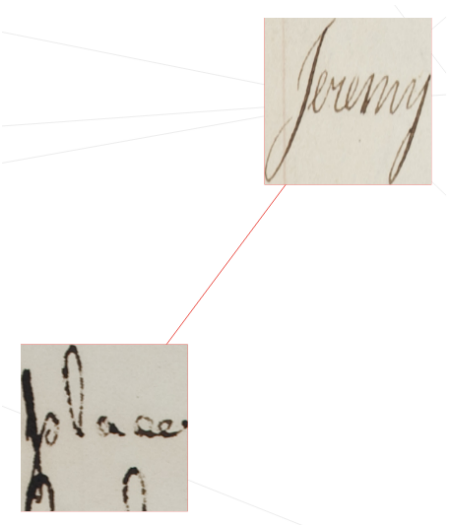


Figure 10.7: Using the same features and distance function as in Figure 10.6, one can observe that an irrelevant adjacency has been created.

10.6.3 Feature evaluation

The proposed visual analysis system can be used to evaluated features only. Indeed, if one uses only one distance function, he could visualise the generated graph over different features. Then, observations can be made either on the whole graphs (*e.g.* which features generate a graph with interesting structures) or be more thoroughly on a local level. Figure 10.8 illustrates a local study. The Itakura distance has been selected, and we observed the relative neighbours of a given word image ("*Majesty*") on the three RNG built considering the three features that are mentioned in Section 10.4. One can observe that the precision is not the same. For BlockHOG features(*cf.* Figure 10.8a), two similar images have been correctly linked. Regarding the column-based features (*cf.* Figure 10.8b), only one among two relative neighbours is relevant. Finally, SSHOG features (*cf.* Figure 10.8c) perform less well: they link only one similar word image among three and this relevant neighbour is the farthest among the three. Thus, based on such local mining on several word images of the graph, one can draw conclusions about the relevance and robustness of the features, regarding the selected distance.

10.6.4 Distance evaluation

In the same way, the proposed visual analysis system can also be used to evaluated distance functions only. Indeed, if one selects a feature, he could visualise the generated graph considering different distance functions. Thus, it is possible to quickly select the most relevant distance function for a given feature but also the most relevant feature for a given distance function. Note that this observation shall be done only on the considered data set and must be tested over other data sets to assess its relevance.

(a) (b) (c)

Figure 10.8: Ranked relative neighbours of a given word image : "*Majesty*" using the same distance, namely Itakura, and using (a) BlockHOG features, (b) Column-based faetures and (c) SSHOG features. One can observe different precisions depending on which feature is considered.

10.6.5 Limitations

During our experiments, we underlined some limitations of the proposed visual analysis system. First, one should not draw some conclusions based only on the global topology of the graph. Indeed, the display of the graph depends on the choice of the drawing algorithm, as discussed previously in Section 7.1. Second, we have considered only a small number of images in the qualitative evaluations. Indeed, it is possible to build the RNG for large collections of images in acceptable times, as seen in Chapter 5. This may be useful to study the distribution of large data sets. However, an issue occurs with the visualisation. Indeed, one can face problems such as the overlapping of nodes, hence word images. These occlusions may be an issue for a thorough local study. Furthermore, one must be aware that the interactions will not be as smooth, due to memory issue.

10.7 Conclusion and future work

In this chapter, we have proposed a visual analysis system that allows one to perform a fast and qualitative assessment of features and distance functions of document entity images. To do so, we took advantage of the proximity graph studied in Chapter 5 and the platform presented in Chapter 7. We have proposed DINet Viewer, a use case in DIA field, to study word images extracted from historical books. Experimentations were done considering a set of given descriptors and distance functions for word images. The graph relevance is underlined and the usefulness of the proposed visual system is highlighted by a set of scenarios.

Future work include the improvement of the platform. Indeed, one can easily think about more visualisations and interactions, such as multifaceted thumbnails to improve the evaluation of either only the features or only the distance function. Feature selection interactions could also be embedded to further understand the impact of features. Besides,

graph partitioning techniques could be used to highlight possible communities. Another line of future work would be the use of our hybrid structure, proposed in Chapter 6, to analyse larger image collections.

Chapter 11

User Evaluation

Abstract

In this chapter, we present the protocol, the design and the results of a user evaluation that has been conducted to evaluate the quality and the relevance of the platform we have proposed in this thesis. The analysis of the results shows us that new users easily learn how to use our platform in a few minutes, and that the exploration that follows is smooth and intuitive. Moreover, thanks to its playful and entertaining features, the platform is considered appropriate by the users for the exploration of image collections .

Contents

11.1 Introduction

In order to evaluate the platform that has been presented in this thesis, we have conducted a laboratory experiment [Carpendale, 2008].

The main objectives of this experiment are:

(i) to study whether the platform is easy to learn and use,

(ii) to get user feedback to see if the platform is appropriate for the exploration of image collections,

(iii) to test if the platform allows a smooth exploration, even when handling large image collections.

In the rest of this chapter, we first present the protocol of the experiment (Section 11.2) and describe the people that have participated (Section 11.3). Then, we detail the apparatus and the experiment design in Sections 11.4 and 11.5 respectively. Finally the results are given in the end of the chapter (Sections 11.6 and 11.7).

11.2 Experiment protocol

The experiment that we have conducted consists in the following steps:

- welcome of the participants,
- presentation of the concept of a user evaluation,
- presentation of the overall objective of the platform and description of the protocol,
- filling of the profile form (name, gender, education level, field of work, computer usage, web usage, screen size usually used, knowledge and usage of tree and graph structures and their visualisations),
- learning step: using the help page of our online platform[1], we have taught the participants how to use the platform for an interactive exploration of an image collection. Then, we allowed them to manipulate the system by seeking a given image in the collection,
- three tasks were performed by the participants. For each task, there was a presentation, the fulfilment and a form filling,
- filling of the synthesis form.

The average duration of one experiment was 45 minutes.

[1] `http://frederic.rayar.free.fr/ice/help.html`

11.3 Participants

Figure 11.1 presents the information gathered from the participants of this experiment: their profiles, some computer usage related information and their notions on graphs and trees. We had 15 participants (5 women and 10 men) between 24 and 38 years old. Most of them (80%) were phd candidates or had a phd degree. Computer science was the field of work for most of them (11 out of 15), and the others are working in industrial design, retail design, law and modern literature.

We also asked a few question to get a general idea of their familiarity with computer and web navigation. All of the participants work with computers, at least 4 hours per day, and more than 8 hours for most of them (10 out of 15). 60% are "experts" (they have either notions or experience in web programming). We also inquired about the size of the screen they usually work with. 13 out of 15 work with screen between 10 and 17 inches, *i.e.* laptops. Regarding, their familiarity with graph and tree structures and how to visualise them, 87% have knowledge of tree and graph structures, 67% used them in their work and 73% have already visualised such structures.

11.4 Apparatus

During the experiment, all the participants have used the same computer that we have provided. It was an Intel Xeon CPU W3520 (quadcore) at 2.66Ghz, with 8GB of RAM. The computer was connected to an external 24 inches monitor, with a mouse and keyboard, and had a wired connection to Internet and our university network. It was running Windows 7 and we used Firefox 48.0.2 as web browser.

Three versions of the platform have been used:

1. Image Collections Explorer (ICE), latest version of the generic platform presented in Chapter 7. The online version was used in this user evaluation (`http://frederic.rayar.free.fr/ice/`). ICE uses the Wang image collection (1000 images).

2. APoD eXplorer, presented in Chapter 9. The online version was used in this user evaluation (`http://frederic.rayar.free.fr/apod/`). APoD eXplorer uses the APoD image collection we have gathered (6,871 images).

3. ImageNet eXplorer (INeX), presented in Chapter 8. A local version was used in this evaluation, to access the image collection stored in a remote server. INeX uses the ImageNet-7M image collection (almost 7 millions images).

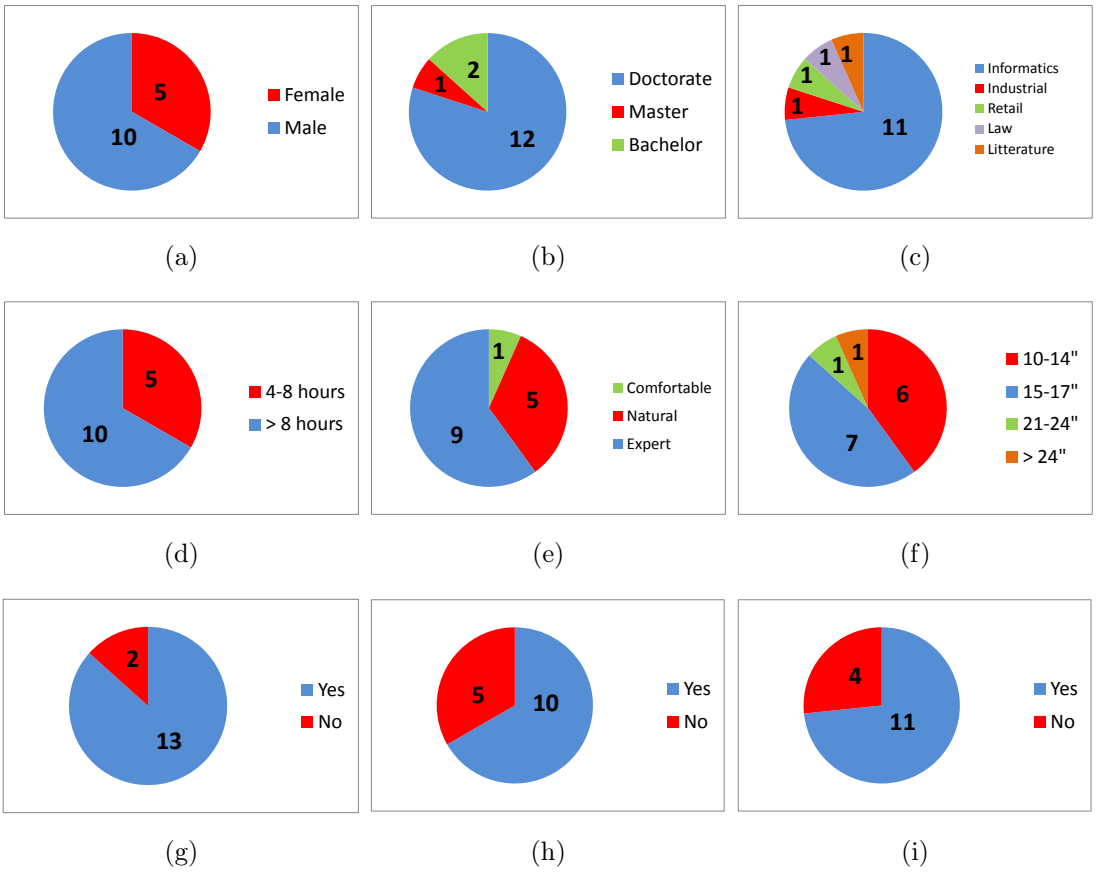


Figure 11.1: Information on the participants: (a) male/female distribution, (b) education level, (c) field, (d) computer usage, (e) web usage, (f) size of the screen that is usually used, (g) knowledge of tree and graphs (in computer science), (h) usage of tree and graph in their work and (i) knowledge how basic visualisation of tree and graph.

11.5 Experiment design

Participants were asked to fulfil three different tasks, that we describe below.

11.5.1 Task 1

The first task T_1 was a goal-oriented task. Such a task has been designed in order to not turn the participants loose in the platform at the beginning of the evaluation. We asked the participants to find in the Wang collection a given image of a dinosaur. The participants were told that they had an infinite time to fulfil the task; however they had the possibility to either stop when a similar picture was found or to simply give up and stop. Thus, this task forced the participants to manipulate the ICE platform to explore the collection.

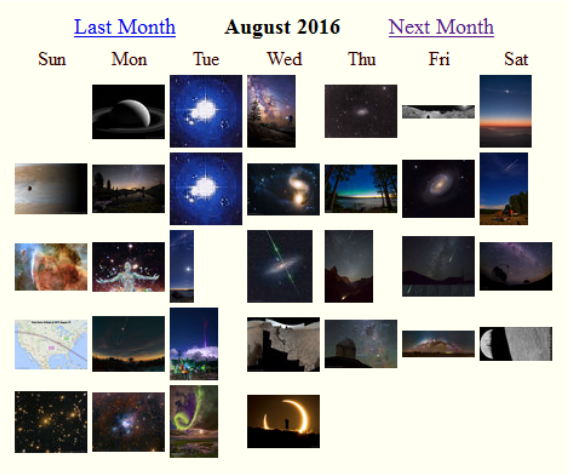


Figure 11.2: APoD Calendar interface.

We measured the task completion time and at the end of the task, the participants were asked to answer a small questionnaire related to the current task. Along with open remarks, participants had to give a score between 1 and 4 regarding the following questions:

- Is the ICE platform easy to learn?
- Is the ICE platform easy to use?
- Is the ICE platform aesthetic?

11.5.2 Task 2

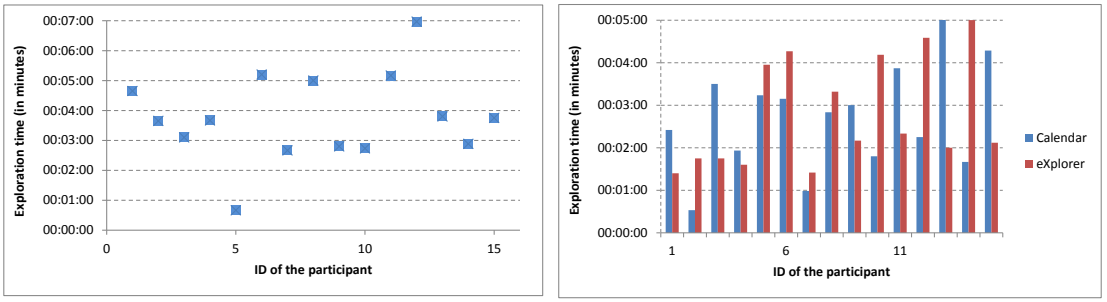
The second task T_2 was an exploratory task with a comparison between two platforms, namely APoD Calendar[2] from the official APoD website (see Figure 11.2) and APoD eXplorer. The APoD project and its archive were presented to the participants. Then, each platform was presented and we asked the participants to find a picture they especially liked in the archive, using Calendar and eXplorer. We have alternated the platforms order so as to avoid a bias. Participants were stopped when they exceeded a 5 minutes exploration on a platform.

At the end of the task, along with open remarks, the participants were asked to answer a small questionnaire to compare the two platforms. The following questions were asked:

- Which platform is easier to use?
- Which platform facilitates the exploration?
- Which platform is the most aesthetic?

We also measured the time of the exploration for each platform, and stored the order of platform usage.

[2] `http://apod.nasa.gov/apod/calendar/allyears.html`



(a) (b)

Figure 11.3: T_1 time results. Exploration time distribution for: (a) T_1 using ICE, (b) T_2 using APoD Calendar and APoD eXplorer.

11.5.3 Task 3

The third and last task T_3 was also an exploratory task. This time, no information related to the image collection was disclosed. The participants were asked to wander around the ImageNet-7M collection without any particular goal. Participants were told that they have an unlimited time for the exploration, but were stopped when they exceeded a 5 minutes session.

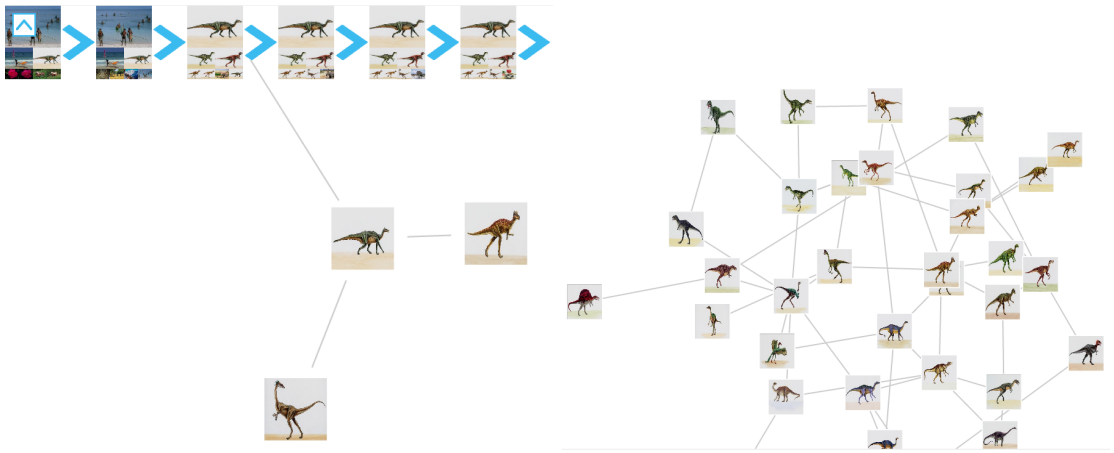
Again, we measured the time of the exploration for each participant, and they had to answer the following questions at the end of the task:

- How many images do you think this image collection contains?
- How fast was the platform? (display/responsiveness)
- Did you felt frustrated by the speed of the platform?

11.6 Results

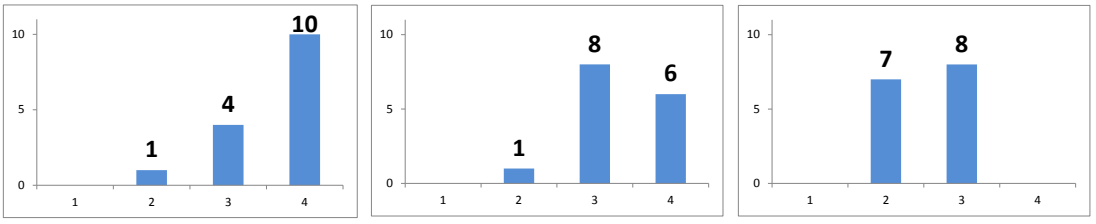
11.6.1 Task 1

In the task T_1, all the participants successfully found the target dinosaur picture in the collection using the ICE platform. The faster participant found it in only 40 seconds, while the longer to complete the task did it in 6m58s. Figure 11.3a presents the exploration time distribution of T_1. Those times are given for information and do not highlight any peculiar observations. Indeed, we have selected a picture that was not totally easy to find. Dinosaur images were spread in three different internal nodes, and each of this node contained several occurrences of dinosaur images (see Figure 11.4). Again, the main idea of this task was to allow the participants to manipulate the interface, and the image to find was only the bait.



(a) (b)

Figure 11.4: T_1 target. (a) dinosaurs could be found on three different internal nodes, and (b) each node had several occurences of dinosaur images.



(a) (b) (c)

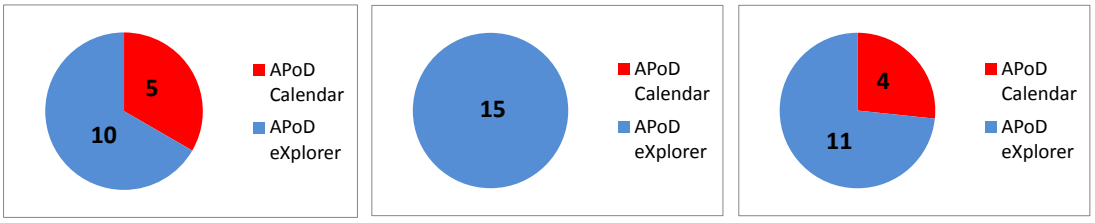
Figure 11.5: Results of T_1: (a) is the ICE platform easy to learn? (b) is the ICE platform easy to use? and (c) is the ICE platform aesthetic?

Regarding the questions that were asked to the participants, the results are displayed in Figure 11.5. Most of them (14 out of 15) found that ICE was easy to learn and use. However, almost half of them found that the aestheticism of the platform could be improved.

11.6.2 Task 2

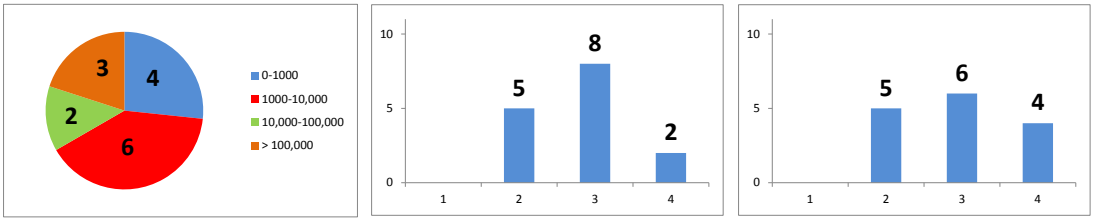
Figure 11.6 presents the results of the second task T_2. 66% of the participants found that APoD eXplorer was the easiest platform to use. All of them found that eXplorer was the platform that facilitates the most the exploration and that supports them to explore the whole collection. Finally, 11 out of 15 found that eXplorer was the most aesthetic platform.

Figure 11.3b presents the exploration times in both platforms. One can note that the exploration time was always shorter when using the second platform (either Calendar or eXplorer).



(a) (b) (c)

Figure 11.6: Results of T_2: (a) which platform is easier to use? (b) which platform facilitates the exploration? and (c) which platform is the most aesthetic?



(a) (b) (c)

Figure 11.7: Results of T_3: (a) how many images do you think this data set contains? (b) how fast was the platform? (display/responsiveness) and (c) did you felt frustrated by the speed of the platform?which platform is the most aesthetic?

One observation that we made is that using Calendar, all the participants were seeking the first pictures based on the date (their birth, or a peculiar event in their life). The participants that used Calendar often browse only 1 or 2 years, month by month before stopping their exploration.

11.6.3 Task 3

The results of task T_3 are given in Figure 11.7. The number of images of ImageNet-7M was not disclosed to the participants, and they were asked to estimate how many images were contained in the collection they explored. We can see in Figure 11.7a that 66% of them thought that the collection did not contain more than 10,000 images. Only 33% of them thought that the exploration was a little bit slow, and felt a little bit of frustration when the images took a few seconds to be displayed.

Hence, the INeX platform succeed to hide the large amount of images explorable, despite the fact that the exploration is a little bit slower. However, feedback from the participants lead us to conclude that in final applications, a visual notification of the number of images contained in the collection must be provided to the users at the beginning of the exploration.

11.7 Subjective results

Overall, the participants felt that the platform was easy to learn and quite intuitive to use. In the last part of the form, the most common positive aspects were *interactive*, *smooth*, *entertaining* and *easy to use*. Most of them (14 out of 15) thought that such a platform was appropriate for the exploration of image collection. The implemented visualisations and interactions were appreciated, especially the breadcrumb and the treemaps, which were qualified of *"very intuitive controls"*. Furthermore, we succeed to conceal the fact that collections could be large, up to millions images (*"We don't realise how many pictures the collection contains"*), despite the fact that the responsiveness is a little bit slower (*"It gets just a bit slow when opening a node and the photograph enlarges, otherwise it's pretty fast."*).

The participants have also highlighted a few bugs and negative aspects of the platform. Among the negative aspects, the most common was the fact that when the image were displayed in the nodes, it was no more possible to distinguish internal nodes, leaf nodes, visited nodes and non-visited nodes. Thus, despite the presence of a log, we had comments such as: *"It's a little bit hard to know where we are in the tree, and to keep in mind which nodes we already visited."*. The visualisation of the MLRNG structure, presented in Chapter 6, could be a solution to provide the user a visual support to assist him in the exploration. Secondly, as mentioned above, the system faces slower responsiveness when dealing with large image collections. This issue could be addressed in future works by using better image load strategies (such as thumbnails loading, parallel web workers, or mipmapping). Finally, one recurring comment was about the images gathered together in a node: *"sometimes unrelated things belong to the same node"*, *"it mixes up the various subjects, example: underwater with monuments, while you are exploring."*. Indeed, even if participants were aware that color features were used to gather similar images, some of them were more focused on image contents. Such feedback motivate us to incorporate available descriptions, keywords or automatically generated captions as features in future works. This would be a straightforward contribution because the modifications to bring to the platform would not be important. However, the choice of such concept-based descriptors and their combination with content-based descriptors will be a scientific challenge.

11.8 Conclusion

We have presented in this chapter the results of the user evaluation that we have conducted to assess the quality and the relevance of the proposed platform. The analysis of the results shows us that new users can easily learn how to use our platform in a few minutes, and that the exploration that follows is quite smooth and intuitive. Moreover, users think that such a platform is appropriate for the exploration of image collections, because of its playful and entertaining features. The platform could also provide a relatively smooth interactive exploration to the users, even when dealing with large image collections, which was a constraint of our endeavour work.

While the results of our evaluation were positive, there are also a few limitations. First, we only had a few number of participants, and most of them were working in the computer science field. Having more users would be more interesting, especially if they are people from various ages and backgrounds. Moreover, the exploration sessions were limited to a few minutes (always less than 7 minutes). We think that one should be able to explore a collection for a longer duration to completely embrace the use of the platform. Finally, we only proposed the exploration of color-based structured image collections. It would be interesting to take into account other image features to *embark* users in more relevant exploration.

Part IV

Conclusion

Chapter 12

Conclusion and future work

12.1 Motivation

This thesis advocates for the need of incremental indexing algorithms and interactive visualisations to support the exploration of large and ever-growing image collections.

We assume that the image descriptors and a distance function are available, to describe and compare images respectively. Then, we wonder about the way to structure the collection, and display it in order to interactively explore the collection.

Two general exploration use cases are considered. First, where one has a peculiar goal and knows (or not) about his image collection (objective-oriented exploration). Second, where one does not have any objective and just aims to wander around his known (or not) image collection (serendipity-oriented exploration).

To address current available image collections characteristics, we endeavoured to propose solutions that meet simultaneously the three following constraints:

1. handling large image collections,
2. handling ever-growing image collections,
3. providing interactive means to explore image collections.

As existing works in the literature do not meet simultaneously the three aforementioned constraints, we proposed a solution that takes advantage and combines the best of the existing paradigms, namely graph-based, clustering-based and projection-based.

This chapter reviews the proposed contributions in Section 12.2 (*what has been done in this thesis?*), discusses the current work and points out its limitations in Section 12.3 (*what can be concluded from this work?*), and outlines possible directions for future research in Section 12.4 (*what can be done in the future?*).

12.2 Contributions

12.2.1 A study on the incremental construction of the RNG

In our first attempt to structure large and ever-growing image collections, we have proposed a study on the incremental construction of the Relative Neighbourhood Graph (Chapter 5).

Following the discussion of an existing work, we have proposed two incremental approaches to build such a graph. First, an approach that is based on a formal proof, and that incrementally yields an exact RNG, is described. Second, we have detailed an approach that incrementally yields a refined approximation of the exact RNG. The proposed algorithms have been stressed out with real-world data sets, including a million images collection.

In this contribution, we have addressed the large and the ever-growing constraints of our endeavour work. However, the interactive visualisation was not easily met. Indeed, early experimentations highlighted some limitations of existing graph visualisation software.

12.2.2 A joint study on the indexing and the exploration of image collections

To address the limitations raised by out first contribution, we have proposed a joint study on the indexing and the visualisation of large and ever-growing image collections. To do so, we have proposed to structure an image collection in a viewable, hierarchical and graph-based hybrid structure. This contribution is two-fold.

First, the incremental algorithm to yield our hybrid structure is detailed (Chapter 6). A hierarchical structure is built and its inner parts are organised using RNG at different levels. Relevant informations are pulled up from the leaves of the tree to allow one to be assisted in the vertical exploration of the hierarchy. Experimentations show that our incremental algorithm can handle images that come in a stream, with a rough estimation of 1.5 million of images per day.

Second, we present our platform for interactive exploration of image collections (Chapter 7). Its architecture, implementation and functionalities are detailed, including the graph formats, the graph layouts, the image graph visualisation, and our hybrid structure exploration. The proposed platform is evaluated in Chapter 11.

12.2.3 Use cases and user evaluation

In this thesis, we have tried to support our claims and the relevance of the proposed solution by presenting three different use cases that use real-world image collection, namely ImageNet eXplorer (Chapter 8), APOD eXplorer (Chapter 9) and DINet Viewer (Chapter 10). The first one supports the scalability property of our work. The second one presents an example of serendipity-oriented exploration. The last one illustrates a scenario of objective-oriented exploration for document image analysis researchers. Moreover, we present the results of a user evaluation, that has been conducted to evaluate our platforms.

12.3 Discussion

In this section, we step back from our work and think about its limitations, and the lessons that can be drawn from it.

First, regarding the study on the incremental construction of the RNG, we can wonder about the relevance of the fully-incremental approach. It appears to us that the best solution to take advantage of the available memory is the mentioned hybrid solution, where the classic algorithm is used to process the beginning of the data set, while our incremental algorithms process the rest.

Second, even if our contribution allows to incrementally structure large and ever-growing image collections in a RNG, we think that it may not be a reliable solution for an interactive exploration of such collections on common computers. However, hardware evolution and graph drawing algorithms may allow to take advantage of this work in the future.

Third, our proposed viewable hierarchical and graph-based hybrid structure is still subjected to the original BIRCH algorithm limitations. The most important one is the parameter T, that has a strong impact on the shape of the hierarchy and the content of its leaves. This parameter is directly linked to the choice of the distance to compare images. To address this issue, existing works in machine learning, such as metric learning, can be used. However, we think that these methods should not be fully automated, and that an interactive solution for the study of image similarity would be appropriate.

Last, the proposed platform is built as a proof-of-concept. However, both the user evaluation and the use cases show promising results with regards to the exploration of image collections. We argue that if one aims at building such platforms as a commercial product, one needs to take into account and incorporate the best practises and the guidelines from several fields, such as Information Visualisation, Design and Human-Computer Interaction. This will allow to build platforms that users will be more disposed to use and take advantage of.

12.4 Future work

This thesis has addressed at best the issue of large and ever-growing image collection interactive exploration. However, many opportunities for improving and extending the presented work remain, both on the indexing and the visualisation parts. We describe some of these directions below.

Platform enhancement

As aforementioned, the platform has been implemented as a proof-of concept. Several improvements can be done on this platform. First, more visualisations and interactions from the literature could be implemented and proposed to the users. For instance, the node overlapping removal for graph drawings or the possibility to see nodes in different layers at the same time. Second, we can think about better strategies to manage the load of the images in the memory cache, *e.g.* when displaying a level, pre-load representative images

of the lower level. This will allow to bring more smoothness during the exploration. Third, new interaction paradigms such as tactile interaction, and even immersed exploration with virtual reality, could bring a richer experience to the users.

User evaluation and use cases

At the time being, only a small user evaluation has been done to assess our platform. Even if positive feedbacks have been given during the evaluation, only a rough idea of the perception of the platform has been obtained. Hence, a full scale user evaluation, involving more people, from different background and computer familiarity would be interesting to conduct. Regarding the objective-oriented exploration, the DINet Viewer use case has given promising results, but it did not lead to real insights and final conclusions up until now. Therefore, we plan to keeping this effort with several other use cases, *e.g.* historical library exploration, biologic research paper library exploration, ontology curation or near duplicates for fraud detection.

Image features study

In this thesis, visual descriptors have been used throughout our experimentations. Hence, the first perspective regarding the descriptors will be to consider image textual descriptors. If they are available, one can use either the annotation or description of the images. Else, one can take benefit from the recent progress in automatic description generation from images. Second, interactive feature selection can also be implemented using our platform. This time, the graph construction could be incremental in the sense of the descriptors. One will (de)select a feature to visually see its impact on generated graph. Third, when considering multiple kinds of features, it would be interesting to build multi networks, where two nodes can have more than one edge (*e.g.* one edge for one type of features). Such a study on the features could also take benefit from coordinated and synchronised views. Last, another interesting direction for future research is an interactive study of distance function used to compare images. Using an interactive interface, the user could tune the parameter T of our hybrid approach, or use metric learning techniques.

Parallelism

Finally, as seen in the experimentations of this thesis, the construction of our structures are tractable. However, the computation times can still be reduced. Without taking into account code optimisation, one interesting direction for future research, will be considering parallelism. Indeed, the proposed algorithms are single-machine algorithms, *i.e.* they are built to be run in one computer. In the big data field, researchers have proposed solutions that leverage parallelism using computer clusters (*i.e.* a group of computers). It would be interesting to take advantage of this paradigm in our work, to reduce the running time of our algorithms and aim at processing not millions, but billions of images, in a few days.

Appendix

Appendix A

Data never sleeps 3.0

In August, 2015, Domo - an american company specialized in business intelligence tools, has released their third infographic regarding digital creation and consumption. The infographic reveals how much data is generated in one minute on the most used online platforms such as Facebook, Snapchat, *etc* ...

Figure A.1 presents the aforementioned infographic. One can find their two previous infographics online:

- How Much Data is Created Every Minute? (June, 2012) [Domo, 2012]
- Data Never Sleeps 2.0 (April, 2014) [Domo, 2014]

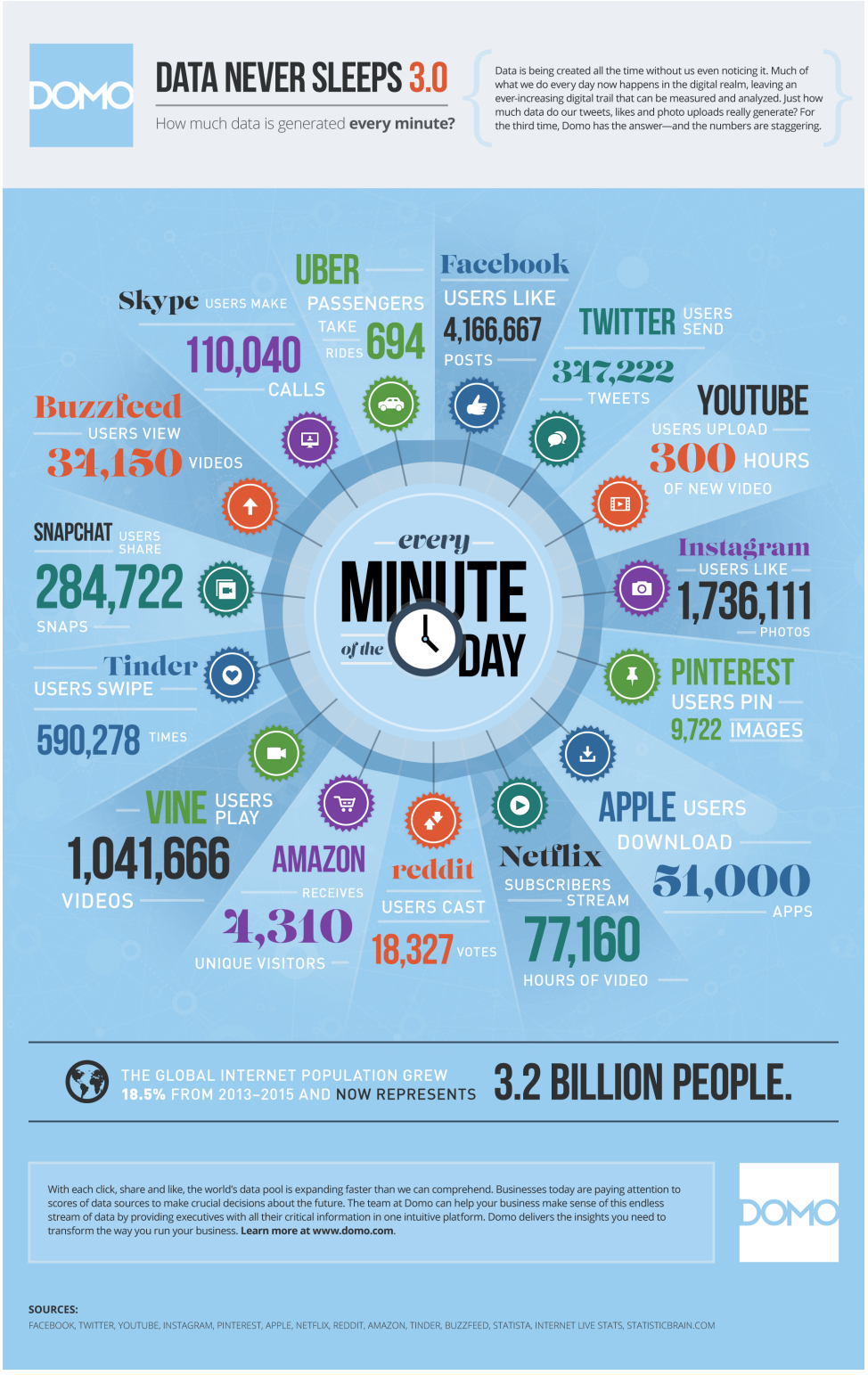


Figure A.1: Digital data creation in one minute. Courtesy of Domo [Domo, 2015].

Appendix B

RNG connectivity proof

B.1 Definition and notations

Let us consider a relative neighbourhood graph $RNG = (V, E)$, with $V \subset \mathbb{R}^d$.
For a given pair $(p, q) \in V \times V, p \neq q$, we denote $\mathcal{L}(p, q)$ the *lune* between p and q.
It is defined by:

$$\mathcal{L}(p, q) = \{r \in \mathbb{R}^d \mid \delta(p, r) < \delta(p, q) \text{ and } \delta(q, r) < \delta(p, q)\}.$$

where $\delta : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a distance function. Figure B.1 illustrates a lune in $\mathbb{R}^2$.

By definition, p and q are relative neighbours, if and only if, the lune $\mathcal{L}(p, q)$ does not contain any other vertex of V: $\mathcal{L}(p, q) \cap V = \emptyset$.

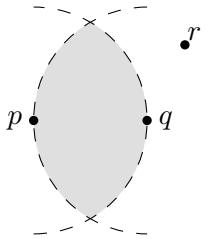


Figure B.1: Lune $\mathcal{L}(p, q)$ (grey area) of two points $p, q \in \mathbb{R}^2$. If no other point lays in this area, then p and q are relative neighbours.

B.2 RNG connectivity

Theorem B.2.1. The Relative Neighbourhood Graph is a *connected* graph.

Proof. Let us consider the list of pairwise vertices (p, q) sorted by ascending distances $\delta(p, q)$. We aim at proving that for each element (p, q) of this list, there exists a path

between p and q. Let us prove this by induction.

Base case: Let (p_0, q_0) the first element of the list. It is the pair with the lowest distance $\delta(p_0, q_0)$. There cannot exist a node r in $\mathcal{L}(p_0, q_0)$, otherwise, we should have:

$$\begin{cases} \delta(p_0, r) < \delta(p_0, q_0) \\ \delta(q_0, r) < \delta(p_0, q_0) \end{cases} \tag{B.1}$$

which is inconsistent with the minimality of $\delta(p_0, q_0)$.
Hence, there exists an edge between p_0 and q_0, and therefore a path between p_0 and q_0.

Induction hypothesis: Assume that, for the fist k elements $(p_i, q_i)_{i \in [\![0\ ;\ k-1]\!]}$ of the list, there exists a path between p_i and q_i.

Induction step: Let (p, q) be the $k+1$ element of the list. We will now show that there exists a path between p and q. There are two cases:

- $\nexists r \in V \mid r \in \mathcal{L}(p, q)$ *i.e.* it does no exists a node r in the lune of p and q. Then, by definition, there exists an edge between p and q, and therefore a path between p and q.
- $\exists r \in V \mid r \in \mathcal{L}(p, q)$ *i.e.* there exists a node r in the lune of p and q. Then, we have:

$$\begin{cases} \delta(p, r) < \delta(p, q) \\ \delta(q, r) < \delta(p, q) \end{cases} \tag{B.2}$$

  From the induction hypothesis, there exists a path between p and r, and between q and r. It follows that there exists a path between p and q.

□

Appendix C

MPEG-7 visual descriptors

In the online available image collections, temporal, geographic and concepts-based descriptors are not always available. To address the most general case, we use in the presented work content-based descriptors, *i.e.* information that is extracted from the images themselves. As the choice of descriptors is not the main issue of this thesis, we decide to fall into step with the MPEG-7 [Pereira and Koenen, 2001] multimedia content description standard.

Following the previous MPEG version, where the emphasis was put on encoding, MPEG-7 aims at describing multimedia content, such as image, audio and video. MPEG-7 specifies, among others, visual descriptors that have been introduced and stressed out in the research community. The objective is to provide a set of standardised descriptors to characterise multimedia content in order to guaranty interoperability and achieve content based multimedia retrieval efficiently.

These content-based visual descriptors [Sikora, 2001] are organised in four categories:

- colour descriptors [Manjunath et al., 2001]
- texture descriptors [Manjunath et al., 2001]
- shape descriptors [Bober, 2001]
- motion descriptors (for videos) [Jeannin and Divakaran, 2001]

One can refer to the references attached to each category to find a thorough description of each categories descriptors.

In this thesis work, two descriptors have been used, namely Color Layout Descriptor (CLD) and Edge Histogram Descriptor (EHD), colour and texture descriptor respectively. Note that the original definition are presented below. One can find some studies that have improve the basic definition of CLD and EHD (*e.g.* in [Won et al., 2002]).

C.1 Colour Layout Descriptor

The CLD is designed to capture the spatial distribution of dominant colours in an image. Colours are expressed in the YCbCr colour space. CLD main advantages are first that it is a very compact descriptor, therefore it fits perfectly for fast browsing and search applications. Second, it is resolution invariant, thanks to its definition.

The feature extraction process consists in four stages:

1. *Image partitioning.* The input picture is divided into $8 \times 8 = 64$ blocks. This guarantees the resolution or scale invariance.

2. *Representative colour detection.* One representative colour is computed for each block. To select the representative colour, any method can be applied, but the average colour of the block is usually used. An 8×8 icon is obtained (see Figure C.1).

3. *DCT transformation.* Each colour channel of the icon is transformed into a series of coefficients by performing a 8×8 Discrete Cosine Transform (DCT). Three series of 64 DCT coefficients are computed.

4. *Quantisation.* A few low-frequency coefficients are selected using zigzag scanning (see Figure C.2) and quantised to form the descriptor.

Hence, we obtained a descriptor with a size of $3 \times 64 = 192$. One can find more details on the the CLD computation in [Manjunath et al., 2001] or [Pereira and Koenen, 2001].

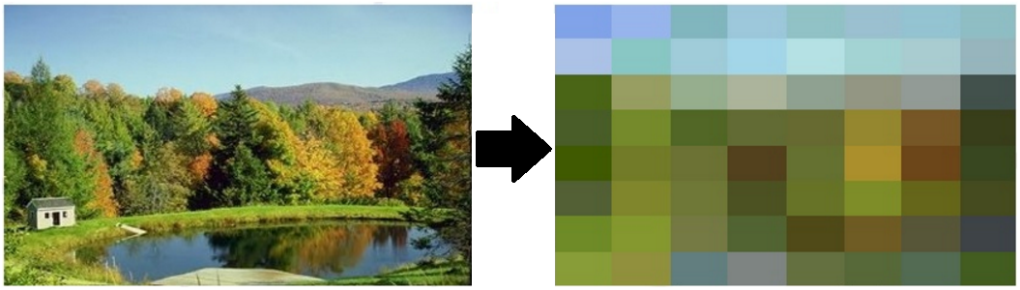


Figure C.1: Color Layout Descriptor. Representative colour selection on a 8×8 partitioned image.

C.2 Edge Histogram Descriptor

EHD [MPEG, 2000] captures the spatial distribution of five types of edges in an image. In that sense, CLD and EHD can be viewed as alike as two peas in a pod. Figure C.3a shows the five types of edges considered.

The EHD feature extraction is quite straightforward: first, an image is divided using a 4×4 blocks. For each block, a histogram of the considered edges distribution is built (see

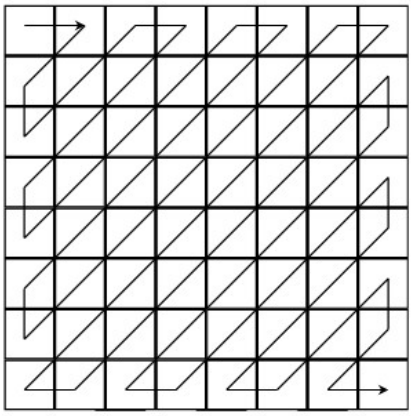
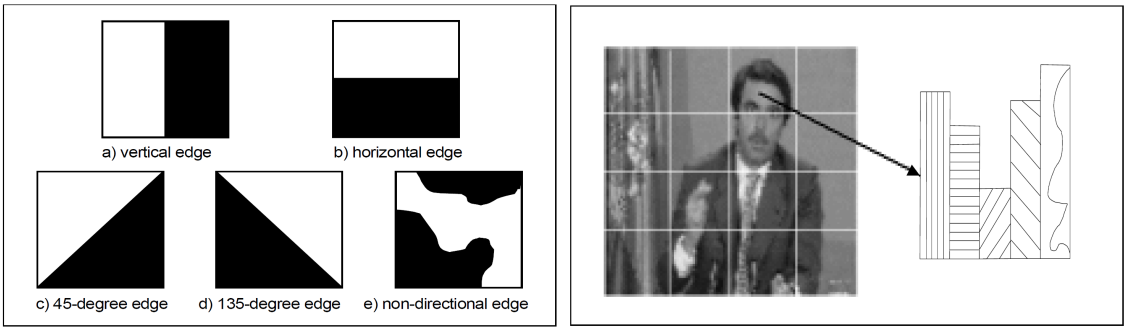


Figure C.2: Color Layout Descriptor. Grid zig-zag scanning.

Figure C.3b). Therefore, we have $4 \times 4 \times 5 = 80$ values that constitute the texture descriptor.

One can find more details on the the EHD computation in [Manjunath et al., 2001].



(a) (b)

Figure C.3: Edge Histogram Descriptor. (a) The 5 considered types of contour and (b) the histogram computation for each image cell.

Appendix D

Visualisation experimentations

D.1 Overview

This appendix summarises early experimentations that have been conducted after our work on the incremental construction of an approximate RNG (described in Chapter 5). Following the structuring of large and ever-growing image collections in RNGs, we have performed some experimentations regarding the visualisation of the yielded graphs using free off-the-shelf graph visualisation software, namely Tulip [Auber et al., 2012] and Gephi [Bastian et al., 2009].

The workflow of these experimentations is given in Figure D.1. It consists in:

1. *Data crawling*: images and metadata are crawled from the web portal of the selected institute.

2. *Description*: visual descriptors are extracted from the images. The MPEG-7's EHD [MPEG, 2000] texture descriptor, described in the previous appendix, is used.

3. *Indexing*: the incremental approximate RNG construction, proposed in this thesis, is used.

4. *Visualisation*: the computed graph is visualised with Tulip and Gephi.

A custom image collection that has been crawled from the *National Gallery of Art* (NGA) of Washington [1] has been used in this early experimentation. We have built a collection of 43,721 *Open Access* images, that contains paintings, sculptures' pictures and drawings. Figure D.2 displays a few images of this collection. Then, we have built a description file for the whole collection. It corresponds to a $43,721 \times 81$ matrix (80 values for the texture descriptor plus an unique id for each image). Our approximative algorithm has computed a graph with 117,925 edges (90 minutes on an Intel Xeon CPU W3520 at 2.67Ghz).

[1] `https://images.nga.gov`

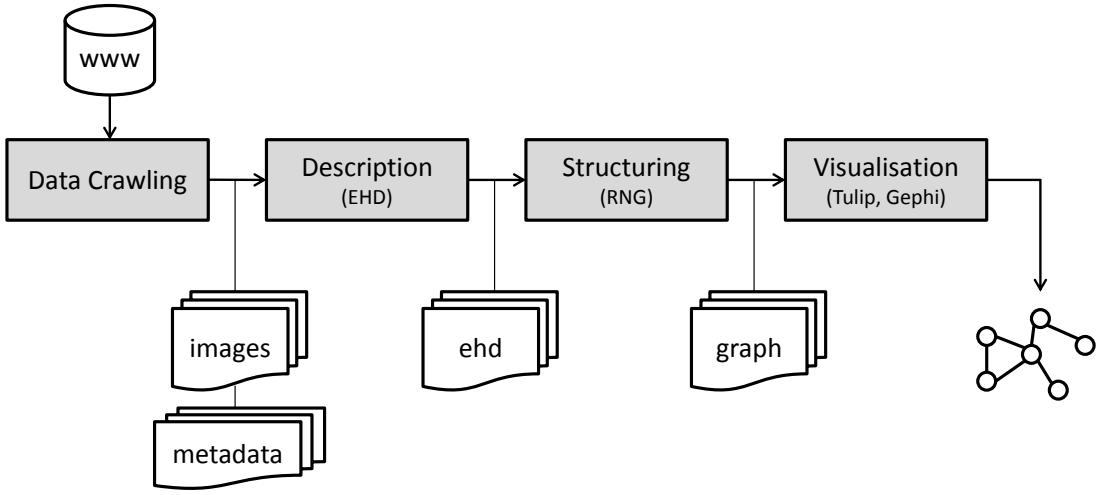


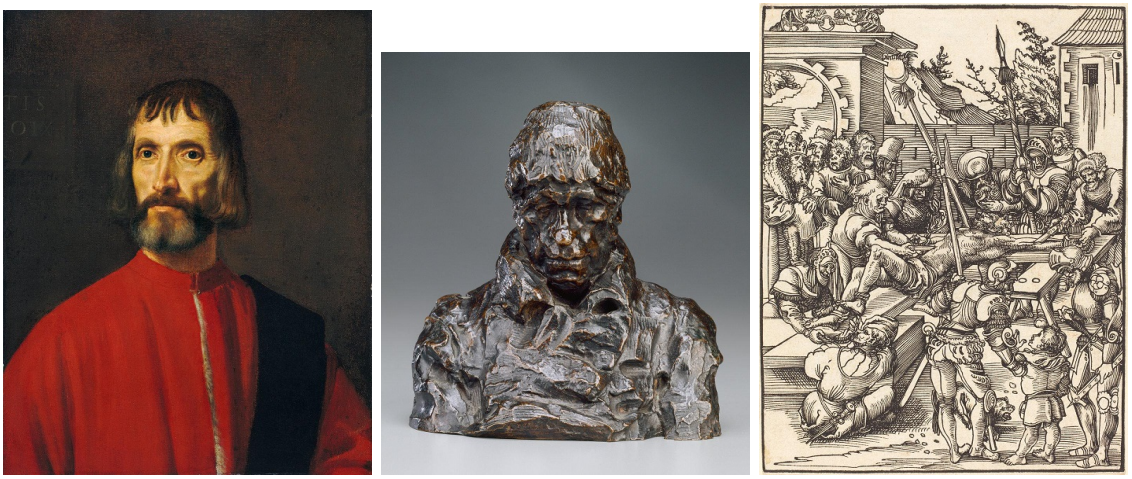
Figure D.1: Workflow of the experimentation.

D.2 Discussion

Several graph drawing algorithms have been used to visualise the graph. We present the results associated with the three following layout algorithms, known for highlighting possible connected components in the drawings: OpenOrd [Martin et al., 2011], Yifan Hu multilevel [Hu, 2005] and FM^3 [Hachul and Jünger, 2005]. Figures D.3a, D.3b and D.4a show the drawing obtained with the three aforementioned layout algorithms.

Based on these results, one can make some observations:

- First, one can see in Figure D.3a (OpenOrd algorithm in Gephi) a dense and homogeneous group which we refer to as the "kernel". Around this kernel, a number of nodes gravitate in a chaotic way.
- Second, one can find the kernel again in Figure D.3b (Yifan Hu multilevel algorithm in Gephi). However, the gravitating nodes seem to be more organised around the kernel this time.
- Third, the kernel observation is found again in Figure D.4a, (FM^3 algorithm in Tulip). However, the kernel seems to contain only a few number of nodes, and in the suburb of the kernel, the nodes are structured in filaments. The first observation is not completely consistent with the two above visualisations.
- Last, using Tulip interactive functionalities, we have realised that the FM^3 visualisation may be deceptive. Indeed, the kernel is composed of 91% of the nodes. Figure D.4b shows a zoom on the kernel drawn by FM^3 in Tulip.



(a) (b) (c)

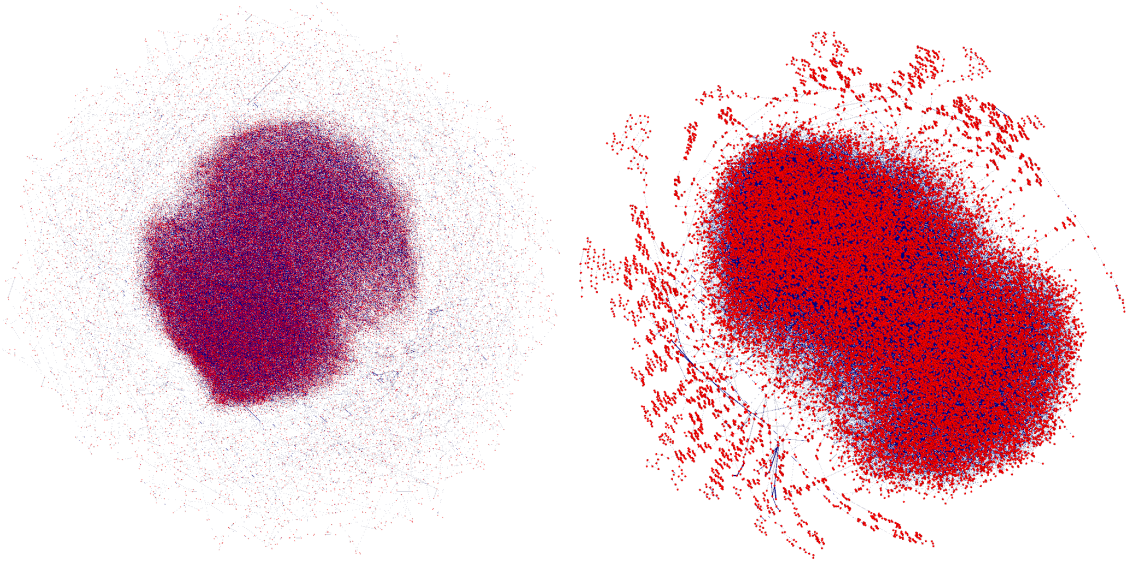
Figure D.2: NGA image collection. (a) A painting, (b) a sculpture picture and (c) a drawing.

Such drawings and observations allow one to follow up two leads:

1. as a majority of the images (91%) are similar in term of texture, one can wonder about the choice of the images' descriptor;
2. one can also focus his attention on the small groups that gravitate around the kernel, and that may be source of information or insights on the image collection.

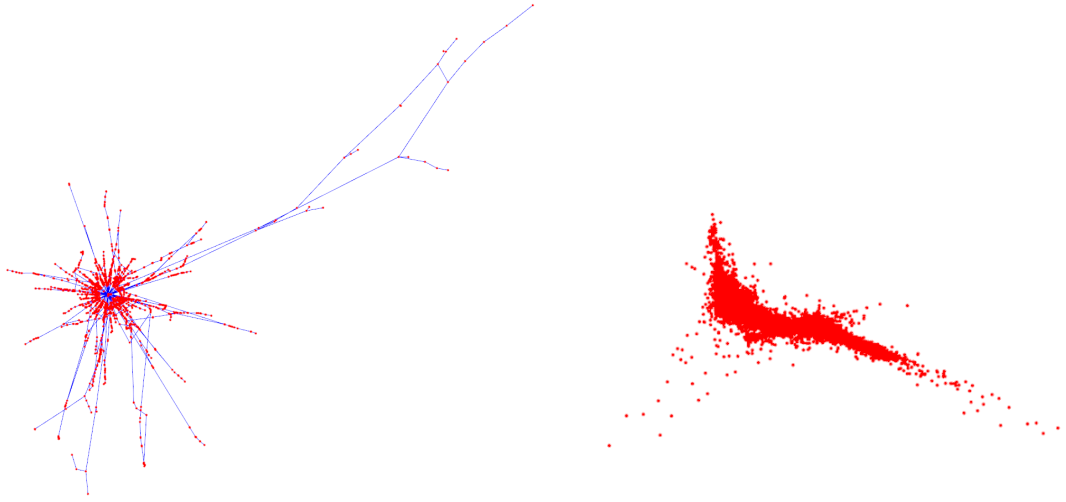
Nonetheless, despite these potential research opportunities for Digital Humanities experts, we have drawn some lessons with regard to our objectives:

- drawing medium-size graphs is difficult for "*common*" computers;
- displaying the nodes' images for medium-scale graphs is difficult for "*common*" computers;
- interactions on medium-scale graphs is difficult for "*common*" computers;
- displaying several thousands of nodes/images may not be relevant for an user;
- the choice of the graph drawing algorithm must not be done without consideration;
- the hairball phenomenon [Kosara, 2012] may be an issue for some graphs;
- one must have at his disposal relevant interactions in order to overcome possible misleading visualisations.



(a) (b)

Figure D.3: Visualisation in Gephi. Drawings of the computed RNG with Gephi and using: (a) the OpenOrd algorithm and (b) the Yifan Hu multilevel algorithm.



(a) (b)

Figure D.4: Visualisation in Tulip. Drawings of the computed RNG with Tulip and using the FM^3 algorithm. (a) displays the whole drawing and (b) displays a zoom on the "*kernel*".

Appendix E

Open research

The works realised during this thesis have generated several resources that may be useful for the community, either for usage, reproducibility, or possible extensions. We list below the resources that will be openly accessible, either in

1. my homepage: `http://frederic.rayar.free.fr`,
2. my GitHub account : `https://github.com/frayar`,
3. the RFAI team website: `http://www.rfai.li.univ-tours.fr/`.

Data sets

- Wang's CLD descriptors file
- NASA APoD's CLD descriptors file
- National Gallery of Art's CLD and EHD descriptors files
- MIRFLICKR-25000's CLD descriptors file
- ImageNet's CLD and EHD descriptors archives
- Wang image collection
- Bentham word images collection

RNG and HRNG

- Chapter 5 RNG and approximate RNG
- Chapter 6 HRNG
- Chapter 10 use case features/distances RNG

Code

- RNG code (Python)
- RNG/iRNG code (C++)
- HRNG code (C++)
- Stress majorisation Linkurious plugin (Javascript)
- Maxent stress Linkurious plugin (Javascript)
- Treemap code (Javascript)
- Breadcrumb Code (Javascript)
- ICE platform (HTML/CSS/JavaScript)
- INex platform (HTML/CSS/JavaScript)
- APOD eXplorer platform (HTML/CSS/JavaScript)
- DINet Viewer platform (HTML/CSS/JavaScript)

Scripts

- NGA Crawler (Python)
- NASA APOD Crawler (Python)
- Graph format conversion scripts (Python)
- Graphs comparison script (Python)

Appendix F

List of publications

1. Frédéric Rayar, Sabine Barrat, Fatma Bouali, Gilles Venturini: *A Viewable Indexing Structure for the Interactive Exploration of Dynamic and Large Image Collections.* ACM TKDD Special Issue on Interactive Data Exploration and Analytics (IDEA). **(under review)**

2. Frédéric Rayar, Sabine Barrat, Fatma Bouali, Gilles Venturini: *APoD eXplorer: Recommendation System and Interactive Exploration of a Dynamic Image Collection.* International Conference on Information Visualisation (IV) 2016: 118-123

3. Frédéric Rayar, Sabine Barrat, Fatma Bouali, Gilles Venturini: *Incremental hierarchical indexing and visualisation of large image collections.* European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN) 2016: 659-664

4. Frédéric Rayar, Tanmoy Mondal, Sabine Barrat, Fatma Bouali, Gilles Venturini: *Visual Analysis System for Features and Distances Qualitative Assessment: Application to Word Image Matching.* International Workshop on Document Analysis Systems (DAS) 2016: 381-386

5. Frédéric Rayar, Sabine Barrat, Fatma Bouali, Gilles Venturini: *Construction incrémentale d'une structure hiérarchique pour l'exploration visuelle et interactive de larges collections d'images.* Conférence internationale sur l'extraction et la gestion des connaissances (EGC) 2016: 327-332

6. Frédéric Rayar, Sabine Barrat, Fatma Bouali, Gilles Venturini: *An Approximate Proximity Graph Incremental Construction for Large Image Collections Indexing.* International Symposium on Methodologies for Intelligent Systems (ISMIS) 2015: 59-68

7. Frédéric Rayar, Sabine Barrat, Fatma Bouali, Gilles Venturini: *Exploration visuelle et interactive d'une large collection d'images en libre accès.* Conférence internationale sur l'extraction et la gestion des connaissances - Atelier Visualisation d'informations, Interaction, et Fouille de données (EGC) 2015: 327-332

Bibliography

[Abello et al., 2006] Abello, J., van Ham, F., and Krishnan, N. (2006). Ask-graphview: A large scale graph visualization system. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):669–676.

[Ackoff, 1989] Ackoff, R. L. (1989). From data to wisdom. *Journal of Applies Systems Analysis*, 16:3–9.

[Akao et al., 2014] Akao, Y., Yamamoto, A., and Higashikawa, Y. (2014). Feasibility study of visualizing diversity of japanese hiragana handwritings by mds of earth mover's distance toward assisting forensic experts in writer verification. In *11th International Workshop on Document Analysis Systems, DAS 2014, Tours, France, April 7-10, 2014*, pages 26–30.

[Alahi et al., 2012] Alahi, A., Ortiz, R., and Vandergheynst, P. (2012). Freak: Fast retina keypoint. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 510–517.

[Albrecht, 2009] Albrecht, T. (2009). Dynamic Time Warping (DTW). pages 69–85.

[Archambault et al., 2008] Archambault, D., Munzner, T., and Auber, D. (2008). Grouseflocks: Steerable exploration of graph hierarchy space. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):900–913.

[Auber et al., 2012] Auber, D., Archambault, D., Bourqui, R., Lambert, A., Mathiaut, M., Mary, P., Delest, M., Dubois, J., and Melançon, G. (2012). The Tulip 3 Framework: A Scalable Software Library for Information Visualization Applications Based on Relational Data. Technical report.

[Bache and Lichman, 2013] Bache, K. and Lichman, M. (2013). UCI machine learning repository. `http://archive.ics.uci.edu/ml`.

[Bastian et al., 2009] Bastian, M., Heymann, S., and Jacomy, M. (2009). Gephi: An open source software for exploring and manipulating networks. International AAAI Conference on Weblogs and Social Media.

[Battista et al., 1998] Battista, G. D., Eades, P., Tamassia, R., and Tollis, I. G. (1998). *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall PTR.

[Bay et al., 2008] Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346 – 359.

[Bederson, 2001] Bederson, B. B. (2001). Photomesa: A zoomable image browser using quantum treemaps and bubblemaps. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, pages 71–80.

[Bernardi et al., 2016] Bernardi, R., Cakici, R., Elliott, D., Erdem, A., Erdem, E., Ikizler-Cinbis, N., Keller, F., Muscat, A., and Plank, B. (2016). Automatic description generation from images: A survey of models, datasets, and evaluation measures. *CoRR*, abs/1601.03896.

[Bertin, 1967] Bertin, J. (1967). *Sémiologie graphique*. Paris:Gauthiers Villars.

[Bober, 2001] Bober, M. (2001). Mpeg-7 visual shape descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):716–719.

[Calonder et al., 2010] Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief: Binary robust independent elementary features. In Daniilidis, K., Maragos, P., and Paragios, N., editors, *11th European Conference on Computer Vision (ECCV)*, pages 778–792.

[Camargo and González, 2009] Camargo, J. and González, F. (2009). Visualization, summarization and exploration of large collections of images: State of the art. *Latin-American Conference On Networked and Electronic Media.*

[Card et al., 1999] Card, S. K., Mackinlay, J. D., and Shneiderman, B. (1999). *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Carpendale, 2008] Carpendale, S. (2008). *Evaluating Information Visualizations*, pages 19–45. Springer Berlin Heidelberg.

[Chau et al., 2013] Chau, P., Vreeken, J., van Leeuwen, M., and Faloutsos, C. (2013). Kdd 2013 workshop on interactive data exploration and analytics. http://poloclub.gatech.edu/idea2013/.

[Chen et al., 2000] Chen, C., Gagaudakis, G., and Rosin, P. (2000). Similarity-based image browsing.

[Chen et al., 1998] Chen, J. Y., Bouman, C. A., and Dalton, J. C. (1998). Similarity pyramids for browsing and organization of large image databases. pages 26–29.

[Cox and Cox, 2000] Cox, T. F. and Cox, M. (2000). *Multidimensional Scaling, Second Edition.* Chapman and Hall/CRC.

[Crockford, 2000] Crockford, D. (2000). Javascript object notation. http://www.json.org/.

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE.

[Delaunay, 1934] Delaunay, B. N. (1934). Sur la sphère vide. *Bulletin of Academy of Sciences of the USSR*, 7:793–800.

[Deng, 2007] Deng, D. (2007). Content-based image collection summarization and comparison using self-organizing maps. *Pattern Recognition*, 40(2):718 – 727.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

[Ding et al., 2015] Ding, H., Wang, C., Huang, K., and Machiraju, R. (2015). Graphie: graph based histology image explorer.

[DM2E, 2015] DM2E (2015). Openglam. http://openglam.org/.

[Domo, 2012] Domo (2012). How much data is created every minute? https://www.domo.com/blog/2012/06/how-much-data-is-created-every-minute/.

[Domo, 2014] Domo (2014). Data never sleeps 2.0. https://www.domo.com/blog/2014/04/data-never-sleeps-2-0/.

[Domo, 2015] Domo (2015). Data never sleeps 3.0. https://www.domo.com/learn/data-never-sleeps-3-0.

[Dwyer et al., 2006] Dwyer, T., Marriott, K., and Stuckey, P. J. (2006). *Fast Node Overlap Removal*, pages 153–164.

[Eades, 1984] Eades, P. (1984). A heuristics for graph drawing. *Congressus numerantium*, 42:146–160.

[ECMAScript, 1995] ECMAScript (1995). Javascript. http://www.ecmascript.org/.

[Eler et al., 2009] Eler, D., Nakazaki, M., Paulovich, F., Santos, D., Andery, G., Oliveira, M., Batista, J., and Minghim, R. (2009). Visual analysis of image collections. *The Visual Computer*, 25(10):923–937.

[Felzenszwalb et al., 2010] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object Detection with Discriminative Trained Part Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645.

[Few, 2004] Few, S. (2004). Perceptual edge. https://www.perceptualedge.com/examples.php.

[Friendly, 2008] Friendly, M. (2008). *A Brief History of Data Visualization*, pages 15–56. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Fruchterman and Reingold, 1991] Fruchterman, T. M. J. and Reingold, E. M. (1991). Graph drawing by force-directed placement. *Softw. Pract. Exper.*, 21(11):1129–1164.

[Gabriel and Sokal, 1969] Gabriel, R. K. and Sokal, R. R. (1969). A New Statistical Approach to Geographic Variation Analysis. *Systematic Zoology*, 18(3):259–278.

[Gansner et al., 2011] Gansner, E., Hu, Y., North, S., and Scheidegger, C. (2011). Multilevel agglomerative edge bundling for visualizing large graphs. In *Pacific Visualization Symposium (PacificVis), 2011 IEEE*, pages 187–194.

[Gansner and Hu, 2009] Gansner, E. R. and Hu, Y. (2009). *Efficient Node Overlap Removal Using a Proximity Stress Model*, pages 206–217.

[Gansner et al., 2013] Gansner, E. R., Hu, Y., and North, S. (2013). A maxent-stress model for graph layout. *IEEE Transactions on Visualization and Computer Graphics*, 19(6):927–940.

[Gansner et al., 2005] Gansner, E. R., Koren, Y., and North, S. (2005). *Graph Drawing by Stress Majorization*, pages 239–250. Springer Berlin Heidelberg.

[Gatos et al., 2014] Gatos, B., Louloudis, G., Causer, T., Grint, K., Romero, V., Sánchez, J. A., Toselli, A. H., and Vidal, E. (2014). Ground-Truth production in the tranScriptorium project. In *11th IAPR International Workshop on Document Analysis Systems (DAS)*.

[GD, 1992] GD (1992). International symposium on graph drawing and network visualisation. http://graphdrawing.org/.

[Gionis et al., 1999] Gionis, A., Indyk, P., and Motwani, R. (1999). Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 518–529.

[Gomi et al., 2008] Gomi, A., Miyazaki, R., Itoh, T., and Li, J. (2008). CAT: A hierarchical image browser using a rectangle packing technique. In *12th International Conference on Information Visualisation, IV 2008, 8-11 July 2008, London, UK*, pages 82–87.

[Google, 2009] Google (2009). Angularjs. https://angularjs.org/.

[Goto et al., 2013] Goto, M., Ishida, R., Feng, Y., and Uchida, S. (2013). Analyzing the distribution of a large-scale character pattern set using relative neighborhood graph. In *12th International Conference on Document Analysis and Recognition, Washington, DC, USA, August 25-28, 2013*, pages 3–7.

[Goto et al., 2015] Goto, M., Ishida, R., and Uchida, S. (2015). Preselection of support vector candidates by relative neighborhood graph for large-scale character recognition. In *13th International Conference on Document Analysis and Recognition, Nancy, France, August 23-26, 2015*.

[Hachul and Jünger, 2005] Hachul, S. and Jünger, M. (2005). *Drawing Large Graphs with a Potential-Field-Based Multilevel Algorithm*, pages 285–295.

[Hachul and Jünger, 2005] Hachul, S. and Jünger, M. (2005). Drawing large graphs with a potential-field-based multilevel algorithm. In Pach, J., editor, *Graph Drawing*, volume 3383 of *Lecture Notes in Computer Science*, pages 285–295. Springer Berlin Heidelberg.

[Hacid and Yoshida, 2007] Hacid, H. and Yoshida, T. (2007). Incremental neighborhood graphs construction for multidimensional databases indexing. In *Canadian Conference on AI*, pages 405–416.

[Hacid and Zighed, 2005] Hacid, H. and Zighed, D. A. (2005). An effective method for locally neighborhood graphs updating. In *DEXA*, pages 930–939.

[Healey and Enns, 2012] Healey, C. and Enns, J. (2012). Attention and visual memory in visualization and computer graphics. *IEEE Transactions on Visualization and Computer Graphics*, 18(7):1170–1188.

[Heesch and Rüger, 2004] Heesch, D. and Rüger, S. (2004). Nn^k networks for content-based image retrieval. In *Advances in Information Retrieval*, volume 2997, pages 253–266.

[Heymann, 2007] Heymann, S. (2007). Graph exchange xml format. https://gephi.org/gexf/format/.

[Heymann, 2013] Heymann, S. (2013). Linkurious. http://linkurio.us/.

[Holten, 2006] Holten, D. (2006). Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):741–748.

[Hu, 2005] Hu, Y. F. (2005). Efficient and high quality force-directed graph drawing. *The Mathematica Journal*, 10:37–71.

[Huiskes and Lew, 2008] Huiskes, M. J. and Lew, M. S. (2008). The mir flickr retrieval evaluation. In *MIR '08: Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval*, New York, NY, USA. ACM.

[Jacomy, 2012] Jacomy, A. (2012). Sigmajs. http://sigmajs.org/.

[Jacomy et al., 2014] Jacomy, M., Venturini, T., Heymann, S., and Bastian, M. (2014). Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. *PLoS ONE*, 9:1–12.

[Jaffe et al., 2006] Jaffe, A., Naaman, M., Tassa, T., and Davis, M. (2006). Generating summaries and visualization for large collections of geo-referenced photographs. In *Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, MIR '06, pages 89–98. ACM.

[Jain and Dubes, 1988] Jain, A. K. and Dubes, R. C. (1988). *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[Jannach et al., 2010] Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. (2010). *Recommender Systems: An Introduction*. Cambridge University Press, New York, NY, USA, 1st edition.

[Jaromczyk and Toussaint, 1992] Jaromczyk, J. W. and Toussaint, G. T. (1992). Relative neighborhood graphs and their relatives. *Proceedings of the IEEE*, 80(9):1502–1517.

[Jeannin and Divakaran, 2001] Jeannin, S. and Divakaran, A. (2001). Mpeg-7 visual motion descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):720–724.

[Jolliffe, 1986] Jolliffe, I. (1986). *Principal Component Analysis*. Springer Verlag.

[Kamada and Kawai, 1989] Kamada, T. and Kawai, S. (1989). An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7 – 15.

[Karpathy, 2015] Karpathy, A. (2015). t-sne visualization of cnn codes. http://cs.stanford.edu/people/karpathy/cnnembed/.

[Keim et al., 2008] Keim, D., Andrienko, G., Fekete, J.-D., Görg, C., Kohlhammer, J., and Melançon, G. (2008). *Visual Analytics: Definition, Process, and Challenges*, pages 154–175. Springer Berlin Heidelberg.

[Keller and Tergan, 2005] Keller, T. and Tergan, S.-O. (2005). *Visualizing Knowledge and Information: An Introduction*, pages 1–23. Springer Berlin Heidelberg.

[Kobourov, 2012] Kobourov, S. G. (2012). Spring embedders and force directed graph drawing algorithms.

[Kobourov, 2013] Kobourov, S. G. (2013). Force-directed drawing algorithms. In Tamassia, R., editor, *Handbook of Graph Drawing and Visualization*, pages 383–408. CRC Press.

[Koffka, 1935] Koffka, K. (1935). *Principles of Gestalt psychology*. Harcourt, New York.

[Kohonen et al., 2001] Kohonen, T., Schroeder, M. R., and Huang, T. S. (2001). *Self-Organizing Maps*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd edition.

[Kosara, 2012] Kosara, R. (2012). Graphs beyond the hairball. https://eagereyes.org/techniques/graphs-hairball.

[Krishnamachari and Abdel-Mottaleb, 1999] Krishnamachari, S. and Abdel-Mottaleb, M. (1999). Image browsing using hierarchical clustering. In *Computers and Communications, 1999. Proceedings. IEEE International Symposium on*, pages 301–307.

[Lab, 2015] Lab, N. U. K. (2015). Timeline js. https://timeline.knightlab.com/.

[Lankford, 1969] Lankford, P. M. (1969). Regionalization: Theory and alternative algorithms. *Geographical Analysis*, 1(2):196–212.

[Larkin and Simon, 1987] Larkin, J. H. and Simon, H. A. (1987). Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11(1):65–100.

[Leutenegger et al., 2011] Leutenegger, S., Chli, M., and Siegwart, R. Y. (2011). Brisk: Binary robust invariant scalable keypoints. In *2011 International Conference on Computer Vision*, pages 2548–2555.

[Li and Wang, 2003] Li, J. and Wang, J. Z. (2003). Automatic linguistic indexing of pictures by a statistical modeling approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(9):1075–1088.

[Listgarten et al., 2005] Listgarten, J., Neal, R. M., Roweis, S. T., and Emili, A. (2005). Multiple Alignment of Continuous Time Series. *Advances in Neural Information Processing Systems*, 17(17):817–824.

[Liu et al., 2004] Liu, H., Xie, X., Tang, X., Li, Z., and Ma, W.-Y. (2004). Effective browsing of web image search results. Technical Report MSR-TR-2004-117, Microsoft Research.

[Lowe, 1999] Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150–1157 vol.2.

[Manjunath et al., 2001] Manjunath, B. S., Ohm, J. R., Vasudevan, V. V., and Yamada, A. (2001). Color and texture descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):703–715.

[Mark J. Huiskes and Lew, 2010] Mark J. Huiskes, B. T. and Lew, M. S. (2010). New trends and ideas in visual concept detection: The mir flickr retrieval evaluation initiative. In *MIR '10: Proceedings of the 2010 ACM International Conference on Multimedia Information Retrieval*, pages 527–536, New York, NY, USA. ACM.

[Marti and Bunke, 2001] Marti, U. and Bunke, H. (2001). Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. *International Journal of Pattern Recognition*, 15:65–90.

[Martin et al., 2011] Martin, S., Brown, W. M., Klavans, R., and Boyack, K. W. (2011). Openord: an open-source toolbox for large graph layout. *Proc. SPIE*, 7868:01–11.

[Matković et al., 2009] Matković, K., Gračanin, D., Freiler, W., Banova, J., and Hauser, H. (2009). Large image collections - comprehension and familiarization by interactive visual analysis. In *Smart Graphics*, volume 5531 of *Lecture Notes in Computer Science*, pages 15–26. Springer Berlin Heidelberg.

[McCormick et al., 1987] McCormick, B. H., DeFanti, T. A., and Brown, M. (1987). *Visualizing Knowledge and Information: An Introduction*, volume 21. ACM SIGGRAPH.

[Mikolajczyk and Schmid, 2005] Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630.

[Miller and Fellbaum, 1985] Miller, G. A. and Fellbaum, C. (1985). Wordnet. https://wordnet.princeton.edu/.

[Mondal et al., 2014] Mondal, T., Ragot, N., Ramel, J.-y., and Pal, U. (2014). Flexible Sequence Matching Technique: Application to Word Spotting in Degraded Documents. In *ICFHR*, pages 210–215. IEEE.

[Moscovich et al., 2009] Moscovich, T., Chevalier, F., Henry, N., Pietriga, E., and Fekete, J.-D. (2009). Topology-aware navigation in large networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2319–2328.

[MPEG, 2000] MPEG (2000). Iso/iec/jtc1/sc29/wg11 - core experiment results for edge histogram descriptor (ct4)), mpeg document m6174. Technical report.

[Munzner, 2014] Munzner, T. (2014). *Visualization Analysis and Design*. A.K. Peters visualization series.

[Nakamoto et al., 2013] Nakamoto, C., Huang, R., Koizumi, S., Ishida, R., Feng, Y., and Uchida, S. (2013). Font distribution observation by network-based analysis. In *Camera-Based Document Analysis and Recognition - 5th International Workshop, CBDAR 2013, Washington, DC, USA, August 23, 2013*, pages 83–97.

[Nemiroff and Bonnell, 1995] Nemiroff, R. and Bonnell, J. (1995). Astronomy picture of the day. http://apod.nasa.gov/.

[Noack, 2007] Noack, A. (2007). Energy models for graph clustering. *Journal of Graph Algorithms and Applications*, 11(2):453–480.

[Nollenburg, 2007] Nollenburg, M. (2007). Geographic visualization. In *Human-Centered Visualization Environments*, volume 4417 of *Lecture Notes in Computer Science*, pages 257–294. Springer Berlin Heidelberg.

[Oka, 1998] Oka, R. (1998). Spotting method for classification of real world data. *The Computer Journal*, 41(8):1–6.

[OpenMP, 1998] OpenMP (1998). Open multi-processing. http://www.openmp.org/.

[Pereira and Koenen, 2001] Pereira, F. and Koenen, R. (2001). MPEG-7: A standard for multimedia content description. *Int. J. Image Graphics*, 1(3):527–546.

[Plant and Schaefer, 2011] Plant, W. and Schaefer, G. (2011). Visualisation and browsing of image databases. In *Multimedia Analysis, Processing and Communications*, volume 346 of *Studies in Computational Intelligence*, pages 3–57.

[Porta, 2006] Porta, M. (2006). Browsing large collections of images through unconventional visualization techniques. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '06, pages 440–444.

[Rodden et al., 2001] Rodden, K., Basalaj, W., Sinclair, D., and Wood, K. (2001). Does organisation by similarity assist image browsing? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, pages 190–197. ACM.

[Roweis and Saul, 2000] Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *SCIENCE*, 290:2323–2326.

[Rowley, 2007] Rowley, J. (2007). The wisdom hierarchy: Representations of the dikw hierarchy. *J. Inf. Sci.*, 33(2):163–180.

[Rublee et al., 2011] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571.

[Rubner et al., 1998] Rubner, Y., Tomasi, C., and Guibas, L. (1998). A metric for distributions with applications to image databases. In *Computer Vision, 1998. Sixth International Conference on*, pages 59–66.

[Scuturici et al., 2004] Scuturici, M., Clech, J., Scuturici, V.-M., and Zighed, D. A. (2004). Modèle topologique pour l'interrogation des bases d'images. In *EGC*, pages 409–414.

[Scuturici et al., 2003] Scuturici, M., Clech, J., and Zighed, D. A. (2003). Topological query in image databases. In *CIARP*, pages 145–152.

[Shneiderman, 1992] Shneiderman, B. (1992). Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.*, 11(1):92–99.

[Shneiderman, 1996] Shneiderman, B. (1996). The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, VL '96, pages 336–. IEEE Computer Society.

[Shneiderman, 1998] Shneiderman, B. (1998). Treemaps for space-constrained visualization of hierarchies - including the history of treemap research at the university of maryland. http://www.cs.umd.edu/hcil/treemap-history/.

[Sikora, 2001] Sikora, T. (2001). The mpeg-7 visual standard for content description-an overview. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):696–702.

[Singhal, 2001] Singhal, A. (2001). Modern Information Retrieval: A Brief Overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 24(4):35–42.

[Tamassia, 2013] Tamassia, R. (2013). *Handbook on Graph Drawing and Visualization*. Chapman and Hall/CRC.

[Terasawa and Tanaka, 2009] Terasawa, K. and Tanaka, Y. (2009). Slit Style HOG Feature for Document Image Word-Spotting. *ICDAR*, pages 116–120.

[Thomas and Cook, 2005] Thomas, J. J. and Cook, K. A. (2005). *Illuminating the Path: The Research and Development Agenda for Visual Analytics*.

[Toussaint, 2014] Toussaint, G. (2014). Applications of the relative neighbourhood graph. *IJCSIA*, 4(2):77–85.

[Toussaint, 1980] Toussaint, G. T. (1980). The relative neighbourhood graph of a finite planar set. *Pattern Recognition*, 12:261–268.

[Toussaint, 1991] Toussaint, G. T. (1991). Some unsolved problems on proximity graphs.

[Treisman, 1985] Treisman, A. (1985). Preattentive processing in vision. *Computer Vision, Graphics, and Image Processing*, 31(2):156 – 177.

[Tufte, 1983] Tufte, E. (1983). *The Visual Display of Quantitative Information*. Cheshire, CT: Graphics.

[Tutte, 1963] Tutte, W. T. (1963). How to draw a graph. *Proceedings of the London Mathematical Society*, 13:743–767.

[Uchida et al., 2015] Uchida, S., Egashira, Y., and Sato, K. (2015). Exploring the world of fonts for discovering the most standard fonts and the missing fonts. In *13th International Conference on Document Analysis and Recognition, Nancy, France, August 23-26, 2015*.

[Uchida et al., 2012] Uchida, S., Ishida, R., Yoshida, A., Cai, W., and Feng, Y. (2012). Character image patterns as big data. In *ICFHR*, pages 479–484.

[Urquhart, 1983] Urquhart, R. (1983). Some properties of the planar euclidean relative neighbourhood graph. *Pattern Recognition Letters*, 1(5-6):317–322.

[Van der Maaten and Hinton, 2008] Van der Maaten, L. and Hinton, G. (2008). Visualizing high-dimensional data using t-sne.

[Vigen, 2014] Vigen, T. (2014). Spurious correlations. http://tylervigen.com/spurious-correlations.

[W3C, 1999] W3C (1999). Css3. https://www.w3.org/Style/CSS/.

[W3C, 2014] W3C (2014). Html5. https://www.w3.org/TR/html5/.

[Wang et al., 2015] Wang, C., Reese, J. P., Zhang, H., Tao, J., Gu, Y., Ma, J., and Nemiroff, R. J. (2015). Similarity-based visualization of large image collections. *Information Visualization*, 14(3):183–203.

[Ware, 2004] Ware, C. (2004). *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc.

[Won et al., 2002] Won, C. S., Park, D. K., and Jeon, Y. S. (2002). Efficient use of mpeg-7 edge histogram descriptor. *ETRI Journal*, 24(1):23–30.

[Zhang et al., 1996] Zhang, T., Ramakrishnan, R., and Livny, M. (1996). Birch: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, SIGMOD '96, pages 103–114, New York, NY, USA. ACM.

Résumé :

Les travaux de recherche présentés et discutés dans cette thèse s'intéressent aux grandes collections d'images numériques. Plus particulièrement, nous cherchons à donner à un utilisateur la possibilité d'explorer ces collections d'images, soit dans le but d'en extraire de l'information et de la connaissance, soit de permettre une certaine sérendipité dans l'exploration. Ainsi, cette problématique est abordée du point de vue de *l'analyse et l'exploration interactive des données*. Nous tirons profit du paradigme de navigation par similarité et visons à respecter simultanément les trois contraintes suivantes : *(i)* traiter de grandes collections d'images, *(ii)* traiter des collections dont le nombre d'images ne cesse de croître au cours du temps et *(iii)* donner des moyens d'explorer interactivement des collections d'images. Pour ce faire, nous proposons d'effectuer une étude conjointe de l'indexation et de la visualisation de grandes collections d'images qui s'agrandissent au cours du temps.

Dans un premier temps, nous nous concentrons sur la construction incrémentale d'un graphe de voisinage, à savoir le Graphe des Voisins Relatifs (GVR), en vue de structurer une collection d'images. Dans un second temps, nous présentons une structure hybride, combinant hiérarchie et graphe, et qui est visualisable pour permettre une exploration interactive de grandes collections d'images. L'algorithme de partitionnement de données BIRCH est exploité pour générer cette structure. Enfin, nous présentons notre plateforme d'exploration interactive de grandes collections d'images. Pour évaluer notre plateforme, nous présentons une évaluation utilisateur, aux cotés de différents cas d'utilisation, tels qu'un système de recommandations ou encore un outil d'analyse visuelle pour l'analyse d'images de documents. L'ensemble de ces cas d'utilisation implique des collections d'images réelles, contenant jusqu'à 7 millions d'images.

Mots clés : Larges collections d'images, Indexation incrémentale, Visualisation, Exploration interactive, Graphe des voisins relatifs, Algorithme BIRCH, Structure hybride, Sérendipité.

Abstract :

The research work that is presented and discussed in this thesis focuses on large and ever-growing image collections. More specifically, we aim at providing one the possibility to explore such image collections, either to extract some kind of information and knowledge, or to wander in the collections. This thesis addresses this issue from the perspective of Interactive Data Exploration and Analytics. We take advantage of the similarity-based image collection browsing paradigm and aim at meeting simultaneously the three following constraints: *(i)* handling large image collections, up to millions of images, *(ii)* handling dynamic image collections, to deal with ever-growing image collections, and *(iii)* providing interactive means to explore image collections. To do so, we jointly study the indexing and the interactive visualisation of large and ever-growing image collections.

Our contribution is three-fold. First, we focus on the incremental construction of a proximity graph, namely the *Relative Neighbourhood Graph* (RNG), to structure the image collections. Second, we propose a hierarchical and graph-based hybrid *viewable* structure to allow an interactive exploration of large image collections. A data partitioning algorithm, namely *BIRCH*, is used to yield this structure. Last, we present our interactive visualisation platform for large image collections. To evaluate the platform, along with an user evaluation, several different use cases are discussed such as a recommendation system for ever-growing image collections and visual analysis system for Document Image Analysis. These use cases are based on real world large image collections, that contain up to seven millions of images.

Keywords : Large and Ever-growing Image Collections, Incremental Structuring, Visualisation, Interactive Exploration, Relative Neighbourhood Graph, BIRCH Algorithm, Hybrid Structure, Serendipity, Visual Analytics.