Bundling : une Technique de Réduction d'Occultation par Agrégation Visuelle et son Application à l'Étude de la Maladie d'Alzheimer

Antoine Lhuillier

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *8/12/2017* **par :**

ANTOINE LHUILLIER

Bundling : une Technique de Réduction d'Occultation par Agrégation Visuelle et son Application à l'Étude de la Maladie d'Alzheimer

---

JURY

DAVID AUBER	Maître de conférences	Rapporteur
DANIEL WEISKOPF	Professeur d'université	Rapporteur
DENIS KOUAME	Professeur d'université	Président du jury
CAROLINE APPERT	Chargé de recherche	Membre du jury
JEAN-DANIEL FEKETE	Directeur de recherche	Membre du jury
EMMANUEL BARBEAU	Directeur de recherche	Invité
CHRISTOPHE HURTER	Professeur	Directeur de thèse
CHRISTOPHE JOUFFRAIS	Directeur de recherche	Co-directeur de thèse

---

École doctorale et spécialité :
MITT : Image, Information, Hypermédia
Unité de Recherche :
UR Laboratoire de recherche ENAC
Directeur(s) de Thèse :
Christophe HURTER et *Christophe JOUFFRAIS*
Rapporteurs :
Daniel WEISKOPF et *David AUBER*

Acknowledgements

First, I would like to deeply thank my PhD advisor Christophe Hurter who encouraged me to pursue a PhD research. Thanks to his immense energy, patience and teaching skills he gave me the motivation and the keys to the scientific research. I could never have imagined any better supervisor, always present in the good as well as the more difficult times of my PhD.

I also would like to thanks all the members of this research project : Christophe Jouffrais, Emmanuel Barbeau and Hélène Amieva, without whom nothing would have been possible. Your various domain of expertise showed me the vast diversity of research and gave me insightful advices and invaluable interdisciplinary experience throughout my PhD. Furthermore, I would like also to acknowledge the support of the Federal University of Toulouse (Université Fédérale Toulouse Midi-Pyrénées - UFT-MiP) and the French Civil Aviation University (ENAC) who funded this project.

I thank very much the reviewers of my thesis David Auber and Daniel Weiskopf as well as all the members of my jury : Denis Kouamé, Caroline Appert and Jean-Daniel Fekete for their critiques, their feedbacks and ultimately for coming to my defence.

Then, I would like to warmly thank Alexandru Telea who helped me refine and challenge my work on bundling by bringing an external perspective during a difficult time of my PhD. I want to thank Ludovic Gardy for his work and help during my work on Alzheimer's disease. I also thank the other PhD students : Almoctar Hassoumi, Maxime Reynal and Michael Traoré as well as all the members of the ENAC research laboratory with whom I always had interesting talks and debates on research topics as well as general knowledge and good laughs. A special thanks to Arnaud Prouzeau, Maxime Cordeil and Emilien Dubois for their encouragements and advices as well as the good times we shared during conferences and summer schools. Thanks to all the proof-readers of this dissertation : Marion, Hugo, Becky and his father.

Throughout my academic education there have been numerous exceptional people whose teaching inspired me and forged me into becoming a scientist. Any specific list would remain incomplete, but hereby I wish to thank each and every one of you.

Moreover, I want to thank all my friends who showed great support during the three years of my PhD, you were always there (even when I was not) to share important moments and recharge my batteries.

Finally, but certainly not least : I would not be the person I am today without my parents, my siblings and the rest of my family. Yet, there is one special person without whom this work and my life would not be the same, my Dear Marion.

Publications

Parts of this dissertation work have been published in international conferences, journals, and workshops and in a French national conference. This section lists all articles published during my PhD:

Papers at International Journals and Conferences

Antoine Lhuillier, Christophe Hurter, and Alexandru Telea (2017). "State of the Art in Edge and Trail Bundling Techniques". In: *Computer Graphics Forum*. Vol. 36.3. Wiley Online Library, pp. 619-645.

Antoine Lhuillier, Christophe Hurter, and Alexandru Telea (2017). "FFTEB: Edge bundling of huge graphs by the Fast Fourier Transform". In: *Pacific Visualization Symposium (PacificVis), 2017 IEEE*. IEEE, pp. 190-199.

Paper at National Conferences

Antoine Lhuillier and Christophe Hurter (2015). "Bundling, graph simplification trough visual aggregation: existing techniques and challenges". In: *Proceedings of the 27th Conference on l'Interaction Homme-Machine*. ACM, p. 10:1–10.

Papers at International Workshops

Antoine Lhuillier, Christophe Hurter, Christophe Jouffrais, Emmanuel Barbeau, and Hélène Amieva (2015). "Visual analytics for the interpretation of fluency tests during Alzheimer evaluation". In: *Proceedings of the 2015 Workshop on Visual Analytics in Healthcare.*. ACM, p. 3:1–3:8.

Antoine Lhuillier, Christophe Hurter, Christophe Jouffrais, Emmanuel Barbeau, and Hélène Amieva (2014). "Cognitive Maps Exploration trough Kernel Density Estimation". In: *EHRVis-Visualizing Electronic Health Record Data. IEEE VIS Workshop 2014*.

I was the main contributor of all those articles. However, several co-authors helped me along the way: first and foremost, Christophe Hurter my PhD advisor, helped me to improve, expand and develop the various ideas and concepts I worked on during my thesis. Alexandru Telea helped me as well to challenge and refine my work towards the improvement and understanding of the bundling landscape. My second PhD advisor, Christophe Jouffrais, helped me understand the cognitive processes at stake in my application to the study of the Alzheimer disease. His help and the help of Emmanuel Barbeau allowed me to identify and fully grasp the wide variety of tasks and use-cases my visual analytics tools should answer to efficiently digitize and clean our new database of verbal fluency tests. Finally, Hélène Amieva helped me during my work by providing us access and resources to digitize the paper-based fluency test of the 3C cohort in Bordeaux.

Contents

Dedicated to the loving memory of Annick Dumond.
1933 – 2012

Chapter 1

Introduction

This thesis is related to the field of Information Visualization (InfoVis). This field of research aims towards the formalization of means for an effective visual communication of information. In other words, this field strives towards improving the understanding of graphical information and optimizing the transmission bandwidth of the visual channel.

This thesis focuses on a problem inherent to dense and multidimensional visualization: *clutter*.

1.1 Context and Motivation

In the field of Information Visualization, big multi-dimensional data-set are analyzed through interactive visualization methods. These processes converting raw-data into visual information can be described through the InfoVis pipeline (Card, Mackinlay, and Shneiderman, 1999, figure 1.1). In this pipeline, raw data is first turned into structured data (*data transformations*), then data structures are converted into visual structures and properties (*visual mappings*), next the visual forms are rendered as image(s) (*view transformations*), to finally allow the perception of data by a human being. InfoVis involves algorithmic approaches to convert raw data into a rendered visualization.

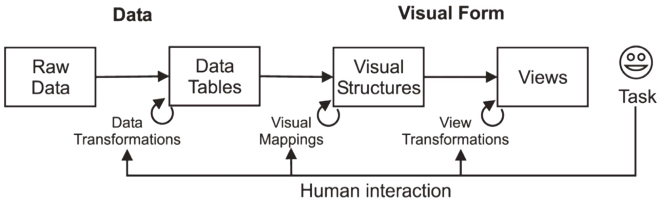


FIGURE 1.1 – InfoVis pipeline according to Card, Mackinlay, and Shneiderman, 1999.

If the efficiency of such approaches have been proved in the past (Card, Mackinlay, and Shneiderman, 1999), they have also shown their limits in processing and visualizing large data-sets (Fekete and Plaisant, 2002). As displays and screens are finite spaces with a limited number of pixels, displaying more and more information quickly introduces denser visualization that results in cluttering problematics. In an effort to solve them, several research approaches and visualization techniques have been developed (Ellis and Dix, 2007), among them *bundling*.

Bundling relates to techniques that provide a visual simplification of a graph drawing or a set of trails, by spatially grouping graph edges or trails (which we next globally call *paths*). This way, the structure of the visualization becomes simpler and thereby easier to comprehend in terms of assessing *relations* that are encoded by such paths, such as finding groups of strongly interrelated nodes in a graph drawing, finding connections between spatial regions on a map linked by a number of vehicle trails, or discerning the motion structure of a set of objects by analyzing their trajectories (as depicted in figure 1.2).

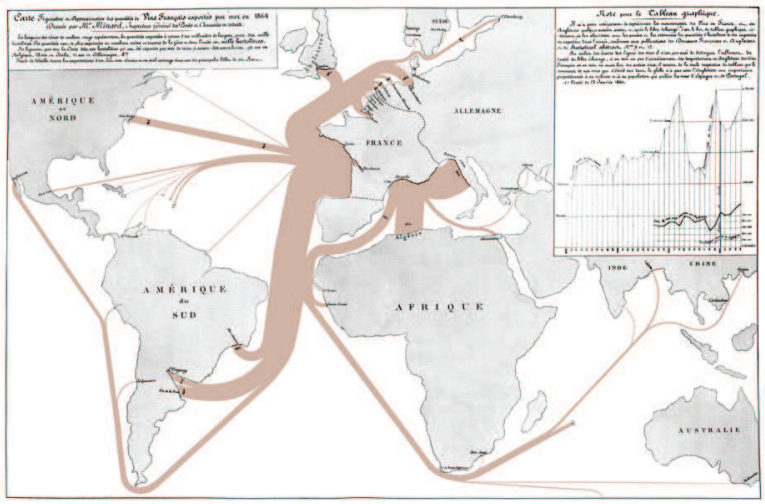


FIGURE 1.2 – Early techniques related to bundling: flow map of French wine exports (Minard, 1864).

Bundling techniques stems from the field of graph simplification such as edge concentration by Newbery, 1989 and the extensions of Sankey graphs by Tufte, 1992. Since then graph drawing simplification has been a major focus of edge bundling techniques, a term introduced by Dickerson et al., 2003 for the reduction of clutter in the drawing of a graph via node placement. The method was next extended to handle a wide variety of layouts and datasets; such as hierarchical graphs drawn in a 2D layout by Holten, 2006 and later 3D layout by the work of Collins and Carpendale, 2007 and Giereth, Bosch, and Ertl, 2008; Holten and Van Wijk, 2009 and Dwyer, Marriott, and Wybrow, 2006 for general graphs; Cui et al., 2008, Ersoy et al., 2011 and Lambert, Bourqui, and Auber, 2010b for spatial trail sets in a 2D embedding and Lambert, Bourqui, and Auber, 2010b on curved surfaces; Hurter, Ersoy, and Telea, 2013 for sequence graphs and Nguyen, Eades, and Hong, 2012 and Hurter et al., 2014a for dynamic and eye tracks; the work of Selassie, Heller, and Heer, 2011, Moura, 2015 and Peysakhovich, Hurter, and Telea, 2015 for generalized directed graphs; Telea and Ersoy, 2010 and Peysakhovich, Hurter, and Telea, 2015 for attributed graphs; McDonnell and Mueller, 2008, Palmas et al., 2014 and Palmas and Weinkauf, 2016 for parallel coordinate plots visualization; Martins et al., 2014 and Rauber et al., 2017 for multidimensional projections; and the work of Yu et al., 2012, Böttger et al., 2014 and Everts et al., 2015 for 3D vector and tensor fields.

Along this growing interest to apply bundling to many data types, a wide array of bundling techniques has been proposed by different authors. Holten, 2006 proposed a technique based on control structures and Gansner et al., 2011 and Telea and Ersoy, 2010 proposed techniques based on graph simplification techniques. Later Holten and Van Wijk, 2009, Dwyer, Marriott, and Wybrow, 2006, Nguyen, Eades, and Hong, 2012 and Everts et al., 2015 based their techniques upon force directed models. Methods proposed by Phan et al., 2005, Cui et al., 2008, Lambert, Bourqui, and Auber, 2010b and Ersoy et al., 2011 are based on computational geometry techniques and Hurter, Ersoy, and Telea, 2012, Böttger et al., 2014, Moura, 2015 and Zwan, Codreanu, and Telea, 2016 use image-based techniques.

Overall, bundling goals largely follow those of early methods for simplifying graphs drawing as detailed in Herman, Melançon, and Marshall, 2000's survey on *Graph visualization and navigation in information visualization.* And since then, bundling has become an established tool for the creation of simplified visualizations of edge and trail data-sets.

However over the last years, the rapid development of the field, coupled with the diversity of its application domains (*e.g.*, graphs, vehicle trails, eye tracking data, vector and tensor fields, all of them attributed or not and time-dependent or not), data types handled and a plethora of different algorithmic approaches, make it hard for users to choose the suitable method for a given use-case, or for researchers to focus on important areas of improvement. Moreover, the ever growing size and multi-dimensionality of data-set coupled with the inherent motive of visualizations to be as interactive as possible (*i.e.* having a low computational speed to provide instant feedback to users), constantly challenges the capabilities of bundling algorithms.

1.2 Research Questions and Approach

In this dissertation, we address the general research question:

➤ ***How can we bridge the gap between the technical complexity of bundling methods and the end-point users?***

Bridging this gap means improving the understanding of bundling methods for users. Here, users can be researchers, engineers or developers with a different goal for each of them. To answer this my effort involves three aspects: a *taxonomy*, a *framework* and a *tasks & uses-cases* analysis.

Taxonomy: First, we must understand how bundling techniques relate to each other. Following the steps of the visual analytics process introduced by Keim et al., 2008 and from a user point of view, we ask the following question:

➤ *How can bundling techniques be classified based on the type of data they work on?*

Framework: Given their large number and diversity, comparing all bundling methods from a technical viewpoint is a difficult challenge. For developers, a technical framework could help to compare in detail specific steps of specific algorithms. For researchers, this could help outlying technical limitations and future improvement areas.

➤ *Can we unify bundling techniques into a generic bundling framework to describe them?*

Tasks & uses-cases analysis: To fully use and understand bundling, we must know which tasks can be supported and in which context. Task taxonomies exist for information visualization in general (Brehmer and Munzner, 2013) as well as for graphs (Lee et al., 2006).

➤ *What are the tasks and proven use-cases bundling can solve?*

Finally, in the context of multidimensional big data analysis, visualization techniques must be able to cope with the ever growing size of data-set. This is valid as well for bundling, where we aim towards fast interactions and aggregation based on user driven criteria (*e.g.* attribute-driven bundling). Here, we can ask ourselves:

➤ *How can bundling cope with very large multidimensional data-sets?*

Application to the Study of Alzheimer's Disease

As part of the multidisciplinary project called *MEMOIRE*, this thesis will report a new usage of edge bundling technique in the field of neuro-psychology.

Currently, approximately 860,000 people are affected by Alzheimer's disease in France. This estimation will possibly reach 2.1 million in 2040. This is why the study of Alzheimer's disease has been identified as a major societal challenge. In the 3C cohort study (Antoniak et al., 2003), 3777 elderly people followed performed a lexical evocation task (*i.e.* a fluency test) by saying the maximum number of city names within one minute. Subjects were followed during a 12 year period and some developed Alzheimer's Disease (AD).

During the fluency test, practitioners handwrote the cited cities on paper. This task is directly related to the concept of cognitive maps (Tolman, 1948). The analysis of the list of cited cities created during the fluency tasks provides a unique opportunity to study the spatial mental representation of geographical space of elderly people, with a cross-sectional study, and the modification during the onset of Alzheimer's dementia, with a longitudinal study. To the best of our knowledge, no previous work has investigated this dataset thanks to interactive visualization methods (Card, Mackinlay, and Shneiderman, 1999). In the following chapters, we present our work which fills this gap. Throughout this application to the study of AD, we will answer the following questions.

First, as fluency tests were written on paper, the first step of our work was to produce an electronic health database as clean as possible. To support this task of cleaning, we asked ourselves the following question:

➤ *How can we use visual analysis tools to support efficiently the digitization and cleaning of handwritten electronic health record data?*

From the build database, we focused our research towards the development of a novel observation tool to analyze both the aging process of elderly and the onset of AD. In the framework of this thesis on bundling, our question was;

➤ *How and to what extend is bundling an efficient tool to generate visualization as an observational and exploratory mean for scientist?*

1.3 Thesis Outline

As a reading guide, each chapters of this dissertation contains a section numbered 0 located just after the chapter's preamble. Section 0 of each chapter is composed of a summary of the given chapter in my native language, French. Non-French speaker can directly jump to the first section of the chapter (numbered 1). Whereas French speakers have access to the outlines of each chapters in their native language. Overall, this dissertation is divided into three parts, seven chapters and finishes with a chapter on conclusions and future work.

Part I: Bundling in the Information Visualization Context

The first part gives an overview of the place of bundling in the context of visualization through two approaches.

Chapter 2: Clutter Reduction for Node-Link Diagrams introduces the major definitions and concepts that relate to bundling techniques. We start by introducing the types of data (and their existing layout) that bundling works on. Then, we detail the basis of existing clutter reduction techniques in order to better understand the scope of bundling in the clutter reduction landscape. Finally, we introduce the main notations and a formal definition of path bundling, which we next specialize for graph and trail bundling.

Chapter 3: A Data-based Taxonomy of Bundling Methods This chapter present our taxonomy of bundling methods according to the type of input data-set they work on (graphs *vs* trails, which we jointly refer to as *paths*). We show how this taxonomy is more clear-cut than the usual *technique-based* taxonomies that group bundling methods into data-driven, geometric, and image-based ones (see *e.g.* Zwan, Codreanu, and Telea, 2016; Zhou et al., 2013; Hurter, Ersoy, and Telea, 2012). Moreover, our taxonomy helps users choose suitable algorithms for their data without having to dive into algorithmic details.

Part II: Unifying, Understanding and Improving Bundling Techniques

The second part of this dissertation works towards solving challenges of bundling techniques through three chapters. The first two chapters solve the challenges of comparing and choosing different bundling algorithms. Then we tackle the computational and scalability challenges of bundling techniques.

Chapter 4: Towards a Unified Bundling Framework In this chapter, we propose a generic framework that describes the typical steps of all bundling methods in terms of high-level operations. We next show how such methods implement these steps. This helps one to compare in detail specific steps of specific algorithms, and complements the above-mentioned high-level taxonomy of bundling algorithms with technical insights. Our framework also helps outlining technical limitations of existing algorithms, suggesting future improvement areas.

Chapter 5: Bundling Tasks and Applications As outlined already, comparing bundling techniques is hard. We address this by proposing a description of *tasks* that bundling aims to address, following Lee et al., 2006's and Brehmer and Munzner, 2013's tasks taxonomies. We discuss how bundling can address these tasks, and also where salient limitations exist, in terms of the operations of our proposed framework. We also discuss ways (and challenges) to compare the results of different bundling methods. By this, we aim to provide a guide to the practitioner in selecting, customizing, testing, and possibly extending existing bundling methods to support optimally a given task set. Next, as bundling has gone far beyond simplified graph drawing we overview a wide set of bundling applications and relate these to the above-mentioned tasks. By this, we address limitations of papers which usually focus either on proposing a new technique or discussing a single application.

Chapter 6: Improving Bundling Scalability In this chapter, we bring bundling a step further towards its ability to cope with increasingly large data-sets. We present the FFTEB technique that addresses the joint edge-bundling challenges of computational speed and data volume, while keeping the quality of the results on par with state-of-the-art techniques. For this, we shift the bundling process from the image space to the spectral space, thereby increasing computational speed. We address scalability by proposing a data streaming process that allows bundling of extremely large data-sets with limited GPU memory.

Part III: Application to the Study of Alzheimer's Disease

This part reports the use of path bundling techniques in the context of the study of Alzheimer's Disease. Using the results of a French cohort study (3C Antoniak et al., 2003), we study both the effect of aging and the evolution of Alzheimer's disease on the human brain through the study of fluency tests (Isaacs and Akhtar, 1972; Isaacs

and Kennie, 1973). The two following chapters expose the results of our work towards a better understanding of Alzheimer's disease.

Chapter 7: A Visual Analytics Approach for Digitizing and Cleaning Health Data Prior to the analysis of bundled cognitive maps, this chapter presents visual analytics tools for electronic health records (EHR). Because the fluency tests were handwritten very quickly by a medical practitioner, some cities are abbreviated or poorly written. In order to analyze such data, we need to digitize and clean the data-set. As these tasks are intricate and error prone, we propose a novel set of tools, involving interactive visualization techniques, to help users in the digitization and data-cleaning process.

Chapter 8: Bundling Cognitive Maps In this chapter we illustrate our exploration and analysis process of our cleaned data-set built in chapter 7. Centered around the clutter reduction capabilities of our bundling technique for node-link diagrams, we show how bundling can efficiently be used as a visualization and exploration tool for neuro-psychologists in our study of Alzheimer's disease. Specifically, we show how bundling allowed us to formulate new hypothesis on Alzheimer's disease.

1.4 Introduction (Français)

Cette thèse s'inscrit dans le domaine de la visualisation d'information (InfoVis – *Information Visualization*). Le domaine de l'InfoVis a pour but d'optimiser la bande passante d'informations du canal visuel entre utilisateurs et visualisation. En d'autres termes, ce domaine vise à optimiser l'utilisation du système perceptif visuel pour optimiser la transmission et la compréhension d'informations graphiques.

Cette thèse se concentre sur un problème inhérent à la représentation de données denses et multidimensionnelles : l'occultation graphique.

1.4.1 Contexte et Motivation

Dans le domaine de la visualisation d'informations, les grands jeux de données multidimensionnelles sont analysés à l'aide de méthodes de visualisations interactives. Ces processus qui convertissent les données brutes en informations graphiques peuvent être décrits à travers la boucle de rendu de l'infovis (ou *pipeline de l'InfoVis*, Card, Mackinlay et Shneiderman, 1999, figure 1.3). Dans cette boucle de rendu, les données brutes sont d'abord transformées en données structurées (transformation de données), puis les données structurées sont converties en structures et propriétés visuelles (représentation visuelle). Ensuite, les structures visuelles sont compilées pour former une ou plusieurs images (transformation visuelle) pour finalement permettre la perception visuelle des données par l'humain. L'InfoVis implique l'utilisation d'approches algorithmiques pour convertir les données brutes en visualisations compilées (ou images).

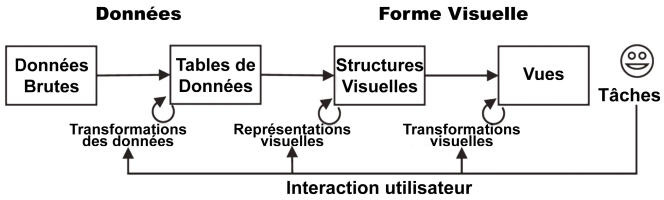


FIGURE 1.3 – Boucle de rendu de la visualisation d'informations d'après Card, Mackinlay et Shneiderman, 1999.

Si l'efficacité de telles approches a déjà été prouvée par le passé (Card, Mackinlay et Shneiderman, 1999), elles ont aussi montré leurs limites quant au traitement et à la visualisation de grandes quantités d'informations (Fekete et Plaisant, 2002). Comme les affichages et les écrans sont des espaces de dimensions finies limités par le nombre de pixels, la représentation visuelle de grande quantités d'informations introduit des représentations visuelles de plus en plus denses qui provoquent des problèmes d'occultation des données. Dans l'optique de réduire les problèmes d'occultation, plusieurs pistes de recherches et techniques de visualisation ont été développées (Ellis et Dix, 2007) et parmi elles, le *bundling*.

Le terme *bundling* fait référence à l'ensemble des techniques de simplification visuelle appliquées aux dessins de graphes ou aux jeux de trajectoires et qui consistent à spatialement grouper les liens d'un graphe ou les trajectoires ensembles. Dans la suite

de cette thèse, nous désignerons par le terme générique *chemins* les liens d'un graphe ou les trajectoires d'un jeu de données. Ainsi, le processus de bundling ou d'agrégation visuelle de chemins permet de simplifier la structure de la visualisation afin de la simplifier et donc de la rendre plus facile à comprendre en termes d'évaluation des relations encodées par ces chemins comme par exemple : la recherche de groupes fortement interconnectés dans un graphe, la recherche de régions fortement interconnectées dans une carte de trajectoires ou encore pour discerner la structure des déplacements d'un ensemble d'objets à travers l'analyse de leurs trajectoires (comme détaillé en figure 1.4).

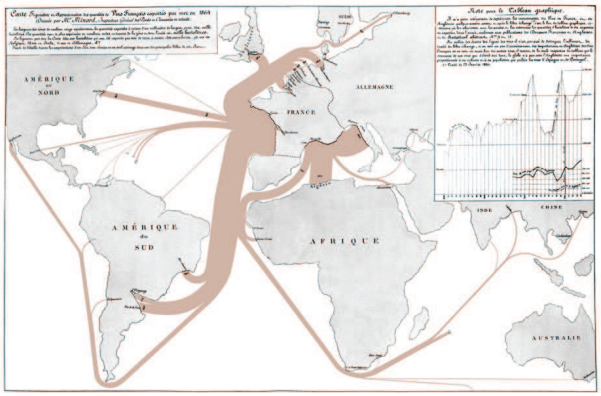


FIGURE 1.4 – Première apparition d'une technique ayant attrait au bundling : la carte des flux d'exportation des vins français (Minard, 1864).

Les techniques de bundling prennent racine dans le domaine de la simplification de graphe à travers des techniques telles que la concentration de liens (*edge concentration* – Newbery, 1989) et les extensions aux graphes de Sankey par Tufte, 1992. Depuis, la simplification du dessin d'un graphe été l'objectif principal des techniques d'agrégation de lien dit de *edge bundling*, terme introduit par Dickerson et al., 2003 pour sa méthode consistant à réduire l'occultation dans le dessin de graphe via le placement de nœuds. La méthode a été ensuite étendue pour gérer un large éventail de techniques d'agencement de graphes et de types de jeux de données tels que : les graphes hiérarchiques dessinés en 2D par Holten, 2006 et plus tard en 3D grâce aux travaux de Collins et Carpendale, 2007 et Giereth, Bosch et Ertl, 2008 ; Holten et Van Wijk, 2009 et Dwyer, Marriott et Wybrow, 2006 pour les graphes génériques ; Cui et al., 2008, Ersoy et al., 2011 et Lambert, Bourqui et Auber, 2010b pour les trajectoires spatialisées en 2D et Lambert, Bourqui et Auber, 2010b sur des surfaces courbées ; Hurter, Ersoy et Telea, 2013 pour les graphes séquentiels et Nguyen, Eades et Hong, 2012 et Hurter et al., 2014a pour les graphes dynamiques et les tracés oculaires ; les travaux de Selassie, Heller et Heer, 2011, Moura, 2015 et Peysakhovich, Hurter et Telea, 2015 pour appliquer le bundling aux graphes attribués ; McDonnell et Mueller, 2008, Palmas et al., 2014 et Palmas et Weinkauf, 2016 l'ont appliquée aux visualisations de coordonnées parallèles ; Martins et al., 2014 et Rauber et al., 2017 pour les projections multidimensionnelles ; et les travaux de Yu et al., 2012, Böttger et al., 2014 et Everts et al., 2015 pour les champs de vecteurs 3D ou de tenseurs.

A travers l'intérêt croissant de l'extension des cas d'application du bundling à plusieurs types de jeux de données, un large éventail de techniques et d'algorithmes de bundling a été proposé par différents auteurs. Holten, 2006 a proposé une technique qui se fonde sur des structures de contrôles et Gansner et al., 2011 ainsi que Telea et Ersoy, 2010 ont proposé des techniques basées sur des techniques de simplification de graphes. Plus tard, Holten et Van Wijk, 2009, Dwyer, Marriott et Wybrow, 2006, Nguyen, Eades et Hong, 2012 et Everts et al., 2015 ont fondés leurs techniques sur des modèles de type interactions de forces. En parallèle, les méthodes proposées par Phan et al., 2005, Cui et al., 2008, Lambert, Bourqui et Auber, 2010b et Ersoy et al., 2011 se fondent sur le calcul de l'agrégation des liens grâce à des approches géométriques. Enfin, Hurter, Ersoy et Telea, 2012, Böttger et al., 2014, Moura, 2015 et Zwan, Codreanu et Telea, 2016 ont proposé des techniques basées image.

Globalement, le bundling suit largement les buts relatifs aux premières méthodes de simplification de graphes comme explicité par l'étude de Herman, Melançon et Marshall, 2000 sur la visualisation de graphes et la navigation dans le domaine de la visualisation d'information (*graph visualization and navigation in information visualization*). Et depuis ce, le bundling est devenu un outil de référence pour générer des visualisations simplifiées d'ensembles de liens ou de trajectoires (*i.e.* d'ensembles de chemins).

Cependant au cours des dernières années, le développement rapide de ce champ de recherche couplé à la grande diversité des domaines d'application (*e.g.* graphes, trajectoires de véhicules, tracés oculaires, champs de vecteurs et tenseurs, tous ces jeux de données pouvant être attribués ou non et temporel ou non), du type de jeux de données traités et la pléthore d'approches algorithmiques, compliquent la tâche des utilisateurs pour choisir la méthode de bundling adaptée à leurs besoins, ou encore compliquent la tâche des chercheurs pour se concentrer sur les axes d'amélioration importants. De plus, l'augmentation constante de la taille et de la multi-dimensionnalité des jeux de données à traiter en parallèle avec le désir constant d'avoir des visualisations aussi interactives que possible (*i.e.* ayant une faible complexité algorithmique afin de fournir un retour rapide aux utilisateurs), challengent constamment les capacités des algorithmes de bundling.

1.4.2 Questions de Recherche et Approche

Au cours de cette dissertation, nous abordons la question de recherche suivante :

➤Comment peut-on combler les lacunes entre la complexité technique des algorithmes de bundling et les utilisateurs finaux ?

Combler ces lacunes reviens à améliorer la compréhension des méthodes de bundling par les utilisateurs. Dans notre cas, les utilisateurs sont soit des chercheurs, des ingénieurs ou des développeurs ayant chacun des objectifs différents. Pour y répondre, mon travail implique trois aspects : une *taxonomie*, un *framework* ainsi qu'une analyse des *tâches et cas d'applications.*

Taxonomie : En premier lieu, nous devons comprendre les relations entre les différentes techniques de bundling. En suivant la démarche des techniques d'analyse visuelle introduite par Keim, 2000 et en abordant un point de vue centré utilisateurs, nous nous posons la question suivante :

➤ *Comment s'organisent et se classifient les techniques de bundling en fonction du type de jeu de données utilisé ?*

Framework : Compte tenu de grand nombre et de la grande diversité des techniques de bundling, il est difficile de comparer ces dernières d'un point de vue technique. Pour les développeurs, un cadre d'application (ou *framework*) permettrait d'aider à comparer en détail des étapes spécifiques d'algorithme de bundling donnés. Pour les chercheurs, un tel framework permettrait de mettre en évidence les limitations techniques et les axes d'amélioration des techniques de bundling.

➤ *Peut-on unifier et décrire les techniques de bundling à l'aide d'une chaine de traitement (ou* pipeline*) générique ?*

Analyse des tâches et cas d'utilisation : Pour pouvoir utiliser les techniques de bundling à leurs plein potentiel, il est nécessaire de connaître les tâches spécifiques supportées par le bundling et leur contexte associé. Pour répondre à cette question, nous nous appuierons sur les taxonomies de tâches liées à la visualisation d'informations (Brehmer et Munzner, 2013) et liées aux dessins de graphes (Lee et al., 2006).

➤ *Quels sont les tâches ainsi que les cas d'utilisation avérés auxquels le bundling répond ?*

Enfin, dans le contexte de l'analyse de grandes quantités de données multidimensionnelles, les techniques de visualisation doivent pouvoir passer à l'échelle et être capable de traiter ces données. Cette assertion s'applique aussi au bundling qui vise à apporter une agrégation rapide des données en fonction de critères spécifiques à l'utilisateur (*e.g.* l'agrégation basée attributs) tout en conservant des interactions toutes aussi rapides. Ici, nous nous posons la question :

➤ *Comment le bundling peut-il faire traiter rapidement de grandes quantités de données multidimensionnelles ?*

Application à l'Étude de la Maladie d'Alzheimer

S'insérant dans le projet multidisciplinaire *MÉMOIRE*, cette thèse rapporte un nouveau cas d'application du bundling au domaine de la neuropsychologie.

Actuellement, environ 860000 personnes sont touchées par la maladie d'Alzheimer en France. Ce chiffre devrait atteindre près de $2,1$ millions d'ici 2040. Ainsi, l'étude de la maladie d'Alzheimer a été identifiée comme un défi sociétal majeur. Dans le cadre de la cohorte 3C (Antoniak et al., 2003), 3777 personnes âgées ont dût réaliser une tâche

d'évocation lexicale (*i.e.* une tâche de fluence verbale) au cours de laquelle ils devaient citer le plus de villes possible en une minute. Ces sujets ont été suivis sur une période de 12 années et certains d'entre eux ont développé la maladie d'Alzheimer (*Alzheimer's Disease* – AD).

Au cours des tâches de fluences verbales, des médecins ont rapporté les villes citées sur papier, ce qui nous donne l'opportunité d'étudier les villes citées. Les études de Tolman, 1948 nous montrent que cette tâche est directement reliée au concept de cartes cognitives. C'est pourquoi l'analyse de la liste des villes citées pendant les tests de fluence verbales est une opportunité unique d'étudier la représentation mentale de l'espace géographique (cartes cognitives) chez les personnes âgées, grâce à une étude transversale, ainsi que d'étudier le développement de la maladie d'Alzheimer, grâce à une étude longitudinale. À notre connaissance, aucuns autres travaux n'ont entrepris l'étude de ce jeu de données à l'aide de techniques de visualisation interactives (Card, Mackinlay et Shneiderman, 1999). Les chapitres suivants présentent mes travaux à ce sujet. À travers cette application du bundling à la maladie d'Alzheimer, cette thèse répondra ainsi aux question suivantes.
Premièrement, comme les tests de fluence verbales ont été rapportés sur papier, nous avons dû créer une base de données informatisée (*Electronic Health Record* – EHR) contenant le moins d'erreurs possibles. Pour répondre à cette tâche complexe, nous avons répondu à la question suivante :

➤ *Comment peut-on efficacement aider la numérisation et le nettoyage de données de santé à l'aide d'outils d'analyses visuelles ?*

À partir de notre base de données construite et nettoyée, nous avons concentré notre recherche sur le développement d'un nouvel outil d'observation permettant à la fois l'analyse de l'impact processus de vieillissement mais aussi de l'impact du développement de la maladie d'Alzheimer sur les tâches de fluences verbales. Dans le cadre de cette thèse consacré aux techniques d'agrégation visuelle, nous avons répondus à la question :

➤ *Comment et dans quelle mesure le bundling est-il un outil de visualisation de données efficace pour générer des visualisations qui permettent aux scientifiques d'explorer et de faire de nouvelles observations de leurs données ?*

1.4.3 Plan de Lecture

L'ensemble de ce manuscrit est rédigé en langue anglaise. Cependant, les points importants de chaque chapitre sont résumés en français. Ces résumés se situent juste après le préambule de chaque chapitre et sont numérotés en section 0 dudit chapitre (*e.g.* la section 2.0 est la section dédiée au résumé en français du chapitre 2). Dans son ensemble, le manuscrit est divisé en trois parties, sept chapitres et se termine par un chapitre de conclusion et de pistes de recherches entièrement traduit en français.

Partie I : Le Bundling dans le Contexte de la Visualisation d'Informations

La première partie nous donne un aperçu de la place du bundling dans le contexte de la visualisation d'information à travers deux approches.

Chapitre 2 : Réduire l'Occultation des Diagrammes Nœuds-Liens Ce chapitre introduit les concepts et les définitions majeures qui ont trait aux techniques de bundling. Nous commençons par introduire les types de jeux de données (et leurs méthodes de dessins associées) que le bundling peut traiter. Puis, nous détaillons les bases des techniques de réduction d'occultation afin de mieux comprendre la place du bundling dans ce paysage. Enfin, nous introduisons les principales notations et notre définition formelle de l'agrégation de chemins, que nous explicitons ensuite aux cas de l'agrégation de graphes ou de trajectoires.

Chapitre 3 : Taxonomie des Méthodes de Bundling Dans ce chapitre, nous proposons une taxonomie des méthodes de bundling fondée sur la structure des données d'entrée (*i.e.* graphes ou trajectoires). Bien qu'il existe d'autres taxonomies des méthodes de bundling purement fondées sur les différences techniques entre ces dernières, comme illustré par Zhou et al., 2013, nous démontrons qu'une taxonomie fondée sur les structures d'entrée est plus claire et précise. De plus, cette approche "structure d'entrée" permet d'aider les utilisateurs à comprendre les techniques de bundling sans pour autant devoir aborder les spécificités de chaque méthode.

Partie II : Unifier, Comprendre et Améliorer les Techniques de Bundling

La seconde partie de ce manuscrit s'inscrit dans un travail de recherche visant à résoudre les défis des techniques de bundling à travers trois chapitres. Les deux premiers chapitres résolvent les défis liés à la comparaison et au choix des différents algorithmes de bundling existants. Puis nous nous attaquons aux défis de la complexité algorithmique et du passage à l'échelle des techniques de bundling.

Chapitre 4 : Un Pipeline Graphique Unifié des Méthodes de Bundling Dans ce chapitre, nous proposons un cadre d'application générique permettant de décrire les étapes nécessaires pour le calcul de l'agrégation visuelle en terme d'opération haut niveau. Puis nous montrons comment les techniques existantes implémentent les différentes étapes de notre pipeline. Ce travail permet de comparer en détail les étapes spécifiques à certains algorithmes de bundling tout en complémentant notre taxonomie haut niveau en apportant des détails techniques. Ce pipeline permet aussi de mettre en évidence les limitations techniques des algorithmes existants pour suggérer des pistes d'améliorations futures.

Chapitre 5 : Tâches et Applications du Bundling Comme détaillé dans les précédents chapitres, il est difficile de comparer l'ensemble des techniques de bundling sans un cadre de référence commun et accepté. Dans ce chapitre, nous analysons comment les techniques de bundling s'intègrent dans la taxonomie de réduction d'occultation de Ellis et Dix, 2007. Puis, nous proposons une description des tâches de visualisation (définies par Lee et al., 2006 et Brehmer et Munzner, 2013) auxquelles le bundling peut répondre. Ensuite, nous montrons comment le bundling peut répondre aux tâches identifiées et montrons les limitations des techniques actuelles. Pour ce faire, nous analysons un large spectre des cas d'utilisations du bundling et les mettons en lumière au regards des tâches de visualisation à résoudre. Ainsi, cette approche nous permet de mettre en lumière les limitations ainsi que les potentiels axes d'amélioration du bundling au regard des besoins utilisateurs.

Chapitre 6 : Une Nouvelle Technique de Bundling : FFTEB Dans ce chapitre, nous améliorons les capacités du bundling à traiter les grandes quantités de données. Nous présentons un nouvelle technique, FFTEB, qui résout les problèmes de passage à l'échelle des algorithmes de bundling. Afin de résoudre ces problèmes de vitesse de calcul et de complexité, nous changeons le paradigme de calcul du bundling en utilisant les propriétés de la transformée de Fourier (FFT). Puis nous résolvons les problèmes liés à la taille des jeux de données en proposant une méthode de transfert de données (*data streaming*) permettant l'agrégation de très larges jeux tout en s'abstrayant de la limite de taille de la mémoire des processeurs graphique.

Partie III : Application à l'Étude de la Maladie d'Alzheimer

Dans cette partie, nous rapportons l'utilisation du bundling dans le contexte de l'étude de la maladie d'Alzheimer. En utilisant les résultats de la cohorte française 3C (Antoniak et al., 2003), nous étudions à la fois les effets du vieillissement et l'évolution de la maladie d'Alzheimer sur le cerveau humain à travers l'étude des tests de fluence verbale (Isaacs et Akhtar, 1972 ; Isaacs et Kennie, 1973). Les deux chapitres suivants exposent les résultats de notre travail visant à améliorer notre compréhension de cette maladie.

Chapitre 7 : Une Méthode d'Analyse Visuelle pour la Numérisation de Données de Santé Avant l'analyse des cartes cognitive agrégées, ce chapitre présente nos outils d'analyse visuelle pour la numérisation des données de santé (EHR). À cause du fait que les tests de fluence verbales ont été rédigés sur papier de façon rapide par un médecin, beaucoup de villes ont été abréviées ou mal écrites. Ainsi, pour pouvoir analyser ces données, nous avons dû numériser et nettoyer notre jeu de données. Comme ces tâches sont complexes et sujettes à des erreurs, nous proposons de nouveaux outils impliquant des méthodes de visualisation interactives, afin d'aider les utilisateurs dans le processus de numérisation et de nettoyage des données.

Chapitre 8 : Application du Bundling aux Cartes Cognitives Au cours de ce dernier chapitre, nous illustrons notre processus d'exploration et d'analyse de notre base de donnée des tests de fluence verbale (construite en chapitre 7). À l'aide de notre technique de bundling (exposée en chapitre 6), nous montrons que le bundling est un outil de visualisation et d'exploration efficace dans le cadre de notre étude sur la maladie d'Alzheimer. Plus spécifiquement, nous montrons que l'utilisation du bundling nous a permis de formuler de nouvelles hypothèses sur l'évolution de la maladie d'Alzheimer.

Part I

Bundling in the Information Visualization Context

Chapter 2

Clutter Reduction for Node-Link Diagrams

Bundling combines two concepts of Information Visualization: path drawing and clutter reduction. This chapter clarifies the main definition and concept for graph drawing and trails as well as clutter reduction techniques and criteria. We start by introducing the types of data (graph or trails) and their existing visualizations that bundling works on. Then we detail the basis of clutter reduction techniques to better understand the scope of bundling in this particular landscape. Finally, we introduce the main definitions and notations we deem necessary to understand the design space of bundling techniques.

2.0 Résumé (Français)
Réduire l'Occultation des Diagrammes Nœuds-Liens

Les techniques de bundling (simplification de liens par agrégation visuelle) combinent deux concepts majeurs de la visualisation d'informations : le dessin de liens et la réduction d'occultation. Dans ce chapitre, nous clarifions les principales définitions et concepts relatifs aux dessins de liens (dessins de graphes ou dessins de trajectoires) ainsi que les techniques et critères relatifs à la réduction d'occultation de la visualisation de données. Nous débutons par l'analyse du type de données utilisables par les techniques de bundling (graphes ou trajectoires) ainsi que les moyens existants pour les visualiser (voir section 2.1). Ensuite nous détaillons les fondations des techniques de réduction d'occultation afin de mesurer la place du bundling parmi ces dernières (section 2.2). Enfin, nous introduisons les principales définitions et notations nécessaires à la compréhension de l'espace de conception des algorithmes de bundling (section 2.3). Dans ce cours résumé en français, nous nous limiterons à définir les concepts et notations relatifs aux techniques d'agrégation visuelle.

Definition de l'Opérateur Bundling

L'espace de conception des techniques de bundling est vaste et complexe contenant pléthore d'implémentations variées et de types de jeux de données utilisables. Dans l'optique d'unifier et d'améliorer la compréhension de cet espace, nous débutons notre travail par la définition des concepts et notations sur lequels se fondent cette thèse.

Les Objectifs du Bundling

Les graphes de grande taille, fortement connectés, issus d'applications concrètes ont beaucoup plus de liens que de nœuds. Au cause de cela, les représentations de ces graphes par des diagrammes nœuds-liens classiques deviennent rapidement inefficaces pour les explorer et les analyser. Ce problème est souvent référencé dans la littérature comme l'encombrement de liens (*edge congestion*, Carpendale et Rong, 2001, l'occultation de liens (*visual clutter*, Ellis et Dix, 2007) ou le problème de la boule de poils (*hairball problem*, Schulz et Hurter, 2013). Ainsi pour résoudre ce problème, plusieurs techniques existent (voir section 2.2 et Ellis et Dix, 2007) et parmi elles le bundling.

De manière informelle, le bundling échange l'occultation pour du "sur-dessin" (Telea et Ersoy, 2010). Cependant, bien qu'il y ait des dizaines d'articles sur le bundling, aucun ne définit formellement ce qu'est le bundling. Dans notre travail, nous argumentons qu'une telle définition est nécessaire si l'on souhaite comprendre le processus, comparer les méthodes et débattre des garanties et des limitations du bundling et plus généralement faire avancer la recherche sur le bundling.

Définition du Bundling

Soit $G = (V, E \subset V \times V)$ un graphe avec des nœuds $V = \{\mathbf{v}_i\}$ et des liens $E = \{\mathbf{e}_i\}$. Soit d la dimension de l'espace de dessin dans lequel nous souhaitons appliquer le bundling (généralement cette dimension est 2 ou 3). En parallèle, soit $T = \{\mathbf{t}_i\}$ ce que nous appellerons un jeu de trajectoires comme définis par Phan et al., 2005 et Zhou et al., 2013. Une trajectoire $\mathbf{t}_i \subset \mathbb{R}^d$ est une courbe orientée. Généralement, les trajectoires décrivent les déplacements d'objets dans l'espace, *e.g.* des avions (Hurter, Tissoires et Conversy, 2009), des tracés oculaires (Peysakhovich, Hurter et Telea, 2015), des bateaux (Scheepens et al., 2011a), ou des personnes (Nagel, Pietsch et Dörk, 2016). Cependant, elles peuvent aussi représenter des courbes non liées à des déplacements, *e.g.* des lignes brisées dans la visualisation de type coordonnées parallèles (*Parallel Coordinates Plots* – PCP) (Inselberg, 2009) ou des tracés de tenseurs de diffusion d'IRM (Everts et al., 2015). Il est important de noter qu'en théorie des graphes la notion de trajectoire a une signification différente (un type de chemin dans un graphe pour lequel tous les liens sont distincts, voir Harris, Hirst et Mossinghoff, 2008). Soient $\mathcal{G}$ et $\mathcal{T}$ l'espace de tous les graphes et respectivement tous les jeux de trajectoires.

L'élément clé unifiant les graphes et les jeux de trajectoires est ce que l'on nomme l'opérateur de dessin D. Pour les graphes, $D : \mathcal{G} \to \mathbb{R}^d$ est typiquement une technique d'agencement ou de dessin des liens d'un graphe (voir Battista et al., 1998). Par analogie, nous noterons $D(\mathbf{e}_i)$ et $D(\mathbf{v}_i)$ comme la transformée (le dessin) des nœuds $\mathbf{e}_i \subset E$, respectivement des liens $\mathbf{v}_i \subset V$. Pour les trajectoires, l'opérateur de dessin est la fonction identité, *i.e.* $D(\mathbf{t}_i) = \mathbf{t}_i$, car les trajectoires sont *déjà* intégrées dans un espace géométrique. Nous unifions les deux espaces $\mathcal{G}$ et $\mathcal{T}$ en définissant $\mathcal{P}$ l'espace des jeux de chemins. Ainsi P dénote soit un graphe G soit un jeu de trajectoires T et par analogie, un chemin $\mathbf{p} \in P$ est soit un lien de graphe $\mathbf{e}$ ou une trajectoire $\mathbf{t}$. Les chemins peuvent avoir des attributs additionnels, tels que la direction, le poids, la temporalité, un nom ou un type générique comme introduit par Peysakhovich, Hurter et Telea, 2015 et Diehl et Telea, 2014. Ainsi, un chemin $\mathbf{p}$ peut être vu comme un vecteur de dimension $n + d$, avec n le nombre d'attributs et d la dimension spatiale du chemin.

Soit $\mathcal{D} \subset \mathbb{R}^d$ l'espace de l'ensemble des dessins de chemins $D(P)$. Soit $B : \mathcal{D} \to \mathcal{D}$ un opérateur denotant l'agrégation (le bundling) d'un jeu de chemins ; et finalement soit $B(D(\mathbf{p}))$ la courbe représentant l'agrégation du chemin $\mathbf{p}$. Alors B est une méthode de bundling (ou d'agrégation) si :

$$\forall(\mathbf{p}_i, \mathbf{p}_j) \in P \times P | \mathbf{p}_i \neq \mathbf{p}_j \wedge \kappa(\mathbf{p}_i, \mathbf{p}_j) < \kappa_{max} \rightarrow$$
$$\delta(B(D(\mathbf{p}_i)), B(D(\mathbf{p}_j))) \ll \delta(D(\mathbf{p}_i), D(\mathbf{p}_j)). \qquad \textbf{Équation du bundling}$$

Dans cette équation, δ est une distance entre les chemins de $\mathcal{R}^d$, *e.g.* la distance de Hausdorff comme défini par Berg et al., 2010. $\kappa : P \times P \to \mathbb{R}^+$ est ce que nous appellerons une fonction de compatibilité dénotant la similarité entre les chemins. En d'autres termes, quand κ tend vers 0 alors les chemins sont similaires, lorsque κ tend vers l'infini alors les chemins sont différents (*i.e.* incompatibles). Dans tous les cas, κ se doit d'être une fonction qui reflète la compatibilité géométrique des chemins dans leur espace

de dessins $D(P)$, *i.e.* lorsque $\kappa(\mathbf{p}_i, \mathbf{p}_j)$ tend vers 0 alors $\delta(D(\mathbf{p}_i), D(\mathbf{p}_j))$ tend lui aussi vers 0. De plus, κ peut inclure des compatibilités sur les n attributs additionnels (basés sur les données) mentionnés plus tôt, *i.e.* κ peut être modélisée par une distance entre chemins en dimension $n+d$ (l'espace géométrique et des attributs, *c.f.* Peysakhovich, Hurter et Telea, 2015). Dans notre équation, seuls les chemins dont la compatibilité est inférieure à un seuil prédéfinis (κ_{max}) doivent être bundlés – sinon le bundling de $D(P)$ subirait des déformations trop importantes pour être correctement analysé. De façon informelle, notre **Équation du bundling** définie que pour deux chemins compatibles alors le dessin bundlé de ces chemins $B(D(\mathbf{p}_i))$ est géométriquement plus proche que le dessin original de ces premiers $D(\mathbf{p}_i)$.

Différences et Similitudes entre Graphes et Trajectoires

Dans la litérature des techniques de bundling, les auteurs font souvent référence aux termes *edge bundling* ou *graph bundling* de manière non discriminée, même dans le cas où les données à bundler sont des trajectoires. Ainsi, il est donc important de clarifier les différences et ressemblances entre le bundling de ces deux entités.

Pour résumer, les graphes et les trajectoires sont deux types de données radicalement différents – le premier n'étant pas lié à un espace géométrique (pour ce faire nous avons besoins du *dessin* du graphe) ; le second étant par définition intégré dans un espace géométrique (Euclidien). Les trajectoires sont typiquement des courbes orientées, alors que les liens d'un graphe ne le sont pas forcément. En théorie, la plupart des algorithmes de bundling peuvent bundler des dessins de graphes ou des trajectoires. Cependant, choisir le correct algorithme de bundling pour agréger un jeu de chemins donné est guidé par des critères plus subtiles que simplement le type de données d'entrée (*e.g.* la définition des fonctions de compatibilités, la quantité d'agrégation à fournir ou le type de structure à mettre en valeur). Tous ces aspects sont importants pour comprendre et utiliser de façon optimale le bundling, comme nous allons le démontrer dans la suite de cette thèse.

Au cours de ce chapitre, nous avons établi le cadre global de visualisation dans lequel les techniques de bundling prennent racine. Nous avons introduit les différents principes et méthodes de mise en forme des différents jeux de données utilisables par le bundling : les graphes et leurs techniques de dessins (voir section 2.1) et les trajectoires. En section 5.1.1, nous avons détaillé et explicité les techniques de réduction d'occultation existantes qui nous permettrons dans les chapitres suivant d'améliorer la compréhension et la place du bundling dans l'ensemble des techniques de réduction d'occultation. Enfin, nous avons défini (et résumé en français) les principes importants et les principales définitions propres au bundling (*e.g.* **Équation du bundling**) nécessaire pour débuter notre analyse de l'espace des techniques de bundling dans les chapitres suivants. Au cours des chapitres à venir, nous exposerons le bundling sous 3 angles différents, en partant d'une taxonomie orientée type de jeu de données d'entrée (chapitre 3) vers la construction d'un pipeline unifié des méthodes de bundling (chapitre 4) et une analyse des tâches et des domaines d'application du bundling (chapitre 5).

2.1 From Data to Point/Line Visualization

Bundling aims at providing a visual simplification by spatial grouping similar lines. This process requires the ability to represent information embedded in a given data-set to be represented as a point/line visualization. In this section, we will detail two formalisms of data-sets structure and how each type of data-set can be plotted on a screen in order to be bundled.

2.1.1 Graphs

"Graphs in general form one of the most important data models in computer science", Beck et al., 2014. Formally, any relational structures consisting of a set of entities and relationships between those entities can be modeled as *graphs*. In the last two decades, graph theory and graph drawing have played a large part in the research themes of Human-Computer Interaction and Information Visualization. In this subsection, we will define and present the basis of graph drawing, specifically, how we transform a graph based data-set into a visual structure. This subsection highlights existing surveys of the field (*i.e.* Von Landesberger et al., 2011, Herman, Melançon, and Marshall, 2000, Battista et al., 1998).

Definitions

What we call *graph* refers to a set of vertices and a set of edges that connect pairs of vertices. A graph is a pair of these sets, $G = (V, E)$ where V is a set of vertices and E is a set of edges between vertices (*i.e.* $E \subset V^2$ *e.g.* an edge $e = (v_1, v_2)$ where $v_1 \in V$ and $v_2 \in V$), see Diestel, 2000. This definition can be extended by attaching attributes to vertices and/or edges to denote their type, size, or some other application related information.

The formal model of graph is further refined and classified into *undirected* and *directed* graphs (Herman, Melançon, and Marshall, 2000). A *directed* graph (or *digraph*) is defined similarly with the exception that the elements of E, called *directed edges*, are ordered pairs of vertices. In other terms, in a *directed* graph, for two vertices v_1, v_2, $(v_1, v_2) \neq (v_2, v_1)$. Conversely, for an *undirected* graph, $\forall v_1, v_2$, $(v_1, v_2) = (v_2, v_1)$. A graph containing both directed and undirected edges is called a mixed graph.

In graph theory, a (directed) *path* in a (directed) graph G is a sequence $(v_1, v_2, ..., v_n)$ of distinct vertices where $v_i \in V$ and $(v_i, v_{i+1}) \in E$. A (directed) path is a (directed) *cycle* if $(v_h, v_1) \in E$. Similarly, a directed graph that has no directed cycles is called a directed *acyclic* graph (DAG).

A *tree* is a connected undirected graph without cycle (Diestel, 2000). A Tree T is called rooted when one vertex r is distinguished as a so called *root* node: $T = (V, E, r)$. In graph drawing, we often called such rooted tree a *hierarchical graph*. However, in the formal framework of mathematical graph theory, a hierarchical graph is a DAG (meaning a node can have several paths to the root node).

Additionally, we call *compound* graph C as a pair (G,T) where $G=(V,E_G)$ and $T=(V,E_T,r)$ that share the same set of vertices, such that:

$$\forall e=(v_1,v_2)\in E_G, v_1\notin path_T(r,v_2) \quad \& \quad v_2\notin path_T(r,v_1) \tag{2.1}$$

Relationships between vertices are expressed by the tree T. Here, vertices sharing a common parent in T belong to the same "branch" (or "group") of T.

Graphs may also *change over time*, forming what we call *dynamic* graphs. *Time* dependent changes can affect the attributes of nodes and edges but also the graph structure. A dynamic graph $G(t)=(V(t),E(t)), t\in\mathbb{R}^+$. A a given time t_i, $G(t_i)$ is called a *frame*. Two different forms of dynamic graphs are known: streaming graphs and sequence graphs.

In a streaming graph $G=(V,E)$, each edge $\mathbf{e}_i\in E$ has a so-called *lifetime* $[t_i^{start},t_i^{end}]$, of duration $\lambda_j=t_i^{end}-t_i^{start}$, where $t_i^{start}<t_i^{end}$. That is, $\mathbf{e}_i$ exists only between t_i^{start} and t_i^{end}. The dynamic graph $G(t)$ thus contains all edges $\mathbf{e}_i\in E$ that are alive at t, *i.e.* for which $t_i^{start}\leq t\leq t_i^{end}$. The same holds for the streaming graph's nodes $\mathbf{v}_i\in V$. Streaming graphs can be available in an online manner – that is, one does not know upfront all moments t_i^{start} and t_i^{end}.

Sequence graphs are, as their name says, ordered sets $G=\{G^i\}$ of static graphs $G^i=(V^i,E^i)$. In contrast to streaming graphs, edges E_i do not have a lifetime, but belong to a single frame i. They typically capture a system's structure at several discrete time moments t_i. Well-known examples are the set of call graphs mined from the several revisions of software system stored in a software repository (Diehl and Telea, 2014 and Reniers et al., 2014). In practice, the frames G^i are usually not unrelated, but have nodes and edges which capture the evolution in time of the same *items*. For instance, two edges $\mathbf{e}_a^i\in E^i$ and $\mathbf{e}_b^{i+1}\in E^{i+1}$ can represent the same call relation in two consecutive frames of a software system. Such links relating information between different sequence frames can be modelled by a so-called correspondence function $c: E^i\rightarrow E^{i+1}\cup\{\varnothing\}$. Here, $c(\mathbf{e}\in E^i)$ yields an edge $\mathbf{e}'\in E^{i+1}$ which logically corresponds to $\mathbf{e}$, if such an edge exists in E^{i+1}, or the empty set, if there is no correspondence, *i.e.* if $\mathbf{e}$ disappears in the transition from G^i to G^{i+1}.

Overall, we can classify existing graphs as summarized in figure 2.1.

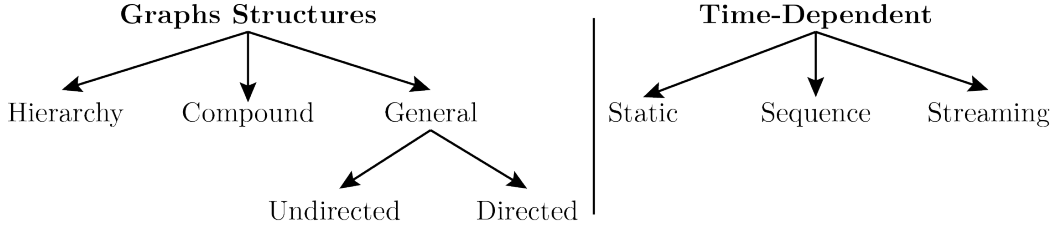


FIGURE 2.1 – Classification of graphs based on their time-dependence and graph structure.

Graph Layout for Node-link Diagrams

Here, it is important to emphasize that a graph and its drawing are quite different objects. Per definition, a graph does not have any drawing and is not embedded in any kind of Hilbertian space. "In general a graph has many different drawings" (Battista et al., 1998), and choosing the correct layout for a given graph is a recurrent research of the Graph Drawing and Information Visualization community. Overall, graph layout techniques aim at optimizing several criteria depicted in three groups in the recent survey "Visual Analysis of Large Graphs: State-of-the-Art and Future Research Challenges" by Von Landesberger et al., 2011 (see Von Landesberger et al., 2011 section 3). These three groups are:

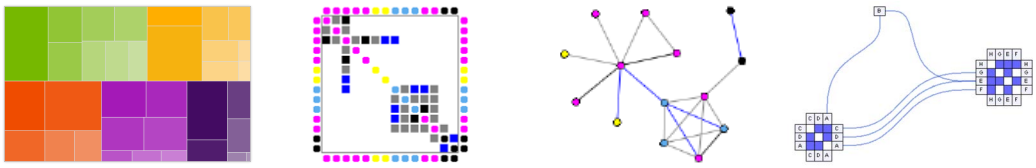
- *General criteria:* They encompass criteria such as reduction of visual clutter or maximization of space efficiency.
- *Dynamic graphs:* Maximization of display stability between time points or reduction of cognitive load.
- *Aesthetic scalability criteria:* These criteria refer to graph readability for larger graphs, *i.e.* scalability in number of vertices, edges, and graphs.

According to Von Landesberger et al., 2011, despite being important these criteria cannot be simultaneously optimized and are not sufficient to design a good layout which is usually data and/or task dependent. Therefore, multiple graph layout are fundamental to exploratory graph visualization.

A plethora of graph layout methods and algorithm exist in the literature and generally a graph layout methods is optimized for a specific type of graph (tree, coumpound, directed graph...). An extensive study of existing graph layout methods is detailled by Von Landesberger et al., 2011 in section 3. We can note a few classes of layout techniques for static graphs studied by Von Landesberger et al., 2011:

- *Space-Filling techniques:* They typically try to use the full area of the display to present the hierarchy of a graph (see figure 2.2a). Here, the spatial position of nodes are employed to display edges between nodes by using either closeness or enclosure.
- *Matrix-Representation:* These techniques displays the adjacency matrix of a given graph. Here each row and column displays a node and the links and attribute are displayed by a cell (see figure 2.2b).
- *Node-Link Techniques:* They are by far the most used layout methods for graphs. Here, the idea is to represents nodes as points and links as edges (see figure 2.2c). Node-link layout are generally embedded in 2 or 3 dimensional spaces. Many techniques exist to optimize the aforementionned layout criteria. Von Landesberger et al., 2011 distinguish *force-based*, *constraint-based*, *multi-scale*, *layered*, and *non-standard* layouts.

- *Hybrid approaches:* As their name suggests, such approaches mix two or more of the different aforementioned layout methods and/or classes. For example, it can mix node-link and matrix layout as depicted in figure 2.2d.



(A) Space-Filling (B) Matrix (C) Node-Link (D) Hybrid

FIGURE 2.2 – Four classes of graph layout techniques. Von Landesberger et al., 2011

Dynamic graphs generally use the static graphs layouts techniques in conjunction with other time-series visualization such as animation or juxtaposition. An extended survey of the layout and visualization techniques for dynamic graphs is provided by Beck et al., 2014 in "The State of the Art in Visualizing Dynamic Graphs".

2.1.2 Trail-sets

Separately, we define a *trail-set* $T = \{\mathbf{t}_i\}$ according to Phan et al., 2005 and Zhou et al., 2013. A trail-set is a collection of trails that typically describe the motion of shapes or objects in space, *e.g.* airplanes (Hurter, Tissoires, and Conversy, 2009), eye tracks (Peysakhovich, Hurter, and Telea, 2015), ships (Scheepens et al., 2011a), or persons (Nagel, Pietsch, and Dörk, 2016). As such a *trail* $\mathbf{t}_i \subset \mathbb{R}^d$ is an oriented curve in a Euclidian space. However, trails can also be curves unrelated to motion, *e.g.* polylines in a parallel coordinate plot (PCP) (Inselberg, 2009) or DTI tracts (Everts et al., 2015).

Conversely to graphs, trail-sets are in essence spatially embedded into an Euclidian space (generally 2 or 3 dimensional). Similarly to graphs, we distinguish different categories of trail-sets: undirected or directed trail-sets and time dependent ones (see figure 2.3). Note that, in graph theory, the term trail has a different meaning, *i.e*, a type of walk on a graph in which all edges are distinct (as defined by Harris, Hirst, and Mossinghoff, 2008).

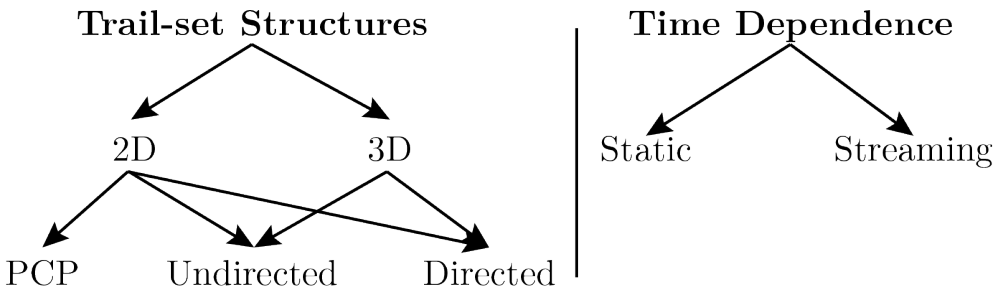


FIGURE 2.3 – Classification of trails based on their time-dependence and dimensional embedding.

2.1.3 Graph *vs* Trail-sets Differences and Similarities

Bundling literature often refers to 'edge bundling' or 'graph bundling' indiscriminately, even when the actual data being bundled are trail-sets. It is thus important to clarify both differences and similarities between (the bundling of) graphs and trail-sets.

Differences

- A graph G does *not* have a *given* spatial embedding. Only a graph *drawing* does. So, one bundles graph *drawings*, not graphs. The distinction is crucial, as the drawing $D(G)$ is an extra degree of freedom – the same G can have multiple drawings $D(G)$. Of course, one can use graph information, *e.g.* edge attributes, to influence bundling – see the discussion of compatibility function in section 2.3.1. Yet, having a layout $D(G)$ is mandatory; without it, we cannot bundle a graph as a non-spatial, abstract object.

- In contrast to graphs, a *trail-set* T is *always* spatially embedded, by definition. This embedding encodes relevant data, *e.g.*, geo-positions of vehicle movements, or variable values in a PCP. Hence, bundling a T is far more delicate than bundling a $D(G)$: Deforming the former can distort spatially-encoded information; deforming the latter only distorts decisions of the graph-drawing algorithm, but none of the raw data G.

- Both graphs G and trail-sets T can be directed or not. However, most trail-sets are directed, as they represent motion trajectories.

- Nodes $\mathbf{v}_i$ in a graph G are typically shared by multiple edges $\mathbf{e}_i$. After all, this defines a graph. In contrast, endpoints of trails $\mathbf{t}_i$ do not *have* to match. Hence, a trail-set T is a less structured dataset than a graph G.

Similarities

- Bundling both graph drawings and trail-sets can be expressed by the same formalism (as defined in section 2.3, **Bundling equation**). Of course, the meaning of the compatibility functions can be different for graphs *vs* trail-sets, but the same can hold for two use-cases featuring graphs or trail-sets.

- Given the above, the same algorithm B can be used to bundle graphs and trail-sets, *if* it delivers a visual simplification (and implicitly, path deformation) which is suitable for the problem at hand. This is well visible in many bundling papers which propose the same bundling method for both graphs and trail-sets.

- The overarching goal of bundling – reducing clutter in a path drawing so that its core structure stands out – is common to both graph drawing and trail-set bundling.

To summarize, Graphs and Trails have a different data types semantic – the former is not spatially embedded data (for that, we need a graph *drawing*); the latter is spatial by definition. Trails are typically directed, while graphs may not be. Many bundling algorithms can technically bundle both graph drawings and trail-sets. However, the choice of a suitable algorithm is driven by more subtle *application* factors such as compatibilities definitions, amount of bundling deformation, and types of emphasized patterns. All these aspects matter for understanding and optimally using bundling, as we shall see next.

2.2 Taxonomy of Clutter Reduction

Dense and complex data-set (*e.g.* be it a graph and one of its drawing or a trail-set) often suffer from occluded items which hinder insight retrieval. This problem is more and more encountered by the ability of governments and commercial organizations to generate vast amount of data. Here, the problem in large data-visualization is simply and precisely stated by Ellis and Dix, 2007: "In essence, too much data on too small an area of display will result in visual clutter, which in turn diminishes the potential usefulness of the visualization, especially when the user is exploring the data rather than posing a specific question".

Over the years, several techniques have proposed to visually simplify cluttered representation. As such, it is important to have a clear understanding of existing techniques in order to understand the role of bundling in the context of clutter reduction. Especially in the context of visual exploration where the user has no specific question regarding the input data-set.

In a survey of 2007, Ellis and Dix, 2007 formalized a taxonomy to understand, express and compare various clutter reduction techniques. In this section, we will summarize the definitions, criteria and results that outcomes from (Ellis and Dix, 2007) survey to understand how bundling integrates in regards to existing techniques. Throughout this section, we will follow the organization of Ellis and Dix, 2007's taxonomy. We start by introducing their clutter reduction classification and then their clutter reduction criteria. Finally, we present their comparison of each classes of clutter reduction techniques based on the aforementioned criteria to formalize a clutter reduction taxonomy.

2.2.1 A Classification of Clutter Reduction Techniques

According to Ellis and Dix, 2007, clutter reduction techniques can be classified into three different categories as detailed next and summarized in table 2.1;

- *Appearance*: is composed of techniques that tends to affect the look of the data item. In this category, we list sampling, filtering, changing point size and opacity and clustering. We note that sampling and filtering differs semantically; sampling is an act of random selection of a subset of the data-set whereas filtering is a

selection of items that meet user-defined criteria. Moreover, Ellis and Dix, 2007 do not make any structure based distinction in the clustering process. Here, the clustering can be done in the data-space (*i.e.* finding clusters of data that share similar attributes) or in a Euclidean space (*i.e.* finding clusters that are relatively closed in a Euclidean space). This distinction can be important for graphs where we distinguish graph clustering and graph drawing clustering. The first being in a graph topological space and the later being in a chosen Euclidean space.

- *Spatial Distortion*: This category includes techniques that displace points or lines in some way. Spatial distortion encompass point/line displacement (*e.g.* adjusting the position of points and lines), topological distortion (*e.g.* stretching non-uniformly or uniformly the geometrical space), space-filling (similar to the graph layout methods where we aim at occupying the maximum space available in the visualization screen), pixel-plotting (*e.g.* plotting data-items as single pixels to pack as much data as possible), and dimensional reordering (multiplexing n dimensions in m dimension with $m \leq n$, *e.g.* parallel coordinates).

- *Temporal*: encompass all animation techniques. This can be for example the temporal evolution of a graph drawing frame by frame.

Table 2.1 – Classification of clutter reduction techniques according to Ellis and Dix, 2007.

	Clutter Reduction Technique	**Examples**
Appearance	sampling	Dix and Ellis, 2002 Rafiei, 2005
	filtering	Ahlberg, Williamson, and Shneiderman, 1992 Stone, Fishkin, and Bier, 1994
	change point size	Derthick et al., 2003 Woodruff, Landay, and Stonebraker, 1998
	change opacity	Johansson et al., 2006 Wegman and Luo, 1997
	clustering	Fua, Ward, and Rundensteiner, 1999 Kreuseler and Schumann, 1999
Spatial Distortion	point/line displacement	Waldeck and Balfanz, 2004 Wong, Carpendale, and Greenberg, 2003
	topological distortion	Carpendale, Cowperthwaite, and Fracchia, 1995 Sarkar et al., 1993
	space-filling	Bederson, Shneiderman, and Wattenberg, 2002 Stasko and Zhang, 2000
	pixel-plotting	Keim, 2000 Rao and Card, 1994
	dimensional reordering	Peng, Ward, and Rundensteiner, 2004
Temporal	animation	De Bruijn and Spence, 2000 Johansson et al., 2006

2.2.2 A Definition of Clutter Reduction Criteria

To compare the defined categories of clutter reduction techniques, Ellis and Dix, 2007 introduce eight clutter reduction criteria. These criteria where defined through an extended survey of the literature and the justification of each criteria is detailed in section 4 of their paper. To summarize, their defined criteria are:

- *Avoids overlap*: This criteria is self explanatory.
- *Keeps spatial information*: self explanatory, the technique conserve the spatial information between points and lines.
- *can be localized*: For techniques that can be applied only in a part of the visualization (*e.g.* only in denser areas).
- *is scalable*: Techniques that can cope with very large data-sets
- *is adjustable*: Allow the user to control the amount of produced de-cluttering.
- *can show point/line attribute*: Self explanatory criterion.
- *can discriminate points/lines*: Self explanatory criterion.
- *can see overlap density*: In the case of overplotting, the visualization can show that there is some overlap between data-items in some specific areas.

2.2.3 Taxonomy

Finally, Ellis and Dix, 2007 define their taxonomy of clutter reduction techniques by comparing each defined techniques with each defined criterion. The resulting taxonomy is transposed in table 2.2. Techniques that meet the criteria have a ✓and those that don't have a ✗. Some special cases are marked as $^+$ or with some text.

TABLE 2.2 – Ellis and Dix, 2007's clutter reduction taxonomy.

Key: ✓satisfies criterion; ✗ does not satisfy criterion; $^+$ some exception/special cases

More complex cases: possibly/partly

		1	2	3	4	5	6	7	8	9	10	11
		sampling	filtering	point size	opacity	clustering	point/line displacement	topological distortion	space-filling	pixel-plotting	dimensional reordering	animation
A	avoids overlap	possibly	possibly	possibly	partly	possibly	✓$^+$	possibly	✓$^+$	✓$^+$	partly	✓$^+$
B	keeps spatial information	✓	✓	✓	✓	partly	7$^+$	possibly	✓$^+$	possibly	✓	✓
C	can be localized	✓	✓	✓	✓	✗$^+$	✓	✓	✗	✗	✗	✓$^+$
D	is scalable	✓	✓	✗	✗$^+$	✓	✗	✗	✗	✗	✗	✓$^+$
E	is adjustable	✓	✓	✓	✓	✓	possibly	✓	✗$^+$	✗$^+$	✓	✓$^+$
F	can show point/line attribute	✓	✓	✓	✗$^+$	partly	✓	✓	✓	✓	✓	✓
G	can discriminate points/lines	✗	✗	possibly	✓$^+$	✓$^+$	possibly	✗	✗	✗	✗	✗
H	can see overlap density	✗	✗	✗	✓$^+$	possibly	✗	✗$^+$	✗$^+$	✗$^+$	✗	✗$^+$

Each problematic technique/criterion assessment is justified by Ellis and Dix, 2007 in section 5. In the following, we will re-explain some assessment we deem useful to our analysis of bundling techniques in the context of clutter reduction.

- *Opacity*

  A4 If change in opacity cannot avoid overlap, it can reveal a small number of underlying or partially overlapping points.

  D4 Healey, Booth, and Enns, 1995 suggest that opacity is only useful when up to five items overlap.

  F4 Reducing the opacity diminishes the significance of the data, especially color.

  G4 Opacity seems to be used with good efficiency in parallel coordinates plots to discriminates overlapping lines especially when combined with clustering (see Johansson et al., 2006, Fua, Ward, and Rundensteiner, 1999).

  H4 When correctly configured, opacity can lead to meaningful density maps (Fekete and Plaisant, 2002 and Wegman and Luo, 1997). In conjunction with aggregation, opacity can be used to promote a visual indication of the overlap density.

- *Clustering*

  A5 Here, clustering consist in the reduction of the number of data items in order to simplify the plot. As such, it can be used to avoid over-plotting by aggregating points or lines.

  B5 By default, clustering loses individual spatial information but it can show aggregate values of clusters (via color, shading or/and opacity).

  C5 Ellis and Dix, 2007 placed a cross against this considering that clustering based on arbitrary similarity measures does not necessarily maintain any spatial locality. However, if the clustering process is based on spatial attributes alone, then it would make sense if the typical cluster diameter was small enough is regards to the locally impacted area.

  F5 Clustering can show aggregate values.

  G5 Clustering techniques can be used to detect outliers as well as allow the emergence of groups (Novotny and Hauser, 2006).

  H5 Clustering does not inherently show the overlap density, but the aggregate value can be displayed (see H4).

- *Point/line displacement*

  A6 Some point/line displacements techniques are designed specifically to locally reduce (rather than avoid) overlap. Such examples are Treemaps (Bederson, Shneiderman, and Wattenberg, 2002), TableLens (Rao and Card, 1994) and EdgeLens (Wong, Carpendale, and Greenberg, 2003).

B6 The amount of point/line displacement often depends on the density of data. The higher the number of overlapping points/lines, the greater the spatial distortion will be. However, Ellis and Dix, 2007 note that the amount of information loss is not necessarily directly dependent on the amount of displacement as relative positions are perhaps a more determining factor.
E6 According to their survey only a few applications allow the user to control the amount of displacement. In the rest of existing applications, the displacement is function of the data density.
G6 Wong, Carpendale, and Greenberg, 2003 claim that curving lines in Edge-lens helps disambiguate the connected nodes of a graph.

Considering the wide scope of Ellis and Dix, 2007's taxonomy of clutter reduction techniques, bundling appears to be a technique to reduce clutter in point/line visualizations where data are already embedded in a Euclidean space and visualizable. In this context, bundling proposes conserving the topology of the visualization space as well as the position of the points and conversely to distort the links.

2.3 Definition of the Bundling Operator

The bundling landscape is a vast and complex one with plethora of various implementations and input types of data-sets. In an effort to unify and improve the understanding of the bundling landscape, we start by introducing the main definitions and notations we will work on throughout this thesis.

2.3.1 Bundling Objectives

Large-scale, strongly connected, real-world graphs have many more edges than nodes. Hence, classical straight-line node-link drawings thereof quickly become ineffective for most, if not all, tasks they address. This problem is often referred in the literature either by edge congestion (see the work of Carpendale and Rong, 2001, Wong, Carpendale, and Greenberg, 2003 and Von Landesberger et al., 2011), visual clutter (Ellis and Dix, 2007, Burch et al., 2011 and Nguyen, Eades, and Hong, 2013) or the hairball problem (Schulz and Hurter, 2013). Bundling is one class of methods that aims to alleviate this problem, along graph clustering and interaction, as further outlined in Sec. 2.3.4.

Informally put, bundling trades clutter for overdraw as detailled by Telea and Ersoy, 2010. However, although there are tens of papers on bundling in the literature, there is – interestingly enough – no formal definition of bundling. Our work argues that such a definition is needed to be able to understand the process, compare methods, reason about guarantees and limitations, and push further research. We propose such a definition next.

2.3.2 Bundling Definition

Let $G = (V, E \subset V \times V)$ be a graph with nodes $V = \{\mathbf{v}_i\}$ and edges $E = \{\mathbf{e}_i\}$. Let d be the dimensionality of the drawing space where the bundled visualization will occur, which is usually 2 or 3. Separately, let $T = \{\mathbf{t}_i\}$ be a so-called trail-set as defined by Phan et al., 2005 and Zhou et al., 2013. A trail $\mathbf{t}_i \subset \mathbb{R}^d$ is an oriented curve. Trails typically describe the motion of shapes or objects in space, *e.g.* airplanes (Hurter, Tissoires, and Conversy, 2009), eye tracks (Peysakhovich, Hurter, and Telea, 2015), ships (Scheepens et al., 2011a), or persons (Nagel, Pietsch, and Dörk, 2016). However, trails can also be curves unrelated to motion, *e.g.* polylines in a parallel coordinate plot (PCP) (Inselberg, 2009) or DTI tracts (Everts et al., 2015). Note that, in graph theory, the term trail has a different meaning, *i.e*, a type of walk on a graph in which all edges are distinct (as defined by Harris, Hirst, and Mossinghoff, 2008). Let $\mathcal{G}$ and $\mathcal{T}$ be the spaces of all graphs, respectively trail-sets.

The key unifying element of graphs and trail-sets is a so-called drawing operator D. For graphs, $D : \mathcal{G} \rightarrow (\mathbb{R}^d, \mathbb{R}^d)$ is a typical graph layout, or graph drawing, method (Battista et al., 1998). By analogy, let $D(\mathbf{e}_i)$ and $D(\mathbf{v}_i)$ be the embedding (drawing) of edges $\mathbf{e}_i \subset E$, respectively nodes $\mathbf{v}_i \subset V$. For trails, the drawing operator is the identity function, *i.e.*, $D(\mathbf{t}_i) = \mathbf{t}_i$, since trails are *already* spatially embedded. Let P denote either a graph G or a trail-set T, called a *path-set*, and $D(P)$ the drawing thereof. A path $\mathbf{p} \in P$ is thus either a graph edge $\mathbf{e}$ or a trail $\mathbf{t}$. Paths can have n additional data attributes, *e.g.* direction, weight, timestamps, name, or type as shown in the work of Peysakhovich, Hurter, and Telea, 2015 and Diehl and Telea, 2014. Hence, a path $\mathbf{p}$ can be seen as $n + d$ dimensional, with n data dimensions and d spatial dimensions.

Let $\mathcal{D} \subset \mathbb{R}^d$ be the space of all path drawings $D(P)$. Let $B : \mathcal{D} \rightarrow \mathcal{D}$ be an operator denoting the *bundling* of a path-set; and finally let $B(D(\mathbf{p}))$ denote the curve representing the bundling of path $\mathbf{p}$. B is a bundling method if

$$\forall(\mathbf{p}_i, \mathbf{p}_j) \in P \times P | \mathbf{p}_i \neq \mathbf{p}_j \wedge \kappa(\mathbf{p}_i, \mathbf{p}_j) < \kappa_{max} \rightarrow$$
$$\delta(B(D(\mathbf{p}_i)), B(D(\mathbf{p}_j))) \ll \delta(D(\mathbf{p}_i), D(\mathbf{p}_j)). \qquad \textbf{Bundling equation}$$

Here, δ is a distance metric between $\mathbb{R}^d$ curves, *e.g.* the Hausdorff distance defined by Berg et al., 2010. $\kappa : P \times P \rightarrow \mathbb{R}^+$ is a so-called compatibility function that captures how dissimilar paths are. That is, low κ values indicate very similar paths, and high κ values indicate dissimilar paths, respectively. κ must, in any case, account for spatial similarity in $D(P)$, *i.e.*, when $\kappa(\mathbf{p}_i, \mathbf{p}_j)$ is small, then $\delta(D(\mathbf{p}_i), D(\mathbf{p}_j))$ is small too. In addition, κ can incorporate any of the other n path data-attributes mentioned above, *i.e.*, it can model distance in the $n + d$ layout-plus-attribute space of paths $\mathbf{p}$ (Peysakhovich, Hurter, and Telea, 2015). Only paths more similar than a threshold κ_{max} should be bundled – otherwise the input drawing $D(P)$ can get too severely distorted to be of any use. Simply put, **Bundling equation** states that the bundled drawings $B(D(\mathbf{p}_i))$ of highly compatible paths are spatially much closer than their unbundled drawings $D(\mathbf{p}_i)$.

2.3.3 Bundling Requirements

To be able to discuss and compare bundling methods in the following sections (sections 3 and 4), we need to define some general requirements on bundling techniques. Distilled from the plethora of reviewed bundling techniques, these requirements are:

Input: A bundling method B accepts a path-set drawing $D(P)$ as input – that is, a set of spatial positions connected by curves. This is in contrast with graph layout methods which typically *compute* such positions from a graph G.

Output: The output $B(D(P))$ of a bundling method is a path drawing having the same endpoints as the input $D(P)$. No bundling method that we are aware of (except Yu et al., 2012 see sec. 3.2.1) changes path endpoints, as these are assumed to contain important information.

Bundle definition: Bundling methods do not explicitly define what a *bundle* is. Bundles are defined implicitly, as sets of paths that share sufficient similarity so as to be represented as a compact graphical shape. Interestingly, this matches the definition of clustering by Jain, Murty, and Flynn, 1999 or image segmentation by Szeliski, 2010 – a segment or cluster shares precisely the same properties. So, bundling can be seen as a clustering or segmentation of the drawing $D(P)$. A constraining criterion in the above is that bundles are assumed to be spatially thin (to reduce edge congestion as explained above) and reasonably low-curvature (so as to be easily visually traceable, see Sec. 5.1).

Density sharpening: All bundling methods we know of aim, implicitly or explicitly, to *sharpen* the spatial edge density ρ of $D(P)$, *i.e.*, the number of paths $D(\mathbf{p})$ drawn per unit screen space. Simply put: In areas where ρ is low (few paths), these paths are shifted to make extra empty space, which declutters the overall image; paths are shifted to close regions where ρ is already high (many paths exist). This essentially trades off clutter for overdraw – the number of intersections of *bundles* should be significantly lower than the number of intersections of input *paths*. This matches the known principle of ink minimization in information visualization first introduced by Tufte, 1992. Overall, this makes visual end-to-end tracing of *bundles* easier than end-to-end tracing of individual *paths*.

Scalability: Bundling becomes interesting for large path-sets (tens of thousands up to millions of paths). For small(er) path sets, classical graph drawing methods suffice (Gansner and North, 2000; Herman, Melançon, and Marshall, 2000; Von Landesberger et al., 2011), as there are too few edges to create the visual clutter discussed above. Bundling *must* thus cope with large path-sets, otherwise its reason to be is not warranted.

2.3.4 From Graph Simplification to Bundling

Graph and trail-set bundling have a long history, stemming from applications related to the visual simplification and clutter reduction in the drawing of graphs using node-link diagram metaphors. We outline next early efforts in the area and how they relate to what is currently understood by bundling.

Graph Simplification

An early approach to simplification and clutter reduction was to simplify the structure of G, by creating a smaller G' (with fewer nodes and/or edges) that captures the main structure of G. Edge concentration, introduced by Newbery, 1989, is such a method used to simplify Sugiyama-style layouts (Sugiyama, Tagawa, and Toda, 1981). Here, edge sets having the same set of start and end nodes are replaced by a so-called concentration node, which effectively presents a simple form of polyline-style bundling. Many other graph simplification methods exist, as surveyed by Schaeffer, 2007. However, such methods display a smaller and/or different graph than G, and as such specific nodes of potential interest are omitted in the drawing. Finding these requires additional interactive exploration methods such as the one proposed by Abello, Van Ham, and Krishnan, 2006, Archambault, Munzner, and Auber, 2008 or Archambault, Purchase, and Pinaud, 2010. Moreover, most graph simplification methods do not fit the scope of bundling, as currently understood by the infovis or graph drawing communities, or as defined in section 2.3, so we do not explore these further.

Graph Drawing Simplification

A separate way was proposed by methods that change the graph drawing $G(D)$, as opposed to the graph G, to reduce clutter. These can be seen as bundling precursors. For example, Brandes and Wagner, 1998 use a mix of straight lines and Bézier splines to draw a general undirected graph where nodes are train stations and edges are train routes, respectively. Spline control points of spatially close train routes are grouped together. This also introduces the concept of a 'control mesh' which is computed to further bundle edges. Dickerson et al., 2003 introduce the notion of confluent drawing, where "groups of edges are merged together and drawn as tracks". Compared to modern bundling methods, this approach can only handle a subset of all possible general graphs. Phan et al., 2005 extend the idea by hierarchically clustering a set of nodes $\mathbf{v} \in V$, positions $D(\mathbf{v})$, and directed edges $\mathbf{e} \in E$ of a graph (V, E), yielding organic, branch-rich images of the graph structure which are similar to early hand-drawn Sankey diagrams showing flows over a geographical map (Figure 1.2).

Interaction

A third way to alleviate clutter is to use interaction. For example, Dunne and Shneiderman, 2013, Abello, Van Ham, and Krishnan, 2006, Van Ham and Perer, 2009 and Henry, Fekete, and McGuffin, 2007 use interaction to navigate a simplification hierarchy. In different approaches, Wong, Carpendale, and Greenberg, 2003, Gansner, Koren, and North, 2005, Tominski et al., 2006 and Wong and Carpendale, 2007 interactively declutter focused areas of interest by removing or bending drawn edges. In contrast to bundling, methods offer a local, on-demand, decluttering and simplification, rather than a global, automatic, one.

In 2006, several papers that introduce bundling as we understand it today were published. Gansner and Koren, 2006 improved circular layouts by clustering edges (drawn as straight-lines) based on spatial proximity, and next bending edges in a cluster towards the cluster's centroid line. Qu, Zhou, and Wu, 2006 proposed bundling for general straight-line graph drawings using NURBS splines whose control points are constructed from a Delaunay triangulation of the graph nodes. Most notably, Holten, 2006 presented hierarchical edge bundling which was able, for the first time, to bundle compound graphs of thousands of edges and having arbitrary node layouts. With this, the age of modern bundling has begun.

2.4 Summary

In this chapter, we have established the overall visualization framework from which bundling techniques take their roots. We have introduced the various principles and layout methods of the data-set structures bundling works on: Graphs and their drawing and trail-sets. In section 2.2, we detailed and expressed existing clutter reduction techniques that will allow us in the following chapter to improve the understanding and the place of bundling among the set of clutter reduction techniques. Finally, we defined the necessary principles and the main bundling definition (*e.g.* **Bundling equation**) to start our analysis of the bundling landscape in the next chapters through 3 different angles; From a data-based taxonomy approach (chapter 3), towards a unified framework of bundling methods (chapter 4) and the tasks and applications of bundling (chapter 5).

Chapter 3

A Data-based Taxonomy of Bundling Methods

In this chapter we propose a *data-based* taxonomy that organizes bundling methods along the type of data they work on – *i.e.* graphs *vs* trails. Although previous bundling taxonomies were solely based upon the technical differences between bundling methods as depicted by Zhou et al., 2013, we argue that our taxonomy is more clear-cut than previous ones. Moreover, we believe our taxonomy helps both researchers and users to understand the bundling landscape without having to dive into the technicalities of specific methods. As outlined in section 2.1, a recent survey by Von Landesberger et al., 2011 proposes a different data-based taxonomy, in this section we will highlight how our taxonomy of bundling methods integrates within the aforementioned work.

Throughout this chapter, we organize bundling methods based on the type of *data* they work on. At the highest level, such data can be split into *graphs* (section 3.1) and *trail-sets* (section 3.2). Further taxonomy levels refine graphs and trail-sets based on additional data characteristics, such as type of graph, direction information, time-dependency, and dimensionality d of the drawing space as presented in chapter 2. Our proposed taxonomy is detailed next and summarized at the end of the chapter in table 3.2 and figure 3.15.

3.0 Résumé (Français) Taxonomie des Méthodes de Bundling

Dans ce chapitre, nous proposons une taxonomie des méthodes de bundling fondée sur la structure des données d'entrée (*i.e.* graphes ou trajectoires). Bien qu'il existe d'autres taxonomies des méthodes de bundling purement fondées sur les différences techniques entre ces dernières, comme illustré par Zhou et al., 2013, nous démontrons qu'une taxonomie fondée sur les structures d'entrée est plus claire et précise. De plus, cette approche "structure d'entrée" permet d'aider les utilisateurs à comprendre les techniques de bundling sans pour autant devoir aborder les spécificités de chaque méthode.

Ainsi, au cours de ce chapitre, nous organisons les méthodes de bundling en fonction du type de données qu'elles utilisent. Au plus haut niveau, les structures d'entrée peuvent être scindées entre les *graphes* (section 3.1) et les *trajectoires* (section 3.2). Les différents niveaux de notre taxonomie raffinent le type de structure de donnée (graphes ou trajectoires) en fonction de caractéristiques additionnelles telles que le type de graphe, les informations de direction des liens, les dépendances temporelles ou la dimension d de l'espace de dessin comme définie et présentée dans le chapitre 2. Notre proposition de taxonomie est détaillée en langue anglaise et résumée dans le tableau 3.1 et la figure 3.1 ci-après.

Le tableau 3.1 permet au lecteur de trouver une liste de méthodes adaptées en fonction du type de jeu de données d'entrée (*e.g.* pour un dessin de graphe orienté, notre tableau suggère au lecteur d'employer les méthodes de bundling de Selassie, Heller et Heer, 2011 et Lambert, Bourqui et Auber, 2010b). De plus, la figure 3.1 procure un point de comparaison visuel des résultats de différents algorithmes de bundling sur un jeu de données test (un dessin de graphes de migrations aux Etats-Unis). Pour exemple, avec la figure 3.1 nous pouvons comparer les styles de rendus visuels de FDEB (Holten et Van Wijk, 2009) et GBEB (Cui et al., 2008) pour une agrégation non-directionnelle (*i.e.* isotrope, voir figures 3.15b et 3.15g). De même, nous pouvons comparer les rendus visuels des méthodes de bundling directionnelles, exemple CUBu (Klein, Van Der Zwan et Telea, 2014) et ADEB (Peysakhovich, Hurter et Telea, 2015), figures 3.15k,l et figure 3.15m.

Dessins de graphes	Statiques	Hiérarchiques, cycliques		Holten, 2006 Telea et Auber, 2008 Holten et Van Wijk, 2008 Telea et Ersoy, 2010 Pupyrev, Nachmanson et Kaufmann, 2010 Trümper, Döllner et Telea, 2013
		Génériques	isotropes	Holten et Van Wijk, 2009 Cui et al., 2008 Lambert, Bourqui et Auber, 2010b Gansner et al., 2011 Nguyen, Hong et Eades, 2011 Luo et al., 2012
			orientés	Selassie, Heller et Heer, 2011 Lambert, Dubois et Bourqui, 2011
			cartes de flux	Phan et al., 2005 Buchin, Speckmann et Verbeek, 2011b Buchin, Speckmann et Verbeek, 2011a
			confluent drawing	Dickerson et al., 2003 Dickerson et al., 2005 Buchin, Speckmann et Verbeek, 2011b Buchin, Speckmann et Verbeek, 2011a Nocaj et Brandes, 2013 Bach et al., 2017
		3D		Giereth, Bosch et Ertl, 2008 Caserta, Zendra et Bodénes, 2011 Böttger et al., 2014
	Dynamiques	Sequentiels		Hurter, Ersoy et Telea, 2013 Hanjalic, 2013 Hurter et al., 2014a
		Continus		Nguyen, Eades et Hong, 2012
Jeux de trajectoires	Statiques	2D	isotropes	Ersoy et al., 2011 Hurter, Ersoy et Telea, 2012 Zwan, Codreanu et Telea, 2016
			orientés	Peysakhovich, Hurter et Telea, 2015 Moura, 2015 Lhuillier, Hurter et Telea, 2017a
			coordonnées parallèles	McDonnell et Mueller, 2008 Telea et Auber, 2008 Zhou et al., 2008b Heinrich et al., 2012 Palmas et al., 2014 Palmas et Weinkauf, 2016
		3D		Lambert, Bourqui et Auber, 2010a Yu et al., 2012 Everts et al., 2015
	Dynamiques (continus)			Hurter, Ersoy et Telea, 2013 Hurter et al., 2014a

TABLE 3.1 – Taxonomies des algorithmes de bundling fondés sur le type de structure d'entrée. Pour des contraintes d'espace, seules les principales méthodes de chaque classe sont listées. Les articles de bundling présentant une application du bundling sans introduire de nouvelles techniques ne sont pas listés ici.

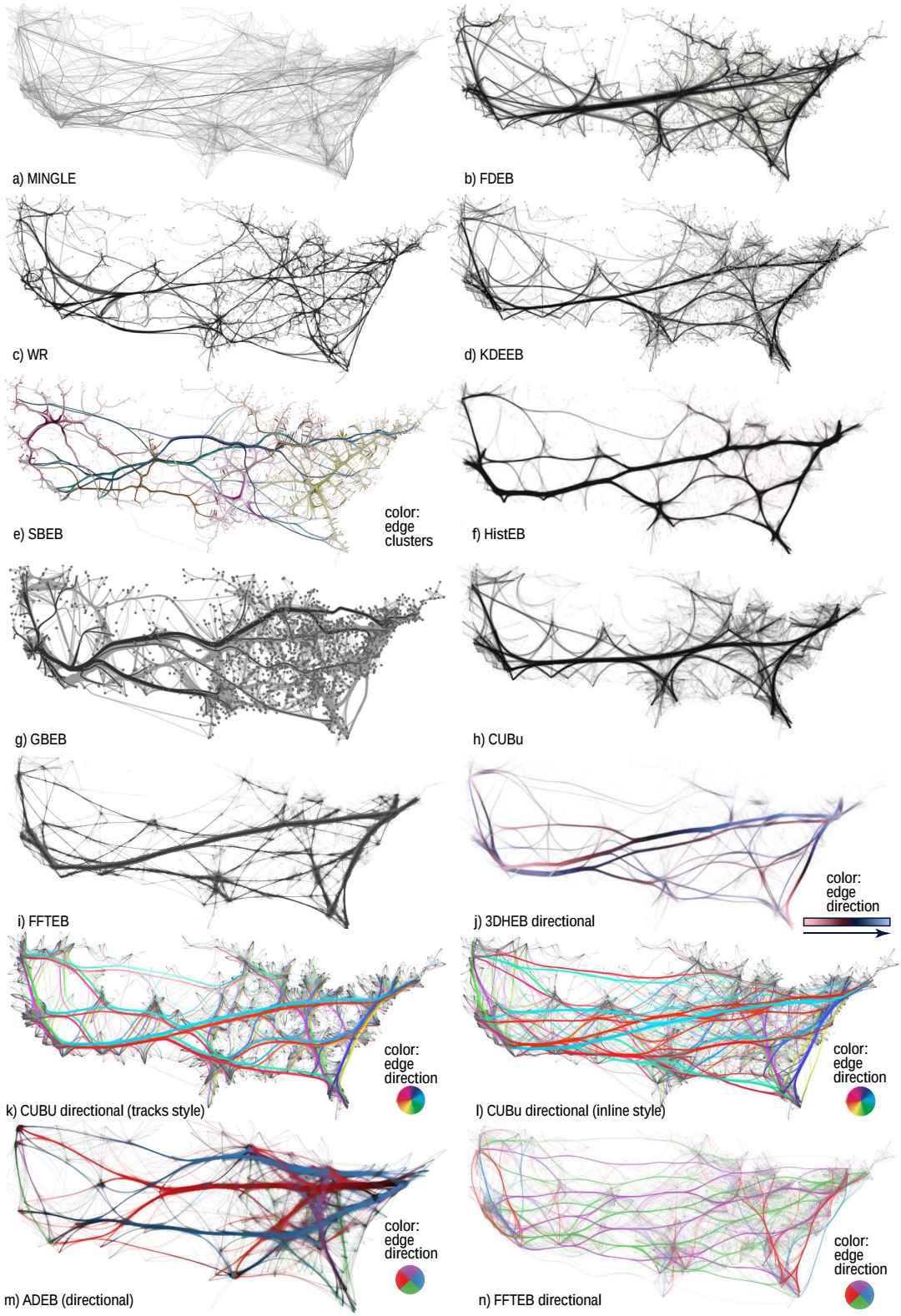


FIGURE 3.1 – Méthodes de bundling génériques isotropes (a-i) et orientés (j-n), jeu de données des migrations aux Etats-Unis ($|N|$=1715,$|E|$=9780). Voir section 3.1.1.

Notre taxonomie démontre que les méthodes de bundling couvrent un large spectre de structures d'entrée : dessins de graphes (arbres, cycles, graphes directionnels acycliques, graphes génériques orientés ou non), trajectoires bidimensionnelles (mouvements de véhicules, tracés oculaires, iso-lignes) et tridimensionnelles (mouvements de véhicules, fibres DTI). Toutes ces structures d'entrée peuvent être agrégées selon des compatibilités fondées sur un ou plusieurs attributs ou non, et leurs dépendances temporelles ou non. Ainsi, nous pensons que la plupart des structures potentiellement utilisables pour réaliser de la simplification visuelle par agrégation de liens sont couvertes par les méthodes existantes et détaillées par notre taxonomie.

D'un point de vue utilisateur, notre taxonomie montre qu'il est possible de choisir parmi la vaste diversité des techniques de bundling en fondant son choix simplement sur le type de structure de données sur lesquelles l'utilisateur souhaite appliquer le bundling. Et ce choix est possible sans forcer l'utilisateur à devoir plonger dans les détails et spécificités techniques de différents algorithmes de bundling.

Cependant, notre taxonomie ne permet pas aux utilisateurs de comparer techniquement l'implémentation des différents algorithmes de bundling. De ce fait, ce manque entrave l'accomplissement de certaines tâches des utilisateurs et des développeurs souhaitant implémenter les techniques de bundling. Pour les utilisateurs, ce manque les empêche de comprendre la paramétrisation des algorithmes de bundling et de comparer les résultats d'un dessin agrégé. De manière similaire, ce manque empêche les développeurs d'implémenter et optimiser les différents algorithmes. Afin de résoudre ces défis, dans le chapitre suivant, nous nous focaliserons sur la formalisation de l'espace des implémentations et techniques existantes.

3.1 Graph Bundling Methods

Graph bundling methods expect as input a graph drawing $G(D)$. Edges $D(\mathbf{e} \in G)$ in such drawings are typically straight lines. When this is not the case, edge drawings do carry information, so deforming them by bundling should be done with great care. Such cases fit better in the class of trail-set bundling, which is discussed separately (section 3.2).

3.1.1 Static Graphs

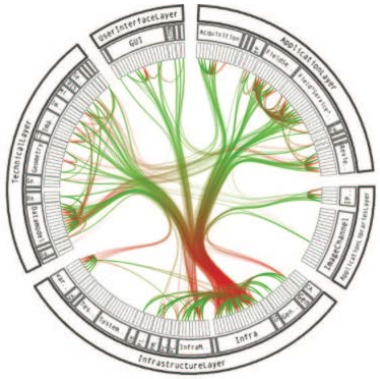
Static graphs have nodes V and edges E which do not change in time. For such graphs, bundling methods can be further classified as follows.

Hierarchical Compound Graphs

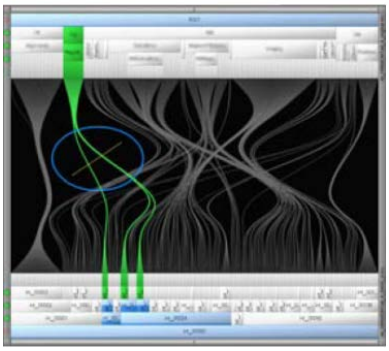
Hierarchical compound graphs are graphs $G = (V, E)$ where V are the leafs of a separate tree $T = (V, E, r)$. Such graphs are often encountered in relational datasets whose items can be organized via a hierarchy, such as dependencies between software components (Diehl, 2007). Edges in E are typically called associations.

The most famous bundling method for hierarchical compound graphs is Hierarchical Edge Bundles (HEB, Holten, 2006). HEB draws the tree T using *e.g.* a radial (also called a chord diagram, Munzner, 2014), balloon, or treemap layout. In case of the radial layout (figure 3.2a), the hierarchy T is shown by a technique known as icicle plots (Kruskal and Landwehr, 1983). This yields a tree drawing $D(T)$ where nodes $\mathbf{v} \in E$ that have a close common ancestor (*e.g.*, father or grandparent) are close to each other. Next, edges $\mathbf{e} \in E$ are drawn as B-splines whose control points are the node positions in $D(T)$ – in other words, $D(T)$ serves as a bundling control mesh (see section 2.3.4). Since control points are shared, the drawn (curved) edges get closer to each other than the original straight-line edges, thus the second part of our **Bundling equation** is respected.

Given the relation between node closeness in the hierarchy T and their placement in $D(T)$, association edges starting or ending in the same subtree of T are bundled together – in other words, the compatibility function $\kappa(\mathbf{p}_1, \mathbf{p}_2)$ (**Bundling equation**) reflects how close the endpoints of two edges $\mathbf{p}_1$ and $\mathbf{p}_2$ are in T. HEB is very simple to implement and scales very well with the sizes of G and T. HEB has been used in many applications in software engineering (Holten and Van Wijk, 2008; Cornelissen et al., 2008; Diehl and Telea, 2014; Reniers et al., 2014), social sciences (Kienreich and Seifert, 2010; Jia, Garland, and Hart, 2011), web ontologies (Hop et al., 2012), text data (Collins and Carpendale, 2007), and life sciences (Böttger et al., 2014; Assaf and Pasternak, 2008; Everts et al., 2015). HEB enhancements that simplify the bundled drawings such as the work of Telea and Ersoy, 2010 (figure 3.2c) are further discussed in section 4.3.2.

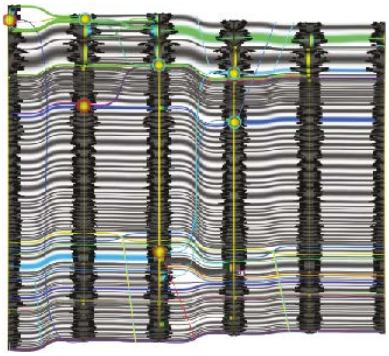


(A) HEB
(Holten, 2006)



(B) Hierarchy comparison
(Holten and Van Wijk, 2008)

(C) IBEB
(Telea and Ersoy, 2010)



(D) CodeFlows
(Telea and Auber, 2008)

FIGURE 3.2 – Hierarchical graphs bundling methods.

The hierarchy T required by HEB can be part of the input data or can be constructed from this data. For the latter, Jia, Garland, and Hart, 2011 use bottom-up graph and data clustering techniques (Schaeffer, 2007; Jain, Murty, and Flynn, 1999) to identify strongly-related node groups, thereby extending HEB to general attributed graphs. In other words, when T is given, κ reflects its structure (as explained earlier in section 2.3); while, when T is computed, this is done based on a given κ defined on the data. Other HEB extensions include the comparison of two hierarchies T_1 and T_2. Here, association edges E link leaf pairs in T_1 and T_2 rather than leafs in the same tree. Applications include the comparison of two (Holten and Van Wijk, 2008, Figure 3.2b) or multiple (Telea and Auber, 2008) software hierarchies (in a kind of 'structural diff' metaphor, figure 3.2d), where T_1, T_2 are given by the software structure; comparison of program execution traces (Trümper, Döllner, and Telea, 2013); and linking related items in coordinated multiple views to show multiple relations and datasets; here, the hierarchies T_i are computed by similarity-based node clustering, as discussed earlier. HEB was also used for 3D bundling, further detailed in section 3.1.1. Image-based simplification of HEB drawings (figure 3.2c) is further discussed in section 4.3.2.

Pupyrev, Nachmanson, and Kaufmann, 2010 improve the classical Sugiyama-style graph drawing algorithm (Sugiyama, Tagawa, and Toda, 1981) for directed acyclic graphs (DAGs) by bundling the curved edges created by Sugiyama in case these have the same start and end nodes. Hence κ reflects the similarity of endpoints of edges. The method is tested on relatively small graphs (tens of nodes). Although this method, strictly speaking, requires a DAG (which is more general than the hierarchical compound graphs required by HEB), we put this method in the same class as HEB, since the Sugiyama algorithm operates by extracting a tree from the input DAG.

Hierarchical compound graph bundling methods are arguably the most successful and best known use-case for graph bundling. Key to this is the ability of bundling to summarize groups of similar relations (edges) *and* the fact that a hierarchy allows a simple, consistent, and scalable way to compute and/or encode the similarity κ. However, the quality of drawings produced by such methods visibly depends on the way the tree T is laid out, as this next influences how bundles are routed. Yet, most papers in this area comment little on different ways to lay out T, beyond the fact that standard force-directed or radial tree-drawing algorithms (Battista et al., 1998) can be used. As an exception, the original HEB method (Holten, 2006) and several of its refinements (Reniers et al., 2014; Reniers, Voinea, and Telea, 2011) propose controlling of the distances between the circles on which the same-depth-from-leaves layers of T get laid out, thereby allowing one to spread or compress the bundled drawing in the available visual space (*i.e.* relaxation of bundle, see section 4.3.5). However, which layouts for T are best for certain tasks or drawing styles, is still a topic for further research study.

General Graphs

General graphs do not have a hierarchy structure. Bundling methods are further specialized on directed *vs* undirected graphs, as follows.

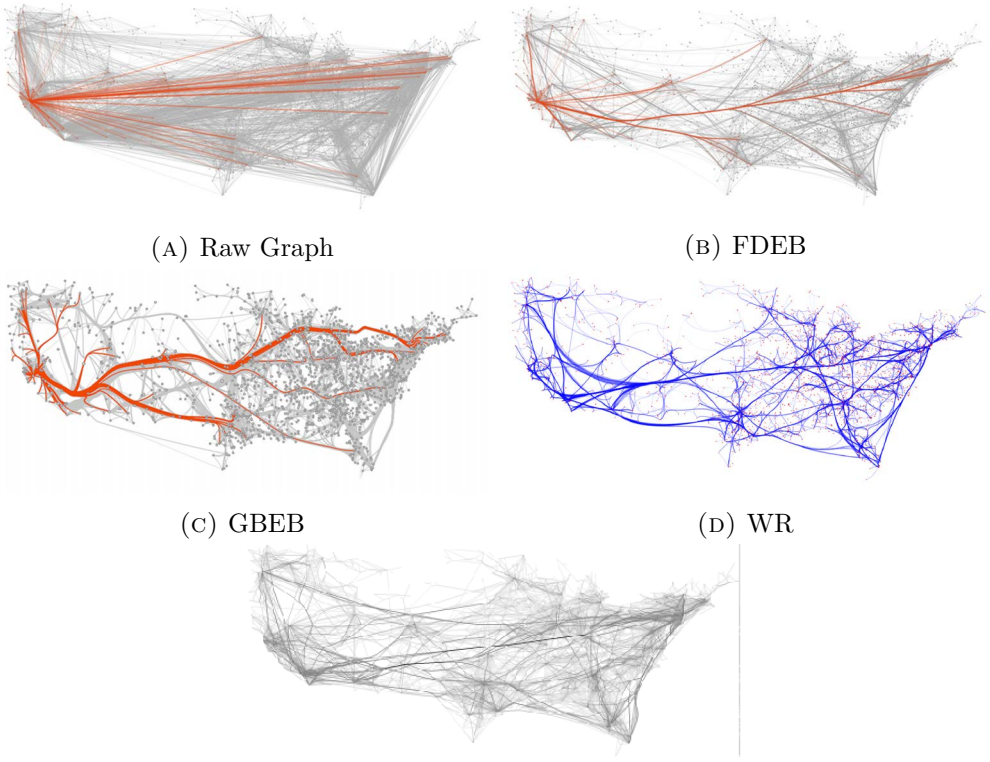
Undirected graphs Undirected graphs are the most general class of static graphs targeted by bundling. The key challenge here is to define the compatibility function κ to take into account both spatial information present in the graph layout $D(G)$ and attribute information present in G itself.

Force-directed edge bundling (FDEB, Holten and Van Wijk, 2009, figure 3.3b), the earliest method in this class, defines κ to include only geometric information in $D(E)$. Given two segments $D(\mathbf{e}_1)$ and $D(\mathbf{e}_2)$ representing the drawings of two edges $\mathbf{e}_1$ and $\mathbf{e}_2$, κ includes their angle, relative distance in $\mathbb{R}^2$, ratio of lengths $\|D(\mathbf{e}_1)\|$ and $\|D(\mathbf{e}_2)\|$, and skewness (for more details, see section 4.1.2). Next, edges are sampled into sample points $\mathbf{x}_i \in D(E)$, and each $\mathbf{x}_i$ is iteratively displaced to get closer to all other sample points $\mathbf{x}_j$ of compatible edges.

Nguyen, Hong, and Eades, 2011 propose TGI-EB to extend the compatibility measures in FDEB to account for importance-based compatibility (defined in terms of a function κ based on the Euclidean distance of n-dimensional edge attributes), and

topology compatibility, based on the position of an edge in the graph G. This richer palette of compatibilities allows more precise control of which edges get bundled, which in turn supports application-dependent analyses and a rich set of drawing styles.

Both FDEB and TGI-EB have no explicit control mesh – edges are drawn closer to each other rather than to a unique skeleton. Contrasting this, Geometry-Based Edge Bundling (GBEB, Cui et al., 2008, figure 3.3c) uses the control mesh strategy: Edges in a graph drawing are clustered in a bottom-up fashion, based on the edges' positions and orientations, yielding a control mesh whose edges tend to orthogonally 'cut' across regions having many similar-orientation edges. Mesh-edge intersections are furthered clustered to yield the shared control points through which the curved edges are finally routed. GBEB allows control meshes to be generated either automatically or with user input, the latter allowing local spatial control over which regions of $G(D)$ one wants to bundle. However, it has been noted that the result quality highly depends on the control mesh's quality, which in turn is hard to guarantee (Luo et al., 2012).



(A) Raw Graph (B) FDEB

(C) GBEB (D) WR

(E) Mingle

FIGURE 3.3 – General undirected bundling methods. US migrations *dataset* ($|N|$=1715, $|E|$=9780).

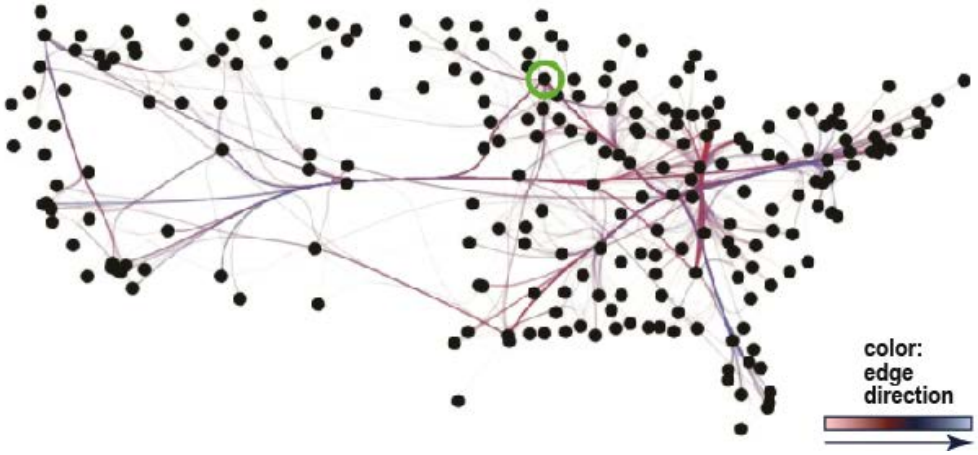
Luo et al., 2012 further refine the idea of control meshes to produce a so-called ambiguity-free bundling. They observe that highly bundled images, such as produced *e.g.* by HEB or FDEB, have difficulties in tracing edges end-to-end. To alleviate this, they propose a simple compatibility κ which is non-null only for edges sharing a node and which are also close in $D(G)$. They also remove ambiguities caused by the bundled edges $B(D(\mathbf{e}))$ passing close to unrelated nodes in $D(G)$ by re-routing

(repelling) the former from the latter. Finally, by using a small number of control points, the smoothness and low-curvature of edges is favored, which also allows their visual following. To reason about the relative positions of nodes and edges, a special quadtree, built from node positions, but storing also which cells are crossed by which edges, is built. The method produces easy-to-follow bundlings on small graphs (under hundred nodes and edges). For large graphs (thousands of nodes or edges), the method is arguably less effective, as its relatively weak bundling will still cause visual clutter.

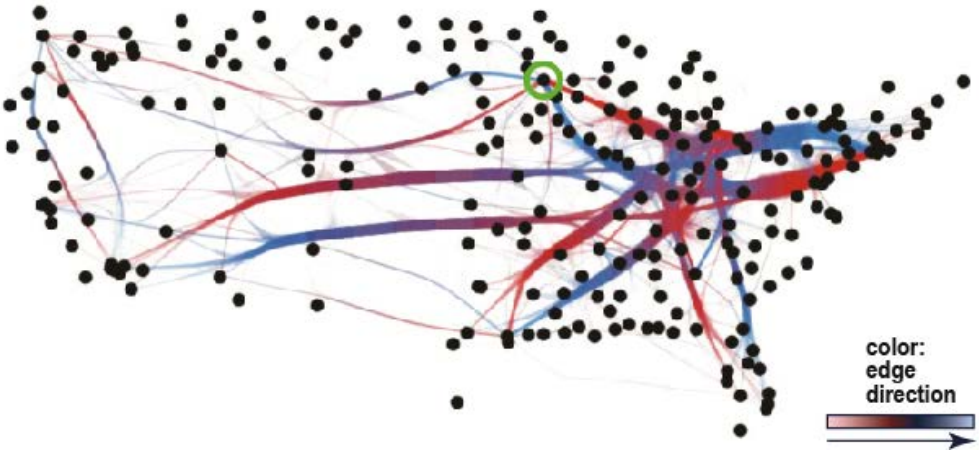
Related to Cui et al., 2008 and Luo et al., 2012, Winding Roads (WR, Lambert, Bourqui, and Auber, 2010b, figure 3.3d) computes a control mesh using quadtrees and Voronoi diagrams. WR also supports routing bundles to avoid unrelated nodes, being the first general-graph method that demonstrates such results for relatively large graphs (thousands of edges).

All general-graph methods discussed so far are quite limited in scalability, either computationally (Holten and Van Wijk, 2009; Nguyen, Hong, and Eades, 2011; Cui et al., 2008) or in terms of the largest graph they can bundle with limited clutter (Luo et al., 2012). MINGLE (Gansner et al., 2011) addresses the former by a computationally-scalable ink-saving principle, similar to the previous one Gansner and Koren, 2006: The ink for drawing a bundle $\bigcup_i B(D(\mathbf{e}_i))$, *i.e.* number of pixels covered by $\bigcup_i B(D(\mathbf{e}_i))$, should be smaller than the ink used for drawing the same unbundled edges $\bigcup_i D(\mathbf{e}_i)$. Note that the latter is roughly equal to $\sum_i \|D(\mathbf{e}_i)\|$, as edges overlap very little in a typical straight-line graph drawing. MINGLE proceeds bottom-up, by finding close edges (that have a high bundling chance, see the **Bundling equation** and related text), and bundle these in a greedy way as long as ink is saved. The process is repeated recursively by adding parts of the so far unbundled edges to existing bundle parts. A polygonal control mesh is thus created, based on the centroids of the edge-sets identified as compatible. The process is very similar to bottom-up hierarchical clustering using average linkage (Jain, Murty, and Flynn, 1999). MINGLE can bundle graphs of up to a million edges that no other general-graph method discussed so far can handle. However, it creates less smooth, and thus harder to visually follow, overall results (see figure 3.3e).

Directed graphs An already early criticism of general-graph bundling methods is that they do not take into account edges' directions: For many applications, finding if two groups of nodes $V_1 \subset D(G)$ and $V_2 \subset D(G)$ are connected by a bundle is not sufficient; we want to specifically see if there are edges from V_1 to V_2, or conversely, or both. This is essential when edge directions carry semantics, such as when exploring a software system's call graph to assess modularity (Diehl and Telea, 2014). A simple way to do this is to color-code edges using a source-to-destination categorical color gradient such as the one used by Holten, 2006, Cornelissen et al., 2008 or Reniers et al., 2014. However, when edges of opposite directions co-exist in a bundle, color mixing occurs, which makes the assessment of categorical colors very hard, let alone seeing how many edges of each direction the bundle has (see further section 4.3.1).



(A) FDEB (Holten and Van Wijk, 2009)



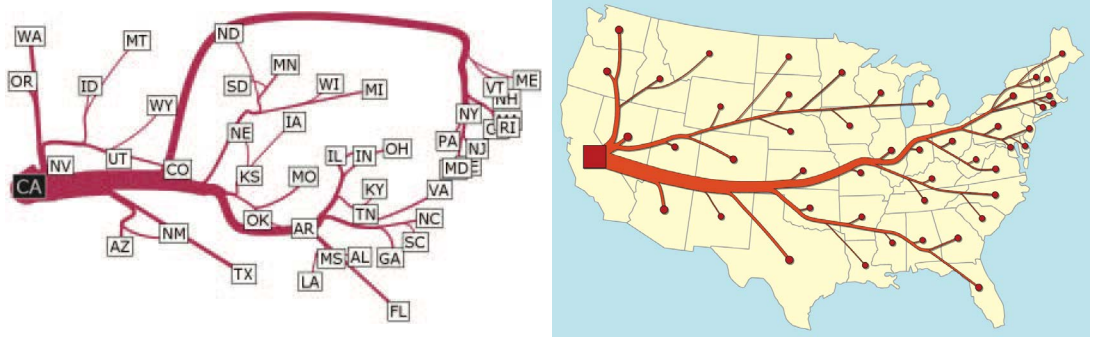
(B) DEB (Selassie, Heller, and Heer, 2011)

FIGURE 3.4 – Comparison of FDEB and DEB for the *US airlines* graph ($|N|$=235, $|E|$=2101). See section 3.1.1.

To solve this, methods for directed graphs have been proposed. Key to these is that the compatibility $\kappa(\mathbf{e}_i, \mathbf{e}_j)$ includes the directions of edges $\mathbf{e}_i$ and $\mathbf{e}_j$. The first method of this type is Divided Edge Bundling (DEB, Selassie, Heller, and Heer, 2011). Simply put, DEB extends FDEB to incorporate edge direction in κ: Same-direction edges have a positive κ, while opposed direction ones have a negative κ, respectively. Since FDEB uses κ as the amount to shift edge sample points during its iterative bundling process (see above in section 3.1.1), same-direction edges are treated as in FDEB, whereas opposed direction ones are repelled from each other. Atop the above, DEB also enhances κ to include edge weights, so that more important edges are bundled less, thus determine the outcoming $B(D(E))$ more than less important ones. Finally, DEB adds a connectivity compatibility term equal to $1/(1+\Delta(\mathbf{e}_i, \mathbf{e}_j))$ for two edges $\mathbf{e}_i$ and $\mathbf{e}_j$, where $\Delta : E \times E \to \mathbb{N}$ is the shortest-path distance in G between the nodes of $\mathbf{e}_1$ and $\mathbf{e}_2$. Overall, DEB can separate opposed direction edges quite well, and makes direction color-coding effective; however, in contrast to undirected methods, *e.g.* FDEB, DEB takes more screen space, and thus increases clutter (figure 3.4). Separately, WR (introduced earlier) is refined by using a quadtree-only control mesh, as opposed

to its more general triangle control mesh, to create directed and orthogonal bundles (Lambert, Dubois, and Bourqui, 2011) following the style of metro map drawing (Wolff, 2007).

Flow maps Flow maps can be seen as a particular subcase of directed graphs. More specifically, these are directed acyclic graphs (DAGs) having a single (or a very few) source node(s). In a transportation or data flow network, they describe how information flows from the source to reach all nodes in the graph. Since the source is unique, flow maps do not need to show edge directions explicitly such as in directed graph bundlings – the direction of data flow can be inferred by doing a visual path tracing from the source to every node of interest. Another feature of flow maps is the ability to show the amout of flow between adjacent nodes, quantified as the (weighted) number of edges linking such nodes. This allows discovering how the total amount of data outflowing from a source is spread over the graph. This is usually done by scaling the thickness of a bundle by the amount of information flowing through it, a technique pioneered by Sankey diagrams (Tufte, 1992). The first automated flow map generation by Tobler, 1981 used a straight-line single-level tree drawing linking a source with all destinations, using edge thicknesses to indicate flow amounts.



(A) Phan et al., 2005

(B) Buchin, Speckmann, and Verbeek, 2011b

FIGURE 3.5 – Flow maps of migration from California (subset of the US migration *dataset*).

The first, and most known, flow map algorithm *using bundling* (Phan et al., 2005, see figure 3.5a) was already mentioned in section 2.3.4. Given a DAG drawing $D(G)$ and a source node $\mathbf{s} \in G$, a spanning tree $ST(\mathbf{s}, G) \subset G$ rooted at $\mathbf{s}$ is constructed. Next, curved edges linking $\mathbf{s}$ to all other nodes in G are created and routed along $ST(\mathbf{s}, G)$, using a control point technique similar to HEB (section 3.1.1). Edge fragments sharing the same path are coalesced to yield bundles of variable thickness. Multiple sources $\mathbf{s}_i$ can be treated by superimposing the flow maps created by each of them – note, though, that this is not the same as having a true multiple source flow. Buchin, Speckmann, and Verbeek, 2011b further reduce the confusing bundle crossings produced by Phan et al., 2005, and produce overall lower-curvature bundles, by using the spiral tree drawing algorithm of Buchin, Speckmann, and Verbeek, 2011a to compute the skeleton to route edges along (see figure 3.5b). Computation of the optimal tree is done by moving its

nodes to optimize a cost function that accounts for avoiding obstacles and producing smooth bundles. Bundled flow maps have been found to scale less well with respect to the number of trails as compared to other visualizations of geographical trail sets, such as OD Maps (Wood, Dykes, and Slingsby, 2010) and MapTrix (Yang et al., 2017).

Confluent drawings Confluent drawings have closely related aims to flow maps, *i.e.*, show end-to-end relations between nodes in a graph drawing with as little ambiguity as possible. They inherently propose bundling in the sense of merging parts of edges that (a) simplify the drawing but (b) do not adversely affect the above-mentioned edge tracing task. Early methods can handle relatively small graphs of a particular category called confluently drawable (see figure 3.6, Dickerson et al., 2003 and Dickerson et al., 2005). Latter methods use a help structure, the power graph (Royer et al., 2008), which can be computed by various heuristics (Dwyer et al., 2014 and Dwyer et al., 2013). In this sense, they resemble flow maps which also use a tree (*e.g.* spanning (Phan et al., 2005), Steiner (Buchin, Speckmann, and Verbeek, 2011a) or spiral (Buchin, Speckmann, and Verbeek, 2011b)) to route edge bundles. In brief, power graphs provide a way to group nodes and edges in a graph in terms of how they are connected. Power graph nodes are next used to route edges between node groups so as to make the group-level interconnections easier separable visually than when using standard bundling. Although older confluent drawings have been restricted to planar graphs, recent work proposed a confluent bundling technique that can effectively handle general graphs (Bach et al., 2017). Separately, the spiral trees in Buchin, Speckmann, and Verbeek, 2011a are further refined by Nocaj and Brandes, 2013 to allow for more inflection points along a bundle, which in turn allows easily tracing of bundles end-to-end.

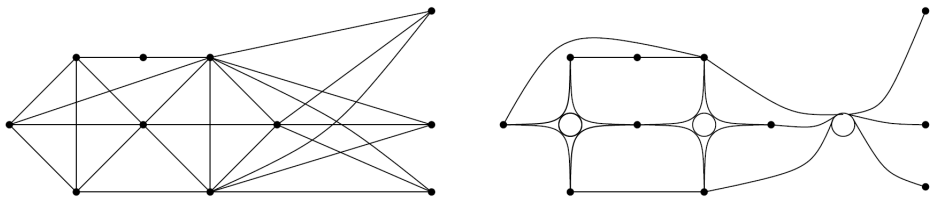


FIGURE 3.6 – Dickerson et al., 2005's confluent drawing. Original graph drawing on the left, bundled drawing on the right.

3D Graph Layouts

All bundling techniques discussed so far expect as input a two-dimensional (2D) graph drawing, *i.e.*, $D(V) \subset \mathbb{R}^2$. However, graph layouts can also produce 3D node positions. These are particularly useful when the underlying graph attributes, or problem to be solved, has a 3D nature. In such cases, one needs to bundle a 3D graph drawing. In theory, all bundling methods presented so far could be extended to handle 3D layouts. However, the many design and implementation decisions they are based on make

this impractical from a computational complexity and/or implementation simplicity perspective. As such, specific 3D bundling algorithms have emerged.

A first way to bundle in 3D is to extend HEB (Holten, 2006) to handle 3D node position introduced by Giereth, Bosch, and Ertl, 2008. This has a low computational complexity, but only works for compound graphs, as outlined in section 3.1.1. In Giereth, Bosch, and Ertl, 2008's bundling technique, the 3D layout is given by a treemap augmented with bar charts to show the structure, respectively quality metrics, of a software system, a metaphor known as a 'code city' (Wettel and Lanza, 2007). A similar idea, called 3D-HEB, used also to visualize compound graphs from software engineering, is described by Caserta, Zendra, and Bodénes, 2011 (Figure 3.7). In contrast to the approach of Giereth, Bosch, and Ertl, 2008, the third dimension is used here to pull the bundles above the 2D treemap layout so as to reduce occlusion. 3D bundling is used also to visualize bike journeys in a city (Nagel, Pietsch, and Dörk, 2016): Given two drawings of the same city map, placed parallel to each other in 3D, straight lines connecting the start and end of journeys (one end in each map) are bundled to yield a simplified traffic view.

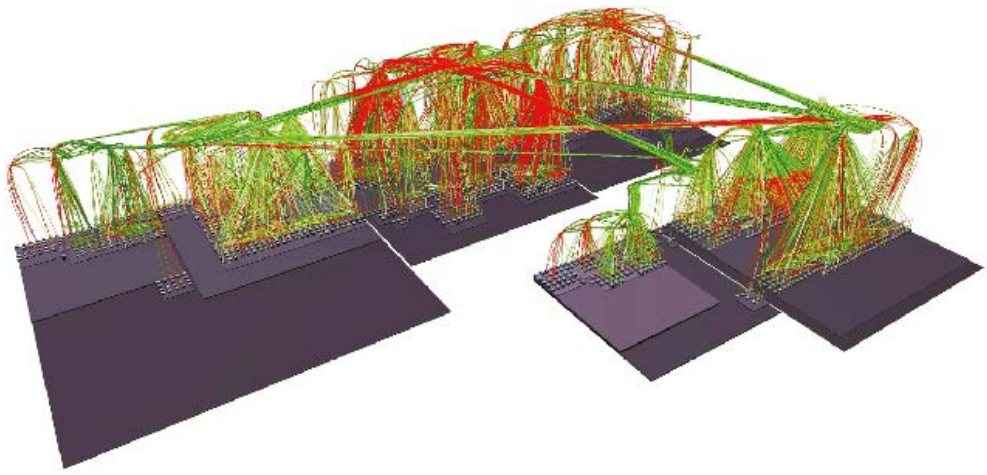


FIGURE 3.7 – 3D-HEB (Caserta, Zendra, and Bodénes, 2011)

In contrast to the above, Böttger et al., 2014 consider a graph capturing functional brain connectivity. Here, nodes represent 3D positions in a human brain. Straight-line edges are bundled using the 3D kernel-density bundling method (KDEEB, Hurter, Ersoy, and Telea, 2012) discussed next in section 3.2.1 adapted to 3D. In contrast to 3D path bundling (section 3.2.1), edge deformation is not a problem, as these edges represent only abstract connections. The method produces convincing results, but is relatively slow, due to the high complexity of 3D kernel density estimation. A similar method is proposed by Zielasko et al., 2016. Here, FDEB is used instead of KDEEB for bundling, and bundling speed is increased by adding a similar edge pre-clustering step, thereby reducing the number of pairs $(\mathbf{p}_i, \mathbf{p}_j)$ on which the compatibility κ needs to be computed (**Bundling equation**).

3.1.2 Dynamic Graphs

A dynamic graph $G(t) = (V(t), E(t)), t \in \mathbb{R}^+$ is a graph where both nodes and edges are time-dependent. That is, at each moment t, we have a potentially different graph $G(t)$ to explore. $G(t)$ is also called a *frame*, by analogy with motion video. Two different forms of dynamic graphs are known: Streaming graphs and sequence graphs. The difference between them, and bundling methods targeted at them, are outlined in section 2.1.1.

However, several aspects are common to bundling both streaming and sequence graphs, so we outline these commonalities first. First, bundling dynamic graphs is strongly related to drawing dynamic graphs (Huang, Eades, and Wang, 1998; Sun et al., 2007; Binucci et al., 2009). A recent survey on the field was proposed by Beck et al., 2014. Two main classes of methods exist here: Small multiple methods draw graphs $G(t_i)$ at a user-selected set of sample moments $\{t_i\}$ side-by-side, using the same visual mapping. To allow comparison, the layout algorithm used to construct $D(G(t_i))$ should be *stable* – that is, small changes in $G(t_i)$, as opposed to close time moments t_j, should correspond to small changes in the drawing $G(t_i)$ as opposed to $D(G(t_j))$. This requirement is also known as maintaining the user's mental map (Beck et al., 2014). Animated methods continuously display $D(G(t))$ for ranges of interest of t, again using the same visual mapping. Layout stability is also required; if met, it allows users to see changes in areas where $D(G(t))$ changes and stable data where the drawing stays unchanged, respectively. Small multiple methods have the advantage that they allow, in principle, comparing any two frames $G(t_i)$ and $G(t_j)$. However, because of screen size limitations, they typically do not scale to more than a few tens of frames. Animated approaches scale, in theory, to an unbounded number of frames. However, users cannot memorize a drawing's evolution over long periods of time, so comparing frames far apart in time requires interactive seek-and-replay of the animation. Separately, animation should be smooth, as too many sharp transitions between consecutive frames are perceived as disruptive.

Streaming Graphs

In a streaming graph $G = (V, E)$, each edge $\mathbf{e}_i \in E$ has a so-called *lifetime* $[t_i^{start}, t_i^{end}]$, of duration $\lambda_j = t_i^{end} - t_i^{start}$, where $t_i^{start} < t_i^{end}$.

Nguyen, Eades, and Hong, 2012 propose StreamEB to deal with streaming graphs. Given two edges $\mathbf{e}_i$ and $\mathbf{e}_j$ of a streaming graph, StreamEB extends the TGI-EB method for undirected graphs (Nguyen, Hong, and Eades, 2011) to add to the compatibility κ a temporal term, based on the lifetime overlap $|t_i^{start} - t_j^{start}| \cdot |t_i^{end} - t_j^{end}|$ and the duration difference $|\lambda_i - \lambda_j|$ of the two edges. After this, FDEB is applied on all edges falling in a time-window $[t, t + \Delta t]$ that slides to cover the entire time-range of $G(t)$. Here, Δt is a time interval small enough so $G(t)$ does not exhibit too many changes, but large enough to show enough interesting changes to the user. If the speed of change of $G(t)$ is small in comparison with the speed of advancing of the sliding window, and

the underlying static bundling method $B(\cdot)$ being used is continuous (in a Cauchy or Lipschitz sense) with respect to small changes in the graph, *i.e.* adding or removing a few edges from $D(G)$ only slightly changes $B(D(G))$, then the dynamic bundling $B(\cdot, t)$ proposed by StreamEB will also be continuous in time. As explained earlier, this is a desirable property for maintaining the user's mental map. However successful in this respect, StreamEB has a very high computational cost: Stable bundling static methods, such as FDEB (Holten and Van Wijk, 2009) or GBEB (Cui et al., 2008) are quite expensive, as already explained. Faster bundling algorithms, *e.g.* MINGLE (Gansner et al., 2011) or Lambert, Bourqui, and Auber, 2010b and Ersoy et al., 2011 are significantly more sensitive with respect to small changes in the input graph drawing. These problems are solved by more recent bundling methods for dynamic graphs and trail-sets, see section 3.2.2.

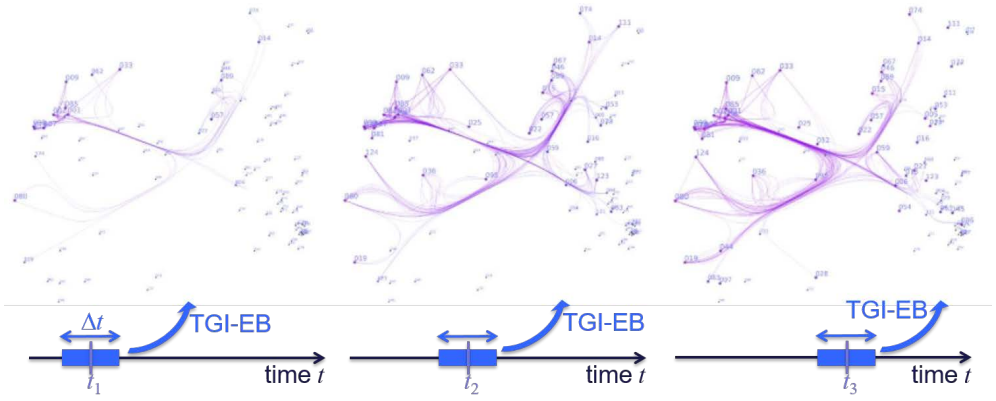


FIGURE 3.8 – Bundling of streaming graphs using StreamEB (Nguyen, Eades, and Hong, 2012) for 3 different frames.

Sequence Graphs

Sequence graphs are ordered sets $G = \{G^i\}$ of static graphs $G^i = (V^i, E^i)$.

Sequence graphs can be bundled by using the sliding-window technique in StreamEB described earlier. A much simpler and faster method is proposed by Hurter, Ersoy, and Telea, 2013 (see figure3.9): A static bundling method is applied to each G^i independently, yielding a sequence of bundled layouts $\{B(G^i)\}$. Next, for each $\mathbf{e} \in G^i$, if $c(\mathbf{e}) \neq \varnothing$, the bundled edge $B(D(\mathbf{e}))$, represented as a polyline, is linearly interpolated towards $B(D(c(\mathbf{e})))$, else $B(D(\mathbf{e}))$ is interpolated towards the straight-line segment linking the endpoints of $D(\mathbf{e})$ and the interpolated edge drawing is faded out. This signals that $\mathbf{e}$ disappears from G^i to G^{i+1}. A symmetric procedure is applied to interpolate edges that appear from G^i to G^{i+1}. Linear interpolation guarantees smooth (piecewise first-order continuous) changes in the bundled edges' positions and opacities, which preserves the mental map. To account for changes in the node set V^i, a single global layout of the union graph $\cup_i G^i$ is done upfront, so node positions $D(V^i)$ do not change. If a stable static bundling technique B is used, the method thus guarantees that big changes in the visualization correspond to appearing (progressively bundled) and

disappearing (progressively unbundled) edges. Edges that do not appear nor disappear move less in the dynamic bundling. Hence, the amount of visual change encodes well the amount of change in the graph. However simple and scalable, this method cannot directly handle streaming graphs, where edges have arbitrary lifetimes, and no explicit frames nor inter-frame correspondence data exists. Hanjalic, 2013 further adapted this method for the visualization of code clones from software repositories.

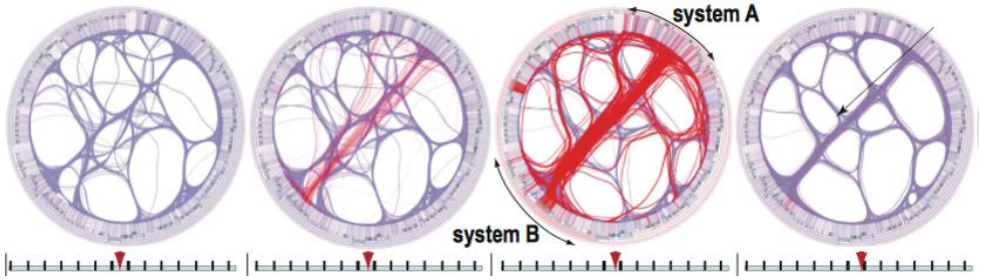


FIGURE 3.9 – Bundling of a sequence graph (see Hurter, Ersoy, and Telea, 2013).

3.2 Trail-Set Bundling Methods

As introduced in section 2.3, trail-sets consist of (typically non-straight) oriented curves $\mathbf{t}$ that describe the motion of objects in Euclidean $\mathbb{R}^2$ or $\mathbb{R}^3$ space. Hence, trail-set bundling methods have extra information and constraints to consider as compared to graph bundling methods. Conversely, the only family of graph bundling methods that uses graph-specific information not present in trail-sets are the hierarchical ones (section 3.1.1). Hence, apart from hierarchical graph bundling, trail-set bundling can be seen as a superset of general graph bundling – one can use trail-set bundling methods to bundle general graph drawings, but, in general, not conversely. We next group trail-set bundling methods based on the data type they work on.

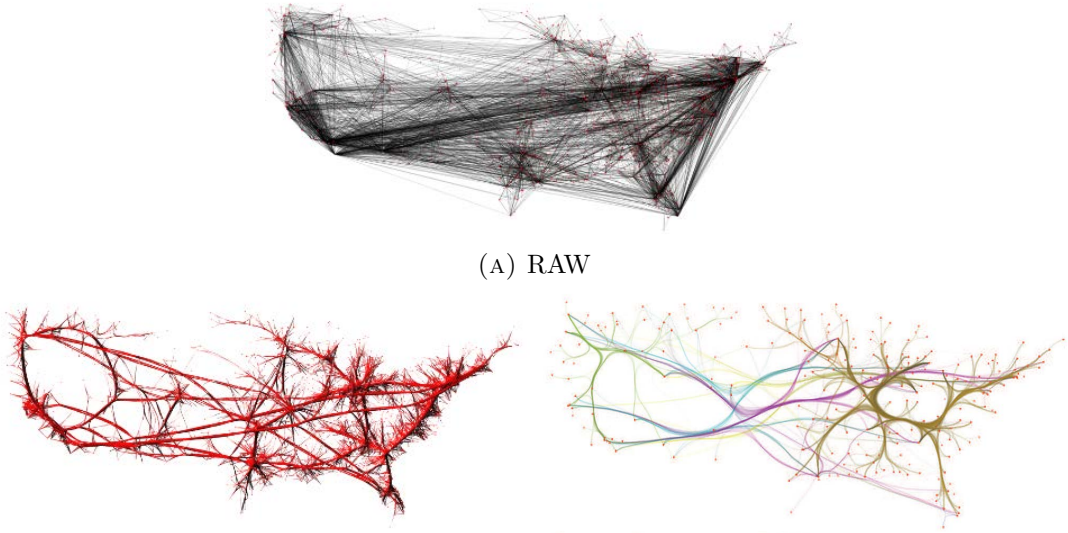
3.2.1 Static Trail-Sets

Static trail-sets are trails which do not change in time. However, time information *can* be present on such trails, *e.g.*, the time moments when a vehicle has reached each point of a trail $\mathbf{t}$ (Hurter, Tissoires, and Conversy, 2009; Scheepens et al., 2011a; Scheepens et al., 2016). In continuity with the static graph methods, we distinguish the bundling of 2D *vs* 3D trail sets, as follows.

2D Trail-Sets

2D trail-sets can be further classified in undirected, directed one. We distinguish a particular category of static trails as Parallel Coordinates Plots (PCP) as being a peculiar case of static trails, as explained here after.

Undirected trails Kernel density estimation edge bundling (KDEEB, Hurter, Ersoy, and Telea, 2012, figure 3.10b) observed, first, that a bundled drawing $B(D(T))$ has a locally either lower (outside bundles) or higher (within bundles) spatial trail density than the unbundled drawing $D(T)$. Hence, B can be cast as a density-sharpening operator, like the well-known clustering algorithm: mean shift (Comaniciu and Meer, 2002). Following this analogy, bundling consists of repeatedly computing the gradient of the density of $D(T)$ and shifting trails $D(\mathbf{t})$ upstream in this gradient until convergence (tight bundles) has been achieved. The method parallelizes well on graphics hardware (GPUs), leading performance increases of over one magnitude order as compared to all earlier general-graph bundling methods. KDEEB also opened the area of so-called *image-based* bundling methods, where B is implemented via image processing operations, as opposed to purely geometry techniques as in earlier methods (detailed further in section 4.1.2).



(A) RAW

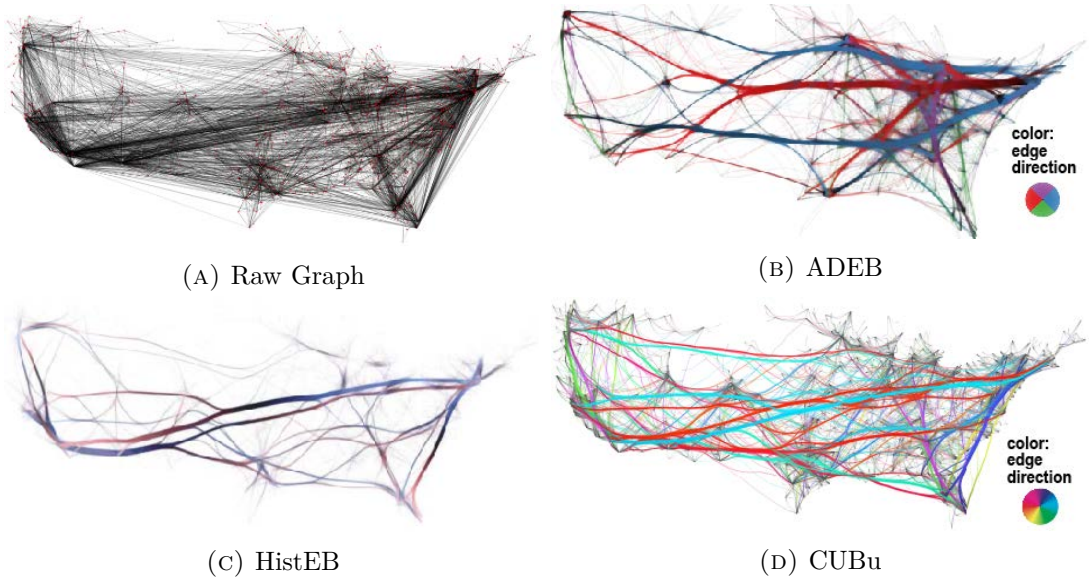
(B) KDEEB

(C) SBEB

FIGURE 3.10 – Comparison of 2D undirected bundling trail-sets methods for the US Air Flight *graph* ($|N|$=235,$|E|$=2101).

Skeleton-Based Edge Bundling (SBEB, Ersoy et al., 2011, figure 3.10c) is another image-based method. SBEB computes a Kernel Density Estimation (KDE) of the drawing $D(T)$ using GPU texture splatting. Next, the KDE map is segmented to obtain a morphologically dilated version $D_{dil}(T)$ of $D(T)$ (Haralick, 1994). Following the observation that bundles should gather trails towards their local center, 2D medial axes, or skeletons (Siddiqi and Pizer, 2008), of $D_{dil}(T)$ are next computed, and trails in $B(T)$ are attracted to the skeleton. SBEB yields smooth and highly-branching bundles, following known properties of 2D medial axes. However, the method relies on a pre-clustering of similar trails in $B(T)$, which is a relatively expensive and parameter-sensitive step.

Directed trails Following the need outlined by DEB in section 3.1.1, directional bundling is also considered, especially for trails capturing vehicle motion, where it is very important to distinguish opposite flows. The main and formost technique that allows the directionnal bundling of trail-sets is Attribute-Driven Edge Bundling (ADEB, Peysakhovich, Hurter, and Telea, 2015, figure. 3.11b). ADEB extends KDEEB by taking edge attributes (direction and/or time) in the definition of the compatibility κ, while keeping KDEEB's high speed. Histogram Edge Bundling (HistEB, Moura, 2015, figure 3.11c) also performs directional bundling, but computes κ by binning the tangent direction space of trails $\mathbf{t}$ and applying KDEEB to each bin separately. CUDA Universal Bundling (CUBu, Zwan, Codreanu, and Telea, 2016, figure 3.11d) further enhances KDEEB and ADEB by proposing a far more efficient density estimation, also implemented on the GPU, making it possible for the first time to bundle sets of up to a million trails at interactive framerates. Separately, Texture Edge Bundling (TEB, Wu, Yu, and Yu, 2015) proposes a GPU-based implementation making heavy use of texture synthesis and processing, optimized for web access.



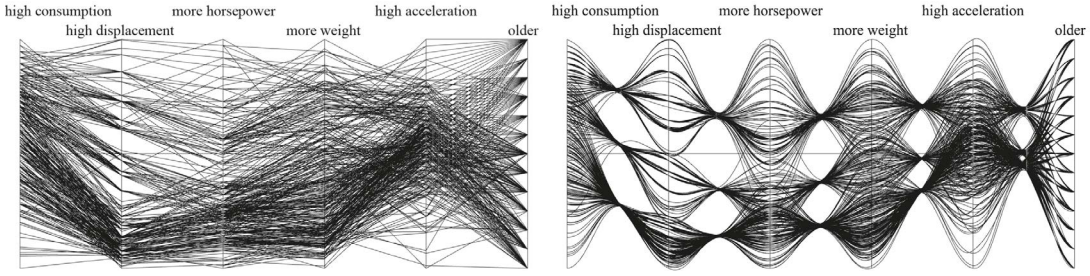
(A) Raw Graph (B) ADEB

(C) HistEB (D) CUBu

FIGURE 3.11 – Comparison of 2D directed trail-set bundling methods for the US Migration graph (($|N|$=1715,$|E|$=9780).

Parallel Coordinate Plots Parallel coordinate plots (PCPs, Inselberg, 2009) can be seen as trail-sets. Given an n-dimensional dataset of N observations $\mathbf{x}_i$, each trail $\mathbf{t}_i$ has n control points representing the n attribute values of observation $\mathbf{x}_i$. For large N, PCPs suffer from the same clutter as large straight-line graph or trail-set drawings (see figure 3.12). Bundling can effectively reduce this and also help one to find clusters of similar observations easier as highlighted by Heinrich et al., 2012 and Holten and Van Wijk, 2010. For this, Zhou et al., 2008b used a method similar to FDEB (section 3.1.1), where PCP trail compatibility (κ, see **Bundling equation**) considers PCP trail distance and angular similarity. In contrast to FDEB, the bundling solution is not computed by iterative gradient descent of the cost function κ, but by linear

programming. Illustrative parallel coordinates (IPCs, Piegl and Tiller, 1997) bundle PCPs by first clustering $D(T)$ via k-means, and next bundle the trails $\mathbf{t}_i$ in the same cluster as B-splines which use shared control points computed from the cluster's average trail (McDonnell and Mueller, 2008). More recently, Palmas et al., 2014 bundle PCPs by explicitly computing averages for each PCP axis and next linking these, for neighbor axes, by compact tubes, whose widths indicate the number of bundled lines. This produces a highly summarized, but largely clutter-free, visualization, which reminds the code flow metaphor used by Telea and Auber, 2008 (see also section 3.1.1). A variant of this technique is also available for Continuous Parallel Coordinates (CPCs, Palmas and Weinkauf, 2016), where the trail density is drawn instead of individual trails, much like it was used elsewhere to show vessel routes (Scheepens et al., 2011a).



(A) Raw PCP (B) Bundled PCP

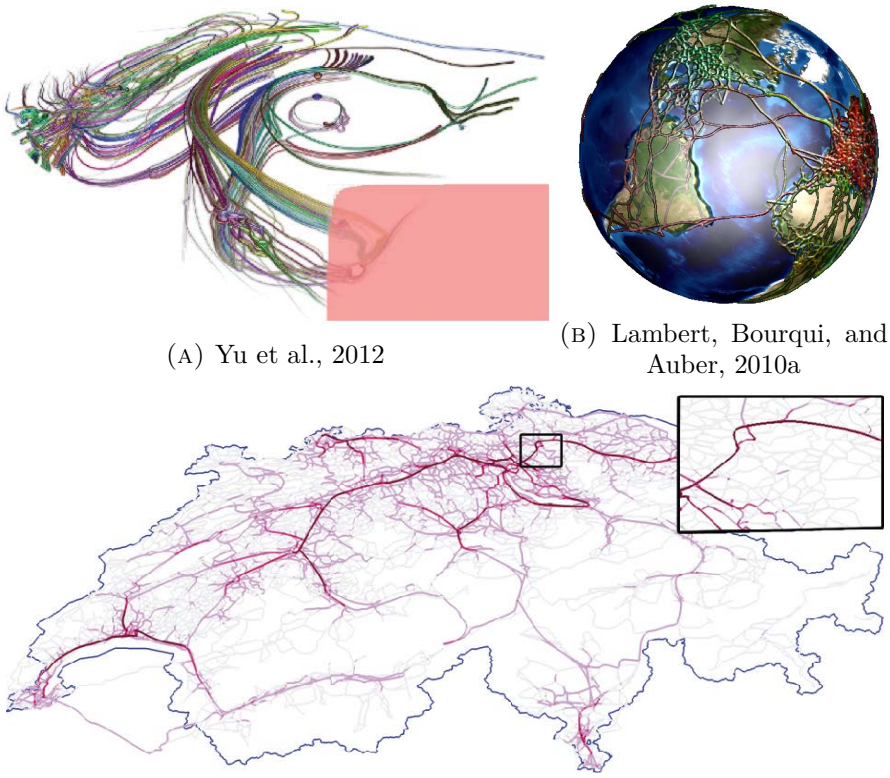
FIGURE 3.12 – Example of a PCP bundling by McDonnell and Mueller, 2008

Bundling is not the only way to reduce clutter and enhance PCP readability. PCP trails can be rendered as smooth cubic curves, allowing one to trace them end-to-end easier (Graham and Kennedy, 2003). Alternatively, parametric transformations can be used to yield similar smooth curves (Moustafa and Wegman, 2002). While yielding smooth curves similar to bundles, these techniques do not explicitly aim at grouping PCP trails following the bundling definition in the **Bundling equation**.

3D Trail-Sets

3D trail-sets are spatial curves embedded in $\mathbb{R}^3$. A good example are Diffusion Tensor Imaging (DTI) trails, or *tracts*, that show anatomical brain structures (Alexander et al., 2007). DTI tracts form a highly complex 3D structure consisting of multiply intersecting surfaces, fanning out into many-direction fibers, so *abstraction* of such visualizations is highly needed to reveal the brain's white matter structure (Tsai et al., 2007). Originally done by fiber clustering (Moberts, Vilanova, and Wijk, 2005), bundling offers advantages in terms of a finer-level simplification control. For this, Everts et al., 2015 adapt the compatibility κ and iterative bundling process proposed by FDEB (section 3.1.1) to include nearest-neighbor distance $\min_{\mathbf{x}_i \in \mathbf{t}_i, \mathbf{x}_j \in \mathbf{t}_j} \|\mathbf{x}_i - \mathbf{x}_j\|$ between two tracts $\mathbf{t}_i$ and $\mathbf{t}_j$ in $D(T)$. A similar approach is illustrated by Telea, 2015 (Ch. 7), where KDEEB (section 3.2.1) is used to bundle DTI fibers by constraining

motion to follow the DTI field anisotropy. Trail bundling is also used to bundle 3D streamlines for multiscale flow visualization (Yu et al., 2012, see figure 3.13a) and 3D geographical routes over a height map (Thöny and Pajarola, 2015, figure 3.13c). A salient difference of this use-case as compared to most other bundling cases discussed here is that a streamline's endpoints are not fixed, as they do not represent important spatial information that needs to be preserved in the visualization. Compared to 3D graph bundling, additional care is taken by all 3D trail approaches above to constrain trail displacement so as to respect, as much as possible, the original data – zones of high linear or planar anisotropy for DTI fields, tancency to the flow for vector fields, and trail fit to a 3D landscape for geographical routes, respectively.



(A) Yu et al., 2012

(B) Lambert, Bourqui, and Auber, 2010a

(C) Thöny and Pajarola, 2015

FIGURE 3.13 – Example of a 3D trail-set bundling

Besides true 3D bundling, methods also exist that bundle trails defined over a curved surface. Lambert, Bourqui, and Auber, 2010a extend the 2D bundling proposed by Lambert, Bourqui, and Auber, 2010b to cover the case of 3D trails representing flight routes (figure 3.13b). The bundles are displayed over the surface of the Earth, offering a better estimation of distances than when a 2D cartographic projection is used. A good review of the challenges of 3D geospatial trail visualization, including a discussion of bundling, is given by Buschmann, Trapp, and Döllner, 2012. 3D trail bundling is also prominently featured by Nocke et al., 2015 in a review of visualization methods for climate networks.

3.2.2 Dynamic Trail-Sets

Dynamic trail-sets parallel the concept of streaming graphs (section 3.1.2) but add explicit spatial information along trails. Examples are paths of vehicles over a given space-time range, such as ships (Scheepens et al., 2011a; Scheepens et al., 2016), airplanes (Hurter, Tissoires, and Conversy, 2009; Ersoy et al., 2011; Hurter et al., 2014a, figure 3.14), or eye tracks (Peysakhovich, Hurter, and Telea, 2015). Compared to static trail bundling, one wants here to show how the set of trails being live at a given time moment changes in time. For this, Hurter, Ersoy, and Telea, 2013 extend KDEEB (section 3.2.1) to use the sliding time-window technique of StreamEB (section 3.1.2). Compared to StreamEB, this approach is much faster, as it continuously bundles, in a loop, the trails present in the current time-window, rather than restating bundling from scratch each time the window is shifted. The method is extended by Hurter et al., 2014a and Klein, Van Der Zwan, and Telea, 2014 to cover use-cases considering eye tracking trails, and animation to show point-like textures flowing in the direction of the trails $\mathbf{t}_i$, to indicate direction.

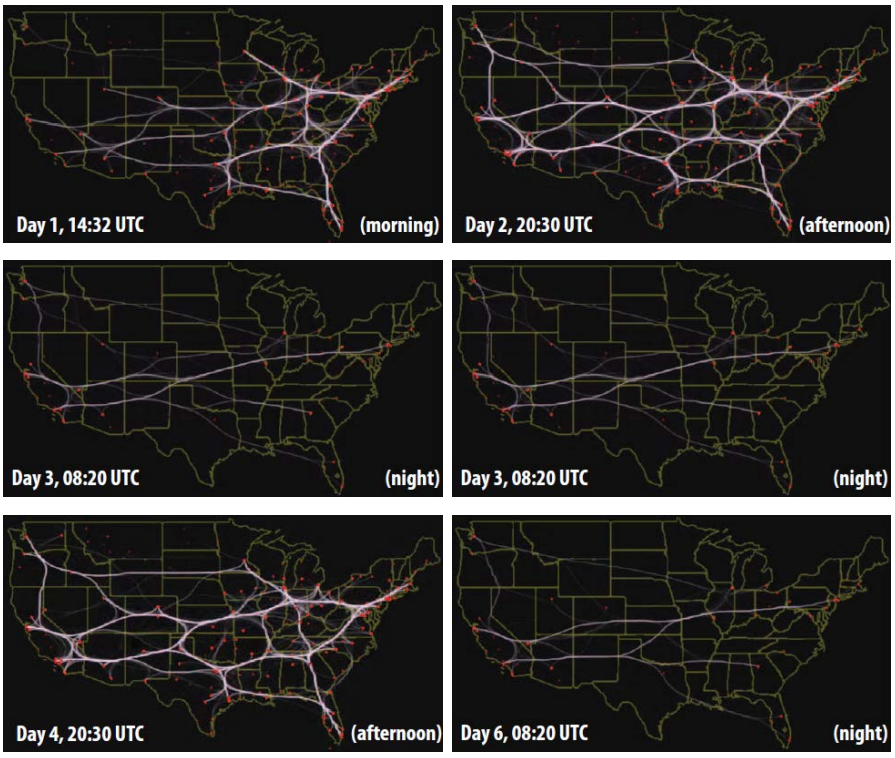


FIGURE 3.14 – Example of dynamic trail-set bundling for 6-days US airline flight dataset ($41K$ flights), Hurter et al., 2014a.

3.3 Summary

In this chapter, we proposed a data-based taxonomy of bundling methods summed up in the next pages in table 3.2 and figure 3.15. Table 3.2 allows users to find a list of suitable bundling algorithms based on the type of their input data-set (*e.g.* for a directed graph drawing, we advise the user to employ Selassie, Heller, and Heer, 2011 or Lambert, Bourqui, and Auber, 2010b). Furthermore, figure 3.15 allows users to compare the visual results of various bundling techniques. For example we can compare FDEB and GBEB bundling styles of undirected graphs (figures 3.15b and 3.15g) or compare CUBu and ADEB directionnal bundling styles (figures 3.15k,l and 3.15m).

Following our taxonomy, bundling methods can cover a wide spectrum of data types: graph drawings (trees, compound, DAGs, general oriented or not); 2D trails (vehicle movements, eye tracks, streamlines); and 3D trails (vehicle movements, DTI fibers, streamlines). All these can be either attributed (with several attributes per path) or not, and time dependent or not. As such, we believe that most data types amenable to bundling are covered by existing methods.

From a user point of view, our taxonomy shows that it is possible to choose between the wide variety of bundling methods depending on the type of input data they work on and without having to dive into the technicalities and algorithmic details of each bundling methods.

However, our taxonomy does not tell how methods relate implementation-wise to one another. In turn, this lack makes it hard for user to understand bundling parameters and compare the results, developers to implement or optimize them and researchers to extend and improve them. Therefore, in the next chapter we will aim at formalizing the technical design space of existing methods to solve the aforementioned challenges.

Graph Drawings	Static	Hierarchical compound		Holten, 2006 Telea and Auber, 2008 Holten and Van Wijk, 2008 Telea and Ersoy, 2010 Pupyrev, Nachmanson, and Kaufmann, 2010 Trümper, Döllner, and Telea, 2013
		General	undirected	Holten and Van Wijk, 2009 Cui et al., 2008 Lambert, Bourqui, and Auber, 2010b Gansner et al., 2011 Nguyen, Hong, and Eades, 2011 Luo et al., 2012
			directed	Selassie, Heller, and Heer, 2011 Lambert, Dubois, and Bourqui, 2011
			flowmaps	Phan et al., 2005 Buchin, Speckmann, and Verbeek, 2011b Buchin, Speckmann, and Verbeek, 2011a
			confluent drawing	Dickerson et al., 2003 Dickerson et al., 2005 Buchin, Speckmann, and Verbeek, 2011b Buchin, Speckmann, and Verbeek, 2011a Nocaj and Brandes, 2013 Bach et al., 2017
		3D		Giereth, Bosch, and Ertl, 2008 Caserta, Zendra, and Bodénes, 2011 Böttger et al., 2014
	Dynamic	Sequence		Hurter, Ersoy, and Telea, 2013 Hanjalic, 2013 Hurter et al., 2014a
		Streaming		Nguyen, Eades, and Hong, 2012
Trail Sets	Static	2D	undirected	Ersoy et al., 2011 Hurter, Ersoy, and Telea, 2012 Zwan, Codreanu, and Telea, 2016
			directed	Peysakhovich, Hurter, and Telea, 2015 Moura, 2015 Lhuillier, Hurter, and Telea, 2017a
			PCP	McDonnell and Mueller, 2008 Telea and Auber, 2008 Zhou et al., 2008b Heinrich et al., 2012 Palmas et al., 2014 Palmas and Weinkauf, 2016
		3D		Lambert, Bourqui, and Auber, 2010a Yu et al., 2012 Everts et al., 2015
	Dynamic (streaming)			Hurter, Ersoy, and Telea, 2013 Hurter et al., 2014a

TABLE 3.2 – Data-based taxonomy of graph and trail-set bundling methods. For space reasons, only the main methods in each class are listed. Bundling papers that present applications, but do not introduce a new bundling technique, are not listed here.

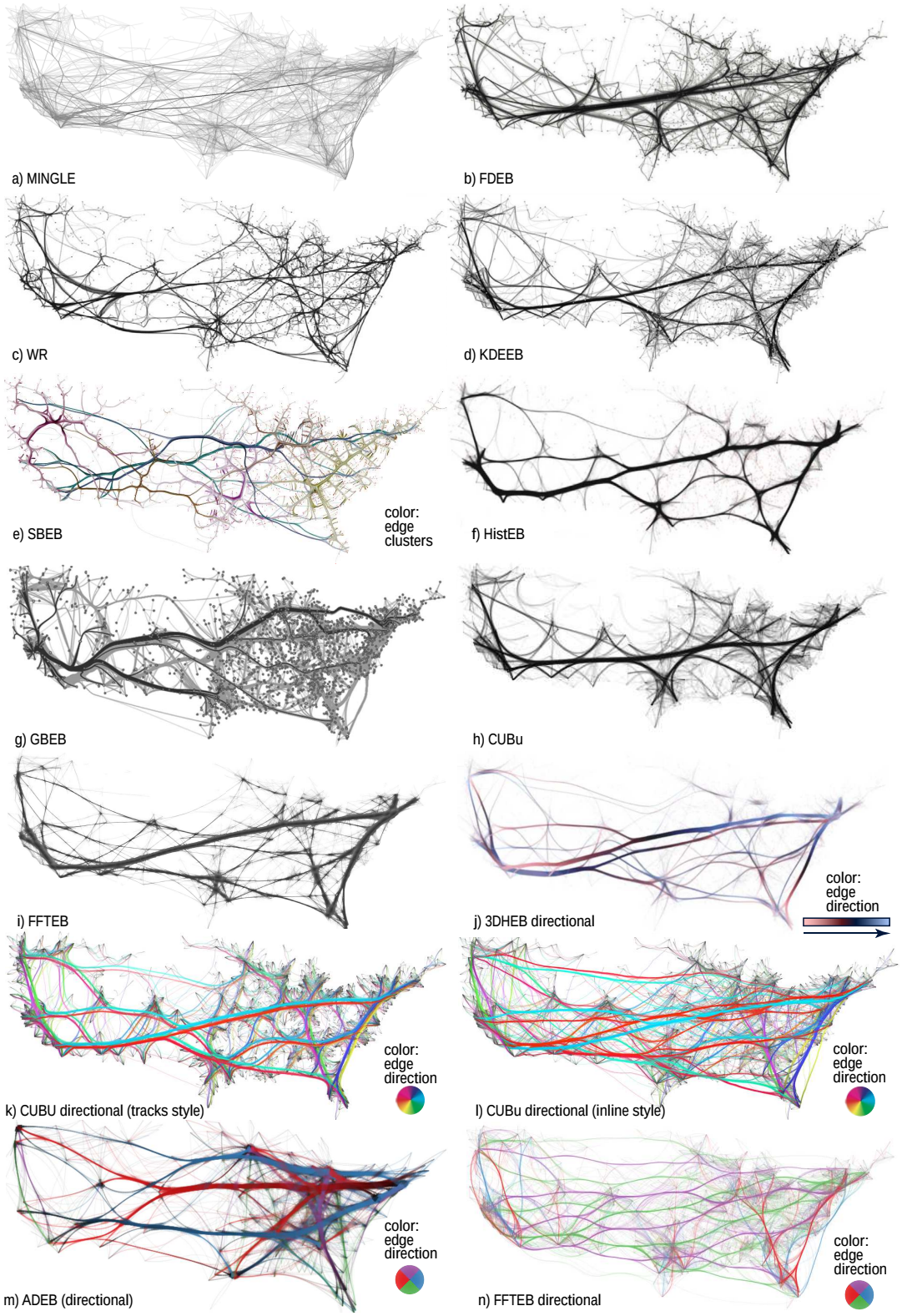


FIGURE 3.15 – General undirected (a-i) and directed (j-n) bundling methods, *US migrations* dataset ($|N|$=1715, $|E|$=9780). See section 3.1.1.

Part II

Unifying, Understanding and Improving Bundling Techniques

Chapter 4

Towards a Unified Bundling Framework

Given their large number and diversity, comparing all bundling methods listed in chapter 3 from a technical viewpoint is a difficult challenge. In this chapter, we propose a generic and unified bundling framework. Based on the notation introduced in section 2.3, our framework describes the main steps and technical choices of most of the existing bundling methods. This is necessary for developers interested to understand how such methods work, and for researchers who aim to extend existing methods. For developers, this technical framework of bundling technique helps one to compare in detail specific steps of specific algorithms, and complements the high-level taxonomy presented in chapter 3. Furthermore, our framework helps researchers by outlying technical limitations of existing algorithms, and by suggesting future improvement areas.

In this framework, we outline the specific advantages and limitations of the various technical choices. Depicted in figure 4.1, the framework has four steps. It starts with a graph G or a trail-set T. In the case of a graph, a graph layout method is used to construct a graph drawing. As this step was already discussed in chapter 2, we consider that we are working directly with the drawing of the graph $D(G)$. For the other case, the trail-set is already embedded in a drawing, as explained in section 2.3. Thus, the proper bundling pipeline starts with a path-set drawing $D(P)$ as input. Next, the similarity functions κ and δ, as formalized in our **Bundling equation**, must be defined. From these, we show in section 4.2 how the bundling operator B can be formalized and applied on the path-set drawing $D(P)$ to yield the bundled drawing $B(D(P))$. In the final step of the pipeline, the bundled path-set is rendered visually, enhanced and explored (section 4.3).

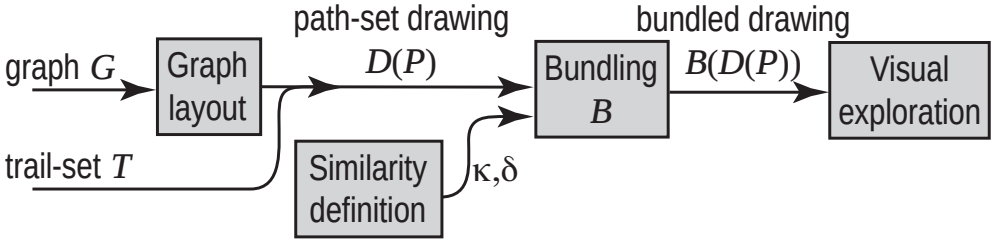


FIGURE 4.1 – Bundling framework steps.

4.0 Résumé (Français)
Un Pipeline Graphique Unifié du Bundling

Compte tenu de leur nombre et de leur diversité, comparer l'ensemble des différentes techniques inhérentes aux méthodes de bundling listées en chapitre 3 est un défi difficile. Dans ce chapitre, nous proposons un pipeline graphique des techniques de bundling. Ce pipeline, générique et unifié, se fonde sur les notations introduites dans la section 2.3. Notre pipeline décrit les principales étapes et principaux choix techniques de la plupart des techniques de bundling existantes. La formalisation d'un tel pipeline est nécéssaire tant pour les développeurs souhaitant comprendre le fonctionnement des méthodes de bundling que pour les chercheurs souhaitant étendre et améliorer les méthodes existantes. Pour les développeurs, le framework technique permet la comparaison détaillée des étapes spécifiques des différents algorithmes et permet de completer notre taxonomie orientée structure de données présentée dans le chapitre précedent (chapitre 3). De plus, notre framework permet aux chercheurs de comprendre les limitations techniques des algorithmiques actuels tout en suggérant des axes d'amélioration.

Dans notre pipeline graphique des algorithmes de bundling, nous soulignons les avantages et limitations spécifiques aux différents choix techniques d'implémentation d'une méthode de bundling. Détaillé en figure 4.2, le pipeline se compose de quatre grandes étapes. Notre framework démarre soit avec un graphe G ou un jeu de trajectoires T. Dans le cas d'un graphe, l'utilisation d'une méthode de tracé de graphe est utilisée pour construire le dessin du graphe $D(G)$ dans un espace Euclidien (étape évoquée dans le chapitre 2). Dans l'autre cas, les trajectoires sont déjà incluses dans un espace Euclidien (comme expliqué en section 2.3). Ainsi, le pipeline de bundling commence avec le dessin d'un jeu de courbes (*path-set drawing*, $D(P)$) comme entrée. Ensuite, deux fonctions de compatibilité κ et δ, comme formalisées en **Équation du bundling** sont définies comme critère d'agrégation des courbes entre elles. A partir de ces compatibilités, nous montrons (voir section 4.2) que l'opérateur d'agrégation B peut être formalisé et appliqué au jeu de courbes $D(P)$ afin de créer le dessin des courbes agrégées $B(D(P))$. Enfin, dans la dernière étape de notre pipeline, la visualisation du jeu de courbes agrégées est générée à l'aide de méthodes améliorant le rendu visuel et permettant l'exploration de ce premier (voir section 4.3).

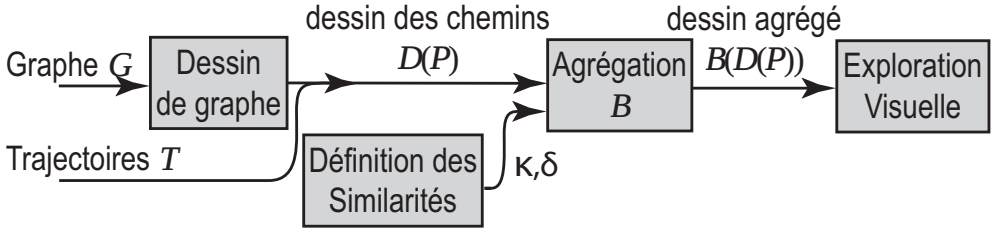


FIGURE 4.2 – Pipeline graphique des méthodes de bundling.

Fonctions de Compatibilité

Afin d'agréger un jeu de courbes donné, il nous faut d'abord spécifier quelles sont les courbes compatibles entre-elles. Nous distinguons deux classes de fonctions de compatibilité entre courbes, détaillées ci-après.

Compatibilités Basées Données

Considérant deux courbes p_i et p_j, les fonctions de compatibilité basées données quantifient la différence entre p_i et p_j dans l'espace des courbes existantes. Ainsi, cette compatibilité peut être :

- basée sur la structure (ou topologie) des courbes p_i et p_j. Les compatibilités basées sur la structure du jeu de données sont simplement définies à l'aide d'une distance de graphe (et non de dessin de graphe) entre les deux courbes. Des exemples de distances sont détaillés en section 4.1.1.
- basée sur les attributs des courbes. Si les courbes possèdent des attributs tels qu'une direction ou une dépendance temporelle, il est alors possible de définir la compatibilité entre deux courbes en fonction de leur attributs respectifs.

Compatibilités Basées Dessins

Si l'utilisation de techniques de compatibilités basées données est optionnelle pour agréger des courbes entre elles, toute agrégation se définit intrinsèquement par une agrégation basée sur la distance euclidienne entre les courbes dans leur espace de dessin. Nous distinguons deux approches pour définir ce type de compatibilité :

- Similarités géométriques. Ces méthodes décomposent les courbes en sommes de lignes droites (poly-lignes) pour évaluer leur compatibilité. Ainsi, plusieurs aspects des courbes peuvent être pris en compte pour la compatibilité comme la distance (euclidienne), l'angle, la taille ou la visibilité entre deux courbes.
- Similarités images. L'approche orientée image de calcul des similarités entre les courbes est en soi une optimisation des similarités géométriques. Plutôt que de chercher pour chaque courbe la courbe associée la plus proche, les similarités basées images utilisent des textures (ou images) pour effectuer une recherche de plus proches voisins plus rapide. Dans ce cas, le dessin des courbes est projeté dans un espace discret représenté par la texture.

Définition de l'Opérateur d'Agrégation

Après avoir défini les règles de compatibilités κ et δ, nous définissons l'opérateur d'agrégation B comme défini dans **Équation du bundling**. Nous distinguons deux approches d'agrégation *implicites* et *explicites*

Méthodes Explicites

Les méthodes explicites calculent une structure intermédiaire I depuis $D(P)$ et définissent B en fonction de I. La structure I est typiquement calculée une seule fois puis donne un cadre à partir duquel les courbes sont agrégées. Nous distinguons plusieurs types de structures I, principalement basés sur le type de données d'entrée (voir section 4.2). Pour exemple, les graphes hiérarchiques utilisent la hiérarchie comme I, les graphes génériques utilisent plusieurs méthodes comme les arbres de Steiner (Buchin, Speckmann et Verbeek, 2011a), un maillage ou mesh (tel qu'un diagramme de Voronoï Lambert, Bourqui et Auber, 2010b).

Méthodes Implicites

A l'opposé des méthodes explicites, les méthodes implicites définissent l'agrégation B entre courbes de manière récursive similairement aux processus d'optimisation de recherche de maximum d'une fonction globale. De ce fait, les courbes sont progressivement agrégées les unes avec les autres. Ces méthodes évitent ainsi le calcul d'une structure intermédiaire I servant de guide.

Amélioration du Rendu Visuel

Sur l'ensemble des techniques de bundling présentées dans la sous-section précédente, toutes utilisent des techniques de traitement d'images afin d'améliorer le rendu visuel. Ces traitements permettent d'améliorer la lisibilité du graphe agrégé ou de coder une information supplémentaire au travers d'une variable visuelle (Bertin, 1981). Ces techniques sont principalement la transparence des liens (*alpha blending*, figure 4.3), la gestion de l'ordre d'occlusion (*z-ordering*) ou encore la gestion de l'éclairage (*shading*, figure 4.4). Dans ce résumé, nous nous concentrons sur les trois étapes d'amélioration du rendu les plus caractéristiques du bundling.

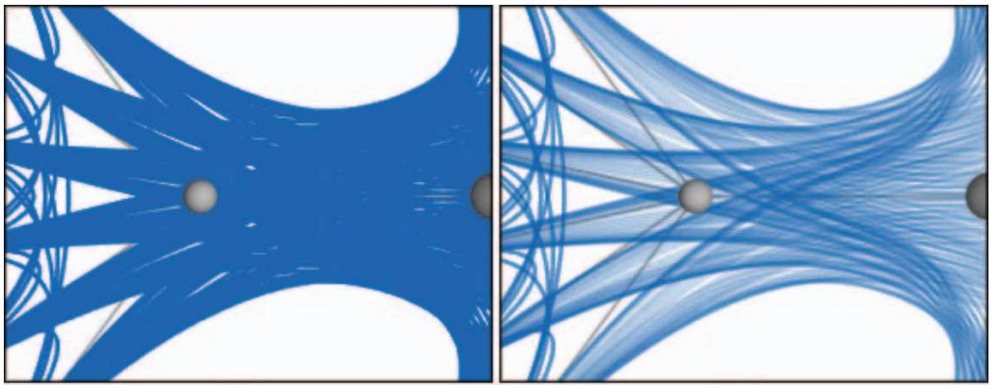
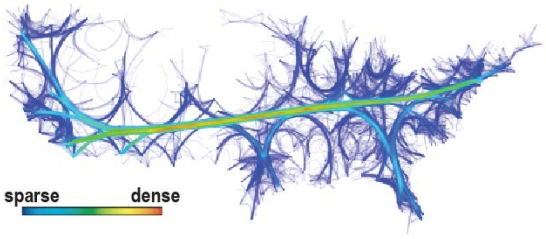
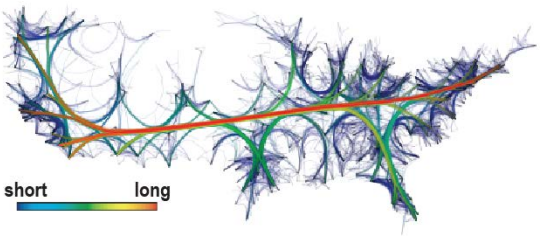


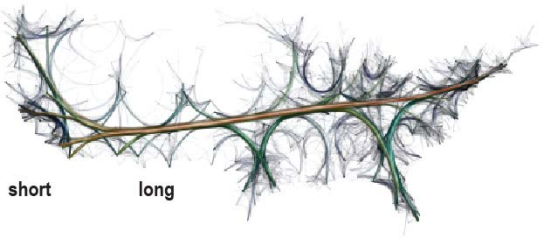
FIGURE 4.3 – Utilisation du mélange de couleur pour renforcer les différences de densité par Holten, 2006. A gauche : dessin $B(D(P))$ sans *blending*. Droite : même dessin avec *blending*.



(A) Densité locale de courbes



(B) Longueur des liens



(C) Longueurs des liens avec ombrage

FIGURE 4.4 – Cartes de couleurs et ombrage utilisés par CUBu (Zwan, Codreanu et Telea, 2016).

Affichage de la Densité

Le bundling ajoute des espaces vides au détriment de zones plus denses (Hurter, 2015). Ainsi, pour permettre à l'utilisateur de visualiser cette agglomération, les liens agrégés affichent la densité de liens originels présents dans un même regroupement (i.e. la densité du nouveau lien). Dans l'ensemble des techniques existantes, deux méthodes de représentation de la densité sont utilisées. La première consiste à coder la densité sur une échelle de couleur. La seconde consiste à représenter des liens dont l'épaisseur dépend de la densité. La densité peut aussi être affichée à l'aide de la technique de *bump-mapping* (Hurter, Ersoy et Telea, 2012 ; Lambert, Bourqui et Auber, 2010b).

Affichage de la Direction

Dans le cas particulier des graphes orientés, plusieurs techniques ont été utilisées pour représenter la direction sous forme de variable visuelle. Ces techniques sont : les dégradés de couleurs, les textures et les animations par particules (Peysakhovich, Hurter et Telea, 2015). Holten et Van Wijk, 2009 ont réalisé des expérimentations permettant de déterminer le choix de design le plus efficace. Ces expérimentations montrent que l'animation et la forme (triangle pointant vers la direction) sont les choix les plus pertinents. Récemment, Scheepens et al., 2016 ont utilisé un système à particules pour visualiser la direction d'un graphe agrégé.

L'Évitement

L'évitement est une méthode plus marginale, qui consiste à insérer des obstacles dans un graphe agrégé. Cette méthode permet d'éviter que des liens ne recouvrent des régions particulières du graphe (Hurter, Ersoy et Telea, 2012).

Dans ce chapitre, nous avons montré que notre pipeline unifie les différentes techniques de bundling existantes. Nous pensons que les techniques de simplification visuelle par agrégation de lien sont arrivées à maturation, grâce aux nombreuses évolutions et améliorations issues de diverses publications. Ces dernières concernent notamment les problèmes de passage à l'échelle et de temps de calcul, comme détaillé en section 2.3.

À travers cette évolution du bundling, nous distinguons quatre générations de techniques de bundling (figure 4.17). Les techniques pré-HEB pouvaient gérer des graphes avec des centaines de liens et fonctionnaient uniquement en CPU (*e.g.* Newbery, 1989 ; Brandes et Wagner, 1998 ; Dickerson et al., 2003 ; Phan et al., 2005). La seconde génération s'est focalisée sur les graphes et jeu de trajectoires possédant quelques milliers de courbes (jusqu'environ 10k courbes) ; à l'exception de MINGLE (Gansner et al., 2011) ayant atteint la capacité de traiter jusqu'à 1 million de courbes. Les techniques de seconde génération fonctionnaient principalement à l'aide de compatibilités géométriques. La troisième génération débute avec SBEB (Ersoy et al., 2011) et KDEEB (Hurter, Ersoy et Telea, 2012), utilisant à la fois le CPU et le GPU, elles ont introduit les compatibilités basées image et permettent de traiter des jeux de courbes allant jusqu'à quelques dizaines de milliers. Avant l'introduction des techniques de quatrième génération, le bundling de grandes quantités de données était un défi tant sur le plan des vitesses de calcul des algorithmes que sur le volume intrinsèque des données à traiter. Ainsi, les techniques de quatrième génération (détaillées dans le chapitre 6) ont résolu ces problèmes en permettant l'agrégation de courbes allant jusqu'à des millions dans des temps de calculs inférieurs à la seconde et résolvent tout problèmes de taille des données en introduisant des méthodes de streaming des données en mémoire. Avec l'apparition de GPU plus rapides et ayant plus de mémoire, nous pensons qu'à travers leur évolution, les techniques de bundling ont réussi à résoudre les problématiques de passage à l'échelle, de telle sorte que la simplification de courbes par agrégation visuelle puisse réellement traiter des *big data*.

Dans le chapitre précèdent, nous avons proposé une taxonomie des méthodes de bundling fondée sur le type de donnée d'entrée à traiter et dans ce chapitre, nous avons définis un pipeline permettant d'unifier les techniques de bundling. Néanmoins, pour être capable d'utiliser et de comprendre l'intégralité des capacités des méthodes de bundling, il nous reste à définir et montrer le type de tâches utilisateurs auxquelles le bundling peut répondre.

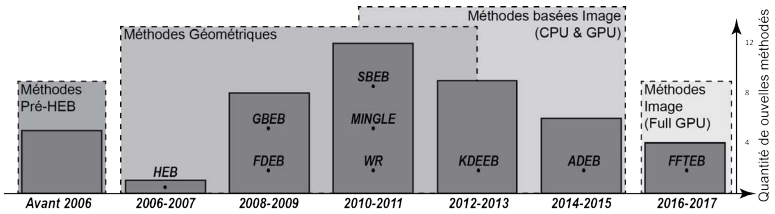


FIGURE 4.5 – Évolution temporelle et générations des principales méthodes de bundling (comme défini en table 3.1).

4.1 Similarity Definition

To bundle a given path-set, we need, first and foremost, to specify which curved edges are compatible with one another in order to bundle them. And then we must specify the amount of bundling to be computed. The first steps correspond to implementing the function κ and the second steps the function δ defined in our **Bundling equation**. As detailled in section 2.3, κ can account for a wide variety of similarities in the *data space* (*e.g.* graph structure and trail attributes), and must in any case account for the similarities in the *drawing space* (generally 2D or 3D). The two components of the existing κ implementations are discussed next.

4.1.1 Data-Based Similarities

Given two paths $\mathbf{p}_i$ and $\mathbf{p}_j$, data-based similarity quantifies the difference between $\mathbf{p}_i$ and $\mathbf{p}_j$ over the space of all possible paths. This can be done in several ways, as follows.

Structure-Based

In contrast to trail-sets, graphs have additional structure (topology). For graphs, κ can be defined by analysing the topology of the input G. Simply put, $\kappa(\mathbf{p}_i, \mathbf{p}_j)$ is set to user defined type of graph theoretic distance between edges $\mathbf{p}_i$ and $\mathbf{p}_j$. The theoretic distance can be extracted either from the graph itself G or from any other structures defined on top of G.

For example, hierarchical bundling methods (see section 3.1.1) use the explicit structure of the graph's hierarchy to define edge similarity in terms of distance, in the given hierarchy, of the nodes of the two edges $\mathbf{p}_i$ and $\mathbf{p}_j$. Intuitively put, edges that start and end at nodes which are close in the hierarchy, are considered to be close, and thus deemed to be bundled together.

Other methods extract such structural data from the input graph, and use it to define edge similarity. Flow maps extract a tree from the graphs with diverse techniques such as a spanning tree (Phan et al., 2005), a steiner tree (Buchin, Speckmann, and Verbeek, 2011a), or a spiral tree (Buchin, Speckmann, and Verbeek, 2011b). Similarly, confluent drawing uses a spiral tree or a power graph to find out which edges in the original graph are strongly related (see Dwyer et al., 2014; Dwyer et al., 2013; Bach et al., 2017).

Separately, TGI-EB by Nguyen, Hong, and Eades, 2011 adds new graph-theoretic metrics to define edge compatibility, such as the centrality of edges in a graph (see Wasserman and Faust, 1994; Van Ham and Wattenberg, 2008), or the fact that edges belong to the same cluster in a simplified version of the graph. Similarly, DEB (Selassie, Heller, and Heer, 2011) proposes a connectivity term based on the graph-theoretic distance between the endpoints of two edges (through a physic metaphor base on Coulomb law). A similar term, called trace compatibility, appears in StreamEB (Nguyen, Eades, and Hong, 2012).

Finally, StreamEB introduces a so-called ego compatibility: For a node $\mathbf{v} \in V$, $ego(\mathbf{v})$ is defined as a small neighborhood in G of $\mathbf{v}$, nodes and edges included. For an edge $\mathbf{e} = (\mathbf{v}_i, \mathbf{v}_j)$, $ego(\mathbf{e}) = ego(\mathbf{v}_i \cup \mathbf{v}_j)$. Next, the ego compatibility of two edges $\mathbf{e}_i$ and $\mathbf{e}_j$ is a decreasing function of $ego(\mathbf{e}_i) \cap ego(\mathbf{e}_j)$. This prevents bundling edges which link weakly-connected communities in G.

Structure-based similarity serves two independent goals. First, it allows specifying which edges are compatible from an application perspective, and thus may end in the same bundle, as outlined above. Separately, it allows constructing groups of possibly compatible edges which are next examined for actual bundling. This essentially reduces the bundling complexity from analyzing all edge pairs in G to only analyzing pairs within a cluster. Clustering can be done using:

- k cores (Brandes, 2005), see Nguyen, Hong, and Eades, 2011;
- bottom-up hierarchical aggregation (Hoon et al., 2004), see Telea and Ersoy, 2010; Ersoy et al., 2011; Zielasko et al., 2016;
- k means, see Piegl and Tiller, 1997;
- or kd-trees, see Gansner et al., 2011.

Structure-based similarities have the advantage to take into account the *actual* data of the graph G and not just the drawing of the graph $D(G)$. This is an important benefit as the simple fact of drawing the graph can lead to the loss of dimensions as explained in section 2.1. Conversely, when using structure-based similarities for bundling the drawing of a graph, we should take care that the drawing of the graph do reflect the similarity κ extracted from the graph G. Finally, structure-based similarities cannot be used readily for trail-sets.

Attribute-Based

When edge or trail data attributes are available, these can be used to compute additional similarity terms. This is important when one does not want to bundle edges of different types together. Good examples are software engineering graphs which contain edges of various kinds, such as inheritance, call, or dependency (Cornelissen et al., 2008; Reniers et al., 2014; Diehl and Telea, 2014); and airline trail-sets which contain trails representing flights with different IDs or types (Hurter et al., 2014c). Each edge type has a different meaning, so edges of different types should arguably not be bundled together. There are mainly three approaches to define attribute based compatibilities;

First, the similarity function can be based on the edge's categorical attribute 'type'. This is probably the simplest implementation of the three. It is used by IBEB (Telea and Ersoy, 2010) as well as the Solid* toolset (Reniers et al., 2014).

Next, the similarity can be based upon edges' quantitative attribute. For example, Trümper, Döllner, and Telea, 2013 and Karran, Trumper, and Dollner, 2013 use the

duration and/or the starting-time for calls in program traces, whereas Peysakhovich, Hurter, and Telea, 2015 use time-stamps for eye tracks to define similarities.

Separately, for dynamic path-sets, time provides an additional compatibility factor, used to bundle only paths that fall within the same (usually small) time-window. This similarity definition is typically used by all dynamic path-sets techniques (see 3.2) like Nguyen, Eades, and Hong, 2012 and Hurter et al., 2014a. This makes bundling computationally and visually scalable to very large streaming datasets.

It is worth noting that this kind of attribute based compatibility is closely related to clustering techniques. Besides implicitly bundling defined on an attributed-based similarity, we can explicitly cluster the path-set based on the same similarity, and then apply any bundling method we want per cluster. This "divide and conquer" strategy is mainly used by Telea and Ersoy, 2010 and Ersoy et al., 2011.

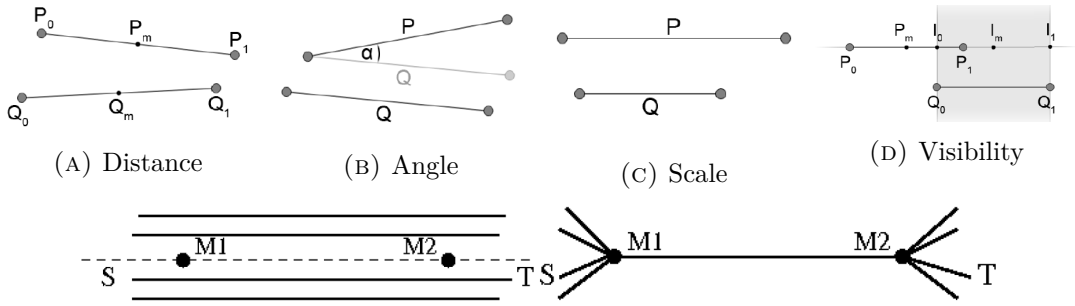
4.1.2 Drawing-Based Similarities

While incorporating *data*-based similarity in the compatibility function κ is optional, all bundling methods must account for the *spatial* similarity of paths to bundle. Indeed, there is no point in bundling paths which are really far away in the path-set drawing $D(P)$. Implementing drawing-based similarities can be done by using either geometric methods (section 4.1.2) or image-based methods (section 4.1.2).

Geometric-Based Similarity

These methods use a piecewise-polygonal sampled representation of the path drawings $D(\mathbf{p})$ to evaluate their similarity. As such, various aspects of a curve can be taken into account. Let $\mathbf{d}_i \subset \mathbb{R}^d$ and $\mathbf{d}_j \subset \mathbb{R}^d$ be two such polyline path drawings. Similarity δ typically is a product (or, for Nguyen, Eades, and Hong, 2012, a weighted sum) of the following terms:

Distance: First, the distance between $\mathbf{d}_i$ and $\mathbf{d}_j$ is considered, as one does not want to bundle far-away path drawings (see the **Bundling equation** and related text). The literature offers different ways to compute the distance between $\mathbf{d}_i$ and $\mathbf{d}_j$. The most common methods used to evaluate the distance between the two poly-lines $\mathbf{d}_i$ and $\mathbf{d}_j$ is to consider it as the distance between the midpoints of $\mathbf{d}_i$ and $\mathbf{d}_j$ (see figure 4.6a). This definition was first introduced by FDEB (Holten and Van Wijk, 2009) and used thereafter by all its refinements (Nguyen, Eades, and Hong, 2012; Nguyen, Hong, and Eades, 2011; Böttger et al., 2014; Zielasko et al., 2016) as well as Winding Roads (Lambert, Bourqui, and Auber, 2010b). Another way to define this distance is used by Gansner et al., 2011. In their implementation, the distance is defined as the sum of distances of the closest endpoint between two poly-lines $\mathbf{d}_i$ and $\mathbf{d}_j$. However simple, such similarities cannot be used for trail bundling, as trails are usually not straight lines.



(A) Distance (B) Angle (C) Scale (D) Visibility

(E) Ink Minimization

FIGURE 4.6 – Classes of geometric-based similarities.

Angle: The absolute value of the cosine of the angle formed by the straight-line segments $\mathbf{d}_i$ and $\mathbf{d}_j$ is used to prevent bundling of orthogonal edges (see figure 4.6b). Angle compatibilities is mainly used by Holten and Van Wijk, 2009, Nguyen, Hong, and Eades, 2011 and Nguyen, Eades, and Hong, 2012. This principle is further refined by DEB (Selassie, Heller, and Heer, 2011) to consider the actual signed angle, which allows directional bundling.

Scale: The length difference of the two segments $\mathbf{d}_i$ and $\mathbf{d}_j$ is used to prevent bundling very long and very short edges (see figure 4.6c). Therefore preventing bundling of edges with different lengths minimizes the deformation of the latter, which are best drawn as almost straight (Holten and Van Wijk, 2009). The same effect is achieved by CUBu (Zwan, Codreanu, and Telea, 2016) by limiting the deformation factor of paths based on their length.

Visibility: In FDEB, Holten and Van Wijk, 2009 propose that edge-pairs which would create a skewed parallellogram should be bundled less than those which create a rectangle (see figure 4.6d). This produces smoother, but less tight, bundles, and as such this similarity term has not been widely adopted by later methods.

Area and ink: Compatible edges can also be defined implicitly as those edges which, when bundled, optimize a global cost function describing the quality of the bundling $B(D(T))$. For instance, Gansner and Koren, 2006 recursively cluster paths bottom-up as long as the resulting bundling improves the usage of the drawing area. Similarly, MINGLE (Gansner et al., 2011) groups edges as long as the amount of ink being used to draw $B(D(T))$ is lower than the amount used to draw the unbundled $D(T)$ (see also section 3.1.1). To achieve this, MINGLE computes the centroids between two sets of edges and then the amount of ink used is minimized by using the golden section search procedure (Press et al., 1989) finding two points along the line linking the centroids, as illustrated in figure 4.6e. Both methods are greedy, so they cannot guarantee a global minimum.

Image-Based Similarity

Apart from assessing which paths can be bundled (κ), one needs to make sure that bundled paths are closer than the unbundled ones. To measure bundled path distance, we need thus to define the function δ introduced in our **Bundling equation**. As stated there, δ is typically a form of Hausdorff distance: Given two sampled paths $\mathbf{d}_i = (\mathbf{x}_i^k)$ and $\mathbf{d}_j = (\mathbf{x}_j^k)$ which are compatible, *i.e.* make sense to be bundled, we have

$$\delta(\mathbf{d}_i, \mathbf{d}_j) = \sum_{\mathbf{x}_i^k \in \mathbf{d}_i} \min_{\mathbf{x}_j^k \in \mathbf{d}_j} \|\mathbf{x}_i^k - \mathbf{x}_j^k\| + \sum_{\mathbf{x}_j^k \in \mathbf{d}_j} \min_{\mathbf{x}_i^k \in \mathbf{d}_i} \|\mathbf{x}_i^k - \mathbf{x}_j^k\|. \tag{4.1}$$

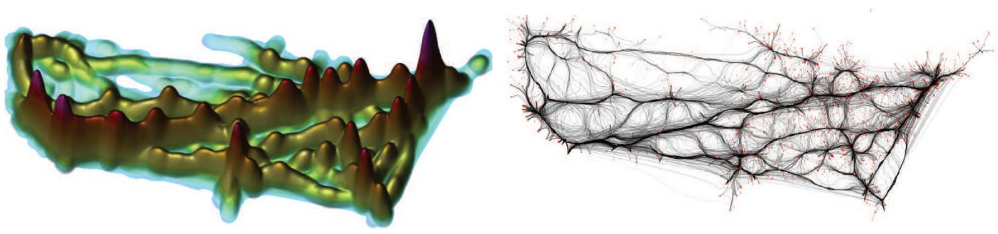
If we do not know upfront which paths are compatible with each other, we thus need to find, for each sample point of a path, the closest sample point of *all* other paths. While this can be accelerated by using various spatial structures such as *kd*-trees (Gansner et al., 2011), quadtrees (Lambert, Bourqui, and Auber, 2010b), this process is very expensive. Image-based methods address the evaluation of Eqn. 4.1 *and* the finding of compatible paths (evaluation of κ) by using imaging and/or GPU methods.

The first method in this class is SBEB (Ersoy et al., 2011) (section 3.2.1). Here, $\delta(\mathbf{d}_i, \mathbf{d}_j)$ is evaluated as the sum $\delta(\mathbf{d}_i, S) + \delta(\mathbf{d}_j, S)$ where S is the medial axis of the group of compatible paths that $\mathbf{d}_i$ and $\mathbf{d}_j$ are part of. Path-to-skeleton distances are evaluated as

$$\delta(\mathbf{d}_i, S) = \sum_{\mathbf{x}_i^k \in \mathbf{d}_i} DT_S(\mathbf{x}_i^k) \tag{4.2}$$

where $DT_S : \mathbb{Z}^2 \to \mathbb{R}^+$ is a pixel map (image) capturing the so-called Euclidean distance transform of the skeleton S (Fabbri et al., 2008). Since there are far fewer clusters (thus skeletons) than paths in $D(T)$, and Eqn. 4.2 can be efficiently implemented using fast-marching methods (Telea and Wijk, 2002; Sethian, 2003), but also GPU methods (Cao et al., 2010), δ can be efficiently computed.

An alternative is proposed by kernel density estimation (KDE) methods. As stated in section 3.2.1, a path density map $\rho : \mathbb{Z}^2 \to \mathbb{R}^+$ is computed from $D(T)$. Next, path sampling points $\mathbf{x}_i^j$ are moved upstream towards the gradient of density map ($\nabla\rho$). This *implicitly* minimizes δ by gradient ascent without having to explicitly find closest sample point-pairs. Informally put, we can see the opposite of the density map (see figure 4.7)



(A) Density Map (B) Bundled Image

FIGURE 4.7 – A density map and its corresponding bundling image by Hurter, Ersoy, and Telea, 2012 for the US migrations graph.

as a map of potential wells were edges are moved towards the closest well. The basic KDEEB method (Hurter, Ersoy, and Telea, 2012) supports only undirected bundling. KDEEB was enhanced by weighting the computation of ρ by various attributes like path directions and data attributes (Peysakhovich, Hurter, and Telea, 2015; Moura, 2015). Moreover, computation speed-ups of ρ are proposed: CUBu improves with respect to KDEEB by using a gathering, rather than scattering, convolution strategy, which parallelizes much more efficiently on GPUs (Zwan, Codreanu, and Telea, 2016). Lastly, FFTEB by Lhuillier, Hurter, and Telea, 2017a improves on CUBu by up to one order of magnitude by executing the convolution in frequency space (using the properties of the Fast Fourier Transform) rather than in image space (see chapter 6).

4.2 Bundling Operator Definition

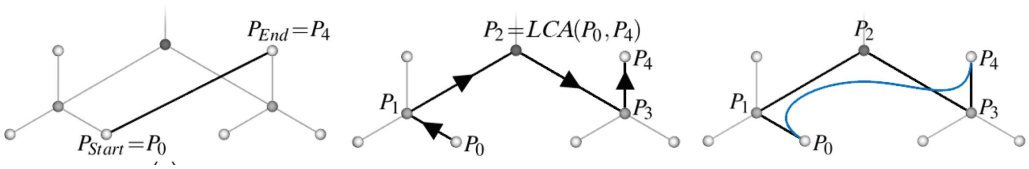
Having the functions δ and κ implemented as described in section 4.1, one can now define the core of a bundling method – the bundling operator B. In this section, we details the existing implementations of the **Bundling equation**. We classify existing bundling methods in *explicit* and *implicit* ones, as follows.

4.2.1 Explicit Methods

Explicit methods compute an intermediary structure I from $D(T)$ and define B based on I. The structure I is computed typically only once, after which paths are routed according to it. As such, explicit methods are in general fast and predictable, but less flexible in terms of bundling control. As already outlined, different types of structure I exist. The choice between structures is mainly dependent on the input data-set;

Hierarchical methods mainly use the explicit hierarchy provided by a compound graph (see Holten, 2006; Cornelissen et al., 2008; Wettel and Lanza, 2007; Giereth, Bosch, and Ertl, 2008; Caserta, Zendra, and Bodénes, 2011). For Holten, 2006 the path along the hierarchy is used as the control polygon of a spline curve as detailed in figure 4.8. Pupyrev, Nachmanson, and Kaufmann, 2010 use a different approach by extracting a tree from a directional acyclic graph with Sugiyama, Tagawa, and Toda, 1981's technique.

General-graph methods extract trees or meshes to define the control structure I. Typically bundling associated with flowmaps or confluent drawing visualization uses



(A) Straight Line (B) Hierarchical Path (C) Spline

FIGURE 4.8 – Holten, 2006's explicit bundling operator for hierarchical graphs. HEB uses the available hierarchy to use it as the control point for the spline.

trees such as spanning trees (Phan et al., 2005), Steiner trees (Buchin, Speckmann, and Verbeek, 2011a), or spiral trees (Buchin, Speckmann, and Verbeek, 2011b).Winding roads (Lambert, Bourqui, and Auber, 2010b and GBEB (Cui et al., 2008) uses a mesh generation such as Voronoï diagrams and/or Delaunay triangulation to route edges.

Once I exists, edges are simply routed along it, using various forms of smooth curves, *e.g.* B-splines (Holten, 2006; Wettel and Lanza, 2007; Giereth, Bosch, and Ertl, 2008; Caserta, Zendra, and Bodénes, 2011; Piegl and Tiller, 1997); Bézier splines (Brandes and Wagner, 1998); NURBS (Qu, Zhou, and Wu, 2006); and cubic curves (Graham and Kennedy, 2003).

4.2.2 Implicit Methods

We discuss next *implicit* methods which define B recursively by an iterative optimization process. Methods in this class are all force-directed techniques working in geometry space $I \subset \mathbb{R}^2$ and image-based techniques working in discrete image space $I \subset \mathbb{Z}^2$.

Implicit methods work in a self-organizing way, without computing an explicit control structure I upfront. As such, they avoid the problems of explicit methods due to computation of a suboptimal I – bundling can 'self-correct' itself during the process. These methods work typically in an iterative way, similar to optimization processes which aim to find the extremum of a global cost function.

Force-based techniques are the first, and best known, class (see Holten and Van Wijk, 2009; Nguyen, Hong, and Eades, 2011; Nguyen, Eades, and Hong, 2012; Selassie, Heller, and Heer, 2011; Böttger et al., 2014). They compute the gradient of a cost function of the form

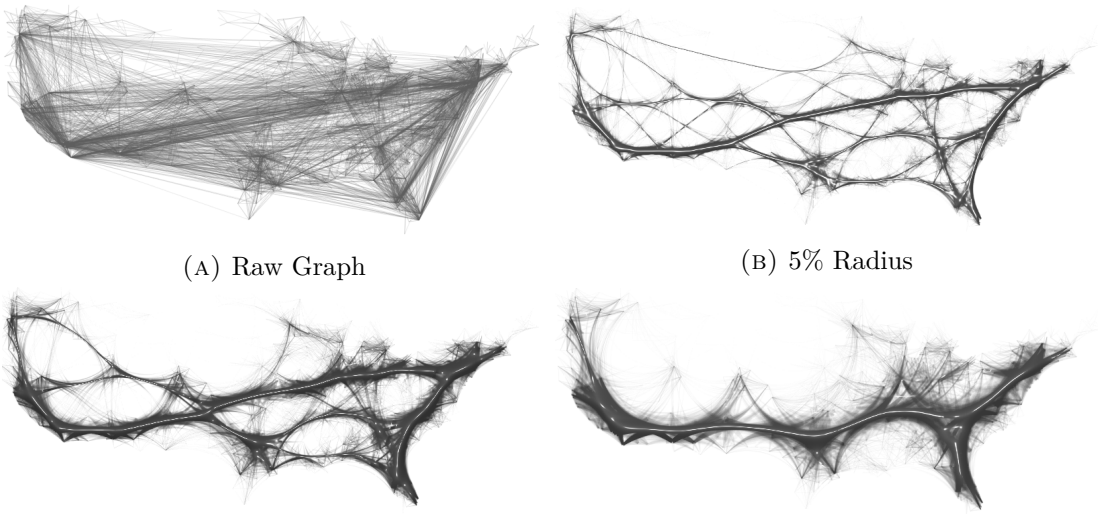
$$C(D(T)) = \sum_{\mathbf{d}_i \in D(T), \mathbf{d}_j \in D(T), \kappa(\mathbf{d}_i, \mathbf{d}_j) < \kappa_{min}} \delta(\mathbf{d}_i, \mathbf{d}_j), \tag{4.3}$$

with δ given by Eqn. 4.1, and iteratively shift, or advect, sample points $\mathbf{x}_i^j$ to minimize C, *i.e.*, yield tight bundles. This idea is very similar to force-based graph layouts (Kamada and Kawai, 1989), albeit using a different cost function. These methods are much slower than explicit ones, as the cost C (equation 4.3) has to be recomputed at each iteration, and computation of δ is expensive, as explained in section 4.1.2. Using spatial search or clustering structures (section 4.1.2) does not alleviate this, as path sampling points are continuously changing over iterations, so such structures would need repeated re-initialization.

Image-based methods are very similar to force-based methods, with the key difference that the cost computation (equation 4.3) is much lower due to the fast GPU-based evaluation of δ (section 4.1.2). Various update ideas are proposed here: Sample points are shifted upstream in the gradient of the skeleton's distance transform ∇DT_S by SBEB; and in the gradient of the density map $\nabla \rho$ by KDEEB, ADEB, CUBu, and FFTEB, respectively. Additionally, recent implicit image-based methods (CUBu,

FFTEB) implement B fully on the GPU (sampling paths $\mathbf{d}_i$ to points $\mathbf{x}_i^j$; computation of the density map ρ; shifting sample points in $\nabla\rho$; and rendering the final results). This yields speed-ups of up to two order of magnitude with respect to earlier image-based methods (KDEEB,ADEB).

Implicit methods add the extra capability of producing *multiscale* bundling in a scale space sense (Koenderink, 1984): Given the κ_{min} constraint in Eqn. 4.3, only edges closer than a certain distance δ_{max} are considered to interact. Setting δ_{max} to small values produces fine-grained bundles, where the amount of deformation of any path point is lower than δ_{max}. Setting δ_{max} to large values produces coarse bundles which exhibit more deformation but reduce clutter more too. Using a δ_{max} bound also massively reduces the costs of computing the all-pair path similarities $\delta(\mathbf{d}_i, \mathbf{d}_j)$ to a small subset. Image-based methods implement multiscale bundling efficiently, as κ_{max} is essentially controlled by the KDE kernel radius R (section 4.1.2). For example, bundling with larger radius yields a much larger density as depicted in figure 4.9 a KDE based multiscale bundling of the US migration graph. Rising the kernel radius yields more bundled results with a higher edge density per bundle. Multiscale bundling is much harder to achieve by explicit methods, as these do not provide a *continuous* parameter to control the scale of the simplification.



(A) Raw Graph (B) 5% Radius

(C) 10% Radius (D) 20% Radius

FIGURE 4.9 – Multiscale bundling of the US Migration graph using a KDE based bundling technique; FFTEB by Lhuillier, Hurter, and Telea, 2017a.

4.3 Bundling Visualization

After bundling B computes a bundled drawing $B(D(T))$ of a path-set $D(T)$, several mechanisms are available for visually presenting $B(D(T))$. These serve multiple purposes: display the bundling, enhance important structural elements thereof, add attribute data atop it, further reduce visual clutter, simplify the displayed image, and get

details on demand. Visualization techniques for bundled data can be grouped into the following classes: blending, data color mapping, shading, smoothing and deformation, animation, and interaction, as described next.

4.3.1 Blending

As outlined in section 2.3.1, "bundling trades clutter for overdraw" (Hurter, Ersoy, and Telea, 2013). Multiple paths (or path fragments) of $D(P)$ become overlapping in $B(D(P))$. However, many tasks require assessing the connection strength, or number of paths, between groups of nodes or endpoints in P. To support this, blending can be used: $B(D(P))$ is drawn using alpha blending or variations thereof. This renders bundles containing many paths more opaque than sparse ones, thereby attracting the user's attention more (Holten, 2006). Essentially, this maps the local bundle density ρ (section 4.1.2) to opacity, color, or shading, an idea pioneerd first by Van Liere and De Leeuw, 2003 for drawing unbundled straight-line graphs, and used next in many other contexts (*e.g.* Scheepens et al., 2011a; Scheepens et al., 2011b; Lampe and Hauser, 2011). All examples in figure 3.15 are rendered using density-based blending. Figure 4.10 shows blending for a detail of HEB (Holten, 2006).

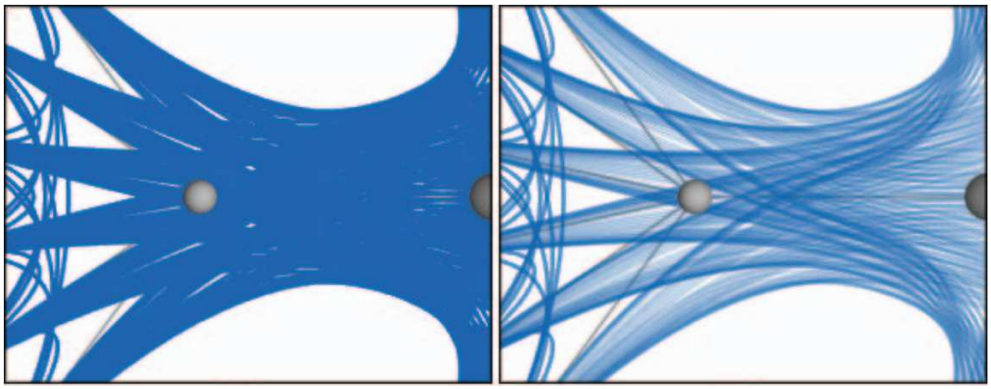
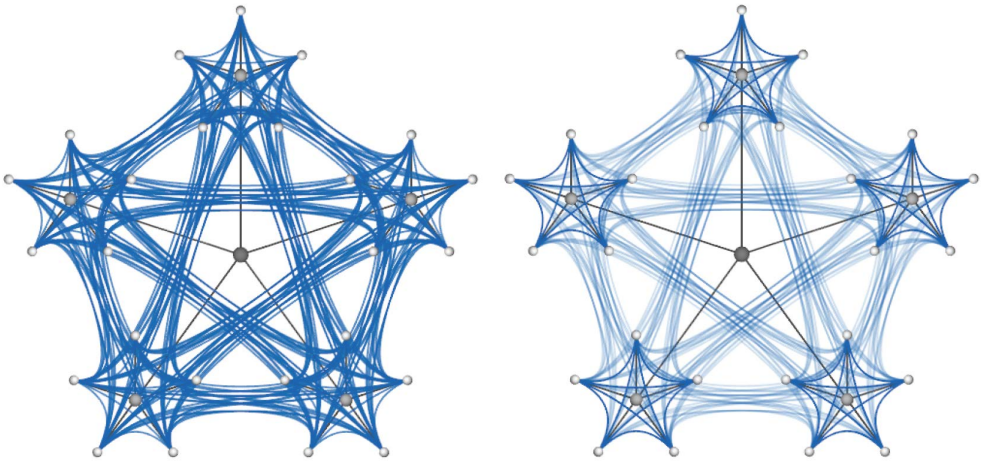


FIGURE 4.10 – Usage of blending to emphasize path density by Holten, 2006. Left: Drawing $B(D(P))$ without blending. Right: Same drawing with blending.

While conceptually simple, blending requires some care: Simply drawing the bundled paths $B(D(P))$ with OpenGL alpha blending will not work, as the typical resolution of an OpenGL alpha channel is 8 bits, which can capture only 256 different values. As many more paths can overlap, the result can easily be fully saturated (opaque). The solution is to map the normalized local bundled-path density ρ, computed using accurate floating-point operations (section 4.1.2), to the 8-bit alpha channel. However, for large bundled path-sets, where the path density can hugely vary between sparse paths and dense bundles, this will make sparse paths disapear. To answer this problem, Zwan, Codreanu, and Telea, 2016 proposes mapping a non-linear transfer function for the bundled-path densities ρ that emphases low densities.



(A) No blending (B) Alpha blending

FIGURE 4.11 – Holten, 2006's example of alpha blending used to emphasize short paths.

Separately, short paths can be easily obscured by long ones. The solution to this is to set path opacity inversely proportional to path length. Holten, 2006 shows a short example reproduced in figure 4.11. Additionally, CUBu draws short paths atop longer ones, so that the former get a fairer chance to stand out in the final image (figure 3.15k,l). However, this requires sorting paths by length, which can be costly ($O(NlogN)$ for N paths in $D(P)$).

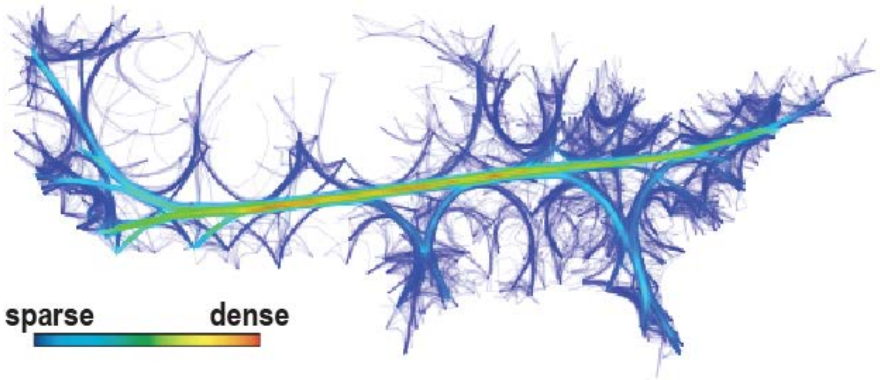
4.3.2 Data Color Mapping

Bundled path colors can be mapped to various path attributes, *i.e.* map data from P to the color visual variable in $B(D(P))$. Data includes:

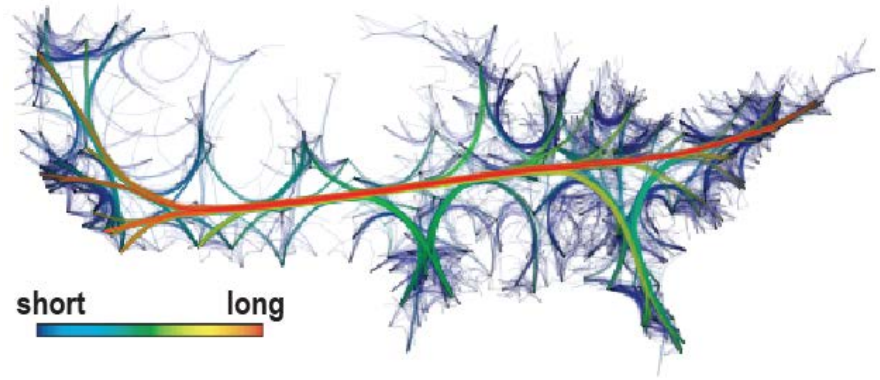
- *Path spatial density* to multiple-hue colormaps. This idea extends the blending idea detailed in section 4.3.1.
- *Path start-to-end direction* using a color gradient as outlined in figure 3.2a and in figure 5.2d. This idea follows earlier methods used for the same goal when drawing straight-line graphs (Diehl, 2007).
- *Cluster* containing the path showed in figure 3.2c, figure 4.13 and illustrated by McDonnell and Mueller, 2008.
- *Type of path* outlined in figure 5.2a by Reniers et al., 2014.
- *Spatial path orientation*, before bundling, using an angle-to-color map such as the one used by 2D directional bundling techniques in figure 3.11.
- *Local height* along spatial airline trails (see Hurter et al., 2014c).

It is worth noting that path directions are rarely indicated by arrows (Reniers, Voinea, and Telea, 2011; Cornelissen et al., 2008) or tappered-line, as these clutter quite easily and become hard to discern when many paths end at the same rather small node, as also discussed by Holten et al., 2011.

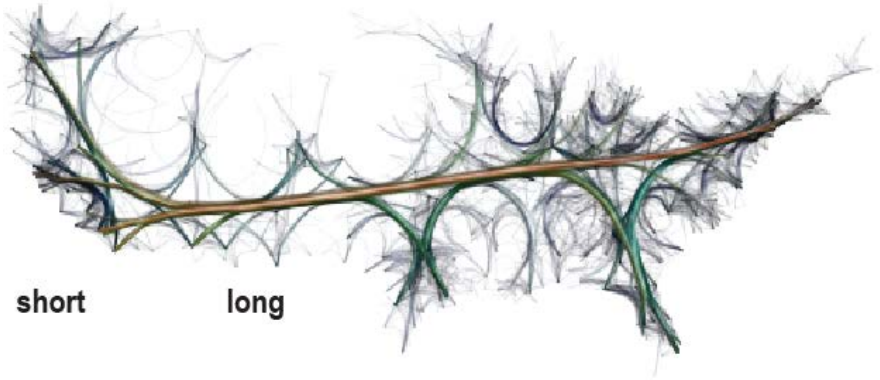
A serious issue here is color blending: When paths overlap, what should be the resulting color? This is especially hard to solve if color maps categorical attributes, which cannot be averaged (as explained by Telea and Ersoy, 2010). To date, no conclusive answer exists to this issue. This is further discussed by Hurter et al., 2014a, Klein, Van Der Zwan, and Telea, 2014 and Zwan, Codreanu, and Telea, 2016.



(A) Local edge density



(B) Edge length



(C) Edge length with shading

FIGURE 4.12 – Color mapping and shading using CUBu (Zwan, Codreanu, and Telea, 2016).

Shading

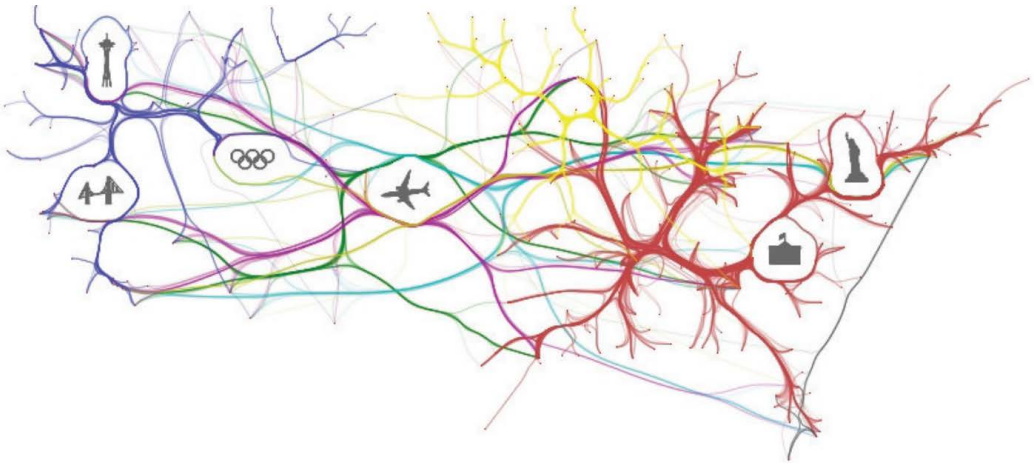
Although bundling reduces the visual complexity of a drawing, it is still hard to discern salient (important) connections, even when using alpha blending (section 4.3.1). To emphasize the spatial extent of a bundle (group of spatially close paths in $B(D(T))$), *pseudo-shading* can be used, which was pioneered by Image-Based Edge Bundles (IBEB, Telea and Ersoy, 2010), see figure 3.2c. For this, IBEB, but also WR, KDEEB, and CUBu create a false height map increasing parabolically from the borders of a bundle to its center, and then shade this by classical Phong shading, yielding similar effects to the well-known cushion treemaps (Van Wijk and Wetering, 1999). This allows better visual separation of crossing bundles, which show up as shaded tubes, based on the luminance variation from the tube borders to their centers, given a 3D stacking effect (see figure 3.2c and 5.4a,b). To do this, IBEB and SBEB need to explicitly cluster the bundled paths, which, as discussed in section 4.1, is delicate. The same explicit path clustering is used by the geometric-shaded bundles by Trümper, Döllner, and Telea, 2013 (figure 5.2c). WR and CUBu do the path grouping implicitly, which makes them the methods of choice. A different approach is to use bump mapping to highlight high-density bundles based on the slope of the density map ρ (introduced in section 4.1.2). While simple to implement, this yields less well-separated bundles, and creates artificial visual discontinuities in their middle (Hurter, Ersoy, and Telea, 2012).

Shading and color mapping can be both combined as outlined in figure 4.12: We can use colors to map either local edge density (figure 4.12a) or edge length (figure 4.12b), thereby emphasizing dense regions, respectively long edges, respectively. Adding shading to the latter combines the effects – salient shaded tubes show dense regions, while color maps edge length, respectively.

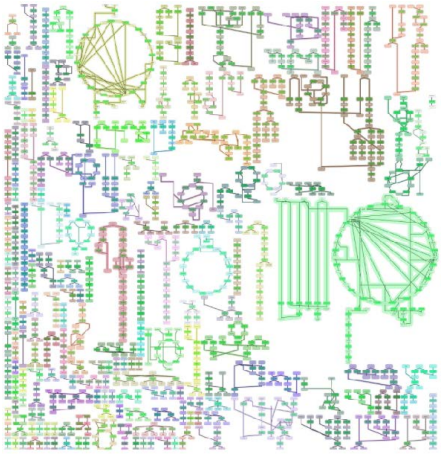
4.3.3 Smoothing and Deformation

Explicit bundling techniques control well the local curvature of resulting bundles, as they route the bundled paths along a precomputed global structure I (section 4.2.1). Implicit techniques (section 4.2.2) have less control, so they postprocess the bundled path-set drawing $B(D(T))$ to achieve smoothness. The by far most common way for this is to apply 1D Laplacian smoothing (Hansen, Douglass, and Zardecki, 2005) on the bundled paths, as introduced by Holten and Van Wijk, 2009. This technique is simple, fast ($O(N)$ for a drawing $B(D(T))$ sampled with N points), and easily controllable.

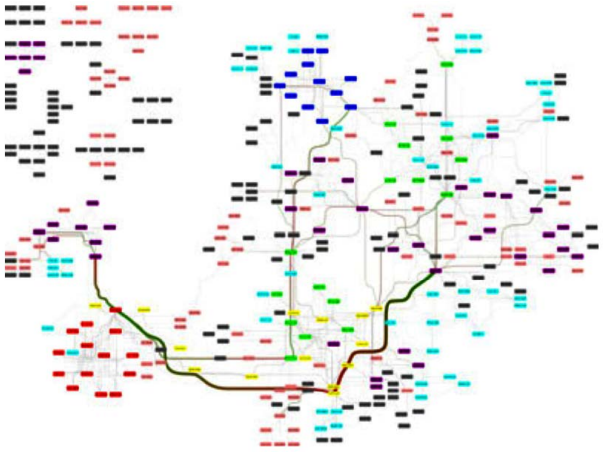
Bundling can also include *deformation* constraints. For instance, KDEEB (Hurter, Ersoy, and Telea, 2012) can route bundles to avoid landmarks in the drawing space $\mathbb{Z}^2$ by using essentially the same principle as "dust & magnets" (Yi et al., 2006), but with repulsion instead of attraction (figure 4.13a). The technique can be easily integrated in any image-based implicit bundling method, either during the iterative bundling or as a postprocessing step. WR (Lambert, Bourqui, and Auber, 2010b) and TGI-EB (Nguyen, Hong, and Eades, 2011) can include bundle orientation constraints, yielding results



(A) Landmarks avoidance, Hurter, Ersoy, and Telea, 2012



(B) Orthogonal drawing, Lambert, Dubois, and Bourqui, 2011



(C) Orthogonal drawing, Nguyen, Hong, and Eades, 2011

FIGURE 4.13 – Bundle deformation to avoid landmarks and favor orthogonal drawing.

that follow the so-called "metro map" drawing style (figure 4.13b and 4.13c). Other deformation mechanisms support interactive exploration (see section 4.3.5 further).

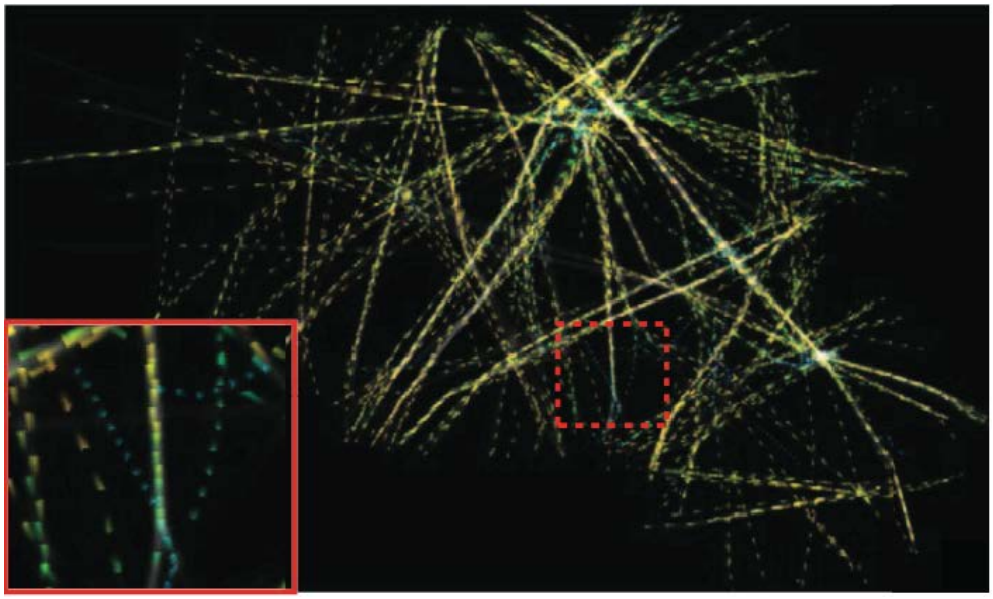
4.3.4 Animation

Adapted to bundling techniques, animation can be used for two main goals:

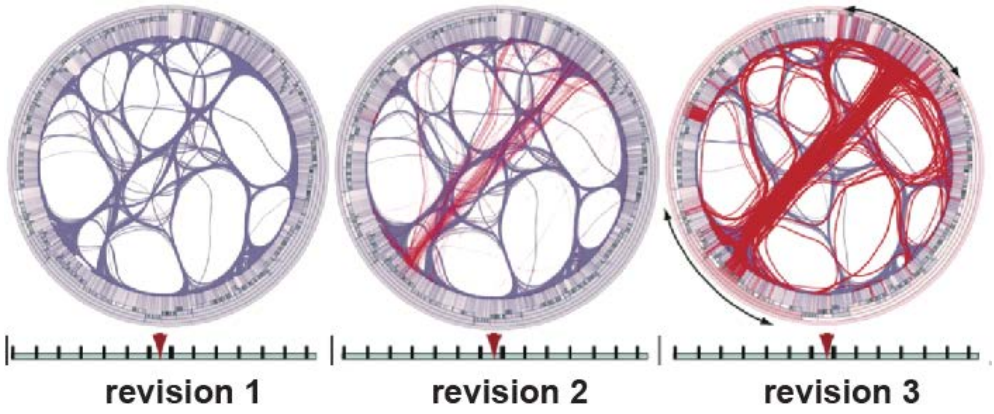
First, it can convey path directions in a trail-set. For this, small point-like textures, also called particle systems, encode the local path direction (Holten et al., 2011) and are drawn at regular intervals along the bundled paths and temporally shifted in the path direction (Hurter et al., 2014a; Klein, Van Der Zwan, and Telea, 2014). As several such textures coherently move towards the same direction, this yields a moving effect along bundles that conveys a bundle's direction. This design is related to older methods for visualizing flow fields (Van Wijk, 2002). Figure 4.14a shows an example using tapered arrow-like textures (Holten et al., 2011) that encode path directions. However, as for data color-mapping (section 4.3.2), animation has the issue that, if a bundle contains paths having different directions, the result can show random noise patterns from which

we cannot discern the fraction of paths going one way *vs* the opposite way. Note that this is not a problem for the underlying technique (IBFV by Van Wijk, 2002) since, for that case, a 2D vector field has a *single* data value per location.

Secondly, animation can show changes in a time-dependent path-set $P(t)$, by interpolating the bundled trails either during keyframes (for sequence graphs, see section 3.1.2) or by continuously morphing a trail-set to account for incoming and leaving trails (for streaming graphs, see section 3.1.2). Figure 4.14b shows three frames from a bundled sequence-graph depicting code clones between software components in three revisions of a software system (Hurter, Ersoy, and Telea, 2013). Red edges show newly appearing clone relations, which are bad for system maintenance, and thus should be spotted and removed.



(A) Textures showing bundle directions (Hurter et al., 2014a).



(B) Sequence graphs (Hurter, Ersoy, and Telea, 2013).

FIGURE 4.14 – Animation techniques used in conjunction with bundling techniques.

4.3.5 Interaction

Interaction is a key technique to the analysis of bundled drawings. Following Brehmer and Munzner, 2013, interaction allows answering *how*-type questions; and enables bundled drawing $B(D(P))$ to support *select*, *navigate*, *filter* and *arrange* tasks. Several types of interaction exist in this context, as follows:

Relaxation: Linear interpolation between the input drawing $D(P)$ and its bundling $B(D(P))$ is used by virtually all bundling methods, starting with Holten, 2006, to control the bundling "tightness". Interactively changing the interpolation parameter to-and-fro allows users to visually link paths in $D(P)$ and $B(D(P))$ and thus resolve (parts of) the bundling ambiguities created by overlap. Similar techniques are used to link items in other visualizations (Hurter et al., 2014b). Figure 4.15a-d shows four frames from a relaxation process;

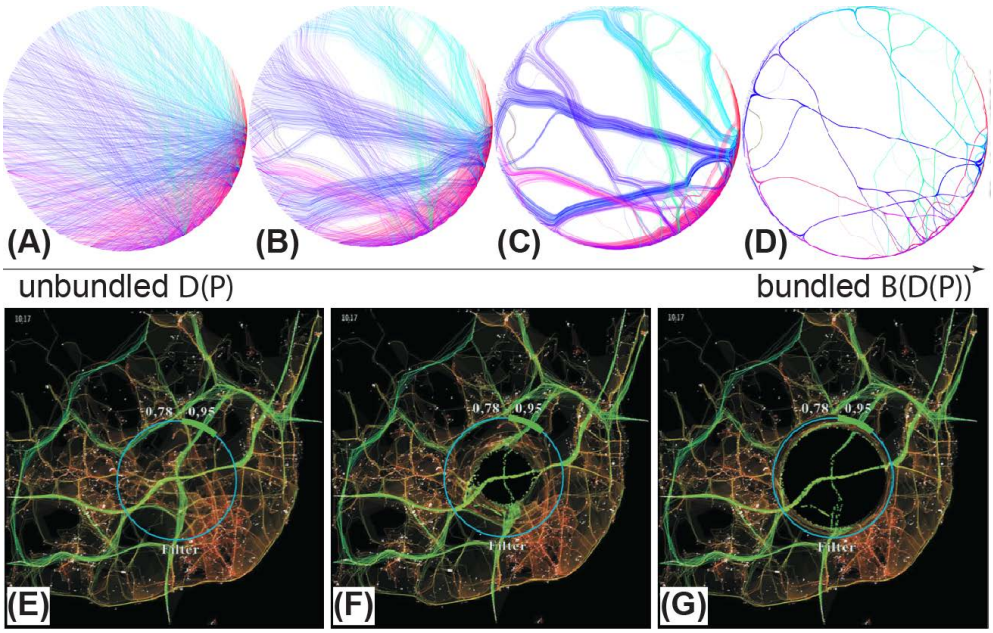


FIGURE 4.15 – Relaxation of bundled drawings: (a-d) Global bundle relaxation (Hurter, Ersoy, and Telea, 2012) and (e-g) local relaxation (Hurter, Telea, and Ersoy, 2011).

Lenses: Bundling can be applied (or prevented) locally on $D(P)$. This way, one can combine views of the unbundled $D(P)$ with the bundled $B(D(P))$. This is essentially a local relaxation variant. Figure 4.15e-g shows three frames from such a local lens, used to get detail on a road-traffic dataset (Hurter, Telea, and Ersoy, 2011). Lambert, Auber, and Melançon, 2010 propose additional variants, such as fisheye and bring & go techniques (Tominski et al., 2006), using GPU-computed splines for interaction fluidity (figure 4.16a). Techniques such as edge plucking (Wong and Carpendale, 2007) can be easily added. The digging lens by Telea and Ersoy, 2010 alleviates occlusion in shaded-tube bundle renderings (section 4.3.2): Bundles are thinned close to the interaction

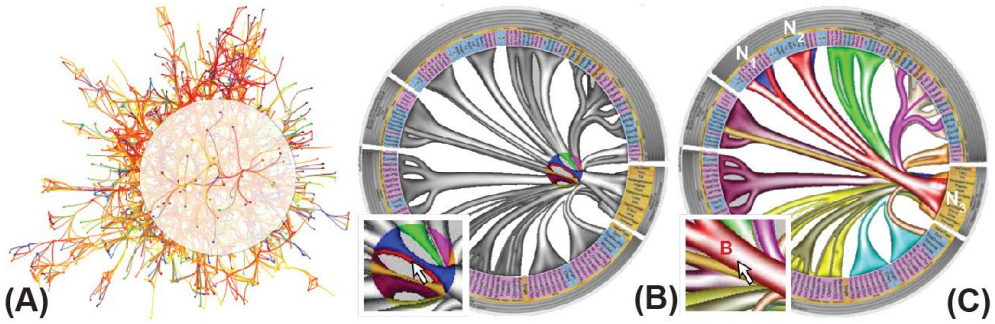


FIGURE 4.16 – Interacting with lenses in bundled drawings: (a) Magic lens (Lambert, Auber, and Melançon, 2010). (b,c) Digging lens (Telea and Ersoy, 2010).

focus using image-processing techniques (figure 4.16b) so that one can see, and bring to front, occluded bundles from beneath (figure 4.16c).

Brushing: Brushing bundles can reveal aggregated attributes of the overlapping paths in $B(D(P))$, thereby alleviating the problem outlined in section 4.3.2 (see Cornelissen et al., 2008; Reniers et al., 2014).

Navigation: HEB-like methods (section 3.1.1) can support navigation of the hierarchy T by interactively collapsing (clustering) or opening (refining) nodes in T. Such operations trigger the display of a different set of bundled edges (Cornelissen et al., 2008; Reniers et al., 2014; Hanjalic, 2013).

Riche et al., 2012 define interactions and design guideline to support node and edge manipulations in bundled drawings. In their implementation of an interactive bundling process, they propose interactive bundling as a technique which intents to allow users to group together arbitrary edges subset that are locally spatialized. Separately, Luo et al., 2012 propose interactive techniques to decrease path overlapping issues.

4.4 Future Challenges

In previous sections, we paved the way towards an unified technical framework of bundling methods. However, despite having explained the various implementations of bundling methods we still lack some design guidelines on how to correctly configure the various parameters of bundling methods. Thus controlling bundling methods is a major technical challenge for the bundling landscape.

Earlier bundling methods have a relatively small, and intuitive, set of parameters. For instance, HEB (section 3.1.1) allows controlling the strength of bundling, the amount of relaxation, detection of the common ancestor in the hierarchy that bundles are routed through, and type of edge blending (for full details, we refer to Holten, 2006). Recent bundling methods have a higher number of parameters. For instance, CUBu (section 3.2.1) allows controlling the resolution of the image used for kernel density

estimation, radius of the kernel, number of sampling points along paths, advection step size, number of bundling iterations, directional bundling style, type of bundle shading, and offers additionally four drawing styles (full details in Zwan, Codreanu, and Telea, 2016).

Concerning parameters, we see the following challenges:

Semantics: Existing parameters have semantics which are typically bound to a specific bundling technique. This makes it hard for users to reproduce results when changing techniques. Our mathematical framework presented in this survey helps this by identifying parameters having identical meanings over different methods, but of course cannot fully solve the problem. A promising direction would be to provide parameters linked to user tasks rather than method technicalities. For instance, if one wants to bundle trails so that a given subset thereof is easily followable, the method could automatically select suitable technical parameter values.

Impact: Changing any of the many bundling parameters produces a different result. While reasonable parameter presets exist, these are not always optimal for all datasets and/or visual insights sought. This issue is less critical with modern bundling methods which work near-real-time (section 3.2.1), as trial-and-error parameter exploration is less costly. Still, supporting users to express their interest more effectively, by offering high(er) level parameter control, similar to work done elsewhere in infovis (Shrinivasan and Wijk, 2008; Shrinivasan and Wijk, 2009), is a potential improvement direction for bundling. Another solution would be setting parameter values based on the characteristics of the input data $D(P)$ to bundle.

Coupling: The same bundling result can be, usually, obtained by setting different parameters to different values. For example, the same bundle tightness in implicit methods (Holten and Van Wijk, 2009; Hurter, Ersoy, and Telea, 2012; Moura, 2015; Zwan, Codreanu, and Telea, 2016) can be obtained by changing either the number of bundling iterations or the advection step size. We call such parameters *coupled*. Few papers discuss coupling, making the parameter space unnecessarily large. This can be alleviated by grouping coupled parameters under a single high-level parameter, similarly to the idea discussed above for impact.

4.5 Summary

To conclude this chapter, we showed that all existing bundling methods can fit into our proposed unified framework. And over the year and improvements, we believe that bundling methods have become quite mature. And as detailed in section 2.3, throughout their maturation bundling techniques have tried to answer the challenge of scalability.

We distinguish four "generations" of bundling techniques (figure 4.17). Pre-HEB bundling techniques were able to handle graphs of hundreds of edges and worked solely on the CPU (*e.g.* Newbery, 1989; Brandes and Wagner, 1998; Dickerson et al., 2003; Phan et al., 2005). Following HEB (Holten, 2006), the second generation targeted graphs and trail-sets having thousands (up to roughly 10K) paths; MINGLE (Gansner et al., 2011) is an exception here, as it targeted graphs up to 1 million edges. These techniques worked mostly geometrically (see Holten and Van Wijk, 2009; Nguyen, Eades, and Hong, 2012; Cui et al., 2008; Nguyen, Hong, and Eades, 2011; Luo et al., 2012; Selassie, Heller, and Heer, 2011) and on the CPU. Starting with SBEB (Ersoy et al., 2011) and KDEEB (Hurter, Ersoy, and Telea, 2012), third-generation techniques are image-based, using a mixed CPU-GPU implementation to massively parallelize both similarity computation (section 4.1.2) and the bundling itself (section 4.2), and can bundle tens of thousands of paths in seconds (see Lambert, Bourqui, and Auber, 2010b; Lambert, Bourqui, and Auber, 2010a; Moura, 2015). Before the introduction of fourth-generation techniques, bundling truly large data-set was still a challenge due to two factors *computational speed* and *data-volume*. Fourth-generation techniques, introduced and detailed in chapter 6, aims at solving these challenges. They work solely on the GPU, can handle multiple GPUs (CUBu by Zwan, Codreanu, and Telea, 2016), can bundle up to a million paths in subsecond time, and remove GPU RAM limitations by data streaming (FFTEB by Lhuillier, Hurter, and Telea, 2017a). With even faster and larger-memory GPUs emerging continuously, we believe that the scalability problem of path bundling has been sufficiently addressed, so that bundling can approach "big data" sets.

In the previous chapter, we proposed a data-based taxonomy of bundling methods and in this chapter we have defined a unified bundling framework. However, to be able to fully use and understand the capabilities of the bundling methods, we still need to define and show what type of tasks it can answer.

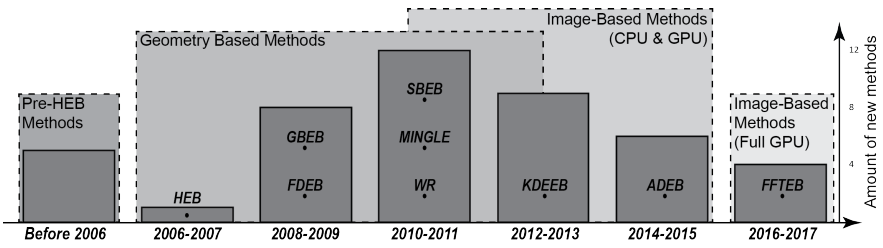


FIGURE 4.17 – Temporal distribution and various generations of the main bundling methods (as defined in table 3.2).

Chapter 5

Bundling Tasks and Applications

As outlined already in the previous chapters, comparing bundling techniques is hard without a common accepted framework of reference (chapter 4). Moreover, relations of bundling with other simplification techniques (*e.g.* data clustering Jain, Murty, and Flynn, 1999 or imager simplification Siddiqi and Pizer, 2008) exist but are not fully analyzed. In this chapter, we propose an analysis of how bundling fits into Ellis and Dix, 2007's clutter reduction taxonomy (as presented in chapter 2) as well as description of a set of tasks bundling supports following Lee et al., 2006 and Brehmer and Munzner, 2013.

We discuss how bundling can fit to these tasks, and also where salient limitations exist, in terms of the operations of our proposed framework. We also discuss ways and challenges to compare the results of different bundling methods. By this, we aim at providing a guide to the practitioner in selecting, customizing, testing, and possibly extending existing bundling methods to support optimally a given task set. We believe this demonstration of bundling task capabilities will allow users to better understand whether bundling can be an answer to a specific (low or high level) visualization tasks.

Moreover, as bundling has gone far beyond simplified graph drawing, we overview a wide set of key bundling applications and relate these to the above-mentioned tasks. By this, we aim to address limitations of bundling papers which usually focus either on proposing a new technique or discussing a single application.

5.0 Résumé (Français) Tâches et Applications du Bundling

Comme détaillé dans les précédent chapitres, il est difficile de comparer l'ensemble des techniques de bundling sans un cadre de référence commun et accepté (*c.f.* chapitre 4). De plus, bien qu'il existe des relations entre le bundling et les autres méthodes de simplification visuelle, ces dernières n'ont pas encore été totalement explorées. Dans ce chapitre, nous analysons comment les techniques de bundling s'intègrent dans la taxonomie de réduction d'occultation de Ellis et Dix, 2007 (*c.f.* chapitre 2). Puis, nous proposons une description des tâches de visualisation (définies par Lee et al., 2006 et Brehmer et Munzner, 2013) auxquelles le bundling peut répondre.

Ensuite, nous montrons comment le bundling peut répondre aux tâches identifiées et montrons les limitations des techniques actuelles. Pour ce faire, nous analysons un large spectre des cas d'applications du bundling et les mettons en lumière au regards des tâches de visualisation à résoudre. Enfin, nous abordons aussi les approches et les challenges inhérents à la comparaison des méthodes de bundling. Ainsi, à travers cette approche, nous proposons aux utilisateurs un guide leur permettant de selectionner, modifier, tester et potentiellement améliorer les techniques de bundling existantes en fonction de la tâche à résoudre. De plus, cette approche nous permet de mettre en lumière les limitations ainsi que les potentiels axes d'amélioration du bundling au regard des besoins utilisateurs.

Un taxonomie de Réduction d'Occultation du Bundling

En suivant la taxonomie de Ellis et Dix, 2007 sur les techniques de réduction d'occultation dans une visualisation, nous montrons en table 5.1 les capacités du bundling tout en le comparant avec trois techniques relativement similaires (opacité, regroupement et déplacement de liens).

	opacité	regroupement	déplacement de liens	**bundling**
évite le chevauchement	en partie	possible	✓	en partie
garde les informations spatialisées	✓	en partie	✗	en partie
peut être localisé	✓	✗	✓	en partie
passe à l'échelle	✗	✓	✗	✓
est ajustable	✓	✓	possible	✓
peut montrer les attributs des courbes	✗	en partie	✓	✓
peut discriminer les courbes	✓	✓	possible	en partie
peut montrer la densité de chevauchement	✓	possible	✗	✓

Table 5.1 – Comparaison du bundling avec trois autres techniques de réduction d'occultation (Ellis et Dix, 2007).

Taxonomie de Tâches

Si l'on fonde notre analyse sur la taxonomie des tâches de graphes introduite par Lee et al., 2006, le bundling permet les tâches suivantes : le suivi de courbes (sous certaines conditions), la visualisation des densités de liens, l'identification de sous-ensembles, l'identification des cliques fortement connectées, la visualization d'ensemble, la recherche de structures et la comparaison de flux.

Les expérimentations de McGee et Dingliana, 2012 ont permis d'évaluer deux types de tâches réalisables avec le bundling sur des jeux de données hiérarchiques. Ils montrent que sans ajout de techniques d'interactions, le bundling n'améliore pas la capacité de suivi d'un lien dans un graphe. Néanmoins, il permet de reconnaître plus rapidement des sous-ensembles dans les jeux de données.

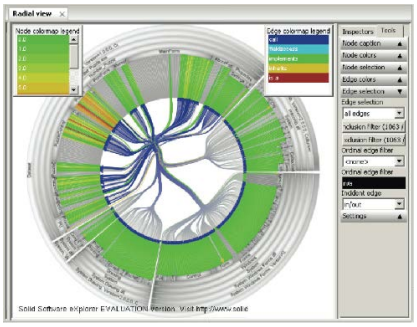
Cependant, le bundling ne permet pas simplement de trouver des informations concernant un nombre réduit de liens. Trouver une anomalie (*e.g.* un lien isolé) n'est pas une tâche supportée par le bundling. De même, connaître le nombre de liens d'un nœud n'est pas possible. Enfin, il est impossible de réaliser des tâches impliquant la notion de distance entre deux nœuds avec le bundling comme par exemple trouver le plus court chemin entre deux nœuds.

Parallèlement, en suivant le mantra de B. Schneiderman "Overview first, zoom and filter, then details-on-demand" [43] (i.e. vue d'ensemble en premier, zoom et filtrage puis détails sur demande), le bundling est un outil pour l'exploration macroscopique de graphes. Il permet la visualisation d'ensemble d'un graphe trop dense pour être analysé dans sa globalité. Et l'utilisation de critères d'agrégation (FDEB et ADEB) permet de réaliser une agrégation sélective en fonction de critères de compatibilité.

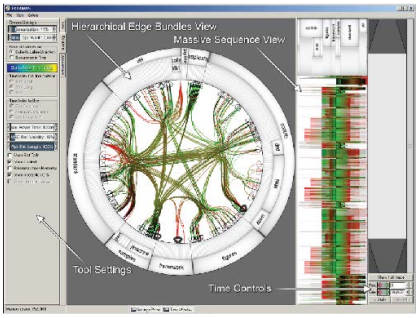
Domaines et Cas d'Applications du Bundling

Les algorithmes de bundling sont utilisés dans différents domaines et cas d'utilisation détaillés en section 5.2, résumé comme suit.

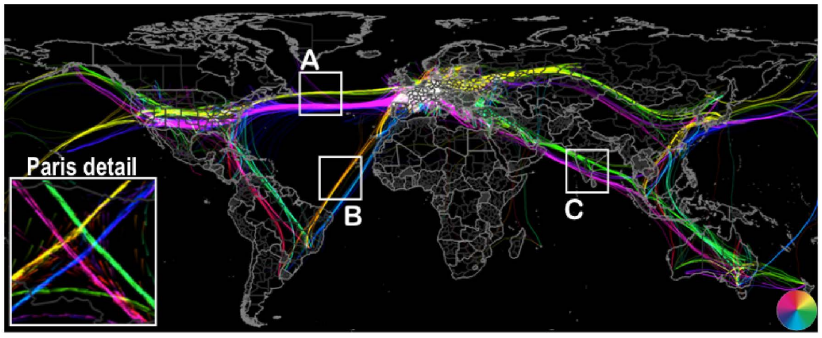
L'un des premiers cas d'utilisation du bundling apparaît avec HEB (Holten, 2006). Ce dernier applique le bundling à la visualisation de programme et les graphes hiérarchiques d'appels de fonctions (*e.g.* Figures 5.1a et 5.1b). Ersoy et al., 2011, Lambert, Bourqui et Auber, 2010b, Hurter, Ersoy et Telea, 2012 et Peysakhovich, Hurter et Telea, 2015 utilisent le bundling pour visualiser des flux. Deux exemples sont très présents dans la littérature : le graphe des migrations de population aux Etats-Unis ainsi que les trajectoires d'avions sur une journée aux Etats-Unis (*c.f.* figure 3.1). CUBu applique le bundling dans l'analyse des flux à travers l'exemple des flux d'avions autour du globe (figure 5.1c). De manière similaire, Mingle (Gansner et al., 2011) analyse la répartition des communications internet. KDEEB (Hurter, Ersoy et Telea, 2012) et ADEB (Peysakhovich, Hurter et Telea, 2015) introduisent un nouveau cas d'utilisation en analysant les tracés oculaires mesurés à l'aide d'un capteur oculaire *eye tracker*. Ce cas d'utilisation exploite l'application du bundling aux graphes orientés (Figure 5.1d).



(A) Solid* toolkit

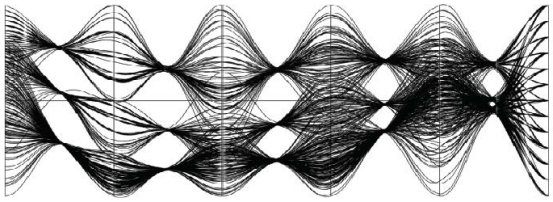


(B) Appels de fonctions

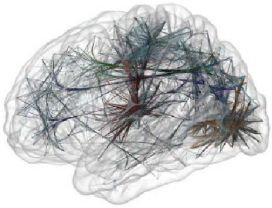


(C) Flux aérien mondial

(D) Tracés occulaires



(E) Coordonnées parallèles Heinrich et al., 2012



(F) IRM Fonctionnelles Böttger et al., 2014

FIGURE 5.1 – Exemples d'application du bundling

Pupyrev, Nachmanson et Kaufmann, 2010 appliquent le bundling pour améliorer la visualisation de graphes de machines à états. Böttger et al., 2014 agrègent les liens représentant des connexions neuronales (figure 5.1f). Enfin, Heinrich et al., 2012 implémentent le bundling afin de diminuer le rapport de surface encre/fond dans la visualisation de coordonnées parallèles *parallel coordinates* (Figure 5.1e). Bien que la plupart des exemples d'utilisation portent sur l'agrégation de graphes en deux dimensions, certaines techniques permettent l'agrégation dans un espace en trois dimensions (Lambert, Bourqui et Auber, 2010a). Pour exemple, ce bundling est possible avec HEB (en prenant une B-Spline 3D), FDEB (avec une distance euclidienne en 3D), KDEEB (avec une carte de densité 3D) ou encore SBEB (avec un squelette 3D).

Qualité et Véracité du Bundling

Dans cette section, nous discutons des défis non résolus inhérents aux techniques de bundling : la mesure de la qualité du resultat bundlé et les problèmes de véracité des informations bundlées.

Notre expérience acquise sur l'utilisation des algorithmes de bundling nous a montré que le résultat final agrégé dépend des valeurs prises pour les paramètres des algorithmes. Ceci est particulièrement vrai pour les paramètres ayant attraits aux structures de contrôle de l'algorithme, influençant fortement le résultat final. Le dessin final peut ainsi être très fortement agrégé ou partiellement. L'expérience montre aussi qu'il est difficile de déterminer quels sont les bons paramètres sans de nombreux essais préalables. De plus, aujourd'hui, il n'existe pas de métriques ou d'outils permettant de quantifier le résultat d'un algorithme de bundling. La seule métrique disponible est le rapport de surface encre/fond proposé par Tufte, 1992 mais qui ne permet pas de qualifier pleinement le résultat d'un algorithme. Pour résoudre ce défi, nous distinguons deux pistes de recherche : adapter le bundling et ses paramètres à la tâche utilisateur et fournir des jeux de données de référence pour comparer les résultats des dessins aggregés.

Pour terminer sur les limitations du bundling, le facteur le plus polémique réside dans la véracité de la visualisation graphique (Nguyen, Eades et Hong, 2013, Nguyen, Eades et Hong, 2017). Nous distinguons deux niveaux d'honnêteté de la visualisation : celui de la véracité théorique du graphe bundlé et enfin celui de la véracité aux yeux de l'utilisateur. Pour ce qui est de la véracité théorique d'un graphe agrégé avec les méthodes de bundling, à ce jour rien ne peut garantir que le résultat de l'algorithme fasse ressortir une information réellement contenue dans les jeux de données. Dans le cas de la véracité des résultats du bundling aux yeux de l'utilisateur, des pistes de résolution existent.

Pour conclure ce chapitre, nous avons présenté comment le bundling s'intègre avec les autres techniques de simplifications visuelles d'après les travaux de Ellis et Dix, 2007. Puis nous avons montré les tâches de visualisation que l'on peut résoudre à l'aide du bundling. Enfin, nous avons présenté des applications mettant en valeur les tâches auxquelles le bundling répond. Aussi, ce chapitre dénote l'adoption des techniques de bundling dans divers champs de recherches. Ainsi, en conjonction avec les deux précédents chapitres, nous avons systématisé et clarifié plusieurs questions de recherches inhérentes aux techniques de bundling. Ce travail permettra à la fois aux utilisateurs de choisir, paramétrer et utiliser les techniques d'agrégation visuelle afin de résoudre des problèmes concrets et aux chercheurs pour comparer, discuter, comprendre et raffiner les futurs algorithmes de bundling.

5.1 Task Support

Using bundling fits very well into Shneiderman, 1996's "overview first, zoom and filter, then details-on-demand" metaphor. The key task a bundled drawing $B(D(P))$ supports is to present an *overview* of $D(P)$ – thus, implicitly of P too – which trades off clutter for overdraw. If users find interesting patterns in it, they can next *zoom* and *filter* on such patterns, and next get *details on demand* on them using the interaction techniques discussed in section 4.3.5. We analyze this further by first discussing how bundling reduces clutter (sub-section 5.1.1). Next, following two well known task taxonomies (Lee et al., 2006 and Brehmer and Munzner, 2013) we detail the tasks supported by bundled drawings (section 5.1.2).

5.1.1 Bundling Clutter Reduction Taxonomy

To understand how well bundling reduces clutter, we analyze it following Ellis and Dix, 2007's clutter reduction taxonomy. From all clutter reduction techniques (see chapter 2, table 2.2 Ellis and Dix, 2007), bundling uses path[1] displacement, clustering, and opacity techniques. Indeed, B displaces paths to create path-clusters and renders them using opacity. In the following, we compare bundling with three clutter reduction techniques according to Ellis and Dix, 2007 defined clutter reduction (presented in chapter 2). Table 5.2, columns 2…4 lists them, showing how opacity, clustering and displacement contribute to each. Column 5 in table 5.2 shows our (arguably subjective) view on how well bundling does this, as detailed below. Simarly to our approach in section 2.2, each clutter reduction criteria is detailed in the following;

	opacity	clustering	path displacement	**bundling**
avoids overlap	partly	possibly	✓	partly
keeps spatial information	✓	partly	✗	partly
can be localized	✓	✗	✓	partly
is scalable	✗	✓	✗	✓
is adjustable	✓	✓	possibly	✓
can show path attributes	✗	partly	✓	✓
can discriminate paths	✓	✓	possibly	partly
can see overlap density	✓	possibly	✗	✓

TABLE 5.2 – Bundling compared with opacity, clustering, and displacement techniques *vs* yielded clutter-reduction benefits (Ellis and Dix, 2007).

a) Avoids overlap: Bundling avoids overlap of *coarse-scale* patterns: Take a set of unbundled, but highly-similar, paths $H \subset D(P)$. Following the **Bundling equation**, their bundling $B(H)$ contains much closer paths (*cf.* the spatial distance δ) than H.

[1] Ellis and Dix, 2007 refer to 'point/line'; as our input is a possibly curved path drawing $D(P)$, we use the term path

Hence, for two such disjoint sets H_i and H_j of $D(P)$, the chance that their bundled versions $B(H_i)$ and $B(H_j)$ overlap is (much) smaller than that of the unbundled sets H_i and H_j to overlap. Bundling yields more overlap of compatible edges (*cf.* the compatibility κ), but less overlap for incompatible ones.

b) Keeps spatial information: B does not change path endpoints, so this type of spatial information is kept. Spatial information of other path points *is* distorted. However, bundle scale control (section 4.2.2), smoothing (section 4.3.3), and relaxation (section 4.3.5) limit the deformation amount.

c) Can be localized: In general, clutter is locally inversely proportional to the bundling amount: Strongly bundled areas exhibit less clutter than weakly bundled ones, assuming an input $D(P)$ with uniform spatial clutter distribution. Hence, local control of the bundling amount can localize the presence of clutter (see Hurter, Telea, and Ersoy, 2011);

d) Is scalable: For bundling techniques, the only scalibility constraint is the computing time. Lastest image-based bundling are scalable to large million-size path-sets as showed in sections 4.1.2 and 6.

e) Is adjustable: Bundles can be widely adjusted in terms of path similarity (κ implementations), tightness, smoothness, shape, obstacle avoidance, and visual appearance (as explained in Sec. 4).

f) Can show path attributes: Bundling can show path local density, direction, and several other data attributes, either implicitly or explicitly. Path attributes are implicitly mapped by their presence in the compatibility κ (section. 2.3) or explicitly mapped to opacity, color, shading, and animation (section 4.3).

g) Can discriminate paths: Directional and confluent bundling techniques do precisely that, favoring different types of paths to discriminate. More example of various path discrimination are detailled in Pupyrev, Nachmanson, and Kaufmann, 2010; Selassie, Heller, and Heer, 2011; Luo et al., 2012; Zhou et al., 2008a; Bourqui et al., 2016);

h) Can show overlap density: Virtually all bundling techniques map local bundled-path density to opacity or color to show precisely that (section 4.3.1,4.3.2).

Overall, we see that bundling supports well the intended benefits of clutter reduction mentioned by Ellis and Dix, 2007. This helps understanding next which tasks bundling can support, and how much. For instance, Ellis and Dix, 2007 mention in Sec. 3 that the *avoid overlap* benefit supports the ability to see and identify patterns (Ahlberg and Shneiderman, 1994); *being localized* helps examining small details while keeping context.

5.1.2 Task Taxonomy

Based upon our previous analyses of the bundling design space through our taxonomy and unified framework, we analyze which tasks bundling supports using two well-known taxonomies, as follows.

Lee et al., 2006's Taxonomy

In their proposed taxonomy, Lee et al., 2006 define and demonstrate how all complex tasks for graph visualization can be seen as a series of *low-level* tasks on graph specific objects (edges, nodes, attributes). They define four main classes of tasks; topology-based, attribute-based, browsing and overview. Amongst the proposed tasks, we next describe the ones we believe bundling can answer. Bundling being a visualization technique focusing on links and clusters, it supports:

- **Path following:** There are bundling techniques specifically designed to allow users to follow a given path. As detailed in the previous sub-section 5.1.1, point (g). Such techniques are mainly directional and confluent bundling techniques (Pupyrev, Nachmanson, and Kaufmann, 2010; Selassie, Heller, and Heer, 2011; Luo et al., 2012; Zhou et al., 2008a; Bourqui et al., 2016). Moreover, with the leverage of interaction, all bundling technique can be extended to support the path following tasks.

- **Edge density visualization:** As explained in section 4.3, all bundling techniques can during the *visual exploration* step of the unified pipeline use color, opacity or even line width to visually convey bundled edge density.

- **Identify edge clusters:** Given an edge (or more generally, path), if we can capture the edge similarity by a function κ then per definition (see **Bundling equation**) bundling does identify edge clusters. It is worth noting that here clusters refers to edges that are spatially close in the drawing space.

- **Identify strongly connected cliques:** This is indeed possible. Refer to section 5.1.1, point (a), last sentence.

- **Overview:** As already explained, B can be seen as a coarsening/simplification operator that keeps *and* enhances the core structure of a drawing $D(P)$. At the correct scale (see section 4.2 and figuure 4.9), bundling alleviates the details and enhances the overall pattern, thus giving an overview of the path-set drawing $D(P)$.

- **Find patterns:** Since B enhances the density distribution of $D(P)$, it follows that patterns which are (vaguely) visible in $D(P)$ will only become clearer in $B(D(P))$. For a formal discussion, see Comaniciu and Meer, 2002; for practical examples, see *e.g.* section 5.2.2. Yet, the converse is not always true: A $D(P)$ having no salient patterns *can* yield a $B(D(P))$ showing false patterns (see next section 5.3).

- **Compare flows:** Directional bundling combined with directional coloring supports this task, see *e.g.* figure 3.11. In conjunction with some interaction, users can efficiently analytically compare similar flows as outlined by Scheepens et al., 2016.

Brehmer and Munzner, 2013's Typology

Conversely to the low-level taxonomy of Lee et al., 2006, Brehmer and Munzner, 2013 proposes a multi-level task typology. In their typology, all mid and low level tasks relate to three high-level questions, *why*, *how* and *what*. This typology allows us to bridge the gap between high-level tasks (*e.g.* overall system requirements) and low-level tasks (*e.g.* selecting data). In the following, we analyse bundling task support for each three high-level questions:

Why

- ***Consume:*** Bundling can *e.g.* *present* big worldwide flight datasets (Klein, Van Der Zwan, and Telea, 2014); help *discover* user patterns in eye-track data (Peysakhovich, Hurter, and Telea, 2015); and *enjoy* organic presentations of complex, abstract data (see Telea, 2015, cover). Bundling can also *produce* new visual artefacts such as the bundled path densities.

- ***Search:*** Bundling covers all defined search queries. For example, users can *explore* the bundled drawing $B(D(P))$ as an overview. Conversely, users *lookup* a specific bundle in a given region such as looking up for outbound flows from California in the bundled drawing of the US migration graph (figure 3.11).

- ***Query:*** Bundling helps *identifying* salient connection patterns, *comparing* path-sets from a sequence or stream (sections 3.1.2, 3.1.2, 3.2.2) and *summarizing* complex path drawings $D(P)$, as clear from all discussions so far.

How

- ***Encode*** is the core of bundling methods. As stated throughout the previous chapters (chapters 3 and 4), bundling encodes the existing path-set drawing into a simplified version that yields the important flows, trends and patterns.

- ***Manipulate:*** Without any interaction techniques, bundling techniques can *aggregate*, *change* and *filter* any input path-set P. The multiscale capabilities of bundling techniques (detailed in section 4.2), bundling can easily change the granularity of the path-set drawing. The implementation of given similarity function κ allows *filtering* (exclude or include) of different paths (section 4.1). Finally, the visual exploration step allows us to *change* the visual encoding of various path attributes (*e.g.* density, direction, length...). The rest of the *manipulate* tasks are answered by bundling in conjunction with interaction techniques, such as lenses, as explained in section 4.3.5.

- ***Introduce:*** Bundling alters existing elements of the visualization. Per definition, bundling transforms the original path-set drawing $D(P)$ into a simplified bundled drawing ($B(D(P))$ yielding clearer patterns of the original path-set P. Moreover, it can compute the density of bundled paths thus *deriving* new data elements from P. Similarly, as bundling generates a new drawing of the original path-set P, it supports *recording* tasks.

What is defined as the inputs and outputs of a visualization technique. Here, this part of the taxonomy was already covered in section 2.3.3.

5.2 Typical Bundling Use-cases

We next illustrate the application of graph and trail bundling in several application areas – software engineering (section 5.2.1), vehicle trajectory analysis (section 5.2.2), eye track analysis (section 5.2.3), multidimensional visualization (section 5.2.4), and vector and tensor field visualization (section 5.2.5). For each area, we outline a few relevant use-cases, including the goals to be addressed, examples of bundling results, and outline some limitations.

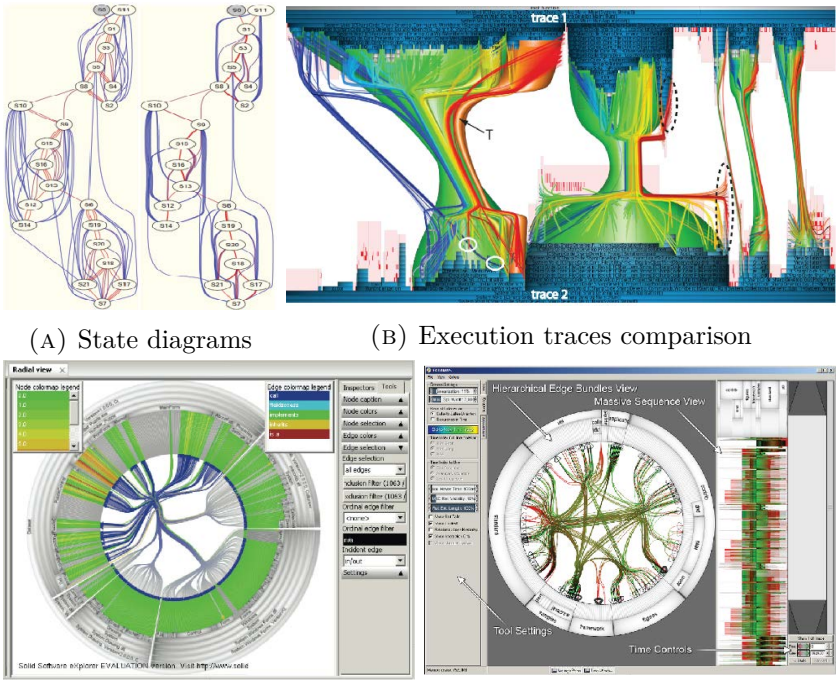
5.2.1 Software Engineering and Data Mining

Some of the earliest applications of graph bundling emerged from program comprehension. A key aim of program comprehension is to help developers understand the structure and execution of large programs. This helps various types of maintenance, such as discovering and fixing performance problems and bugs, refactoring the software, and recovering its architecture (Chapin et al., 2001). Static and dynamic program mining produces a wealth of data, of which an important component are attributed compound graphs, whose nodes describe software entities (*e.g.* methods, classes, and packages) and edges describe inter-entity relations (*e.g.* call, inherit, data transfer, compilation dependency) (described by Emanuelsson and Nilsson, 2008 and Cornelissen et al., 2009). Such graphs can have up to hundreds of thousands of nodes and edges, so displaying them using classical straight-line node-link drawings is not effective (Telea et al., 2009). Drawing software graphs is an important subfield of software visualization, for which good surveys exist such as Koschke, 2003 and Diehl, 2007. Bundling produces simplified drawings, where it answers tasks such as finding how groups of related software entities (*e.g.*, in the same package) are connected with other similar groups.

Early applications include the simplified visualization of relatively small state diagrams, based on a DAG layout, such as the bundling method by Pupyrev, Nachmanson, and Kaufmann, 2010 (figure 5.2a, see also section 3.1.1). HEB by Holten and Van Wijk, 2008 yielded a major breakthrough, allowing tens of thousands of edges to be bundled. The original method (figure 3.2a) was next enhanced to handle graphs of hundreds of thousands of elements by allowing the interactive opening and collapsing of hierarchy nodes and automatic aggregation of children edges. Such extensions are Telea et al.,

2009 (figure 5.2b) and the solid* toolset (figure 5.2c, Reniers et al., 2014). Besides using a radial layout for the graph nodes, treemaps were also used, with edges bundled in 3D (figure 3.7, Wettel and Lanza, 2007; Caserta, Zendra, and Bodénes, 2011). The main advantage here is that treemap cells can be used to show more data, such as software metrics, than the (small) cells in a radial icicle plot. Another extension allowed for the comparison of two code bases (Holten and Van Wijk, 2008) or multiple versions of the same code base (Telea and Auber, 2008) (see figures 3.2b and 3.2d). The key task here is to find how elements in one code base correspond to the other one, and thus spot structural changes. HEB was also used to visualize code duplication (clones) in a software system (Voinea and Telea, 2014; Reniers et al., 2014) and how these evolve in time (Hanjalic, 2013; Hurter et al., 2014a). This supports the planning of clone removal with minimal impact on system architecture. HEB-like bundling has also been used to visualize program execution, *e.g.* to compare two or more traces of a program (Trümper, Döllner, and Telea, 2013) (to detect anomalous behavior) or to visualize the behavior of multi-threaded programs (Karran, Trumper, and Dollner, 2013) (to find performance problems).

Bothorel, Serrurier, and Hurter, 2013 used a general-graph bundling method to display frequent itemsets. These are arranged on multiple (rather than a single) circular layouts, so as to minimize edge lengths. Bundling was used to highlight groups of strongly-connected itemsets. The method was found to help knowledge engineers in extracting association rules from the itemsets.



(A) State diagrams (B) Execution traces comparison

(C) Solid* toolkit (D) Execution traces

FIGURE 5.2 – Bundling in program comprehension.

5.2.2 Vehicle Trajectory Analysis

Vehicle trajectory analysis is important in many applications such as air-traffic (Wise, Hopkin, and Smith, 1990), nautical vessel (Scheepens et al., 2011a; Scheepens et al., 2011b; Scheepens et al., 2016), and roadmap (Thöny and Pajarola, 2015) planning and control. Early on, Holten and Van Wijk, 2009 used bundling to simplify the depiction of large trail-sets so as to help inferring the main vehicle routes over a country . This use-case, as well as the *US airlines* dataset featured by Holten and Van Wijk, 2009 (figure 3.4), stayed visible in most trail-bundling papers since then (Cui et al., 2008; Lambert, Bourqui, and Auber, 2010b; Ersoy et al., 2011; Gansner et al., 2011; Hurter, Ersoy, and Telea, 2012; Zwan, Codreanu, and Telea, 2016). To support source-to-target trail analysis, directional bundling methods were proposed by Selassie, Heller, and Heer, 2011 and Moura, 2015.

Hurter et al., 2014a extend the above to handle streaming trail-sets, obtained from monitoring flights over a given spatial region over a period of time (US territory, 6 days). The obtained animation allows the detection of variations in position and density of the main flight routes depending on the time of the day, and comparing same-time patterns over several days. Recently, this method was extended, by using the fast GPU-based CUBu technique (Zwan, Codreanu, and Telea, 2016), to visualize around 800K flights collected from the entire world over June 2013 (Klein, Van Der Zwan, and Telea, 2014), see figure 5.3 top. Given the streaming nature of Hurter et al., 2014a, this approach can be used to visualize large-scale trail-sets that evolve over unbounded time ranges.

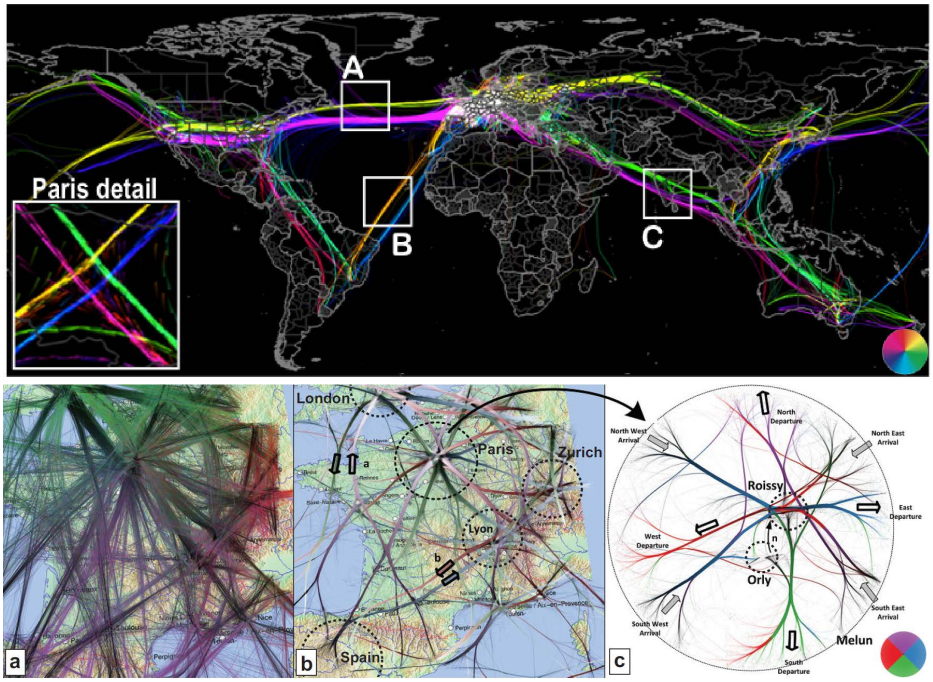


FIGURE 5.3 – Aircraft trail analysis – Top: Worldwide flights (Klein, Van Der Zwan, and Telea, 2014). Bottom: (a-c) Raw trails, directional bundling over France, and zoom-in over Paris area (Peysakhovich, Hurter, and Telea, 2015).

A more involved use-case is described by Peysakhovich, Hurter, and Telea, 2015. The data consists of 24 hours of flight traffic over France (18K trails). The bundled visualization (figures 5.3 bottom) helps air-traffic controllers to cross-reference actual flows with known theoretical air routes (on a global scale in figure 5.3b) and theoretical approach routes to the Paris airports (locally in figure 5.3c). This way, problems can be spotted early on, and traffic planning can be adjusted next.

For the same use-case (worldwide flight analysis), Lambert, Bourqui, and Auber, 2010a extend Winding Road (Lambert, Bourqui, and Auber, 2010b) to bundle 2000 flight trails over the Earth surface. They show how 3D bundling is superior to bundling on a 2D map for assessing flight lengths and for more exact trail-to-trail distance assessment (figure 5.4 top). Finally, Vector Maps by Thöny and Pajarola, 2015 bundle commuter paths on the 3D Swiss road network. The key parts of the road network are used as a 'skeleton' to attract bundles. The visualization shows which main routes (*e.g.* highways) are important for which parts of the traffic (figure 5.4 bottom).

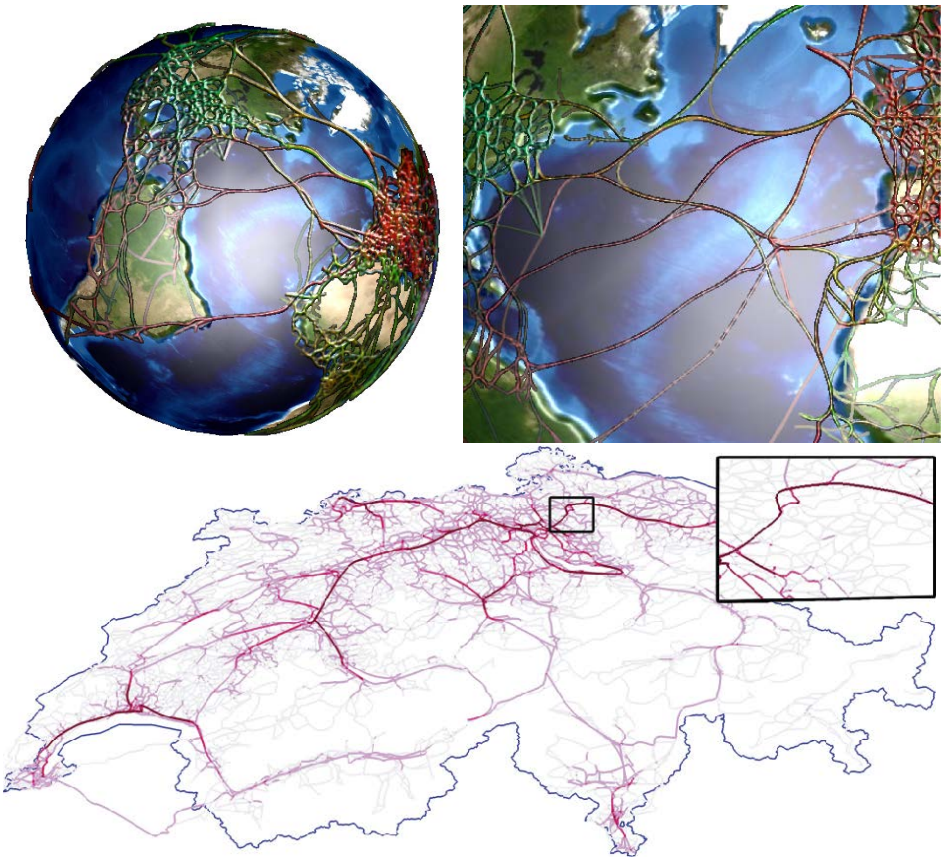


FIGURE 5.4 – Top: 3D bundling of 2000 worldwide flight trails left, and detail right Lambert, Bourqui, and Auber, 2010a. Bottom: Bundling commuter trails along the Swiss road network Thöny and Pajarola, 2015 (detail top-right).

5.2.3 Eye-Track Analysis

Eye tracking delivers datasets consisting of 2D points (so-called fixation points of the human gaze) linked by transitions (called saccades) (Tatler et al., 2010). Analyzing such tracks is important for many applications, *e.g.* assessing user performance when using new interfaces (Çöltekin, Fabrikant, and Lacayo, 2010; Kim et al., 2010) and finding how users read a display and whether they do it efficiently (Larkin and Simon, 1987; Krygier et al., 1997). An important part of the analysis of eye trails is detecting the so-called fixation areas (FAs), defined as groups of many close fixation points; and finding how FAs are linked by saccades.

Trail bundling perfectly suits such tasks. Recent works in eye-tracking also strongly consider bundling as a viable solution to reduce clutter induced by the large amount of ocular trail sets (Blascheck et al., 2014). Application-wise, ADEB (Peysakhovich, Hurter, and Telea, 2015) was used to show how bundled trails can be used to assess the proficiency of users (figures 5.5a,b). Here, the ocular behaviors of a novice and expert user during a multi-task experiment are compared. The background image shows the GUI that the users had to monitor and interact with during the experiment. From this, the authors show how and where the expert performed better, which can lead to improvements in either the training procedure or the GUI being proposed to users.

Separately, Hurter et al., 2014a considered the time information present in eye trails, by bundling only trails that are compatible with respect to occurring close to each other in time using a streaming method. Figures 5.5c,d shows how dynamic bundling highlights salient eye-movement patterns of an aircraft pilot during a landing

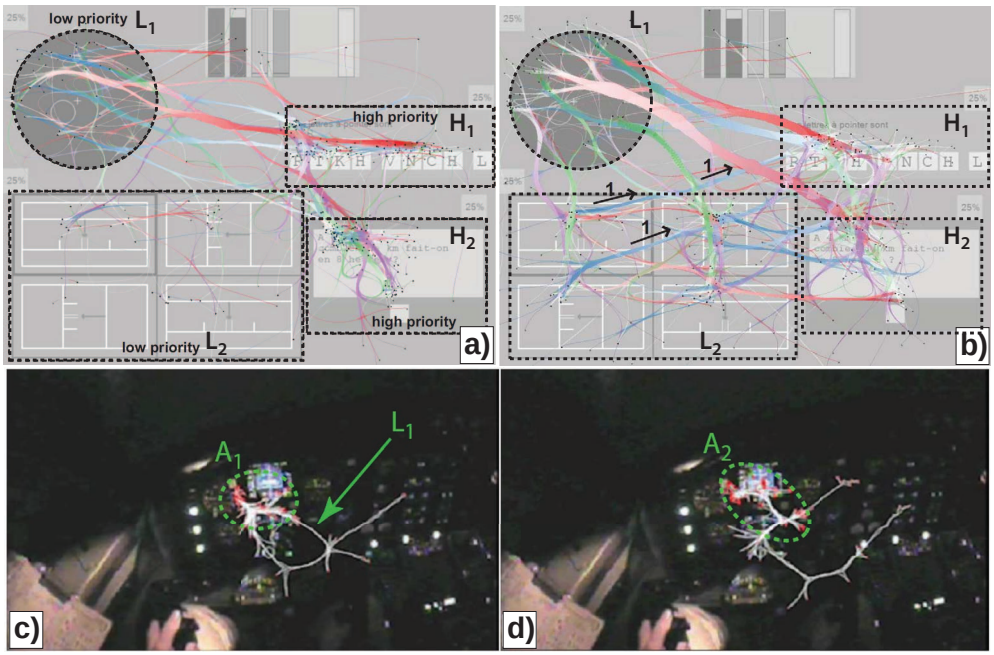


FIGURE 5.5 – Bundled eye-tracking trails of novice (a) and expert (b) users in a multitask experiment (Peysakhovich, Hurter, and Telea, 2015). (c-d) Visual analysis of pilot eye-tracking data with dynamic bundling (Hurter et al., 2014a).

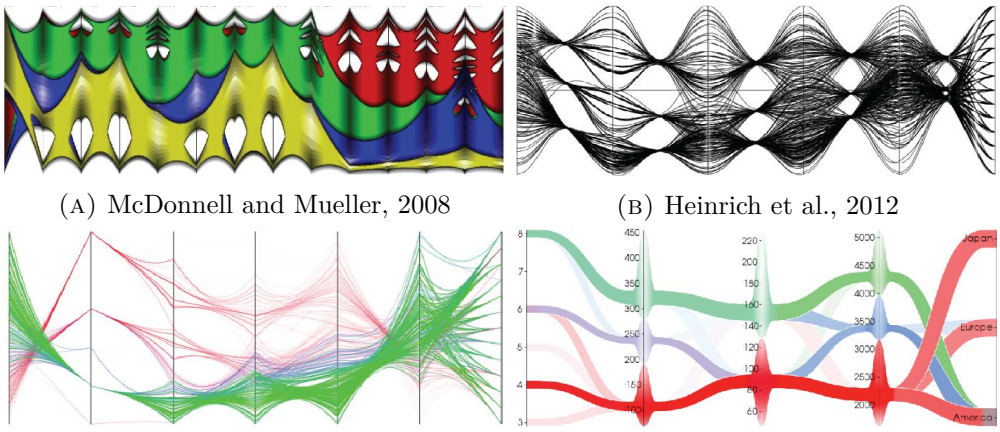
sequence. The background images show actual views from the cockpit, with dashboard instruments in focus. Bundles show the pilot's so called main visual strategies during landing. This helps pilots to analyze and correct their behavior (improves training) but also designers of new aircraft landing-assisting cockpit instruments in finding if such instruments have been effectively used by pilots.

5.2.4 Multidimensional Data Exploration

Multidimensional data exploration is a very challenging field: Datasets whose samples have many dimensions (tens or even hundreds) have to be mapped to 2D or 3D. Relevant tasks include finding groups of similar samples, outlier samples, and correlations and trends of subsets of the existing dimensions.

Such data can be visualized by Parallel Coordinate Plots (PCPs) (see Sec. 3.2.1). As explained there, PCPs for thousands of samples or more quickly become cluttered. Bundling can help here, much in the same way it helps declutter straight-line graph drawings. Figure 5.6 shows four bundling methods for PCPs discussed in section 3.2.1. As visible, each method targets different tasks: McDonnell and Mueller, 2008 (figure 5.6a) separate the data into compact clusters, emphasizing data cluster differences (which sample groups are different). Coloring and shading are very similar to IBEB (Telea and Ersoy, 2010) (section 4.3.2). Heinrich et al., 2012 (figure 5.6b) emphasize the continuity of the PCP polylines, helping end-to-end tracing, as opposed to all other methods. Zhou et al., 2008b offer a proposition similar to McDonnell and Mueller, 2008, but with less overlap (figure 5.6c). Finally, Palmas et al., 2014 offer the strongest clutter reduction (but overdraw increase) (figure 5.6d). Their visual design is very similar to HEB (Holten, 2006) – just as HEB lets one see how *groups* of nodes in a graph are connected, so do they show how *ranges* of variables occur together in a multidimensional dataset.

Dimensionality-reduction (DR) methods are another way to plot high-dimensional data. Given an n-dimensional dataset, a DR method creates a 2D scatterplot where



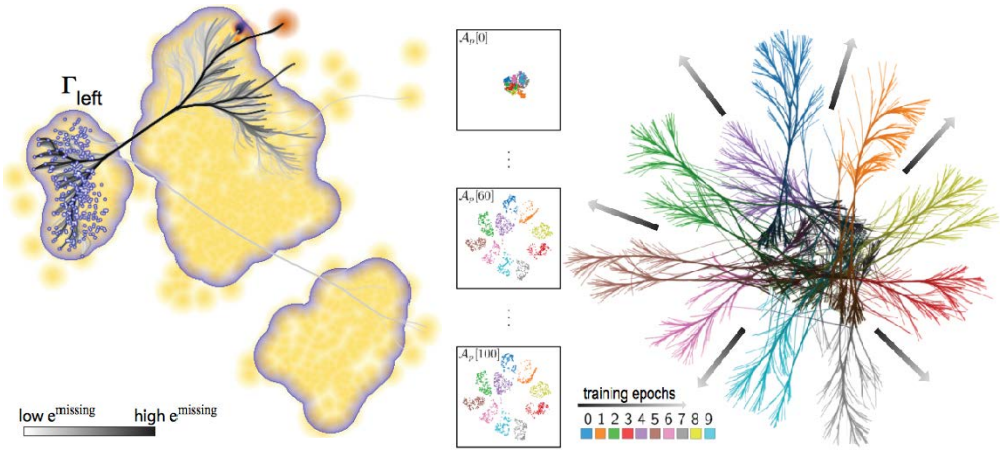
(A) McDonnell and Mueller, 2008 (B) Heinrich et al., 2012

(C) Zhou et al., 2008b (D) Palmas et al., 2014

FIGURE 5.6 – PCP bundling with four methods.

inter-point distances reflect the corresponding nD distances (Sorzano, Vargas, and Montano, 2014). It is well known that DR techniques cannot perfectly map nD distances to 2D. So, ways are needed to assess errors in such 2D scatterplots. Martins et al., 2014; Martins, Minghim, and Telea, 2015 classify such errors into false neighbors (points too close in 2D *vs* nD) and missing neighbors (points too far in 2D *vs* nD). They next extend these notions to groups (clusters) of points representing concepts. Bundling, done with CUBu (Zwan, Codreanu, and Telea, 2016) helps showing missing (group) neighbors: All 2D scatterplot point-pairs which are farther apart, as compared to their nD counterparts, than a given value, are connected by edges, whose opacity reflects the 2D-nD distance discrepancy. Next, these edges are bundled. This effectively shows which zones in a DR plot miss information, and where this information is. For example, in figure 5.7a, bundles show that many of the elements of the group Γ_{left} in the scatterplot miss neighbours which the DR method erroneously places in group Γ_{top}, but miss no neighbours with respect to the group Γ_{bottom}. Bundling is also used to compare two plots created by different DR methods.

Bundling is also used to visualize training of artificial deep neural networks (DNNs). Understanding how such networks learn from examples is notoriously hard, as they usually operate as black boxes (Mühlbacher et al., 2014). Rauber et al., 2017 use bundling to help this: Imagine that all neurons of a DNN are points in nD space, with coordinates given by their so-called activations. Such a DNN is typically trained by feeding it a sequence of N examples. Hence, activations change with each learned example. This can be visualized by projecting all neuron activations to 2D (using the well-known t-SNE DR technique by Maaten and Hinton, 2008) and next constructing trails linking the N 2D positions of the same neuron. This yields a streaming trail-set, which next can be bundled, as in Fig. 5.7b. Here, luminance encodes training time, and color encodes neurons specialized for the same task (for Rauber et al., 2017, this is classifying images). The bundle structure, highlighted by the arrows, shows how



(A) Visualizing errors in multidimensional projections (Martins et al., 2014).

(B) Visualizing learning in a neural network (Rauber et al., 2017).

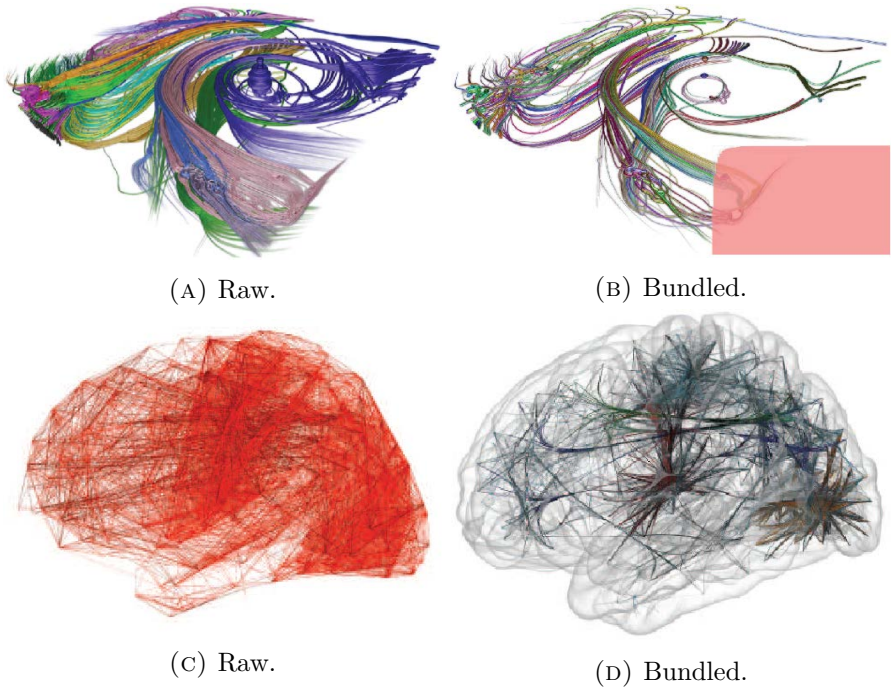
FIGURE 5.7 – Bundling applied to dimensionality reduction methods.

originally untrained neurons (image center, dark) progressively diverge from each other. This serves in assessing the training performance: For a DNN, we want indeed that neurons progressively differentiate from each other and specialize for doing different tasks.

5.2.5 Vector and Tensor Fields

Trail bundling has also been used to simplify displays of vector and tensor fields. For vector fields, Yu et al., 2012 bundle 2D and 3D streamlines to produce simplified visualizations of the respective fields (figure 5.8a and 5.8b). The tasks addressed cover reduction of clutter and occlusion (in 3D) and easily spotting salient field patterns such as separatrices, laminar flow regions, and turbulent regions (Post et al., 2003). Compared to other hierarchical vector field simplification methods (Telea and Wijk, 1999), this approach can better capture salient vector field structures at similar levels of detail. It is however important to note that bundling is done here *purely* to find groups of very similar streamlines (following a similarity definition analogous to δ, equation. 4.1). After such groups are found, an actual physically correct streamline best representing each group is rendered. Thus, no geometric deformation takes place.

Böttger et al., 2014 bundle 3D trails to help neuroscientists visualize brain connectivity captured by fMRI techniques. The input data is a graph G, with nodes V representing 3D locations in the brain and edges E linking locations that are related with respect to function. Straight-line drawings of G produces highly cluttered pictures (figure. 5.8c). KDEEB bundling (adapted to 3D) on G massively reduces occlusion and



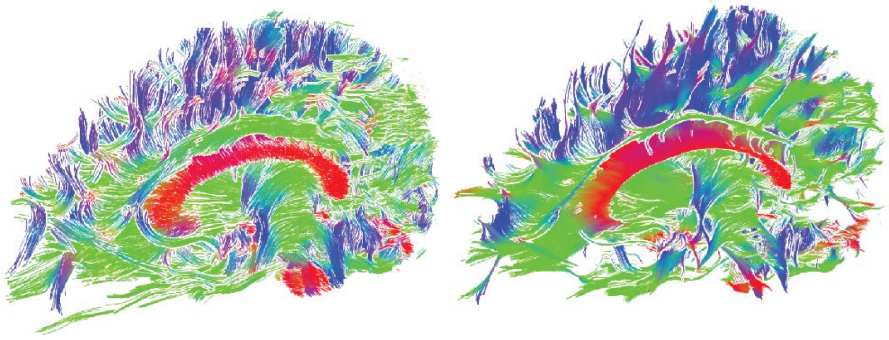
(A) Raw. (B) Bundled.

(C) Raw. (D) Bundled.

FIGURE 5.8 – Vector and tensor fields. Top: 3D streamlines (Yu et al., 2012). Bottom: Functional brain connectivity (Böttger et al., 2014).

allows one to see how groups of spatially close nodes inter-relate (figure. 5.8d). For this use-case, edge deformation is not an issue, as edges only carry connectivity information.

Bundling is also applied to Diffusion Tensor Imaging (DTI) fields. From these fields, trail (or tract) sets indicating the locations of important white-matter neural fibers are extracted by tractography (Assaf and Pasternak, 2008). Rendering the raw tracts yields relatively cluttered images, depending on actual tract extraction settings (figure 5.9a). Everts et al., 2015 bundle tracts (for details, see section 3.2.1) to yield simplified, less cluttered, images (Fig. 5.9b). Tracts are next colored to indicate 3D orientation (for details, see Telea, 2015, Ch. 7). In contrast to Böttger et al., 2014, deformations are now constrained, as actual track locations are important. The bundled views are typically used to assess how (strongly) different anatomical regions in the brain are connected, which can next help planning minimally disruptive surgery.



(A) Raw. (B) Bundled.

FIGURE 5.9 – DTI tracts (Everts et al., 2015).

5.3 Quality and Faith-fullness of Bundling

In this section, we discuss next unsolved challenges of bundling techniques: The quality assessment of bundled drawings and the issue of faithfulness, or how much information is kept (or not) in a bundled drawing.

Quality Assessment

There is no accepted way to measure the quality of a bundling. The core problem is that it is hard to define objective criteria for what a *good* bundling is. Quality metrics have been discussed, and advocated for, since long in information visualization (see the work of Brath, 1997; Miller et al., 1997; Bertini and Santucci, 2006). Closer to our scope, these include the ink-ratio and *lie factor* of a visualization (Tufte, 1992); defining visual clutter (Ellis and Dix, 2007); and measuring edge congestion (Carpendale and Rong, 2001), edge crossings (Kobourov, Pupyrev, and Saket, 2014), readability (Dunne and Shneiderman, 2009; Eades et al., 2015), aesthetics (Purchase, Cohen, and James, 1995; Ware et al., 2002), and faithfulness (Nguyen, Eades, and Hong, 2013; Nguyen, Eades, and Hong, 2017) in graph visualizations. However, no such set of metrics fully covers path bundling. For bundled drawings quality, only a handful of studies exist, touching upon the visual navigability of small bundled drawings (Pupyrev, Nachmanson,

and Kaufmann, 2010), comparing the effectiveness of bundled *vs* unbundled drawings (Telea et al., 2009), studying the comprehensibility (McGee and Dingliana, 2012) and ambiguity (Bach et al., 2017) of bundled drawings, and visualizing edge deformation (Hurter, Ersoy, and Telea, 2012).

Quality assessment can be approached by defining what quality is. Following well-established principles in software engineering (Ken, 2003), we can define the quality of a bundled drawing by either measuring its *fitness for purpose* (how well it helps solving a certain problem) or by comparing it to a *ground-truth* whose quality is known. Both paths have, however, challenges, as outlined next.

Fitness for purpose: To quantify this, we need first to define what the goal(s) of a bundled drawing are. Section 5.1.2 outlines the tasks that bundling aims to cover. This (or a similar) proposal could be next used to scope, and assess the value of, their contributions. This can be done by user studies where the percentage, correctness, and time of completing, a given task is measured. Besides the know challenges that organizing large-scale user studies (and generalizing their results) have, an extra difficulty is that the same method B can generate a wealth of different drawing styles from the same dataset $D(P)$, see *e.g.* Zwan, Codreanu, and Telea, 2016. The data-based taxonomy proposed in section 3 can help here in narrowing the focus of methods to be compared against each other based on the type of input data they work on.

Ground truth comparison: Quality can be measured by computing the difference between a bundling $B(D(P))$ produced by the method under study and a so-called ground truth $B_g(D(P))$, *i.e.*, a bundled drawing known to be good for a certain task. Most existing bundling papers do this implicitly, by comparing their results with earlier methods on the same dataset $D(P)$. Yet, in most (if not all) cases, the comparison is only visual, such as shown in figure 3.15. This can be improved by using quantitative metrics to compare two bundled images, *e.g.* by measuring their Hausdorff distance, or metrics to assess global quality parameters of a drawing $B(D(P))$, *e.g.* amount of overdraw, spread of path displacements, spatial path-density distribution, amount of overlap of different-direction bundles, amount of bundle crossings, and bundle curvature distribution. We can next infer good values for such metrics *e.g.* by extrapolating from the graph-drawing and graph-aesthetic principles mentioned earlier, and compare actual metric values with the desired good values. Another way to measure quality is to compare $B(D(P))$ with the unbundled drawing $D(P)$, using a similar set of quality metrics. A good bundling is, in this case, one that increases the metrics' values.

A separate problem for ground truth comparison is that there is, so far, no established benchmark of graphs and trail-sets (and bundling method implementations) that the infovis community could use. Creating such a benchmark is reasonably easy and should be highly useful for the community. A starting point for graphs (including node layouts) can be the well-known Florida collection introduced by Davis and Hu, 2011.

Bundling Faithfulness

A separate issue regards the information alteration or loss produced by bundling, or the so-called faithfulness of the produced drawings (Nguyen, Eades, and Hong, 2013; Nguyen, Eades, and Hong, 2017). Simply put, we need to measure (a) how much of the original information conveyed by $D(P)$ is kept by $B(D(P))$, and (b) how much incorrect information $B(D(P))$ adds as compared to $D(P)$. In other words, we need to measure the precision and recall for B. The above are related to the concept of inferrence affordance that measures the informational equivalence of two displays (Larkin and Simon, 1987; Çöltekin, Fabrikant, and Lacayo, 2010; Tatler et al., 2010) – in our case, $D(P)$ and $B(D(P))$.

Concerning (a), it is clear that bundling loses some of the information present in the original $D(P)$. This is a higher problem for trail-sets than for graphs, since the former contain relevant spatial information. To assess this, we can measure (1) the amount of trail distortion created by B, *i.e.*, $\sum_{\mathbf{p}_i \in D(P)} \delta(\mathbf{p}_i, B(\mathbf{p}_i))$, with δ given by equation 4.1. Next, depending on the task, we can decide whether this amount is acceptable or not. To help this, distortion can be visualized on $B(D(P))$ (Hurter, Ersoy, and Telea, 2012). When distortion is too large, relaxation can be used (section 4.3.5), or bundling can be selectively stopped when it reaches a (local or global) maximum value. Further, a ratio of clutter reduction to amount of distortion can be computed, analogously to Tufte's ink-space ratio (Tufte, 1992) to measure the faithfulness of B. Another way to increase faithfulness is to animate $D(P)$ towards $B(D(P))$ during the bundling (Hurter et al., 2014a). This helps users match the input and output of B, thus reducing the information loss.

Concerning (b), it has been shown that, for some datasets, bundling *can* yield false insights. Figure 5.10 illustrates this for KDEEB (Hurter, Ersoy, and Telea, 2012): A random graph whose nodes and edges are uniformly spread over a 2D region is bundled. The result shows emerging structures which, however, do not reflect any actual patterns in the input data. This effect is mainly due to the fact that implicit bundling methods expect that their input *has* salient structures; when this is not the case, the bundling's underlying density-sharpening principle (section 2.3) will accentuate small-scale noise.

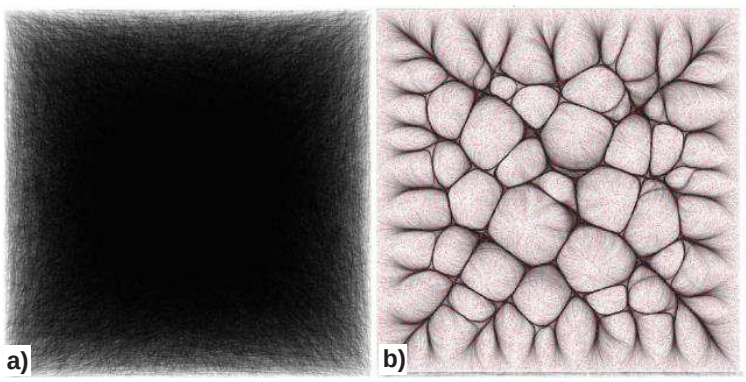


FIGURE 5.10 – Pseudo-random graph (a) and its KDEEB bundling (b). Red dots represents graph nodes; Hurter, Ersoy, and Telea, 2012.

5.4 Summary

In this chapter, we have presented how bundling supports clutter reduction following Ellis and Dix, 2007's taxonomy presented in chapter 2. We proposed a taxonomy for the tasks it supports and presented a wide sample of applications that rely on bundling.

This chapter denotes the adoption by various research field of bundling techniques. The development of bundling techniques has grown parallel to widening their application. Bundling has been arguably best accepted in software engineering and geospatial trail analysis (sectoin 5.2). Other salient application fields are eye-tracking analysis, DTI tract visualization, and network visualization. Several mature software tools offer bundling, *e.g.* GraphViz (Gansner, 2017), Tulip (Auber et al., 2017), Gephi (Bastian, Heymann, and Jacomy, 2009), D3 (Bostock, 2017), and Protégé (Tu, 2017; Hop et al., 2012). Yet, except a few methods like HEB (Holten, 2006), FDEB (Holten and Van Wijk, 2009), and MINGLE (Gansner et al., 2011), most other bundling methods have still not been integrated in such mainstream packages.

To conclude, we believe that in conjunction with the two previous chapters, we have systematized and clarified several so-far open points in the bundling literature. This will serve both practitioners in understanding how to choose, parameterize, and use bundling techniques to solve concrete problems and also researchers to compare, discuss, understand, and refine future bundling algorithms.

Chapter 6

Improving Bundling Scalability

In this chapter, we propose a technique that addresses the joint bundling challenges of computational speed and data volume, as follows. First, we re-cast bundling from convolving in the image domain to the frequency domain, using the Fast Fourier Transform (FFT), which gives the name to our technique, FFTEB. FFTEB significantly reduces the convolution cost inherent to all image-based edge bundling techniques. It also allows us to formulate directional and attribute-based edge bundling with a lower computational complexity than existing techniques. Secondly, we present a streaming scheme that allows efficient transfer of data between the CPU and GPU. This makes bundling scalable to very large datasets, beyond what current methods can efficiently handle. Overall, FFTEB produces similar-quality results as state-of-the-art bundling methods, while allowing one to bundle faster, as well as bundle much larger datasets, than it has done ever before. We compare our bundled results with nine well-known edge-bundling methods, while allowing on a set of path-sets ranging from thousands up to a million edges, and edge-sampling resolution up to a *billion* sample points.

The structure of this chapter is as follows. Section 6.1 overviews recent image-based techniques that we aim to surpass in terms of scalability and speed. Section 6.2 presents the mathematical model, design rationale, and implementation details of FFTEB. Section 6.3 presents results of FFTEB on very large datasets and compares it with existing bundling methods. Section 6.4 compares our scalability with recent bundling techniques and also discusses our method. Section 6.5 concludes the chapter.

6.0 Résumé (Français)
Une Nouvelle Technique de Bundling : FFTEB

Dans ce chapitre, nous proposons une nouvelle technique de bundling qui résout deux défis majeurs des techniques de bundling (*i.e.* la vitesse de calcul et la taille des données à traiter). Afin de résoudre ces défis, notre première étape consiste à changer le paradigme de calcul du bundling en transformant la convolution du domaine image vers le domaine fréquentiel à l'aide de la transformée de Fourier (Fast Fourier Transform - FFT). Ce changement de paradigme est aux fondations de notre technique que nous avons ainsi baptisée FFTEB (Fast Fourier Transfrom Edge Bundling Technique). En plus de significativement accélérer le temps de calcul nécessaire au bundling, cette approche nous permet de formaliser une technique de bundling basée sur n'importe quels attributs d'agrégation tout en procurant une complexité bien inférieure aux précédentes techniques. Dans une seconde étape, nous présentons un schéma de transfert de données nous permettant un transfert efficace des données entre le CPU et le GPU. Cette approche nous permet de s'abstraire des limites physiques inhérentes à la mémoire disponible sur les GPU. De ce fait, cela nous permet de calculer le bundling de graphes de taille bien supérieure à ce que les précédentes techniques pouvaient agréger. Au final, notre technique FFTEB produit des courbes agrégées aux résultats similaires à l'ensemble des autres techniques de bundling, tout en permettant une agrégation plus rapide de jeux de données bien plus grands qu'auparavant. Enfin, dans la section 6.3 de ce chapitre, nous comparons les résultats de notre technique avec neuf autres techniques de bundling reconnues à travers l'étude de plusieurs jeux de données de tailles variées (allant du millier aux millions de courbes agrégées).

Amélioration de la Complexité du Bundling

Afin de comprendre les améliorations introduites par notre nouvelle technique de bundling, il nous faut analyser les spécificitées techniques des méthodes de bundling fondées sur les techniques d'estimation par noyau Gaussien (KDE - Kernel Density Estimation).

Pour calculer la version aggregée d'un graphe, les méthodes basées KDE fonctionnent comme suit. En premier lieu, elles calculent une carte de densité de lien $\rho : \mathbb{Z}^2 \to \mathbb{R}^+$ en convoluant l'image $D(P)$ des chemins p echantillonés (x) avec un noyau Gaussien K modulé par une fonction de compatibilité $\kappa : \mathbb{R}^2 \times \mathbb{R}^2 \to [0;1]$. Ici, la fonction de compatibilité κ définit si deux chemins sont similaires ou non d'après leurs différents attributs (*e.g.* leur direction, leur temporalité). Ainsi, cette carte de densité ρ est definie comme :

$$\rho(\mathbf{x} \in D(P)) = \sum_{\mathbf{y} \in D(P) | \|\mathbf{x}-\mathbf{y}\| \leq h} \kappa(\mathbf{a}(\mathbf{x}), \mathbf{a}(\mathbf{y})) \cdot K(\mathbf{y}-\mathbf{x}) \tag{6.1}$$

Ensuite, l'ensemble des points des chemins de P sont déplacés suivant le gradient de la carte de densité afin de déplacer les chemins vers les zones où la densité de chemins est plus importante (processus d'agrégation). L'agrégation complète s'obtient après plusieurs itérations du processus.

$$D^{new}(P) = \left\{ \mathbf{x}^{new} = \mathbf{x} + \epsilon \frac{\nabla \rho}{max(\|\nabla \rho\|, \iota)} \middle| \mathbf{x} \in D(P) \right\}, \tag{6.2}$$

L'analyse de la complexité algorithmique des techniques basées KDE nous montre que la majeure partie du temps de calcul de ces techniques réside dans la convolution de l'image des chemins (et de leurs compatibilité) avec le noyau Gaussien K. Ainsi la complexité algorithmique est de l'ordre de $O(I(NR^2 + R^3))$ (*c.f.* section 6.1.3).

Pour résoudre ces problèmes de complexité, FFTEB propose d'exploiter les propriétés du produit de convolution. Ainsi, considérant deux fonctions de $\mathbb{L}^2$ et leur transformée de Fourier respective ($\mathcal{F}[f]$ et $\mathcal{F}[g]$), le produit de convolution $f * g$ équivaut à la transformée inverse de la multiplication des transformée de Fourier de f et g, *i.e.* $\mathcal{F}[f * g] = \mathcal{F}[f] \cdot \mathcal{F}[g]$. Ainsi, en remarquant que l'équation 6.1 peut s'écrire comme le produit de convolution de deux fonctions (*c.f.* section 6.2), nous pouvons réécrire le gradient de la carte de densité. Soient $\kappa : \mathbb{R}^m \times \mathbb{R}^m \to [-1; 1]$ un fonction de compatibilité bilinéaire de dimension m, $\{e_i\}$ et $\{u_j\}$ des bases orthonormées respectivement de $\mathbb{R}^m$ et $\mathbb{R}^n$, alors le gradient de la carte de densité $\nabla \rho(\mathbf{x} \in D(P))$ s'exprime comme :

$$\nabla \rho(\mathbf{x} \in D(P)) = \sum_{j=1}^{n} \kappa \left(\mathbf{a}(\mathbf{x}), \sum_{i=1}^{m} \mathcal{F}^{-1}[\mathcal{F}[\mathbf{a}_i] \cdot \mathcal{F}[\nabla_j K]](\mathbf{x}) \cdot e_i \right) \cdot u_j, \tag{6.3}$$

avec $\mathbf{x} \in \mathbb{R}^n$, $D(P) \in \mathbb{R}^n$, $\nabla K \in \mathbb{R}^n$ et $\mathbf{a} \in \mathbb{R}^m$ et $\mathcal{F}[\mathbf{a}_i]$ la transformée de Fourier de la composante (ou dimension) i de la carte des chemins $D(P)$. Grâce à cette reformulation du calcul de la carte de densité, la complexité de notre technique est drastiquement réduite et devient de l'ordre de $O(I(N + R^2 \log R))$. Concernant l'implementation de notre technique (*c.f.* section 6.2.2), FFTEB peut se résumer à travers les étapes suivantes (figure 6.1).

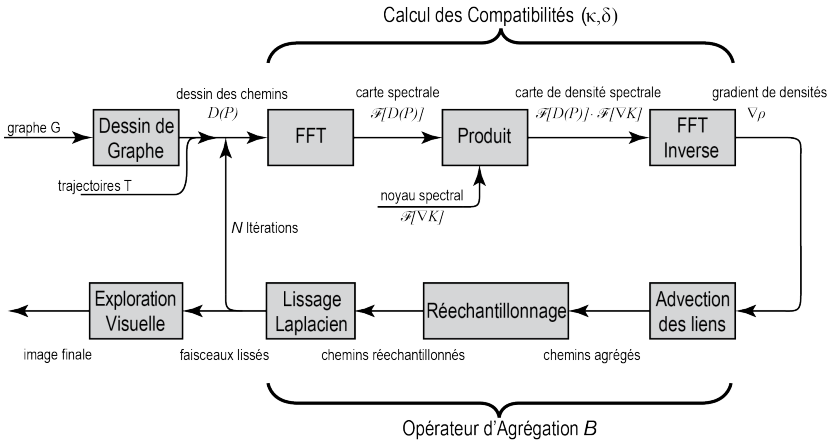


FIGURE 6.1 – Pipeline de notre technique de bundling FFTEB.

Un Nouveau Schéma de Transfert de Données pour le Bundling

Comme détaillé dans l'introduction, le second défi relatif aux techniques de bundling réside dans la taille des jeux de données à traiter. Notamment pour les techniques de bundling utilisant les cartes graphiques (GPU) pour accélerer les temps de calculs. Afin de résoudre ce problème, nous proposons un schéma de transfert de données (*streaming scheme*) simple mais adapté aux techniques de bundling basées KDE. Ce schéma de transfert des données se déroule comme suit. Tout d'abord, les chemins echantillonés de $D(P)$ (stockés en CPU) sont partagés en C parts c_i. Ce partage s'efffectue de telle sorte que chaque part possède environ N/C échantillons dont la taille est compatible avec la mémoire GPU disponible. Ensuite, chaque part est transférée vers le GPU et la carte de densité $\nabla\rho_i$ (calculée selon l'équation 6.3) formée par les chemins de la part en mémoire est accumulée dans la carte de gradient des densités globale $\nabla\rho$. Après avoir transferé l'ensemble des parts du jeu de donnée, chaque part est à nouveau transférée en GPU, afin d'agréger, lisser et rééchantillonner les chemins qui la composent. Enfin, chaque part est retransférée vers le CPU pour laisser de l'espace pour la part suivante. Ainsi, au cours de ce processus, l'algorithme effectue $2C$ transferts CPU vers GPU et C transfert GPU vers CPU, *i.e.* un transfert de données total d'une taille de $2N$ (CPU vers GPU) et N(GPU vers CPU). En comparaison du cas idéal où l'ensemble du jeu de données tient en mémoire GPU, notre processus de streaming ajoute seulement un coût supplémentaire de N par itération (*c.f.* sections 6.2.3, 6.3.4 et 6.4 pour plus de détails).

Examples d'Aggrégation de Simples Graphes

Nous illustrons les résultats de notre technique à travers quatre jeux de données de tests aggrégés selon leur direction ou non (figure 6.2). Les couleurs des chemins représentent leur direction et chaque jeu de donnée contient environ 2000 chemins. À travers ces 4 exemples, nous illustrons l'utilité d'agréger les liens selon des critères spécifiques (*e.g.* la direction des chemins) afin de correctement montrer les structure sous-jascentes.

En conclusion, au cours de ce chapitre, nous avons présenté une nouvelle technique de bundling capable d'agréger à l'aide d'attributs des jeux de données de très grandes tailles baptisée FFTEB. Le point clé de notre technique réside dans le passage à l'échelle tant sur l'amélioration du temps de calcul que sur la taille des jeux de données analysables. FFTEB exploite les propriétés de la transformée de Fourier ainsi que d'un processus de transfert de données (*streaming scheme*) pour permettre l'agrégation de jeux de données de taille bien supérieure à ce que les techniques précédentes étaient capable de faire tout en diminuant le temps de calcul. De plus, FFTEB généralise le bundling de chemins selon n'importe quel nombre et type d'attribut sans pour autant impacter la complexité de l'algorithme. Nous avons démontré les bénéfices de notre approche en comparant cette dernière avec neuf techniques de bundling au travers de jeux de données allant jusqu'à un milliard de points ; Un ordre de grandeur qu'aucune autre technique n'avait encore put atteindre. Notre travail a aussi démontré que les capacités

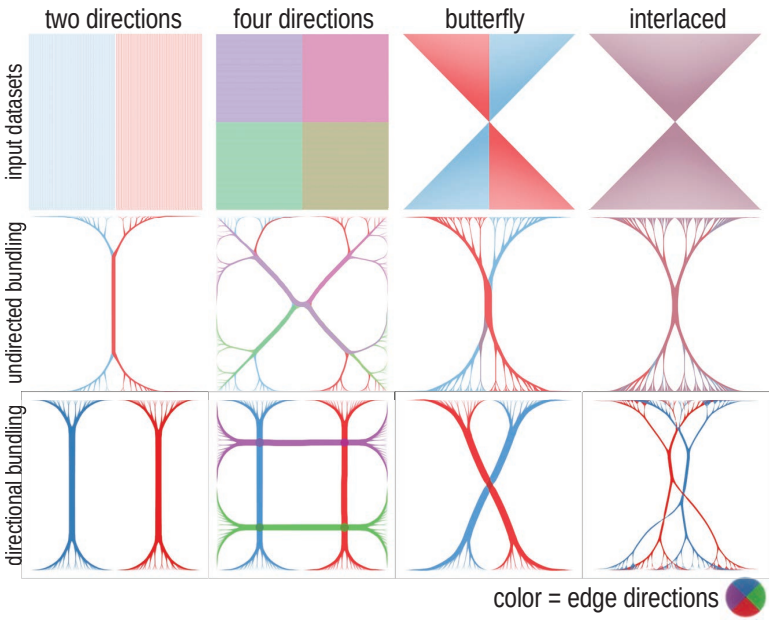


FIGURE 6.2 – Bundling non-directionnel et directionnels de quatre jeux de données avec FFTEB.

de passage à l'échelle de notre technique étaient essentielles pour générer des images agrégées avec une meilleure résolution. L'amélioration de la résolution des images agrégés permet de prendre en compte les détails des structures sous-jacentes à un jeu de données et par la même permet d'améliorer compréhension de la structure des chemins. Malgré les capacités de notre technique, certains aspects pourraient être encore améliorer. Par exemple, les derniers algorithmes de calcul de transformée de Fourier (Wang, Chandrasekaran et Chapman, 2016) pourraient permettre d'accélérer encore plus les capacités de notre technique. De plus, la faible complexité de notre agrégation basée attributs ouvre de nouvelles possibilité pour explorer virtuellement n'importe quelles fonctions de compatibilité entre chemins et ce en fonction de n'importe quel type d'attribut ou critères d'agrégation. Enfin, le caractère générique et les capacités de passage à l'échelle de FFTEB ouvre la voie pour une implémentation efficace d'une technique de bundling en 3-dimensions à travers de multiples cas d'application (*e.g.* le bundling d'IRM fonctionnelles dans un cadre médical, Böttger et al., 2014).

6.1 Kernel Density Estimation Based Bundling Methods

In this section, we outline the technicalities of existing bundling methods based on Kernel Density Estimation (KDE) from which FFTEB is inspired. First, we detail the peculiar technical steps of KDEEB (Hurter, Ersoy, and Telea, 2012). Then, we detail two existing attribute-driven variant of KDEEB: ADEB (Peysakhovich, Hurter, and Telea, 2015) and CUBu (Zwan, Codreanu, and Telea, 2016). Finally, we outline the kernel density based bundling related scalability issue.

6.1.1 Kernel Density Bundling

Kernel density bundling methods basically follows our unified bundling framework (introduced in chapter 4) as described next.

Similarity Definition

KDE-based bundling methods define the paths similarity by capturing path distance from an image (*i.e.* a density map). The undirected version of KDE (KDEEB, Hurter, Ersoy, and Telea, 2012) defines all paths as being potentially compatible, thus the compatibility function κ is defined as the unit function. As such for all $p_i \in D(P)$, $\kappa(p_i, p_j) = 1$. On the other hand, KDEEB still needs to define the similarity function δ. To do so, KDE methods first compute an edge-density map $\rho : \mathbb{Z}^2 \to \mathbb{R}^+$ by convolving all sampled points x of the paths p from $D(P)$ with a decaying radial kernel $K : \mathbb{R}^2 \to \mathbb{R}^+$ of support radius h. The kernel K is typically defined as a Gaussian or Epanechnikov (*i.e.* parabolic) (Epanechnikov, 1969). Thus, the edge density map ρ is defined as;

$$\rho(\mathbf{x} \in \mathbb{Z}^2) = \sum_{\mathbf{y} \in D(P) | \|\mathbf{x}-\mathbf{y}\| \leq h} K(\mathbf{y} - \mathbf{x}). \tag{6.4}$$

Put informally, the similarity of a path $p \in D(P)$ is defined as the closest maximum value of the density map (*i.e.* the gradient of the density map). As such, the similarity function δ is defined as the gradient of $\rho(\mathbf{x})$, *i.e.* $\delta(\mathbf{x}, D(P)) = \nabla\rho(\mathbf{x})$.

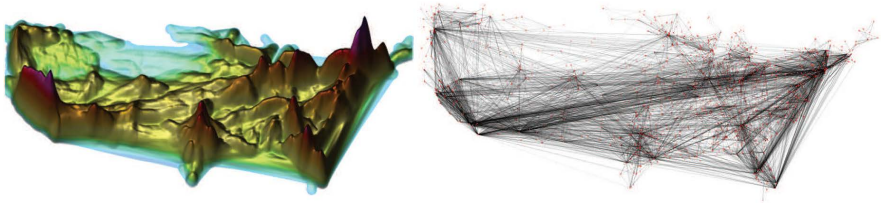
Bundling Operator

All KDE methods use an implicit image-based bundling operator (section 4.2.2). Here, the sampling points $\{\mathbf{x} \in p | p \in D(P)\}$ are advected along the gradient of the density map ρ with a small distance ϵ, to yield a new drawing of the path-set:

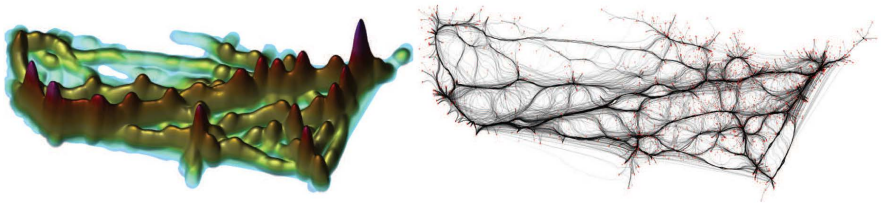
$$D^{new}(P) = \left\{ \mathbf{x}^{new} = \mathbf{x} + \epsilon \frac{\nabla\rho}{max(\|\nabla\rho\|, \iota)} \middle| \mathbf{x} \in D(P) \right\}, \tag{6.5}$$

The density gradient $\nabla\rho$ is normalized in a regularized manner and the term $\iota \simeq 10^{-5}$ takes care of null gradients. Moreover, normalizing the gradient constrains the movements $\mathbf{x^{new}}$ to a factor ϵ of the kernel bandwidth h. This limits paths from "jumping" from one side to the other of the local density maxima. Next, a 1D Laplacian smoothing pass is applied over all paths $\mathbf{p} \in D^{new}(P)$. This smoothing pass is needed

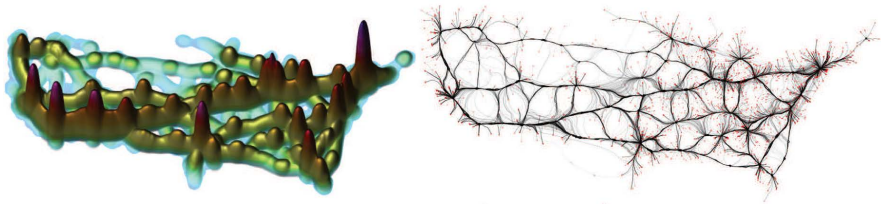
to regularize the estimation of $\nabla\rho$. All above steps can be efficiently done on the GPU: equation 6.4 maps to convolving $D(G)$ with a 2D texture encoding K; equation 6.5 and the Laplacian smoothing are mapped to simple computations done on a point-sampling of $D(P)$. Finally, the drawing $D(P)$ is resampled at each iteration to ensure a good spatial sample density, needed for a good estimation of ρ by equation 6.4. As all implicit methods, this process is repeated a number of I iterations (typically around 10) to yield the final bundled drawing $B(D(P))$. The idea is with each iteration to sharpen (a bit more) the edge density map thus yielding tighter bundles as shown in figure 6.3.



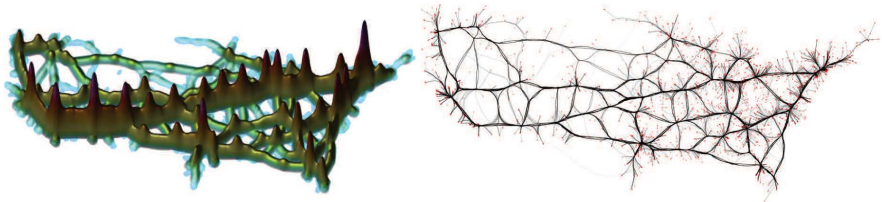
(A) Iteration 0



(B) Iteration 3



(C) Iteration 6



(D) Iteration 10

FIGURE 6.3 – Evolution of density map and corresponding bundling for the US migrations dataset ($|N|$=1715,$|E|$=9780) by KDEEB (Hurter, Ersoy, and Telea, 2012).

Visual Exploration

Between each iteration, the new path drawing $D^{new}(P)$ can be visually represented, displaying an animation of the bundling process. Bundled drawing can be visually explored using a wide variety of visual enhancement as presented in section 4.3. For example, KDEEB uses opacity map directly computed from ρ as well as shading and

bump mapping to emphasize high-density bundles. KDEEB also introduces an obstacle constrained bundling which is directly computed by modifying the density map ρ (as previously shown in section 4.3.3 and figure 4.13). More visual exploration details are presented by Hurter, Ersoy, and Telea, 2012 or Zwan, Codreanu, and Telea, 2016.

6.1.2 Attribute Driven KDE Bundling

As outlined in section 4.1, a high-interest case for bundling concerns datasets where the path compatibility κ is a function of both spatial position and data attributes of paths. Two bundling methods implementing the kernel density estimation technique define path compatibility based on directions and/or data attributes: ADEB (Peysakhovich, Hurter, and Telea, 2015) and CUBu (Zwan, Codreanu, and Telea, 2016). Essentially, these two methods replace equation 6.4 with

$$\rho(\mathbf{x} \in D(P)) = \sum_{\mathbf{y} \in D(P) | \|\mathbf{x}-\mathbf{y}\| \leq h} \kappa(\mathbf{a}(\mathbf{x}), \mathbf{a}(\mathbf{y})) \cdot K(\mathbf{y}-\mathbf{x}) \tag{6.6}$$

where $\mathbf{a} : E \to \mathbb{R}^2$ is a 2-dimensional attribute defined on the graph edges, and $\kappa : \mathbb{R}^2 \times \mathbb{R}^2 \to [-1;1]$ is a compatibility function that tells how similar two such attributes are. Typically, setting $\mathbf{a}$ to the tangent vector of a path at a given sampling point $\mathbf{x} \in p$ and κ to the scalar product over $\mathbb{R}^2$ yields directional bundling (Peysakhovich, Hurter, and Telea, 2015; Zwan, Codreanu, and Telea, 2016).

Apart from path direction, ADEB (Peysakhovich, Hurter, and Telea, 2015) proposes to use any path attributes to define the compatibility function κ. For instance, ADEB proposes bundling by time attribute value $\{t_i(\mathbf{x}) \in \mathbb{R} | \mathbf{x} \in p, p \in D(P)\}$ of a path-set by mapping the attribute t_i to a vector $\mathbf{t}_i = (\mathbf{t}_i^x, \mathbf{t}_i^y) \in \mathbb{R}^2$ using polar coordinates, *i.e.*

$$\mathbf{t}_i^x(\mathbf{x}) = cos\left(\frac{t_i(\mathbf{x}) - t_{min}}{t_{max} - t_{min}}\right), \mathbf{t}_i^y(\mathbf{x}) = sin\left(\frac{t_i(\mathbf{x}) - t_{min}}{t_{max} - t_{min}}\right) \tag{6.7}$$

with $t_i \in [t_{min}; t_{max}]$ over the entire path-set P. Again, by using the scalar product of $\mathbb{R}^2$ for this vector yields a temporally-based bundling. Here we note that the use of the polar coordinates ensure time folding properties (as detailed by Peysakhovich, Hurter, and Telea, 2015).

More generally, for any given set of attributes (*e.g.* of dimension m) of the path-set, as long as we can define a similarity function $\kappa : \mathbb{R}^m \times \mathbb{R}^m \to [0;1]$, KDE based techniques can yield attribute-based bundled drawings.

6.1.3 Complexity Analysis

The complexity of KDE based bundling methods strongly depends on the chosen computational model. Consider bundling a path-set drawing $D(P)$ having N sampling points, using I bundling iterations, a kernel K of diameter h pixels, and an image of $R \times R$ pixels.

For undirected bundling, KDEEB (Hurter, Ersoy, and Telea, 2012) implements equation 6.4 by convolving (scattering) the kernel K over all sample points of the path-set drawing $D(P)$, yielding a complexity of $O(INh^2)$. As h should be proportional to R to ensure good results, the complexity becomes $O(INR^2)$. CUBu (Zwan, Codreanu, and Telea, 2016) speeds this up by about 50 times by using a gathering strategy where equation 6.4 is evaluated at each image point based on nearby sampling points . Additionally, CUBu splits the 2D convolution in equation 6.4 into two 1D convolutions, since the kernel K is separable, *i.e.*, $K(x \in \mathbb{R}, y \in \mathbb{R})$ can be written as the product $K_x(x)K_y(y)$ of two 1D kernels K_x and K_y. This yields a complexity of $O(I(N+R^2h))$. Given the relation of h with R, this is equivalent to $O(I(N+R^3))$. Here, the cubic complexity in image size creates serious scalability issues when creating high-resolution bundled images. For instance, CUBu requires over 0.7 seconds to bundle a graph of about 4M sample points on a modern GPU (NVidia GTX 690) at 2048^2 screen resolution. Bundling a graph of one million paths (not huge by nowadays big data standards), each sampled with 50 points on average, would cost CUBu over eight seconds. These are too slow rates if we aim at interactive exploration of large graphs.

For attribute-driven KDE techniques, the computation of κ yields a higher complexity for both ADEB and CUBu in regards to KDEEB's complexity. Moreover, equation 6.6 is *not* always separable in two 1D convolutions as equation 6.4 was. Hence, the cost of ADEB's and CUBu's attribute-driven bundling is $O(I(N+R^2h)$ (cost of computing the undirected density map, needed for shading) plus $O(INh^2)$ (cost of Eqn. 6.6, identical to KDEEB), *i.e.* a total of $O(I(NR^2+R^3))$. The speed of CUBu is thus largely lost for attribute-driven bundling.

6.2 FFTEB Framework

We next describe our FFTEB method for bundling very large graphs using the spectral domain. Section 6.2.1 introduces our method. Sections 6.2.2 and 6.2.3 cover implementation details and our proposed GPU streaming technique for handling very large datasets, respectively.

6.2.1 Fourier Transform Approach

To address the scalability issues outlined above in section 6.1.3, we exploit the properties of the convolution theorem. Let f and g be two function in $\mathbb{L}^2$ (*i.e.* f and g are square integrable) with Fourier transforms $\mathcal{F}[f]$ and $\mathcal{F}[g]$ respectively. Then the convolution $f*g$ equals the inverse Fourier transform of the point-wise multiplication of the Fourier transforms of f and g, *i.e.*

$$\mathcal{F}[f*g] = \mathcal{F}[f] \cdot \mathcal{F}[g]. \tag{6.8}$$

If we note that the density ρ in equation 6.4 is nothing but the convolution $D(G) * K$, we derive from equations 6.4 and 6.8 that

$$\rho(\mathbf{x}) = \mathcal{F}^{-1}[\mathcal{F}[D(P)] \cdot \mathcal{F}[K]](\mathbf{x}), \tag{6.9}$$

where $\mathcal{F}^{-1}$ denotes the inverse Fourier transform and $\cdot$ denotes point-wise signal multiplication. We can even go a step further by showing that the gradient of the density map $\nabla\rho$, is in fact the convolution of the path-set drawing with the gradient of the kernel as follows

$$\delta(\mathbf{x}, D(P)) = \nabla\rho(\mathbf{x}) = [D(P) * \nabla K](\mathbf{x}). \tag{6.10}$$

Hence, we re-define the similarity function of KDE as

$$\delta(\mathbf{x}, D(P)) = \mathcal{F}^{-1}[\mathcal{F}[D(P)] \cdot \mathcal{F}[\nabla K]](\mathbf{x}). \tag{6.11}$$

As the kernel K generally needs to be computed only once, equation 6.11 shows that we can directly compute the gradient of the kernel ∇K once. This avoids the computation of the density map's gradient at each iteration. By using the convolution theorem, we shift the bundling process from the image space (*i.e.* the drawing $D(P)$) to the spectral space (*i.e.* the frequencies $\mathcal{F}[D(P)]$).

Attribute-Driven Bundling

Our FFT approach becomes even more valuable for attribute-driven bundling, if we restrict the compatibility function κ to be *bi-linear*. For example, directional bundling, where κ is a scalar product, meets this condition. In detail, for directional bundling in a 2-dimensional drawing space, from equations 6.6 and 6.11, we compute the gradient for $\{\mathbf{x} \in \mathbf{p} | \mathbf{p} \in D(P)\}$ as;

$$\nabla\rho(\mathbf{x}) = \sum_{\mathbf{y} \in D(P) | \|\mathbf{x}-\mathbf{y}\| \leq h} \kappa(\mathbf{a}(\mathbf{x}), \mathbf{a}(\mathbf{y})) \cdot \nabla K(\mathbf{y} - \mathbf{x}). \tag{6.12}$$

As κ is bi-linear, we can simplify equation 6.12 to

$$\begin{aligned} \nabla\rho(\mathbf{x}) = \Bigg(& \kappa\Big(\mathbf{a}(\mathbf{x}), \sum_{\mathbf{y} \in D(P) | \|\mathbf{x}-\mathbf{y}\| \leq h} \mathbf{a}(\mathbf{y}) \cdot \nabla_x K(\mathbf{y} - \mathbf{x})\Big), \\ & \kappa\Big(\mathbf{a}(\mathbf{x}), \sum_{\mathbf{y} \in D(P) | \|\mathbf{x}-\mathbf{y}\| \leq h} \mathbf{a}(\mathbf{y}) \cdot \nabla_y K(\mathbf{y} - \mathbf{x})\Big)\Bigg). \end{aligned} \tag{6.13}$$

For directional bundling, $\mathbf{a}$ is a 2-dimensional vector, we can rewrite equation 6.13 as

$$\nabla\rho(\mathbf{x}) = \Big(\kappa\Big(\mathbf{a}(\mathbf{x}), \sum_{\mathbf{y}\in D(P)|\|\mathbf{x}-\mathbf{y}\|\leq h} \sum_{i=1}^{2} \mathbf{a}_i(\mathbf{y}) \cdot \nabla_x K(\mathbf{y}-\mathbf{x}) \cdot e_i\Big), \\ \kappa\Big(\mathbf{a}(\mathbf{x}), \sum_{\mathbf{y}\in D(P)|\|\mathbf{x}-\mathbf{y}\|\leq h} \sum_{i=1}^{2} \mathbf{a}_i(\mathbf{y}) \cdot \nabla_y K(\mathbf{y}-\mathbf{x}) \cdot e_i\Big)\Big), \tag{6.14}$$

where $\{e_i\}$ are the unit vectors of the orthonormal basis of $\mathbb{R}^2$ (*i.e.* e_x and e_y). Using the convolution theorem (equation 6.8), we can rewrite equation 6.14 as

$$\nabla\rho(\mathbf{x}\in D(P)) = \Big(\kappa\Big(\mathbf{a}(\mathbf{x}), \sum_{i=1}^{2} \mathcal{F}^{-1}\left[\mathcal{F}\left[\mathbf{a}_i\right] \cdot \mathcal{F}\left[\nabla_x K\right]\right](\mathbf{x}) \cdot e_i\Big), \\ \kappa\Big(\mathbf{a}(\mathbf{x}), \sum_{i=1}^{2} \mathcal{F}^{-1}\left[\mathcal{F}\left[\mathbf{a}_i\right] \cdot \mathcal{F}\left[\nabla_y K\right]\right](\mathbf{x}) \cdot e_i\Big)\Big), \tag{6.15}$$

where $\mathcal{F}\left[\mathbf{a}_i\right]$ is the Fourier transform of the i^{th} component of the attribute signal $\mathbf{a}$ defined over $D(G)$. For directional bundling, the two component are $\mathbf{a}_x$ and $\mathbf{a}_y$.

Generalization

Our approach can be extended to any m-dimensional bilinear compatibility function $\kappa : \mathbb{R}^m \times \mathbb{R}^m \to [-1;1]$ and any n-dimensional projection space of the path-set (although it is typically 2 or 3). Let $\{e_i\}$ be the orthonormal basis of $\mathbb{R}^m$ and $\{u_j\}$ the orthonomal basis of $\mathbb{R}^n$, then δ can be generalized from equation 6.15 as

$$\delta(\mathbf{x}, D(P)) = \sum_{j=1}^{n} \kappa\left(\mathbf{a}(\mathbf{x}), \sum_{i=1}^{m} \mathcal{F}^{-1}[\mathcal{F}[\mathbf{a}_i] \cdot \mathcal{F}[\nabla_j K]](\mathbf{x}) \cdot e_i\right) \cdot u_j, \tag{6.16}$$

where $\mathbf{x} \in \mathbb{R}^n$, $D(P) \in \mathbb{R}^n$, $\nabla K \in \mathbb{R}^n$ and $\mathbf{a} \in \mathbb{R}^m$. As in equation 6.15, $\mathcal{F}\left[\mathbf{a}_i\right]$ is the Fourier transform of the component i of the attribute signal $\mathbf{a}$ defined over $D(G)$.

Complexity Improvements

Equation 6.9 can be implemented on the GPU more efficiently than equation 6.4. First, we note that Eqn. 6.4 needed to find all points $\mathbf{y} \in D(G)$ within a radius h from every image pixel $\mathbf{x}$. For this, CUBu computes a special map recording the number of sample points of $D(G)$ falling over each pixel $\mathbf{x}$, and this computation needs atomic write operations to parallelize (Zwan, Codreanu, and Telea, 2016, section 3.1). In contrast, equation 6.9 only reads and writes every image pixel once. Secondly, approximating $\mathcal{F}$ with the Discrete Fourier Transform (Cooley, Lewis, and Welch, 1967) for an input image of R^2 pixels requires a spectral map of only $((R-1)/2)^2$ pixels – which is roughly four times less than the image size needed for equation 6.4. Overall, computing ρ with equation 6.9 requires a direct and an inverse Fourier transform for each of the I bundling iterations, followed by a simple point-wise signal multiplication. Using the

CUFFT CUDA library for FFT computations, this amounts to $O(R^2 \log R)$ operations per input image of $R \times R$ pixels (Podlozhnyuk, 2007) per iteration, thus a total FFTEB complexity of $O(I(N + R^2 \log R))$, which is lower than CUBu's $O(I(N + R^3))$.

For the attribute-driven version of bundling, several differences between our model and CUBu and ADEB exist, as follows. First, equation 6.15 has the same complexity as equation 6.9, *i.e.*, $O(I(N + R^2 \log R))$, which is smaller than CUBu-directional and ADEB's $O(I(NR^2 + R^3))$ (see section 6.1.1). In particular, the costly NR^2 term in the latter is not present in our formulation. This is due to the efficiency of FFT-based convolution, which is higher than classical convolution for non-separable large kernels, and also due to the bi-linearity property of κ. Secondly, our choice of compatibility functions is more flexible than ADEB – for example, for directional bundling, we can use the scalar product of path tangents (like CUBu) which attracts compatible paths and repels incompatible ones. In ADEB, only the attraction part was present.

6.2.2 Implementation Details

FFTEB's pipeline is similar to KDEEB (figure 6.4). In detail, to bundle paths with attributes $\mathbf{a} \in \mathbb{R}^2$ in a two dimensional drawing space (*i.e.* $D(P) \in \mathbb{R}^2$) FFTEB implements the following steps.

First we compute the similarity between paths. For this, the 2 scalar components (a_x, a_y) of $\mathbf{a}$ are stored in 2 floating-point textures, and a DFT $\mathcal{F}[a_i]$ is computed for each of them. The kernel gradient DFT $\mathcal{F}[\nabla K]$ is computed once as it does not change during bundling. Next, the point-wise products of each $\mathcal{F}[a_i]$ and the two components $\mathcal{F}[\nabla_x K]$ and $\mathcal{F}[\nabla_y K]$ (with K repeated to reach the same size as ρ) are computed, yielding $2 * 2$ textures. The inverse Fourier transform of each such texture is next computed, yielding another $2 * 2$ textures. Next, $\nabla\rho$ is computed (equation 6.15) by

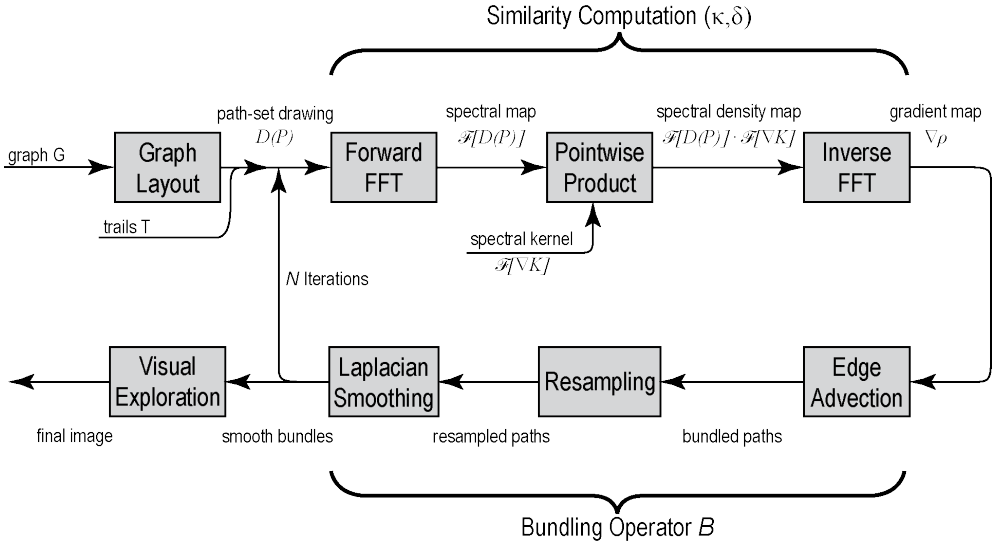


FIGURE 6.4 – Flowchart of FFTEB.

evaluating two 2-dimensional compatibilities of $\kappa(\mathbf{a})$ with the two corresponding 2-dimensional components of the convolutions of $\mathbf{a}$ with the kernel gradient components $\nabla_x K$ and $\nabla_y K$, respectively. This process yields $\delta(\mathbf{x}, D(P))$. Informally put, for a given sampling point of a path $\mathbf{x} \in D(P)$, it yields the gradient towards the most similar path in the surroundings of the sampling point $\mathbf{x}$.

Secondly, we apply the bundling operator. Thus, the sampling points are advected as in equation 6.5. Next a Laplacian smoothing, and a resampling of $D(P)$ are computed. This part of the process is similar to older KDE-based techniques bundling and implemented as detailled in section 6.1.1.

Finally, in the visual exploration step, bundled paths are rendered as colored using the pseudo-shading based on the density map height described by Zwan, Codreanu, and Telea, 2016. Here, the color-mapping is user defined upon a chosen attribute. We additionally emphasize important (high-density) bundles by scaling the thickness of each drawn bundled path with the local density value $\rho(\mathbf{x})$. This way, important bundles are rendered *both* thicker and with a more prominent shading. This lets us quickly discover these even in complex visualizations, as we shall see in section 6.3.1.

All of the above steps are implemented in NVidia's CUDA programming language using the CUFFT library for fast computation of direct and inverse DFT's. An implementation of FFTEB is provided at Lhuillier, Hurter, and Telea, 2016.

Generalizing, the computation of the similarity between paths for m-dimensional attributes $\mathbf{a}$ in a n-dimensional space yields more textures. In details, each scalar component of $\mathbf{a}$ are stored in m textures and the DFT is applied. The point-wise product between the n component of the kernel gradient and the m scalar component yields nm textures and we compute nm inverse DFTs. Finally, $\nabla\rho$ is evaluated as the n-dimensional m-dimensional compatibilities of $\kappa(\mathbf{a})$. The rest of the process remains unchanged.

6.2.3 Scalability and Streaming

As outlined in our introduction, GPU-based bundling is challenged by large path-sets. Let's consider the *amazon* graph (used by Gansner et al., 2011 and Zwan, Codreanu, and Telea, 2016), which has over 900K edges. A dense point sampling of this graph at a resolution of 2000^2 pixels (1 sample point every few pixels) easily yields $N > 1$ billion points. Per point, we need to store at least its two floating-point coordinates (8 bytes of storage), yielding a total need for at least 8GB VRAM. Edge attributes $\mathbf{a}(\mathbf{x})$ add extra storage. Such large datasets do not fit in the VRAM of current consumer-grade graphics cards.

We address this data volume challenge by a simple but efficient streaming scheme for FFTEB bundling, as follows. We divide the sampled paths of $D(P)$, which is stored on the CPU, into C chunks c_i, each having roughly N/C sample points, so that a chunk fits in the available VRAM. We next stream each chunk to the GPU, and accumulate its density-map gradient $\nabla\rho_i$ (computed as described in section 6.2.1), in a global gradient map $\nabla\rho$. After all chunks have added their contribution to $\nabla\rho$, we stream them again,

one by one, to the GPU, to advect, smooth, and resample their paths. After a chunk is processed, we stream its new sample points back to the CPU to make space for the next chunk. This process requires $2C$ CPU-to-GPU and C GPU-to-CPU chunk transfers, *i.e.* a total data transfer of sizes $2N$ (CPU-to-GPU) and N (GPU-to-CPU), respectively. Compared to the case where the entire sampled graph fits in VRAM, our streaming process only adds an extra cost of size N per iteration. Examples of using streaming for very large graphs and streaming performance considerations are given in section 6.3.4 and 6.4, respectively.

6.2.4 Simple Bundling Example

Similarly to the tests used by ADEB (Peysakhovich, Hurter, and Telea, 2015), we illustrate FFTEB via four simple datasets bundled using the undirected, respectively, directional compatibility criteria (Fig. 6.5). Paths are color-coded to indicate direction. Each dataset has 2000 paths. As visible, undirected bundling (middle row) creates patterns which do not convey an intuitive simplification of the data – for instance, the undirected bundlings of the *two directions*, *butterfly*, and *interlaced* datasets look very similar, while the corresponding datasets are quite different. The bottom row shows the FFTEB directional bundling, using the scalar product on $\mathbb{R}^2$ as compatibility function. As visible, the bundled graphs now capture well the structure of the input datasets. In particular, we can see how paths with opposite directions are repelled from one another to yield almost orthogonal crossings of incompatible paths. Overall, thus, FFTEB reaches the same bundling quality as ADEB but with larger test-sets (3 orders of magnitude).

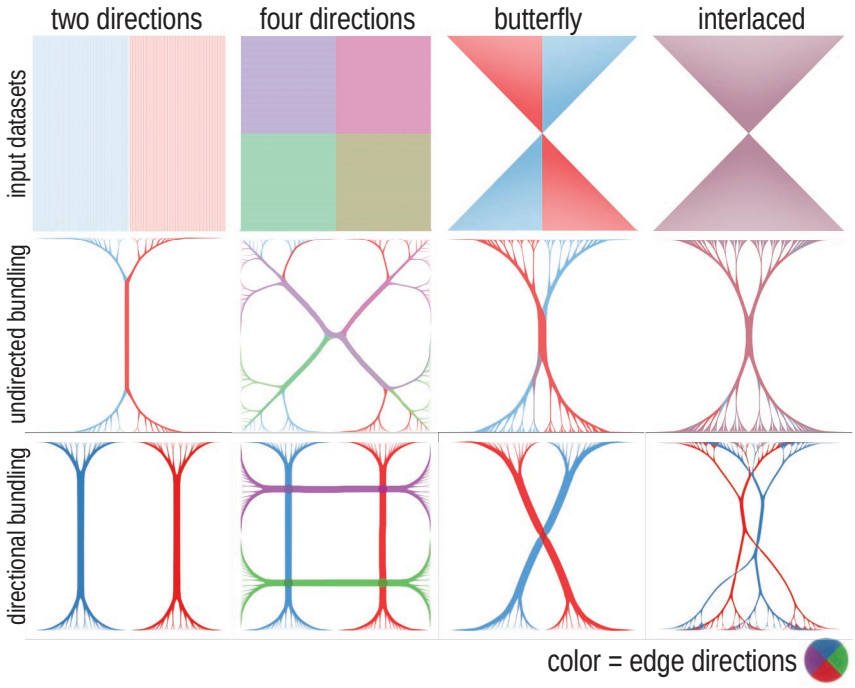


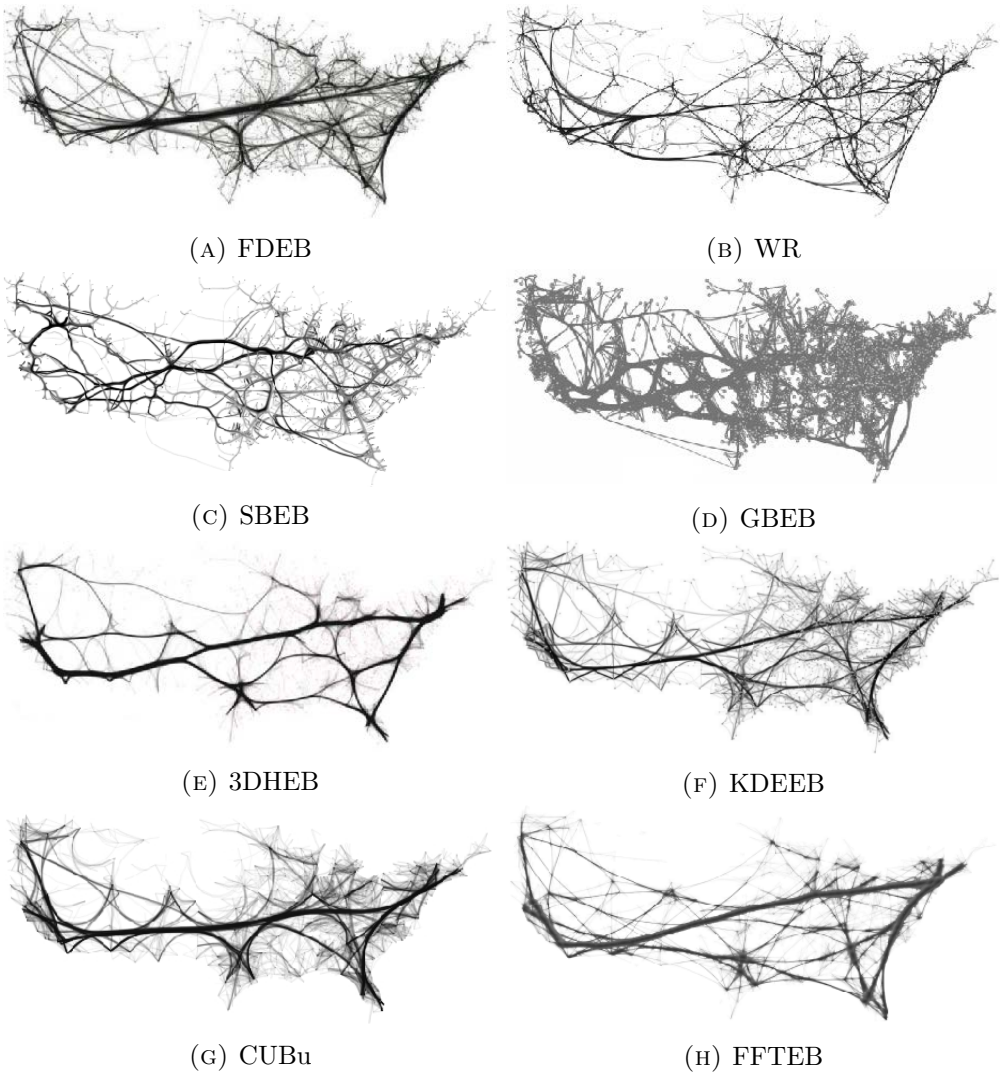
FIGURE 6.5 – FFTEB undirected and directional bundling of four datasets.

6.3 Example Applications

We next illustrate FFTEB with four studies: bundling quality as a function of resolution (section 6.3.2); attributed-driven bundling of eye-tracking data (section 6.3.3); and directional bundling of huge graphs that do not fit in VRAM using streaming (section 6.3.4). These use-cases outline together the three main capabilities of FFTEB – computational speed, handling attributed graphs, and handling large data volumes.

6.3.1 Comparison

Figure 6.6 compares the undirected FFTEB bundling with eight other well-known bundling methods (FDEB, WR, KDEEB, SBEB, 3DHEB, GBEB, CUBu, and MINGLE). As datasets, we used the *US migrations* (9780 edges). For the relatively small *US migrations* dataset, FFTEB produces results which are very similar in terms of



(A) FDEB (B) WR

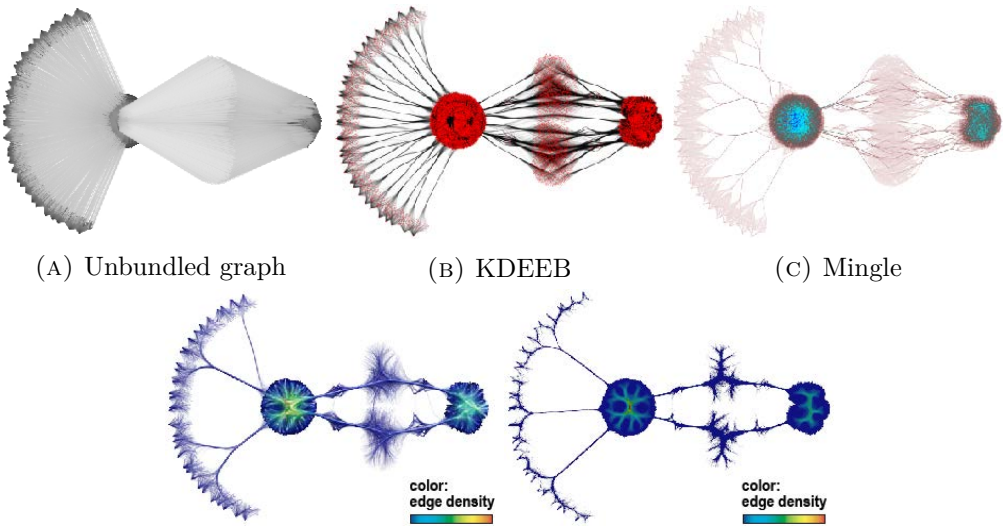
(C) SBEB (D) GBEB

(E) 3DHEB (F) KDEEB

(G) CUBu (H) FFTEB

FIGURE 6.6 – Comparison of undirected FFTEB with several state-of-the-art bundling methods for the *US migrations* dataset (see section 6.3.1)

positions and structure of the main bundles to FDEB, KDEEB, 3DHEB, and CUBu. The small differences are explained by different parameter values used for the kernel size, amount of Laplacian smoothing, advection speed, and shading style (which are not exactly replicable from the aforementioned papers). These similarities are expected, as FFTEB shares the same bundling principle with all the named methods. SBEB and GBEB produce, as expected, different results since they use a different model for edge routing. In figure 6.7, we compare FFTEB with three bundling methods on the *net50* dataset (928K edges). For this much larger dataset, FFTEB again produces results which are very similar with KDEEB and CUBu, and notably a much stronger simplification of the graph structure than KDEEB and MINGLE. This is explained by the fact that, for this example, both FFTEB and CUBu use a kernel size h (equation 6.4) about three times larger than KDEEB. In turn, this is due the quadratic complexity of KDEEB in h (section 6.1.1), which makes this method impracticable for strongly simplifying very large graphs like *net50*. The lower simplification of MINGLE is explained by its decision to bundle an edge with maximally $k = 10$ nearest neighbors thereof (see Gansner et al., 2011, section 4). As noted there, using higher bundling factors does not decrease the saved ink to draw the bundled graph. However, this also decreases the amount of simplification one can achieve.



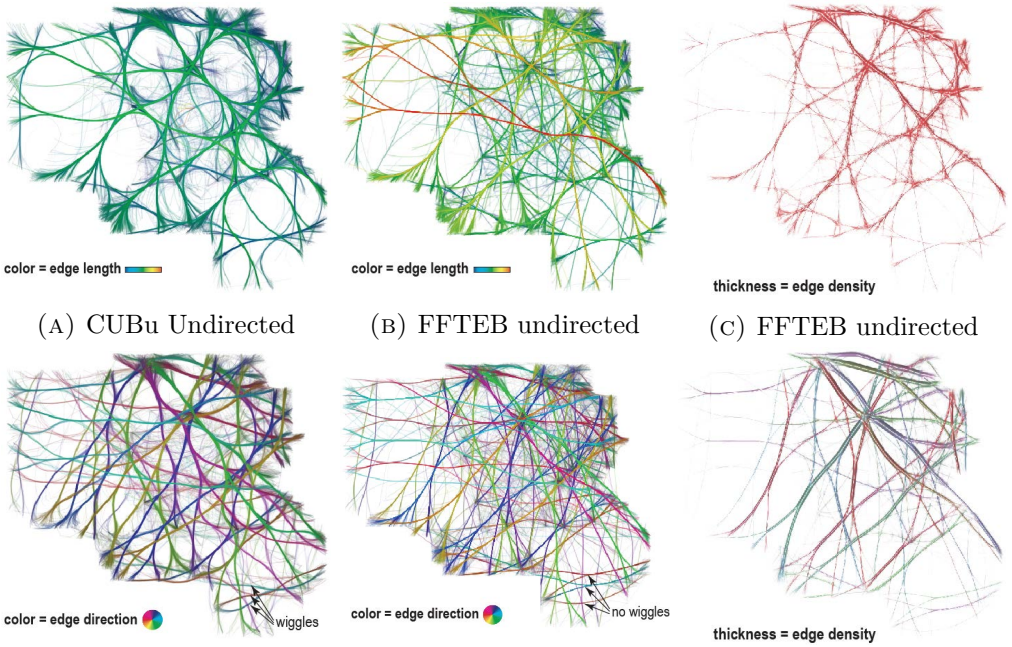
(A) Unbundled graph (B) KDEEB (C) Mingle

(D) CUBu (E) FFTEB

FIGURE 6.7 – Comparison of undirected FFTEB with several state-of-the-art bundling methods for the *net50* datasets (see section 6.3.1)

Figure 6.8 compares both the directional and undirected FFTEB bundling with the respective variants of CUBu for the *France airlines* dataset. We see that FFTEB and CUBu methods deliver similar results, which is explained by the fact that they use the same underlying algorithm (kernel density bundling, explained in section 6.1.1). In theory the two bundling results should be the same but further investigation should be done to validate this theoretical hypothesis. The differences may come from the data transformation accuracy between spectral and image space. The less wiggling bundles in FFTEB allow a better end-to-end tracing of directed bundles, see *e.g.* the

marked details in figures 6.8e and 6.8f. As before, such differences are explained by small parameter-value differences, and should not be interpreted as an inherent higher-quality of the FFTEB method. The rightmost images (figures 6.8b and 6.8c) show a strong simplification of the directional and undirected FFTEB bundlings, done by rendering the respective bundled graphs with an opacity proportional to the local edge-density and by setting edge line-widths to the same density (section 6.2.2). This both reduces clutter caused by spurious edges or low-importance bundles and emphasizes the largest bundles in the graph.



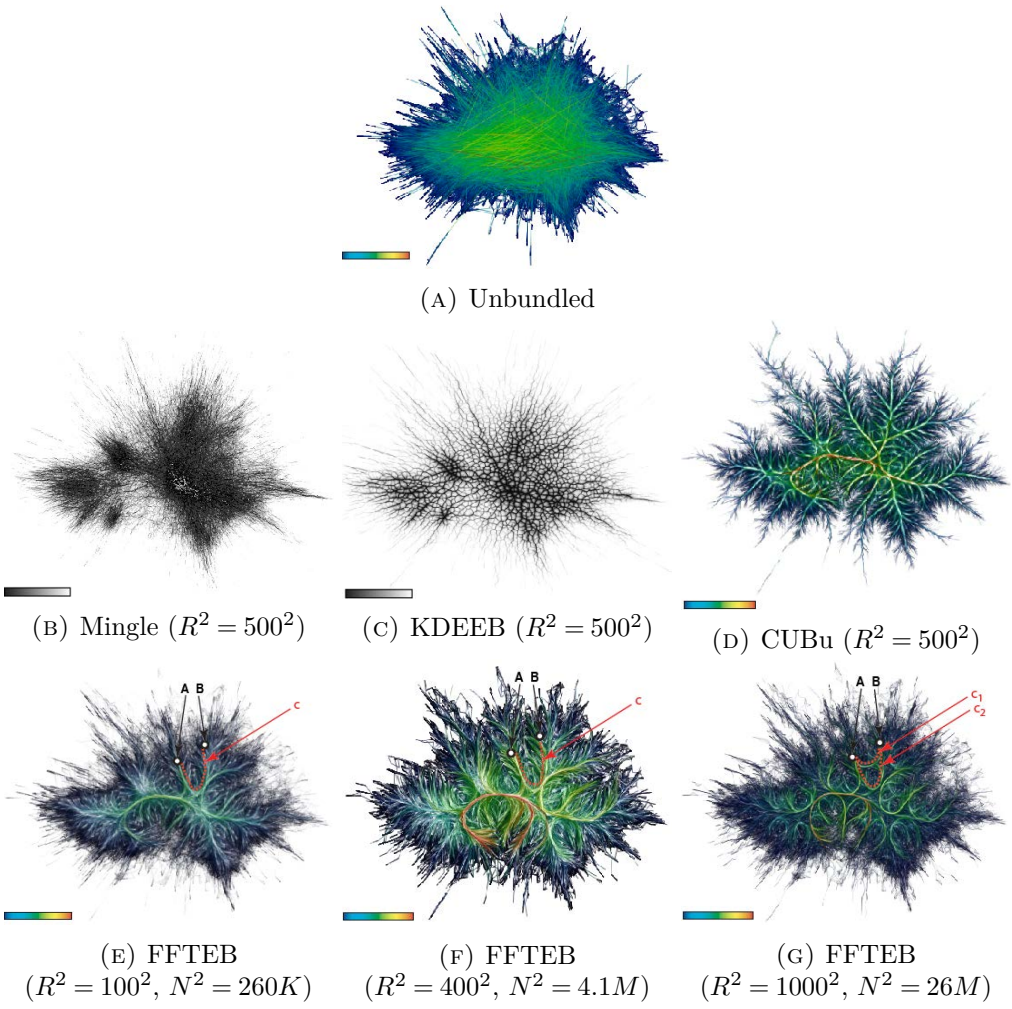
(A) CUBu Undirected (B) FFTEB undirected (C) FFTEB undirected

(D) CUBu (E) FFTEB directional (F) FFTEB directional

FIGURE 6.8 – Comparison of directional and undirected FFTEB with the corresponding CUBu variants for the *France airlines* dataset. See section 6.3.1.

6.3.2 Bundling Quality *vs* Resolution

As stated in section 6.2.1, FFTEB's complexity is $O(I(N + R^2 \log R))$ for N samples, I iterations, and a resolution of $R \times R$ pixels. Notably, R has the highest influence on computational speed. It seems tempting to use lower resolutions R to accelerate computations. However, for very large graphs, this can create problems in distinguishing finer-scale details, as discussed next. To study the effect of R on the results, we consider the *amazon* graph, which contains records of over 900K co-purchase relations between 520K items on *amazon.com* (Gansner et al., 2011). We bundle this graph at three different resolutions ($R^2 \in \{100^2, 400^2, 1000^2\}$ pixels). We set the kernel size h (equation 6.4) to a value of $R/10$, following Hurter, Ersoy, and Telea, 2012; Peysakhovich, Hurter, and Telea, 2015 and Zwan, Codreanu, and Telea, 2016, and also adapt the sampling step of edges to be about 3 pixels for each R value, again in line with earlier studies such as CUBu (Zwan, Codreanu, and Telea, 2016).



(A) Unbundled

(B) Mingle ($R^2 = 500^2$)

(C) KDEEB ($R^2 = 500^2$)

(D) CUBu ($R^2 = 500^2$)

(E) FFTEB ($R^2 = 100^2$, $N^2 = 260K$)

(F) FFTEB ($R^2 = 400^2$, $N^2 = 4.1M$)

(G) FFTEB ($R^2 = 1000^2$, $N^2 = 26M$)

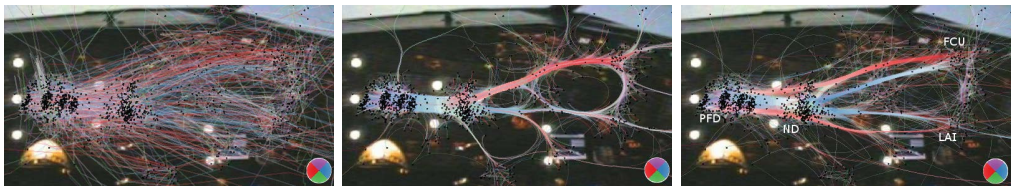
FIGURE 6.9 – Bundling of *amazon* graph Gansner et al., 2011 for different screen resolutions R and total number of edge sampling points N. Colors map edge density.

Figure 6.9 shows the original unbundled graph and bundling results using MINGLE, KDEEB, CUBu, and FFTEB. As visible in figures 6.9b and 6.9c, both MINGLE and KDEEB do not achieve much in terms of highlighting structure in the bundled result, apart from being able to show high-density areas. As explained in section 6.3.1 for the *net50* graph, such limitations are due to their relative low bundling strength, which is in turn justified by computational cost (KDEEB) and the desire to minimize edge bending (MINGLE), respectively. For large graphs like *amazon*, such constraints are not very effective. The density-based shading of CUBu highlights quite well the most important (densest) bundles at a resolution of 500^2 pixels (figure 6.9d). At similar resolutions, FFTEB and CUBu generate quite similar results, which is not surprising, given that they are based on the same kernel-density process (equations 6.4, 6.5). The last three images, computed with FFTEB show, the added-value of using a high resolution: Consider, for instance, two points A and B which appear to be part of the same bundle. Using low-to-middle resolution bundlings (figures 6.9e and 6.9f) lets one see a path c between A and B indicated by the respective red dotted lines. However,

when we increase the resolution, we see that c actually splits in two paths c_1 and c_2 (figure 6.9g), where c_2 is a shorter path connecting A and B which was not visible at lower resolutions. Hence, high resolutions are needed to obtain detailed bundling insights for very large graphs. Since FFTEB scales better than all other recent fast bundling methods in terms of high resolution R and graph size (number of sample points N), this method is clearly more suitable for generating such detailed images. We further present figures to support this higher scalability of FFTEB in section 6.4.

6.3.3 Attribute-Driven Bundling

Our next study considers a dataset containing eye tracking data. In the underlying use-case, described in full detail by Peysakhovich, Hurter, and Telea, 2015; Hurter et al., 2014a, an eye tracker was used to record the motion of the eyes of a pilot that looks at a plane's cockpit dashboard while performing a landing manoeuvre in a flight simulator. The recorded data can be represented as a graph where vertices are points to which the eyes where drawn (so-called fixation points), and the connecting edges indicate eye-movement between the vertices (so-called saccades). As detailed by Hurter et al., 2014a, the aim of analyzing such data is to detect high-level repetitive patterns in the eye movement, which in turn let one understand how well the subject visually scanned the dashboard instruments to perform the required maneouvres. Drawing the raw saccades between fixation points generates a completely cluttered image, from which high-level connections or connection patterns between fixation points cannot be inferred (figure 6.10a). Undirected bundling can be used to reduce clutter and extract a few simple eye-motion patterns from the data (figure. 6.10b). However, such an image is actually too much simplified, as it bundles together saccades that run in opposite directions. Directional bundling corrects this by separating saccades that link the same spatial areas but have opposing directions. Figures 6.10c and 6.10d show the results of ADEB and CUBu for directional bundling. While these images show more detail than undirected bundling, as explained above, they still suffer from a certain amount of clutter, visible in terms of stray saccades which have not been bundled succesfully. The explanation of these is technical: Since ADEB and CUBu do not scale very well for directional bundling, as explained earlier and also detailed by timings in section 6.4.1, the number of sample points N and the kernel size h cannot be too high if we want to obtain a high bundling speed. Hence, trails which are too far away will not be bundled, and will appear as stray curves. In contrast, FFTEB allows using large values for N and h without sacrificing performance (see Sec. 6.4.1 further for performance figures). In turn, this allows a tighter bundling that yields cleaner, more simplified, images – compare figure 6.10e with figures 6.10c and 6.10d. We should stress that FFTEB's higher quality bundling is a consequence of its ability to use parameter ranges for which earlier methods become impractical, and not a result of it using a different bundling heuristic – for the same parameter values, FFTEB, ADEB, and CUBu yield very similar images, up to local differences caused by implementation details.



(A) Raw eye trails (B) KDEEB undirected (C) ADEB directional

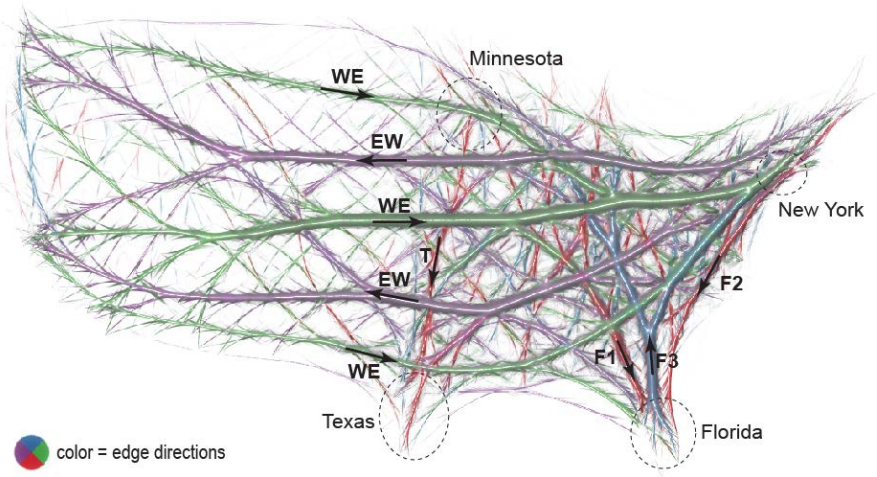
(D) CUBu directional (E) FFTEB directional (F) FFTEB directional & time

FIGURE 6.10 – Bundling of eye trails for airline pilot training.
See section. 6.3.3.

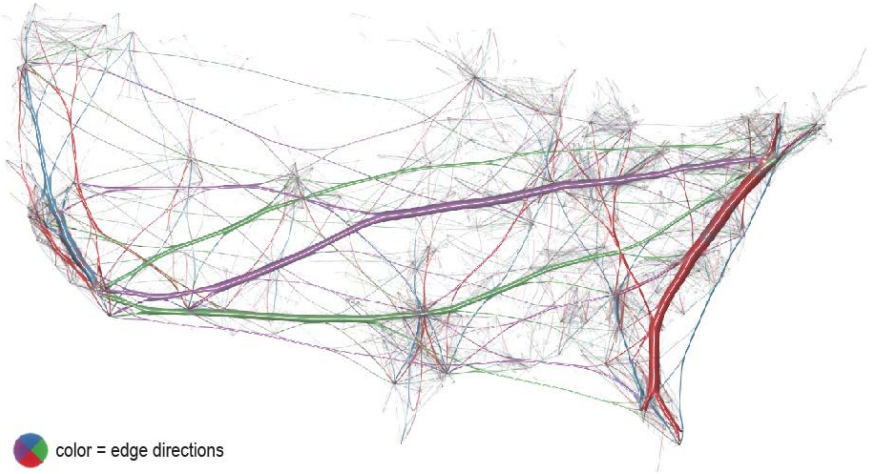
Figure 6.10f shows a new variation of attribute-based bundling done with FFTEB. Here, the edge similarity function κ (equation 6.12) combines both direction and time-information from the trails, each having the same weight. That is, trails are bundled if they run in similar directions and at similar time instants. The result is visually quite similar to the direction-only bundling. This tells us that eye saccades that run in different directions in the image, *i.e.* present in different bundles, occur at quite different moments during the recorded sequence – if not, then such trails would be bundled together due to their high time similarity, and thus figure 6.10f would look quite different from figure 6.10e.

6.3.4 Very Large Directional Graph Bundling

Our last study shows how FFTEB (directionally) bundles very large graphs whose point-samplings do not fit in VRAM. For this, we use a *migrations* graph of 600K edges that describes the relocation of people in the US over one year (U.S. Census Bureau, 2003). Sampling this graph at a screen resolution of 1000^2 pixels, using the sampling step constraints of KDE-based methods discussed earlier, yields over 63 *billion* points. This is three *orders of magnitude* larger than the largest sampled graph we know to have been bundled (Zwan, Codreanu, and Telea, 2016, *amazon* graph, 19M sample points). While earlier bundling techniques considered subsets of this graph (Gansner et al., 2011, 9660 edges; Zwan, Codreanu, and Telea, 2016; Lambert, Bourqui, and Auber, 2010b; Holten and Van Wijk, 2009, 9780 edges; Cui et al., 2008, 9798 edges), this is the first time the entire dataset is bundled. Also, this is the largest example of *directional* bundling known to us so far (600K edges *vs worldflights* graph by Zwan, Codreanu, and Telea, 2016, 26K edges). Our 63 billion point sampling requires over 15GB of memory, so the streaming feature of FFTEB is needed to handle this dataset on any current GPU. Performance considerations for this graph are discussed in Sec. 6.4.



(A) large US *migration* graph U.S. Census Bureau, 2003
(600K edges, 63 billion sampling points)



(B) small US *migration* graph
(9780 edges, 290K sampling points)

FIGURE 6.11 – Directional bundling of very large datasets.

Figure 6.11a shows the large US *migrations* graph bundled by FFTEB, with edge density encoded into bundle width and edge directions encoded by colors. The image reveals three main migratory west-to-east fluxes (figure 6.11a , thick green bundles WE) and two main fluxes in the opposite direction (figure 6.11a, thick purple bundles EW). On the vertical direction, we find that the eastern half of the country has much more migration than the western half. For the former, migration can be summarized by a flow going to Texas (figure 6.11a, red bundle T), two flows ending in Florida (figure 6.11a, red bundles F1 and F2); and one flow leaving Florida towards the Minnesota and New York areas (figure 6.11a, blue branching bundle F3). As visible, FFTEB can succesfully produce an image that conveys a simplified, yet clear, view of the main migratory patterns in this very large dataset.

As a comparison, Figure 6.11b shows the small US *migrations* graph, directionally bundled by FFTEB. This is the same graph as used in figure 6.6 (9780 edges), but bundled directionally this time. As visible, the small bundled graph conveys a quite

different insight from the full graph: The overall picture in terms of connection patterns is different, and the distribution of bundle thicknesses is very different. This indicates that a *subsampled* graph cannot always convey the same insight as the full graph. Hence, using an entire, non-subsampled dataset is the best option to avoid such subsampling issues. The added-value of FFTEB becomes clear here, as our method lets us do this even for very large graphs (large N values).

6.4 Discussion

We next discuss several aspects of FFTEB. First, we compare in section 6.4.1 the computational performance of FFTEB with the fastest-known bundling techniques we know – KDEEB, CUBu, and ADEB. As a benchmark, we use the six graphs in tables 6.1, 6.2 and 6.3, which appear in all recent bundling papers. In section 6.4.2, we analyze FFTEB's performance as a function of its two free parameters – image resolution R and graph sample-size N. We summarize the advantages and limitations of FFTEB in Sec. 6.4.3

6.4.1 Perfomance Comparison

For a start, we must note that the compared bundling methods treat attributed graphs differently. KDEEB cannot do attribute-based bundling. CUBu offers attribute-based bundling in terms of edge directions, apart from undirected bundling. FFTEB does attribute-based bundling by definition. For a fair comparison, we thus compare KDEEB (which does no attribute-based bundling) with FFTEB (for which we set the compatibility function κ (equation 6.15) to identity to emulate undirected bundling) and with CUBu (undirected version); and separately ADEB with the directional version of CUBu and with FFTEB.

2D undirected bundling: All tests were done using a single-GPU 4GB NVidia GTX 690 graphics card. The tests presented here use an image resolution of 1000^2 pixels and a kernel size of $h = 21$ pixels. We adapted the edge-sampling parameters in all tested methods (KDEEB, ADEB, CUBu, FFTEB) so as to generate roughly the same number of sample points N for a graph. Exactly identical N values for the three tested methods (KDEEB, CUBu, FFTEB) could not be obtained, since N is not an explicit parameter of any of these methods, but a result of setting an edge-sampling-density parameter. Table 6.1 shows the obtained timings. We see that both FFTEB and CUBu are 50 to 100 times faster than KDEEB. The important message, however, is that FFTEB is up to 2 times faster than CUBu, especially for larger datasets. This makes FFTEB the fastest undirected-bundling method in existence.

TABLE 6.1 – Timing comparison of isotropic bundling methods: KDEEB (Hurter, Ersoy, and Telea, 2012), CUBu (Zwan, Codreanu, and Telea, 2016), and FFTEB.

Graph	Edges	**2D undirected bundling methods**					
		KDEEB		CUBu		FFTEB	
	$\|E\|$	Samples	Time(ms)	Samples	Time(ms)	Samples	Time(ms)
US airlines	2099	86K	500	86K	14	105K	10
Poker	2127	50K	400	50K	11	60K	8
Radial	4021	290K	1500	290K	23	290K	13
US migration	9780	220K	1500	221K	24	300K	15
France airlines	17275	330K	1800	330K	25	330K	16
Amazon	899791	19M	8053	19M	152	19M	93

2D directional bundling: For these tests, we use edge directions as compatibility criterion. Parameter settings are identical to the undirected bundling evaluation presented above. Table 6.2 shows the results. ADEB results for *amazon* miss, as this graph is too large for the implementation by Peysakhovich, Hurter, and Telea, 2015. FFTEB is 10 to 50 times faster than ADEB, with a larger speed-up for larger graphs. Also, we see that the directional version of FFTEB has basically the same cost as the undirected version. Compared to CUBu, FFTEB is roughly 3 times faster.

TABLE 6.2 – Timing comparison of directionnal bundling methods: ADEB (Peysakhovich, Hurter, and Telea, 2015), CUBu (Zwan, Codreanu, and Telea, 2016), and FFTEB.

Graph	Edges	**2D directional bundling methods**					
		ADEB		CUBu		FFTEB	
	$\|E\|$	Samples	Time(ms)	Samples	Time(ms)	Samples	Time(ms)
US airlines	2099	80K	150	86K	28	88K	11
Poker	2127	50K	200	50K	22	50K	6
Radial	4021	300K	450	290K	46	300K	13
US migration	9780	260K	650	221K	48	290K	15
France airlines	17275	250K	400	330K	50	250K	13
Amazon	899791	n/a	n/a	19M	304	19M	98

3D undirected bundling: Here we used two implementations of 3 dimensional bundling techniques based on KDE. We compare a 3D GPU-based implementation of KDEEB with FFTEB in 3D but implemented in CPU. Bundling is computed only on the path geometrical proximity (*i.e.* without any attribute compatibility) on the same computer at an image resolution R of 100^3 (*i.e.* 10^6 texels) and a kernel K of 10 pixels. Table 6.3 shows our results. We see here that even in CPU, FFTEB is faster by 2 to 3 orders of magnitude than the 3-dimensional version of KDEEB implemented in GPU. This result is a direct consequence of the lower complexity of FFTEB compared to KDEEB.

TABLE 6.3 – Timing comparison of 3D undirectionnal bundling methods: 3D implementation of KDEEB in GPU and 3D implement of FFTEB in CPU.

Graph	Edges	**3D undirected bundling methods**			
		3D-KDEEB		FFTEB (CPU–3D)	
	$\|E\|$	Samples	Time(ms)	Samples	Time(ms)
US airlines	2099	80K	5300	81K	207
France airlines	17275	543K	38800	515K	293

Given the results for undirected (2D and 3D) and directional bundling presented above, we can conclude that FFTEB is the fastest undirected *and* directional method in existence.

The advantage of FFTEB extends beyond the above results. Recent work has shown that *approximate* FFTs can be efficient parallelized on CUDA, yielding speeds one order of magnitude higher than exact FFT's such as CUFFT (Wang, Chandrasekaran, and Chapman, 2016). Since we only use the FFT to compute a density map, whose gradient does not need to be very precise for the iterative advection used by the KDE model (Eqn. 6.5), such techniques are directly applicable to FFTEB.

6.4.2 Parameter Analysis

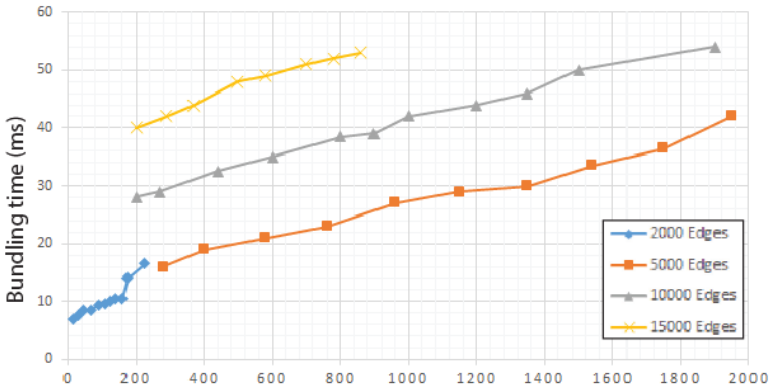
As stated in section 6.2.1, FFTEB's complexity is $O(I(N + R^2 \log R))$, lower than CUBu's complexity of $O(I(N + R^2 h))$. Also, FFTEB's complexity is independent on the kernel size $h \sim R$, while CUBu's complexity is not. Apart from this, both FFTEB and CUBu are linear in the number of bundling iterations I which, as explained, is set to a small fixed value $I = 10$.

We next study how FFTEB's performance depends on its two free parameters – number of sampling points N and image size R. Recall that N depends on the image size R, so the sampling *density*, *i.e.*, ratio N/R, is invariant. As such, we proceed as follows. We assume a fixed sampling density of a few pixels between consecutive sampling points on each edge. Next, we benchmark FFTEB for four graphs having

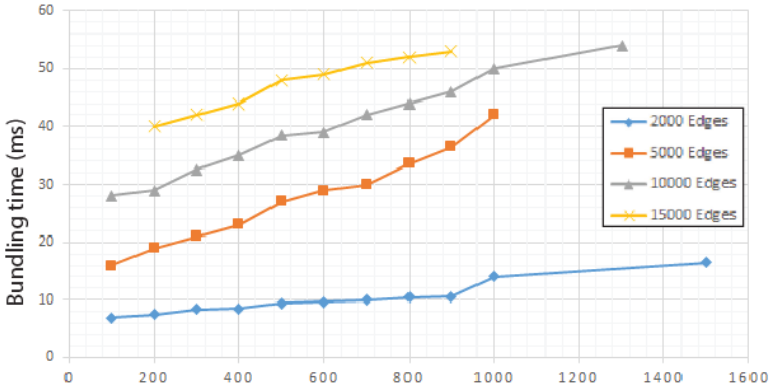
between 2K and 15K edges, bundled at resolutions varying from 100^2 to 1500^2 pixels. For each run, we record the computational time and produced sampling-point count.

Figure 6.12 shows how FFTEB's speed depends on both R and N. As expected, the computational time is linear in N (figure 6.12a). More interestingly, however, is figure 6.12b, which shows an almost linear dependency of time on the size (width, height) R of the image, while, as we have seen, the computational complexity in R is $O(R^2 \log R)$. In contrast, CUBu's time dependency on R is clearly superlinear (see Zwan, Codreanu, and Telea, 2016, figure 12a). This positive result shows that FFTEB scales in practice very well with the image size R.

This can be explained by the very efficient parallelization of FFTEB in CUFFT, the large number of parallel cores (3072) of the used GPU (NVidia 690 GTX) *vs* the amount of data to process, and the fact that we only evaluated performance up to a relatively low resolution ($R = 1600$). Quadratic increase of computation time with image size may actually become visible, on this GPU, only at higher image resolutions. Finally, we see that the slopes of the four graphs are quite similar (figure 6.12b). This tells that FFTEB has a throughput (number of bundled sample points per unit time) which does not strongly depend on the data size. This practically confirms the FFTEB complexity of $O(I(N + R^2 \log R))$, which is linear in the sample point count N.



(A) Edge sampling points N (**x**1000)



(B) Image size (width, height) R (pixels)

FIGURE 6.12 – Scalability of FFTEB as function of number of sample points N and image size R.

6.4.3 Advantages and Limitations

FFTEB solves the scalability and computational speed challenges of earlier bundling methods, for all known bundling variants – undirected (2D and 3D), directional, and general attribute-based. By itself, this is an important results, as it opens the possibility to *efficiently* handle datasets of arbitrary size and number of attributes. In particular, FFTEB strongly alleviates the main scalability problems of KDE-based methods, which, as discussed in Sec. 6.1.1, strongly depend on the image resolution R, kernel-size h, and a graph's sampling points count N. Putting it simply, FFTEB can generate high-resolution images (large R), with strong bundling (large h), for big graphs (large N) faster than all known methods we are aware of.

Additionally, our proposed modulation of bundle width by local edge density emphasizes important bundles much better than shading and/or opacity can. More specifically, when looking at an image, it is easier to quantitatively compare the widths of bundles, especially when these are rendered as shaded tubes as we do (introduced by Telea and Ersoy, 2010, refined in CUBu, Zwan, Codreanu, and Telea, 2016), than quantitatively compare their relative opacities. This is related to the well-known power of encoding of the size (width) and opacity visual variables (Bertin, 1983; MacEachren, 2004).

FFTEB by itself does not solve several challenges related to the *effective* interpretation of edge-bundled drawings. Following certain edge groups end-to-end is hard, especially for large graphs – directional bundling helps here only partially. Designing compatibility functions κ that effectively capture the similarity of multivariate edges highly depends on the specific nature and meaning of edge attributes for the problem at hand. Controlling and/or showing the amount of distortion that bundling produces is an important factor that must be further considered when bundling data with spatial meaning, such as trails. Finally, quantifying the quality of a bundled layout in terms of how effective it is to solve a specific problem, or based on ground truth in terms of its ability to preserve the underlying data structure, is still an open grand challenge in the edge bundling arena. Potential ideas to address this are (1) organizing controlled experiments to test the accuracy and/or speed of performing a certain task using different bundling methods; and (2) measuring the amount of distortion and/or displacement of the bundled edges with respect to the unbundled ones, or to a relevant subset hereof, to quantify how data structure is preserved.

Given all above, we conclude that FFTEB effectively solves the bundling efficiency challenges of current methods and is a good framework to base further research in measuring and improving the effectiveness of edge-bundled drawings of large datasets.

6.5 Summary

We have presented FFTEB, a method to bundle very large attributed graphs. FFTEB's key asset is its scalability, both in size of the input graph (number of edges or edge sampling-points) and in the resolution of the final image. FFTEB exploits the properties of the Fast Fourier Transform, and of a GPU streaming design, to bundle graphs which are much larger than what state-of-the-art methods can handle, and with less time. Additionally, FFTEB generalizes attribute-based graph bundling with no computational penalties, something that aforementioned methods cannot do. We demonstrate our approach by comparing FFTEB with nine state-of-the-art methods and using graphs that have up to *billions* of sample points – an order of magnitude that not current method can handle. Additionally, we show that the high scalability delivered by FFTEB is essential to generate images which can capture fine details of the underlying datasets, which in turn helps their better understanding.

Many possible extensions for FFTEB exist: Using recent results in computation of sparse FFT can accelerate our method one order of magnitude (Wang, Chandrasekaran, and Chapman, 2016), which would make FFTEB cope with most big-data challenges envisable today. Next, FFTEB's efficient attribute-based bundling opens new ways to explore virtually any kinds of edge-compatibility criteria based on any nature and/or number of edge attributes. Finally, the overall generality and scalability of FFTEB makes its extension to interactive 3D bundling a low hanging fruit prospect, with application *e.g.* into medical imaging (Böttger et al., 2014).

Part III

Application to the Study of Alzheimer Disease

Context

In this part of the thesis, we report the usage of edge bundling technique in the Memory project. This project began in September 2014 and involved neuro-scientists, cognitive and visualization experts.

According to a ministerial review of 2004, approximately 860,000 people are affected by Alzheimer's disease in France. This estimation will possibly reach 1.3 million in 2020 and 2.1 million in 2040. This is why the study of Alzheimer's disease has been identified as a major societal challenge. In the three-city 3C cohort study (Antoniak et al., 2003), 3777 elderly people performed a lexical evocation task by saying the maximum number of city names within one minute. They were longitudinally followed during a 12 year period and some of them developed Alzheimer's Disease (AD). The 3C project also contains many other Electronic Health Record (EHR) and has been used to extract relevant information regarding personal health (*e.g.* more than 80 papers since 2002 on various subjects such as nutrition, hypertension and Alzheimer disease, Alperovitch et al., 2002).

The results of the 3C project are an excellent occasion for studying both the effects of aging on the human brain through the cognitive map of each person, with a cross-sectional study, and modification during the progression of Alzheimer's dementia, longitudinal study. To the best of our knowledge, no previous work has investigated this dataset thanks to interactive visualization methods (Card, Mackinlay, and Shneiderman, 1999). In the following chapters, we present our works which fills this gap.

Cognitive Maps

The concept of cognitive map was introduced by Tolman, 1948 in his article "Cognitive maps in rats and men" and echoed by O'keefe and Nadel, 1978 in their book "the hippocampus as a cognitive map" published in 1978. Tolman's suggested that a rat placed in a labyrinth in order to find its food (*e.g.* figure 6.13) builds a mental representation of this labyrinth rather than answering to a flow of internal and external stimuli. This mental representation allows new and complex behavior (Tolman, 1948 – for example changing route when the usual one is blocked).

A cognitive map is then considered, among other things, as a sort of matrix in which the episodes of life can be recorded to be then accessed when needed by mental processes (O'keefe and Nadel, 1978 p 390, chapter 14). What's interesting is that according to the authors, hippocampic lesions distort spatial *imaging* (O'keefe and Nadel, 1978 p 415, chapter 15).

Formally, cognitive mapping is "the capacity of a subject to reorganize spatial information in order to develop cognitive representation of the environment beyond its field of perception". It allows one to mentally co-localizing several points in an organized and coherent configuration. It is an allocentric (or exocentric, depending on the authors) representation of space, i.e. independent from the subject position or point of view. To build a cognitive map, one may need to create a subjective representation of space

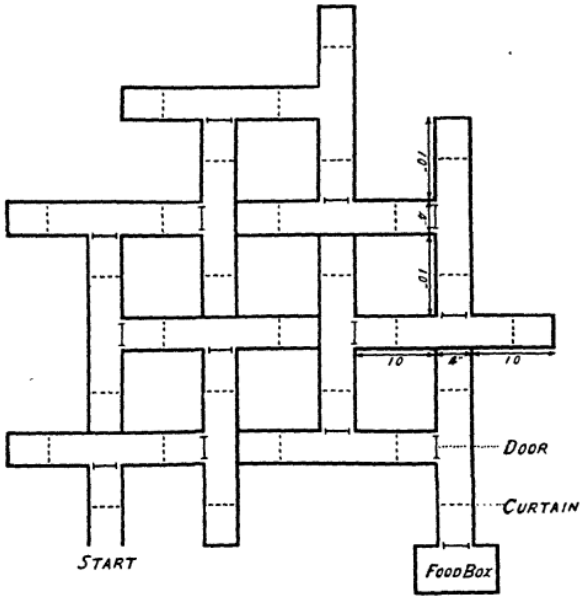


FIGURE 6.13 – Tolman's labyrinth example

with a frame of reference, as well as a processing task. Historically and as opposed to the behaviorist point of view, cognitive map is the ultimate mental representation of space.

The 3C data is a reservoir to visualize and analyze verbal renderings of individual cognitive maps, as well as their evolution in time and during Alzheimer's disease.

3C Electronic Health Record Database

The 3C project is a population cohort study in which 3777 people aged 65 or more are involved. This study aims to evaluate the medical, cognitive and functional levels of those people through their lifetime; the study has been ongoing for the past 22 years. During those years, some of the subjects have developed Alzheimer's dementia as confirmed by clinical tests with psychologists and physicians.

During this study, patients have been subjected to a variety of task such as a verbal fluency exercise. This exercise consists of naming a maximum of cities that they can in one minute. This task was repeated during each check-up in the cohort, between 18 months and 3 years.

The gathered data allows us to understand how subjects apprehend geographical space through city names. It is a particular type of cognitive map based on both the subject living history (autobiographic memory), semantic knowledge (the geographical shape of France and the localization of known French cities), memory capacities, and capacity to elaborate appropriate strategies to collect and organize this knowledge in a restricted time.

The qualitative study of this fluency test gives us the occasion to comprehend cognitive maps of geographical space in a large cohort. A great variety of information

can be analyzed such as: city population, cognitive map's size in terms of geographical perimeter, the center of the cognitive map, the order in which people name the cities.

Approach

With the 3C data, those different aspects of cognitive maps can be studied longitudinally during the aging process of the subjects. Even more interestingly, we can assess the cognitive maps of subjects before and after the onset of AD. Alzheimer's disease affects first the hippocampic regions of the brain in which cognitive map takes a large place. Studying the evolution of those maps through the aging process of a patient with Alzheimer's disease is a great opportunity to improve the scientific community's understanding of the disease.

As the fluency tests were written on paper, the first step of our work was to produce an electronic health database as clean as possible of the tests. This process is detailed in chapter 7. Once the Electronic Health Records (EHR) database was digitized, we were able to visualize the node-link diagram of two populations: the subjects who will develop Alzheimer's disease and the control subjects. Since these graphs quickly became cluttered with numerous edges, we used our bundling technique to simplify the visualization. Hence, we used bundling to find new geospatially-based insight into our data-set. Our analysis and exploration of the fluency tests EHR data-set is detailed in chapter 8.

Parts of this work were conducted in cooperation with: Hélène Amieva (ISPED, Bordeaux France) and her team who did the digitization of the fluency tests and build the database, Ludovic Gardy (M.Sc. in Biology) who worked on the cleaning and the statistical analysis of the data-set, and Emmanuel Barbeau (CNRS director of research at CerCo, Toulouse France) who helped and gave us meaningfull leads regarding the field and research direction of neuro-psychology.

Chapter 7

A Visual Analytics Approach for Digitizing and Cleaning Health Data

In the previous section, we established that performing the digitization and cleaning of the fluency tasks is a great opportunity to study the effect of aging as well as the effect of Alzheimer's disease on the human brain.

In order to analyze and study the evolution of cognitive maps in the 3C cohort study (Alperovitch et al., 2002), we first need to convert each paper-based fluency tests into an Electronic Health Record (EHR). In this chapter, we propose a visual analytics approach to build an electronic database of fluency tests. As notes are taken by hand by a medical practitioner some cities are abbreviated or poorly written (as highlighted in figure 7.1), this digitization process is not an easy task and requires special care.

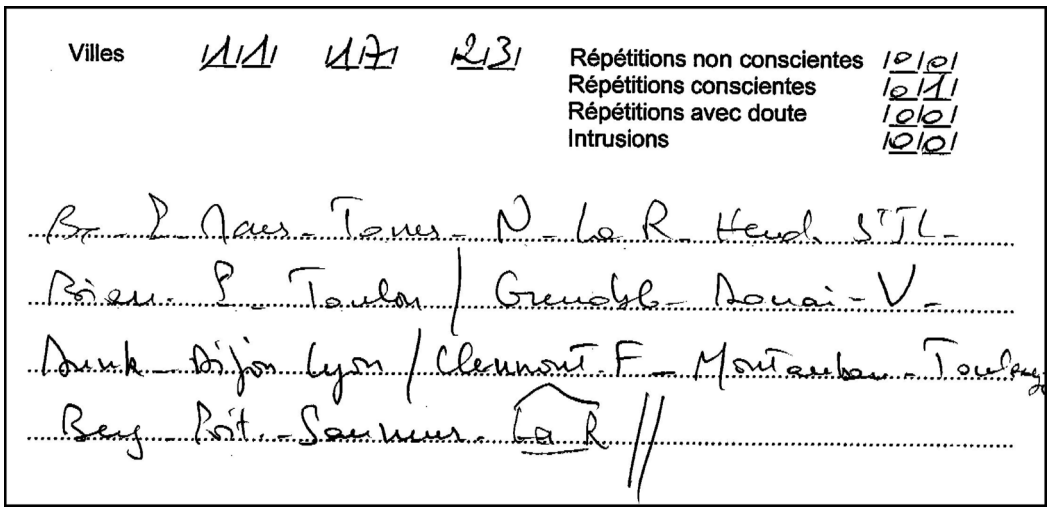


FIGURE 7.1 – Sample of a fluency test result. The slashes represent timestamp separators (15, 30 and 60 seconds). The first word is *Bx* an abbreviation for the city of Bordeaux, France.

The first step toward analyzing this data is the digitization. This process involves first to transform the handwritten information into a list of cities with timestamps. Subsequently, we need to clean the dataset from digitization errors and to remove

ambiguities such as misspelled cities or any other kind of confusion of data. Due to the complexity of these two tasks, no automatic method can be applied. Furthermore, some errors in the list of cities can also contain relevant information regarding the cognitive state of the patient. For example, a patient could cite a word that is not a city but this error is still an insight to be kept and identified in our database (see section 7.4).

We used a User Centered Design (UCD – Schuler and Namioka, 1993) process and developed a set of tools to help medical practitioners during the digitization and cleaning process. This set of tools comprises an auto-completion algorithm, a confidence-based cleaning algorithm and a visualization to help knowledge extraction.

The main contribution presented in this chapter is our interactive visualization system to help practitioners clean the patient records, trace modifications and remove ambiguities between digitization error and actual insights. In the following section, we present existing digitization and cleaning approaches relevant to our study (section 7.2). Then, we list the design requirements to support the digitization and cleaning tasks (sections 7.4 and 7.5). Next, we describe our software features and justify our implementation choices (section 7.6). Finally, we outline the strength of our design with specific digitizing and cleaning scenarios (section 7.7).

7.0 Résumé (Français)
Une Méthode d'Analyse Visuelle pour la Numérisation de Données de Santé

Dans l'introduction de cette thèse (section 1.4.2), nous avons montré que la numérisation et le nettoyage des tests de fluence verbale était une excellente opportunité d'étudier les effets du vieillissement ainsi que les effets de la maladie d'Alzheimer sur le cerveau humain.

Avant d'analyser et d'étudier l'évolution des cartes cognitives dans le cadre de la cohorte 3C (Alperovitch et al., 2002), il est nécessaire de convertir chaque test de fluence verbale sur papier vers des données de santé informatisées (EHR – *Electronic Health Data*). Ainsi, dans ce chapitre, nous proposons une approche orientée analyse visuelle (*Visual Analytics*) permettant la construction d'une base de donnée informatisée de ces tests. Comme les notes ont été prises manuscritement par des médecins au cours du test, certaines villes ont été abrégé ou mal écrites (comme visible en figure 7.2). De ce fait, l'étape de numérisation n'est pas triviale et nécessite un traitement spécifique dans l'objectif de créer une base de donnée la plus précise possible.

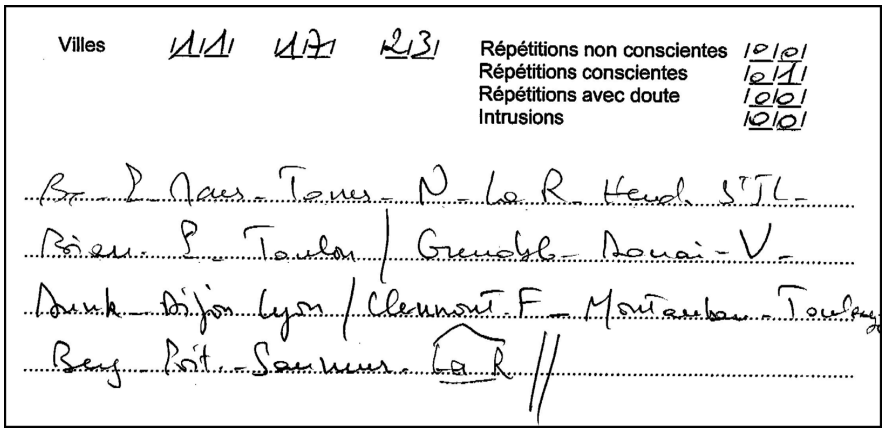


FIGURE 7.2 – Exemple de test de fluence verbale. Les barres obliques séparent les différents intervalles temporels du test (15, 30 et 60 secondes). Le premier mot *Bx* est l'abbréviation de la ville de Bordeaux.

La première étape vers l'analyse de nos données consiste donc à numériser ces dernières. Ce processus nécessite d'abord de transformer les informations manuscrites en une liste de villes avec un intervalle temporel, puis de nettoyer la base de données de potentielles erreurs de numérisation, et enfin de supprimer les ambiguïtés de saisie comme par exemple des villes mal épelées ou d'autres types de confusion dans les données. À cause de la complexité de ces deux tâches, aucune méthode de saisie automatique n'a pu être utilisée. En outre, certaines erreurs dans la liste des villes cité pouvant contenir des informations importantes sur l'état cognitif de notre patient il est nécessaire de les prendre en compte sans les classer comme erreur de numérisation. Pour exemple, un patient pourrait potentiellement citer un mot qui n'est pas une ville mais ce mot reste un indice potentiel à garder et à identifier dans notre base de données (voir section 7.4).

Afin de proposer des outils adaptés aux acteurs de la santé pour leur tâche de numérisation et de correction, nous avons adopté une approche centrée utilisateur (*User Centered Design* – UCD, Schuler et Namioka, 1993). Notre ensemble d'outils comprend entre autres un algorithme d'auto-complétion, un algorithme de nettoyage des données basé sur des indices de confiance et une visualisation permettant d'extraire des informations sur le test de fluence verbale. Parr exemple, notre algorithme d'auto-complétion se fonde sur une base de donnée des villes de France (GeoNames, 2015a) ainsi qu'une base de données des villes mondiales de plus de 15000 habitants (GeoNames, 2015b). De ce fait, nous avons pu améliorer la tâche de saisie de villes en réduisant le nombre de frappes clavier à effectuer ainsi qu'en évitant des erreurs de saisie (détails en section 7.6). De façon similaire, la définition d'indice de confiance des villes numérisée (section 7.4.2) a permis aux utilisateurs de limiter le nettoyage des villes cités aux villes ayant un faible indice de confiance.

La contribution principale de ce chapitre réside dans notre système interactif de visualisation permettant d'aider les médecins à nettoyer, tracer les modifications et enlever les ambigüités entre erreurs de numérisation et informations importantes. Dans les sections suivantes du chapitre, nous présentons les approches existantes pour numériser et nettoyer des bases de données qui se rapprochent de notre étude (section 7.1). Ensuite, nous explicitons les différents besoins liés à nos tâches de numérisation et de nettoyage (sections 7.4 et 7.5). Puis nous décrivons les spécificités de nos outils et justifions nos choix d'implémentation (section 7.6). Enfin, nous explicitons les points forts de notre système à travers des scénarios de numérisation et nettoyage spécifiques (section 7.7). Dans ce résumé du chapitre en français, nous nous limiterons à présenter les résultats de nos étapes de numérisation et de correction.

Résultats

Au cours de l'étape de numérisation, nous avons pu numériser et nettoyer un peu plus de 1500 tests manuscrits, chacun contenant en moyenne une vingtaine de villes citées. Au total, nous avons numériser un peu moins de 45000 citations. Au cours de la création de cette base de données informatisée, chaque test a été numérisé, puis nettoyé et vérifié. Les deux premières étapes nous ont permis de construire la base de données et la corriger. A l'inverse, l'étape de vérification nous a permis d'assurer une estimation correcte des erreurs potentiellement encore présentes dans la base de données. Les résultats des erreurs trouvées et corrigées au cours de chaque étape sont résumés dans le tableau 7.1.

Au total, nous avons informatisé deux populations (*i.e.* déments et non-déments) contenant 394 sujets. Les sujets sont appariés un pour un, c'est à dire que des sujets appariés (*i.e.* un dément et un non-dément) partagent les mêmes valeurs des facteurs de confusion connus de le maladie d'Alzheimer (*cf.* Alperovitch et al., 2002, *i.e.* même âge, genre, niveau académique.). Ainsi, nous avons deux jeux de données contenant chacun 197 patients appariés un pour un avec un taux d'erreur résiduel inférieur à 1% par citation et 5% par test.

TABLE 7.1 – Résultats des erreurs de numérisation et de nettoyage ainsi que les erreurs résiduelles après vérification de la base de données.

Étape	**Type d'Erreur**	**Quantité**	**% (par test)**	**% (par citation)**
Numérisation	Date du test	26	1.4%	NA
	Id. du Patient	7	0.4%	NA
Nettoyage	Mauvaise ville	404	23%	0.9%
	Ville inconnue	4	0.2%	$\leq 0.1\%$
Vérification	Mauvaise ville	21	$\leq 5\%$	$\leq 1\%$

D'après la construction de la cohorte 3C, nous savons que les sujets ont passé leurs tests de fluence verbale au cours de multiples sessions (allant jusqu'à six sessions) qui se sont déroulées sur plusieurs années. Ces sessions sont numérotées de $S1$ à $S6$ et deux sessions sont généralement espacées par deux ou trois années pour un même sujet. Quant au cours d'une session donnée, un patient était déclaré comme dément de la maladie d'Alzheimer alors les médecins arrêtaient de faire passer des sessions au sujet concerné (*i.e.* si un patient était déclaré dément à la session $S2$ alors les seuls tests de fluence de ce patient sont $S0$, $S1$ et $S2$). Comme nous souhaitons étudier l'évolution de la maladie avant l'entrée des patients au stade démentiel de la maladie, nous avons choisi de formaliser notre base de données de la manière suivante : la session Si au cours de laquelle un sujet a été diagnostiqué comme dément est numéroté $T0$. Ensuite, les sessions sont numérotées à rebours telle que $T-2$ (pour $S(i-1)$), $T-4$ et ainsi de suite jusqu'à $T-12$ (si le sujet a passé 6 sessions). Pour les sujets non-déments appariés, nous définissons leur $T0$ comme la session Si (session à laquelle son sujet dément apparié a été déclaré dément). Ainsi, une paire de sujets a toujours le même nombre de tests (*cf.* figure 7.12). Ce changement de paradigme dans la numérotation des tests nous permet de comparer chaque population depuis un point de référence commun où les patients ont officiellement été déclarés atteints de la maladie.

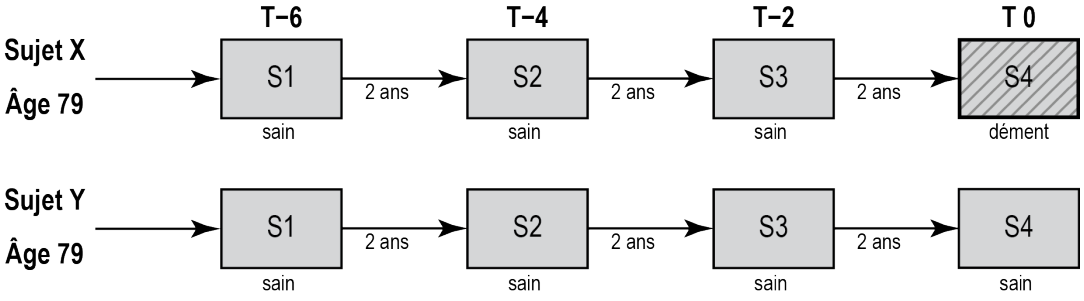


FIGURE 7.3 – Illustration du décalage entre T_i (*i.e.* temps avant démence) et le numéro de session.

Enfin, dans la mesure où la cohorte 3C a porté sur l'étude de multiple démences séniles (*e.g.* vasculaires, fronto-temporelle, Parkinson ou Alzheimer), nous avons dû réduire la taille finale de notre base de données pour inclure uniquement les patients atteints par Alzheimer. Ainsi, notre étude se limite à une base de données finale contenant 127 sujets déments de la maladie d'Alzheimer et 127 sujets non-déments appariés.

En conclusion de ce chapitre, la principale difficulté liée à la construction d'une base de données des tests de fluence verbale réside dans l'indentification et la résolution d'ambiguïté entre les erreurs introduites par nos étapes de numérisation et de correction et les erreurs faites par le patient durant la tâche de fluence verbale. A cause de la grande variabilité de ces erreurs, leur identification et leur résolution ne peut se faire avec une méthode de numérisation automatique. Ainsi, cette numérisation doit être réalisée par un opérateur humain. Cependant, il nous est possible d'aider l'opérateur en lui fournissant plus d'informations pour résoudre les problèmes d'ambigüité. Ainsi, l'implémentation de nos outils a permis aux utilisateurs de répondre plus aisément à ces problèmes (comme par exemple choisir la bonne ville parmi une liste de villes ayant tout le même nom, *e.g.* Chalons). Notre travail peut bien sûr être encore amélioré et étendu à travers plusieurs pistes. Nous pourrions, par exemple, envisager la création de différents indices de confiances (un test d'homonymie ou de toponymie). De plus, nous pensons que les outils présentés dans ce chapitre pourront être utile pour numériser d'autres types de fluences verbale dans le cadre de l'étude de la maladie d'Alzheimer. Enfin, grâce à nos outils, nous avons réussi à rassembler une base de donnée contenant plus de 1500 tests de fluence verbale pour 394 sujets. A travers notre processus, nous avons évité, autant que faire se peut, les erreurs de numérisation et avons corrigé les erreurs restantes au cours d'une étape de nettoyage des données. Enfin, notre étape de validation a montré que le taux d'erreur résiduel dans le jeu de données était inférieur à 5% par test de fluence verbale avec une probabilité d'erreur pour une citation inférieure à 1%. Ainsi, nous avons démontré que notre base de données était suffisamment précise pour être utilisée et analysée.

7.1 Related Work

Our first approach towards the digitization of the fluency tests, laid in evaluating text recognition algorithms. We investigated text recognition algorithms like A2iA by Menasri et al., 2012. However, in our case, the hand writing process associated with the time restriction produced too many abbreviations and poor quality trails which hinder and prohibit the usage of automatic text recognition.

Data cleansing, also called data cleaning or data scrubbing, is the process of correcting data by removing errors. According to the work of Damerau, 1964, the quality of the resulting cleaning is highly dependent on the intervention of a human domain expert to perform accurate cleaning. We believe that it is also the case for the dataset we will have to clean where only an expert can perform the cleaning process.

Many tools focus on the data cleaning processes, such as TimeCleanser (Gschwandtner et al., 2014), which is specialized in time oriented data. Our data-set is also oriented since the city citations are ordered by time. However, the majority of the cleaning process will operate on the city names and not on the temporality of the city citations. In our case, the cleaning process will deal mostly with the geographical aspect of the dataset. And more generally, if different taxonomies of data quality problems exist (*i.e.* Barateiro and Galhardas, 2005, Bose, Mans, and Aalst, 2013, Kim et al., 2003, Müller and Freytag, 2005, Oliveira, Rodrigues, and Henriques, 2005 and Rahm and Do, 2000), none of them proposes a solution in which inconsistent spatial data can be automatically cleaned.

Among the existing data cleaning tool, we note that AJAX (Galhardas et al., 2000) provides a valuable framework including expressive and declarative language specifications to perform the data cleaning process. For instance, it supports mapping, matching, clustering, and merging of data entries. However, in our study, very few errors are generic and only the individual investigation of the records can help to determine if they contain errors or ambiguities.

Bensalem and Kholladi, 2010 and Buscaldi and Rosso, 2008 in their toponym disambiguation base their analysis on the geographical as well as hierarchical relations (e.g. Paris is the capital of France and France is a country of Europe). In their work, the hierarchical relations are based upon the WordNet ontology (Miller, 1995). Although extremely interesting for our case, the WordNet ontology was not precise enough to help us disambiguate toponyms for small villages of France (*e.g.* the WordNet ontology do not know smaller French town nor villages such as *Blagnac*).

7.2 Definitions

To understand the various concepts of our visual analytics approach, we need to define the following important medical and technical terms:

Fluency Test: This is a task part of the Isaac test set (Isaacs and Akhtar, 1972, Isaacs and Kennie, 1973). In the fluency test, patients are asked to cite the largest

possible number of city names within a minute. As the patient lists city names, the practitioner writes the cited cities and adds a marker at 15, 30 and finally 60 second intervals (the end of the test).

Citation: We make a distinction between two types of citations: *Verbal* and *Digitized* citations. Verbal citations are cited by the patient and then noted down on paper. Digitized citation are verbal citations that have been digitized.

Intrusion: A word that is definitely not a city (*e.g.* Banana). This is one of the errors that can be present in verbal and/or digitized citations. More details regarding errors and ambiguities are provided in section 7.4.

Paraphasia: A speech defect characterized by incoherence in the arrangement of a syllable or word. There are different types of paraphrases depending on the type of incoherence (*e.g.* semantic, verbal or phonemic). For example, saying "tiger" instead of "lion" is a semantic paraphasia.

Patient: The subject who performs the fluency test. The patient may be diagnosed as having dementia or not, as a result of medical testing process which are outside of the scope of this chapter.

Practitioner: The person who supervised the fluency tests and annotated the cited city on paper. She is a medical expert.

Digitizer: The person who takes the paper based record of the fluency test and enters the list of city citations into the software. This software contains many features to help the digitizer during the digitization process (these features are detailed in section 7.6).

Cleaner: The person who employs our tool to perform the data cleaning. The cleaner may also be the practitioner, however, it is important that she has specific knowledge about cognitive processes. She must be an expert to determine whether the investigated record contains errors or ambiguities. Finally, she needs to perform possible corrections.

Inspector: The person in charge of checking the correctness of the cleaned data. The inspector is involved in our process after the cleaner. She must be a medical expert regarding cognitive processes.

7.3 A User Centered Design Process

During the design and implementation of our digitization and cleaning tools, we applied a user centered design process (UCD – Schuler and Namioka, 1993). In this design process, we involved users as much as possible in the tool developments.

Different kinds of users were involved in this project: the practitioners who ran the tests, the patients who took the tests, the digitizer who carries out the digitization, the cleaner who cleans the data, the inspector who validates the cleaned data and finally the neuropsychologist who analyzes the results. We recorded interviews with practitioners and digitizers and we had numerous discussions with one specific inspector and one neuropsychologist.

Overall, we spent two days with the practitioners located in Bordeaux, France, and we conducted many discussions with three practitioners, two cleaners and one inspector. We spent up to twenty hours of discussion time on the phone to assess our tools. We conducted one brainstorming session with two HCI experts, one cognitive science expert, a practitioner and a cleaner. We conducted one design walk-through (*i.e.* human computer interface validation) with one cleaner. Finally, we spent three hours installing and detailing our tools for one cleaner.

These interviews and brainstorming sessions helped us to analyze potential errors and to refine the design requirements of our solutions.

7.4 Errors and Uncertainties

Exploring, analyzing and building statistics on a large cohort of patients requires taking into account errors during the digitization and cleaning process. Therefore, we need to quantify, analyze and justify each error that may occur during the processes from the digitization to the analysis phase. This led us to define ways to compute a confidence value for digitized cited city names with regards to possible errors.

7.4.1 Types of Error

Thanks to the UCD process and the numerous discussions with the practitioners, we have identified three different types of errors detailed hereafter in table 7.2.

Table 7.2 – Error types identified in our study.

Type	**Sub-type (with details)**
Patient errors	Semantic paraphasia (lexical substitution)
	Verbal paraphasia (phonetic substitution)
	Intrusion
	Repetition
	Undetermined other errors
Uninterpretable errors	Unreadable city name (on the paper-based fluency test)
	Incomplete test (*e.g.* a whole part of the test is missing)
Digitization errors	Wrong city (input error)
	Unknown city (city not in our database)

In addition to correcting the *digitization* and *uninterpretable* error types, the data cleaning step can also be used to identify patient specific errors. For example, in our particular case, some digitization errors (such as a wrong city) can in fact be a patient error. For example, *Toulouse* and *Toujouse* are two existing French cities. A digitized citation of *Toujouse* can be seen as a digitization error (*i.e.* wrong city), a patient error (*i.e.* a verbal paraphasia) or a correct citation. Here, the disambiguation between the three possible cases can only be achieved with the help of an expert (our *cleaner*). However, we can support the cleaner's work by defining a confidence value of the *potential* correctness of the digitized citation.

7.4.2 Confidence Values

To support the cleaner work, we defined a *confidence* value describing the correctness of a given digitized citation. We define *confidence* as a function f whose codomain is the set $[0;1] \in \mathbb{R}$. The closer it is to 1 and the more confident we are that a given digitized citation is correct. *Confidence* value can be computed using multiple criteria. In our study we have identified three types of confidences: spelling-based, phonetic-based and Euclidean-distance-based. Each confidence is defined as follows:

Spelling-Based

The first confidence value is spelling. In France multiple cities have the same name; for example there are more than 10 different cities whose name start by *Chalons* and are orally referred as *Chalons* without the rest of their proper name. There is a large possibility of typing errors (*e.g.* misspelling). Thus, a digitized citation containing the string *Chalons*, has an higher chance of being erroneous than *Paris*. We define our spelling-based confidence value as the probability of a given digitized citation c_d of having a lot of similar spelling. To compute this probability, we first need to find the number of similar city names from c_d. For this, we use the Damereau-Levenstein distance (Damerau, 1964, Levenshtein, 1966) d, and the number of city with similar spelling is;

$$n_{spelling}(c_d) = Card(\{city | d(c_d, city) \leqslant k\}), \tag{7.1}$$

where k is a integer value defined in $[0; length(c_d)]$. From the number of similar cities, we define the confidence as:

$$f_{spelling}(c_d) = \frac{max(n_{spelling}(c_d), l)}{l}, \tag{7.2}$$

where l is the upper limit of the maximum number of similar cities. Our spelling-based confidence is thus based upon the definition of the parameters k and l. Typically, from our experience, we set $k = 3$ and $l = 10$.

Phonetic-Based

Relatively similar to the spelling-based confidence, we define the phonetic-based confidence as the probability for a given digitized citation to have a lot of phonetically similar city names. In the case of the phonetic-based confidence, city names are map to a unique key via the metaphone algorithm (Philips, 1990 and Philips, 2000). We choose the double metaphone algorithm as it is the more suitable for French language (Philips, 2000). Then we compute the number of phonetically similar city names by comparing the city keys using the Damereau-Levenstein distance. Let χ be the function that transform a city name into the unique token using the double metaphone algorithm, we define the number of phonetically similar city names as,

$$n_{phonetic}(c_d) = Card\left(\{city | d(\chi(c_d), \chi(city)) \leqslant k\}\right), \tag{7.3}$$

where d and k are respectively the Damereau-Levenstein distance and a threshold value in $[0; length(c_d)]$. By extension, our phonetic confidence is

$$f_{phonetic}(c_d) = \frac{max(n_{phonetic}(c_d), l)}{l}. \tag{7.4}$$

In the case of phonetic distance, we set $k = 1$ and $l = 10$. Setting k to 1 means that for a given digitized citation we look for all city names which phonetic pronunciation has no more than 1 different syllable (*e.g. Toulouse* and *Toujouse* differ from only 1 syllable). Setting l to 10 means that we consider a digitized citation to have a confidence null if the city has more than 10 cities with a similar pronunciation.

Euclidean Distance-Based

From our experience, only lexically-based confidences may not be sufficient to detect potential errors. During the task of verbal fluency, people tend to list cities near to one another's following routes itinerary as theorized in the vista space concept (Meilinger, 2008) and revealed during our interviews with practitioners. Considering a given digitized citation c_d, the previous cited city c_{d-1} and the following one c_{d+1}, this means that c_d should be geographically located in an area not too far from c_{d-1} and c_{d+1} but not too close to c_{d-1} nor c_{d+1} as illustrated in figure 7.4.

Let $Center$ be the geographical position between the preceding city c_{d-1} and the following one c_{d+1} of our cited city and c_d, the geographical position of this city. We define our Euclidean-based confidence as:

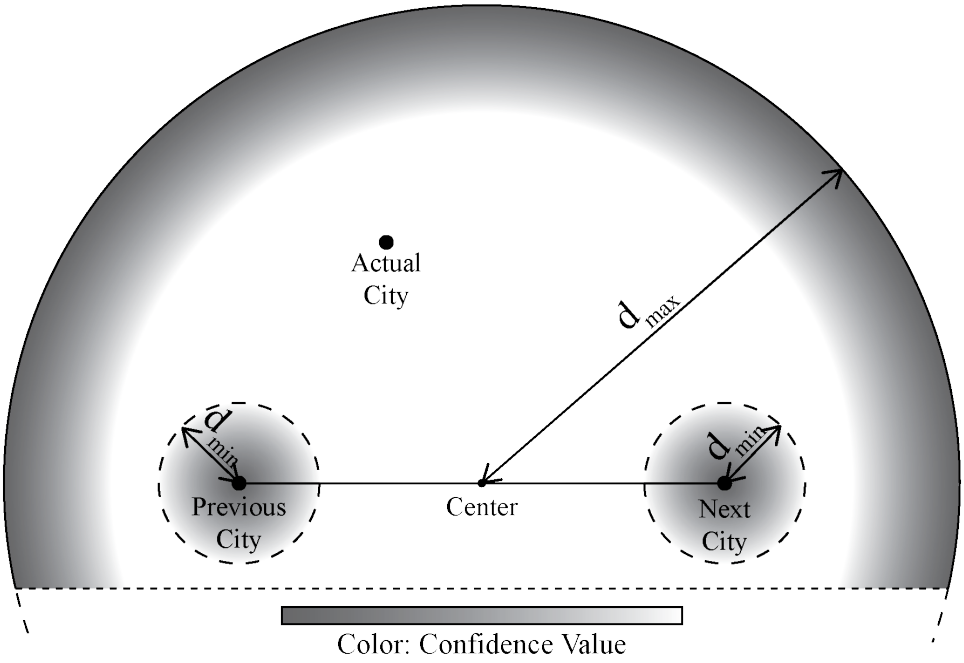


FIGURE 7.4 – Illustration of our Euclidean-based confidence. The associated confidence value is encoded in a grey-scale gradient.

$$f_{Euclidian}(c_d) = \begin{cases} \dfrac{\delta(c_d, c_{d-1})}{\delta_{min}} & \text{if } \delta(c_d, c_{d-1}) \leq \delta_{min}, \\ \dfrac{\delta(c_d, c_{d+1})}{\delta_{min}} & \text{if } \delta(c_d, c_{d+1}) \leq \delta_{min}, \\ max\left(\dfrac{\delta_{max} - \delta(c_d, Center)}{\delta_{max} - \delta(c_{d-1}, c_{d+1})}, 0\right) & \text{if } \delta(c_d, Center) \geq \delta(c_{d-1}, c_{d+1}), \\ 1 & \text{else ,} \end{cases} \tag{7.5}$$

where δ_{min} and δ_{max} are threshold values set as a factor of $\delta(c_{d-1}, c_{d+1})$. From our experience, we use $\delta_{min} = 0.2 * \delta(c_{d-1}, c_{d+1})$ and $\delta_{max} = 1.2 * \delta(c_{d-1}, c_{d+1})$. Setting δ_{min} and δ_{max} to a factor of the distance $\delta(c_{d-1}, c_{d+1})$ means that we consider the distance of the digitized citation over the local space of the previous and following cities. For example, if c_{d-1} and c_{d+1} are two cities distant from 50 kilometers, we expect the digitized citation c_d to be in a radius of 60 kilometers from the center position of c_{d-1} and c_{d+1}. Moreover, our Euclidean-based confidence has the advantage of showing the discontinuities in the citation strategies of a subject. For example, a subject citing city from a country (*i.e.* a country-level vista space) then capitals of various country (*i.e.* a continent-level vista space) will be clearly identified by this metric.

7.5 Design Requirements

This section presents the design requirements to achieve the digitization and the cleaning tasks. First, we analyze the digitization requirements and then the cleaning one's. Requirements associated with the digitization process are abbreviated ***Rd*** and then those with the cleaning process ***Rc***. All the requirements associated with the two tasks were collected during our UCD process

7.5.1 Digitization Process Requirements

During the digitization process, the digitizer needs to retrieve the list of digitized cited cities. We summarize with the following digitization requirements:

Rd1: Multiple solutions were available to us such as character recognition, verbal recognition or typing. Due to the poor writing quality of the tests (figure 7.1), we were not able to use the character recognition. Concerning verbal recognition, as the tests are in French, we needed a French-based grammar recognition. However, we were unable to find an efficient and open source French-based grammar. Thus, finally we chose to digitize the fluency test by manual typing.

Rd2: Manual typing does not solve by itself the need to digitize a correct data-set. Some cities might have the same name (e.g. London, UK and London, CA). To avoid such ambiguities, the user must be able to digitize a city to a unique name.

Rd3: The digitizer must specify the separators between cities listed during a period of 0 and 15 seconds, 15 and 30 seconds and 30 and 60 seconds

Rd4: The digitizer must be able to specify an intrusion.

7.5.2 Cleaning Process Requirements

Similarly, we summarize the cleaning requirements as follows:

Rc1: Each digitized and cleaned fluency test must be validated by an inspector. To enhance his validation work, we want him to be able to look through any modification that the cleaner might have made and to validate or reject each test. Thus, we need to provide him with the list of each modification as well as allowing him to display the original digitized dataset.

Rc2: The cleaner must be able to correct a digitized city and specify the error type. Thus, he must have interaction means to achieve his task. In the continuity of our digitization process, the cleaner must be able to solve any city-name ambiguity problem (*i.e.* the system must allow the addressing of a unique city name).

Rc3: However, allowing him to fill-in a unique city name, means using a database of existing cities. If a city is missing from such a database, the cleaner need to be able to add one directly.

Rc4: Finally, as some errors are not known in advance, the system must give the cleaner the opportunity to find new errors. Thus, he must be able to explore the data-set in order to study prospective analysis methods.

7.5.3 Fluency Test Dataset

The fluency test data-set records the cities listed by a patient during a test (*e.g.* "Paris, Bordeaux, Toulouse, Lyon…"). This list of cited cities is separated into three sub-lists which consists of the cited cities during three sequential intervals (see table 7.3). Different tests are linked through the patient's Id.

TABLE 7.3 – Names and respective semantic of fluency test's attributes.

Field name	**details**
TestId	The unique Id of the test
PatientId	The unique Id of a patient
BirthDate	The birth-date of a patient
TestDate	The date of the test
City15	The cited cities between 0 and 15 seconds
City30	The cited cities between 15 and 30 seconds
City60	The cited cities between 30 and 60 seconds

To digitize a city, we used an open source city database composed of all the world city have more than 15000 inhabitants (GeoNames, 2015b) and all the existing French towns (GeoNames, 2015a). In this city database, each city is represented by its true-name (*i.e.* with accent or other language-specific characters), an ASCII-name (*i.e.* the city name without any accent or language-specific characters), the city's population and its geographical position (*i.e.* latitude and longitude).

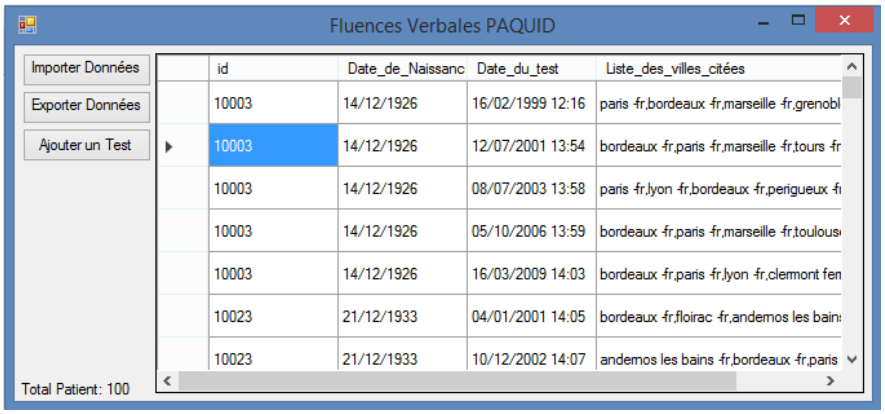
7.6 Tools

From our analysis of the basics requirements for our tasks, we developed two tools. In this section, we detail the basic features of each tools and how they relate to our defined requirements.

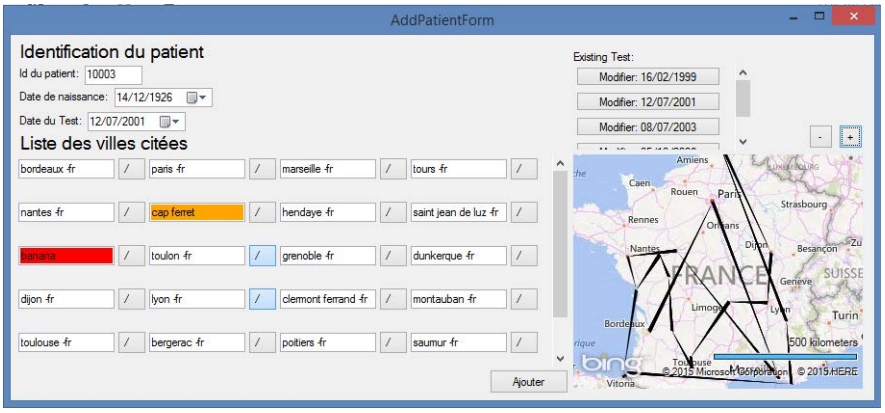
7.6.1 Digitization Tool

The digitization tool has two panels. The main panel shows the list of the fluency test records with values (figure 7.5a). The digitizer can import and export the digitized data at any time via two buttons.

To add test results, the digitizer uses a form and manually enters the names [Rd1] and fills in the data of the fluency test (see table 7.3). To allow the digitizer to bind a city name with a unique city, we have implemented an auto-completion algorithm based on our city database (section 7.5.3) [Rd2]. When a town is missing, the digitizer can still override the auto-completion. In that case, the digitizer is warned with an orange feedback. Moreover, he can specify an intrusion with a right click on the text box [Rd4].



(A) The list of digitized fluency tests.



(B) The input form to add a fluency test. Each city is typed in combo-boxes. Unknown cities are orange, and intrusions words red.

FIGURE 7.5 – The digitization tool.

While entering the listed cities, the digitizer can see the geographical polyline formed by the fluency test (figure 7.5b). This tapered polyline shows the location of each city (known in the database) and the direction of the polyline (using the results of experimentations by Holten et al., 2011). With this visualization, he is able to see whether a city seems erroneous or not (*i.e.* a city too far from the previously listed ones).

The timestamp of the fluency test (each 15, 30 and 60 seconds) is displayed in the interface with slash buttons allowing the digitizer to put the three slashes initially in the paper-based test [Rd3]. Finally, by typing an existing test ID, the digitizer is able to load an existing test and modify it using the top-right box.

7.6.2 Cleaning Tool

In the subsection, we detail the component used in our cleaning tool (figure 7.6). Our cleaning tool is composed of three main panels: A histoSelector panel, the citation panel and the detail panel.

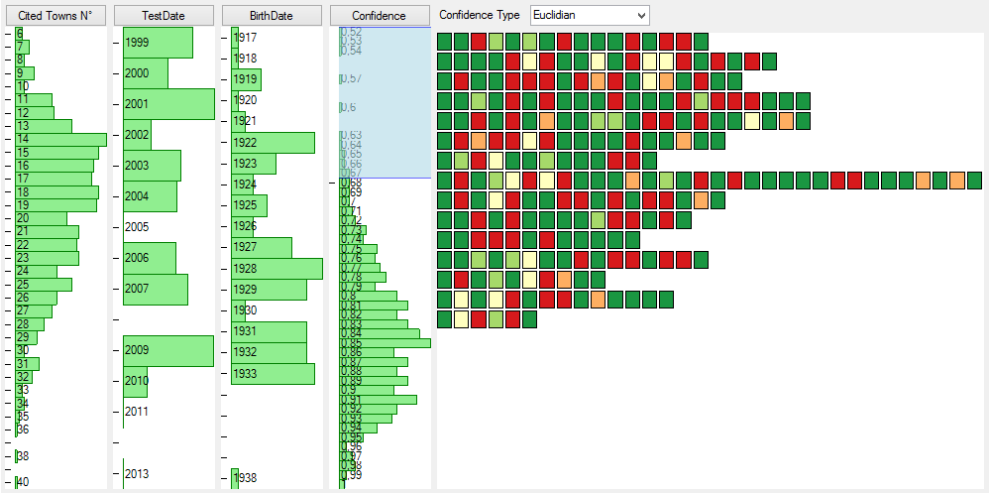


FIGURE 7.6 – The cleaning tool. On the left, the HistoSelector panel, and on the right the Citation Panel. The cleaner can choose the confidence criteria with a combo-box above the citation panel.

The HistoSelector Component

To filter and explore the dataset during the cleaning process [Rc4], we display histograms showing the distribution of values of each dataset attribute (figure 7.7). The cleaner can select the dataset attribute to be displayed by right clicking on the histogram name.

To find new insights or errors in the dataset, we have chosen a selection paradigm based on a brushing interaction (Li and North, 2003). Users draw a bounding box (a one dimension brushing technique) and can modified it by sliding the bounds of the selection. Multiple boxes can be specified, and they can be removed with a brush stroke with the right button of the mouse pressed (i.e. the erase mode).

We maintained the classical reading pattern (left to right) in such a way that the selections or modifications of one histograms directly modify its neighbor's values. Nesting multiple histogram allows the user to refine a query depending on multiple attributes of the dataset.

Thus, the fluency tests selected through the histograms are displayed in the citation panel.

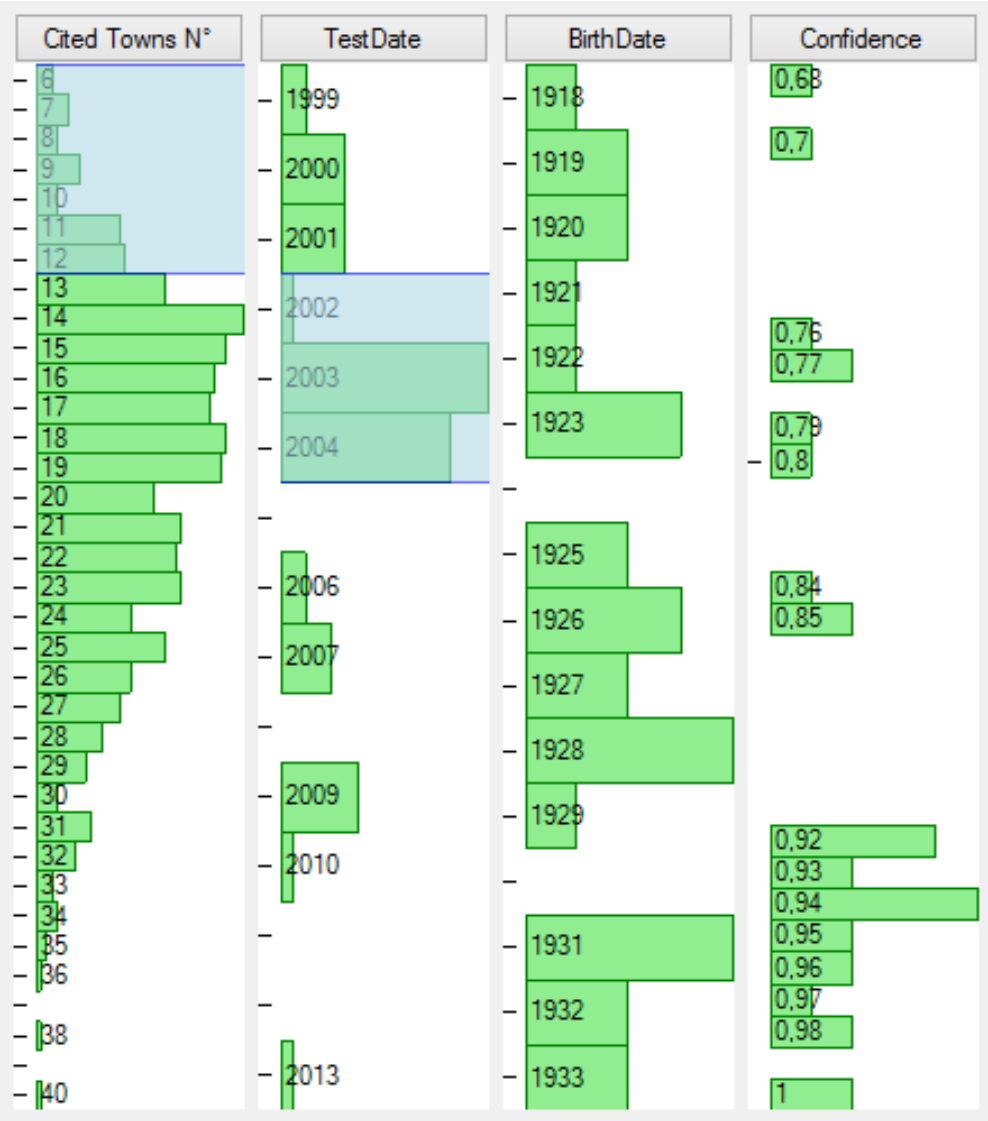


FIGURE 7.7 – Histogram panel. By drawing boxes on the histograms, the cleaner select all patients who cited a low number of towns during the 2002-2004 tests. A selection in one histogram updates all histograms to its right.

Citation Panel

The citation panel is the representation of each fluency test in our cleaning tool (figure 7.8) [Rc1].

One row corresponds to one fluency test. The row is composed of coloured squares which correspond to each cited city with its confidence value. Two parameters are displayed in the squares.

First, the saturation of each square is bound to the state of validation of the city. If the city has been validated then it has a low saturation value. On the other hand, if it is not validated, the saturation is at its full value. As saturation is a dissociative visual parameter (Bertin, 1981), encoding the validation state of a city through saturation allows the visual emergence of the not validated cities.

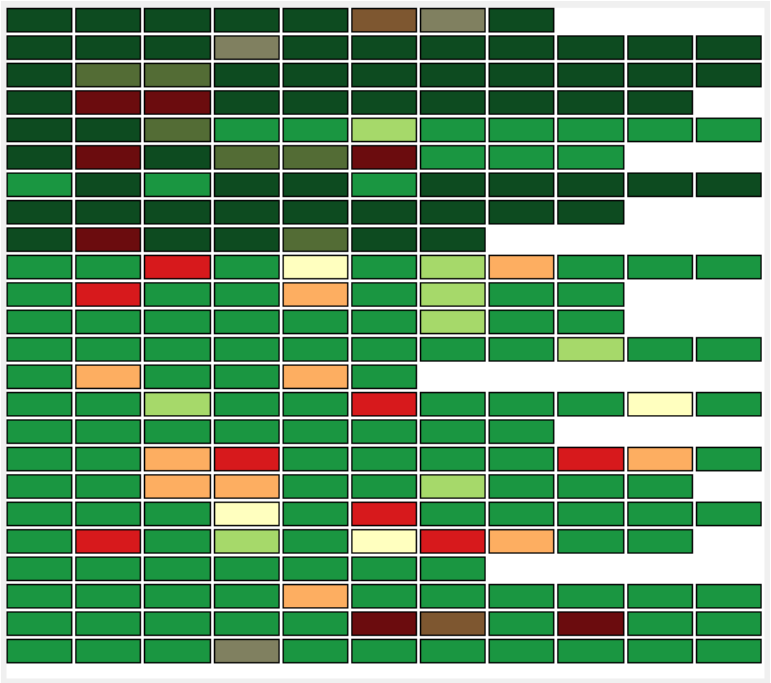


FIGURE 7.8 – Citation panel. Displays the confidence of each digitized city encoded in five levels of color (red to green). A validated city is de-saturated (top of the figure).

Secondly, the color of each square is bound to the confidence value of the city. The associativity of the color improves pre-attentive selection (Bertin, 1981) allowing users to instantly visualize a group of low confidence cities that need to be reviewed. Our color range is defined by five levels of confidence with a color generator (Brewer, Harrower, and University, 2015).

The colors are green for confidence values over 80%, then lightgreen (60-80%), yellow (40-60%), orange (20-40%), and red for confidence values lower than 20%.

Finally, clicking on a colored square opens up the detail panel.

Detail Panel

The detail panel allows the user to correct a potential error in a digitized city [Rc2]. This panel is composed of four main parts (see figure 7.9).

The first part, at the top of the form, displays the values associated with the city (PatientId, BirthDate, TestDate, Confidence value and the actual digitized cited city). In this first part, the user is able to correct the city name (via the Correct City textbox) and fill-in the error type (Auriacombe et al., 2010). Each text-boxes has personalized auto-completion algorithm. The cleaner is also able to add a city to the database via the "City not in DB" button [Rc3].

The second part displays the list of cities lexically similar to the cited one.

The third part shows the cited cities preceding and following the selected city and displays the distance between them. This geographical distance is enhanced by displaying the polyline formed by the fluency test associated with the selected city centered on

the selected one. With this visualization, the user is able to geographically place and check the location of the city and to compare it with the following and preceding one as well as with the whole fluency test.
The last part of the detail panel (bottom of figure 7.6) is an image display of the original paper-based test. This allows the cleaner to check if the city was really written or not, especially in the case of a city with a low confidence value but in which the digitized city was indeed cited by the patient (refer to figure 7.10) for an example).

Once the correction of the city has been performed, the user can validate or cancel his changes.

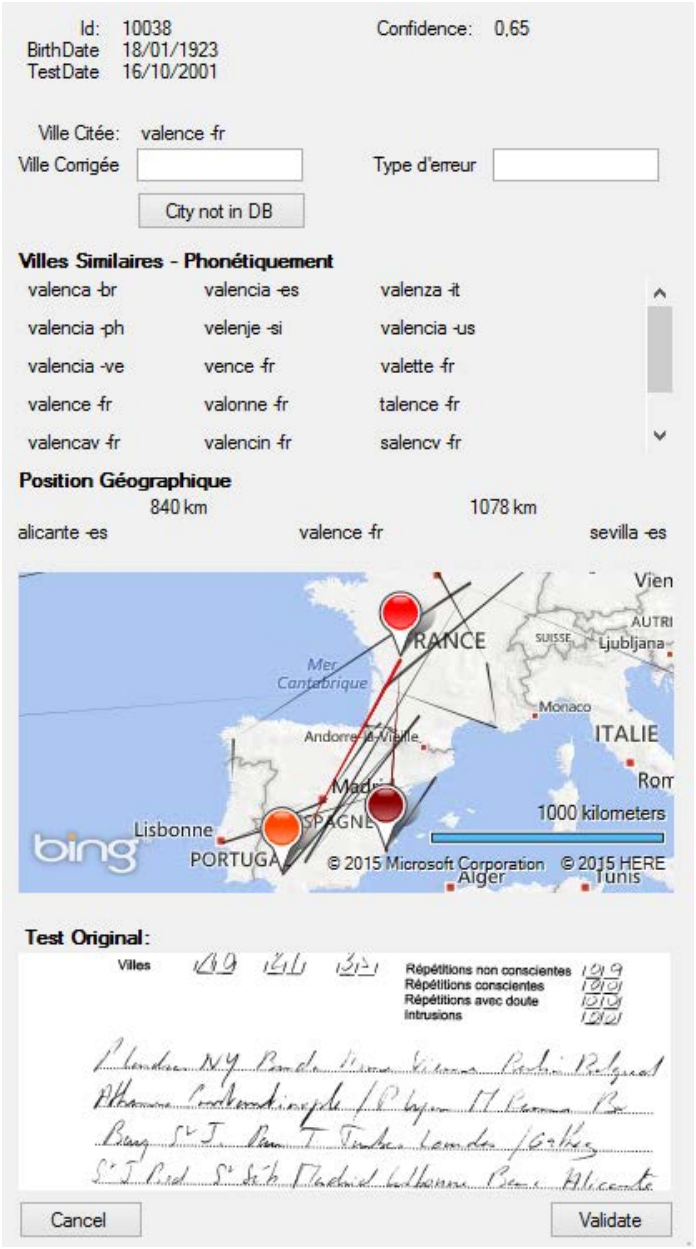


FIGURE 7.9 – Detail panel. On the top are the test information, then the phonetically similar cities and the geographical position within the fluency test. Finally, we display the original fluency test in the bottom of the panel.

7.7 Scenario

This section describes four scenarios of use. The first two describe a true-positive and a false-positive detection. The third one shows how the cleaner can explore the data-set and find unrevealed erroneous data. Finally the last scenario illustrates the validation process performed by one inspector. The typical meta-scenario is as follows:

While a patient V executes the task of verbal, a practitioner W writes down each cited city on a paper. At 15", 30" and 60" the practitioner writes down a slash to timestamp the cited cities. After the test, a digitizer X manually types each cited city and marks slashes with our digitization tool as explained in section 7.6.1. Once the data is loaded into our numeric database, a cleaner Y uses our cleaning tool to correct the database. The cleaning tool help the cleaner by showing the most error-prone cited cities according to the selected type of confidence. The cleaner then has to check whether the detection of an erroneous value by the system was justified (section 7.7.1) or not (section 7.7.2). Finally, an inspector Z checks each correction made by the cleaner in order to validate or invalidate the changes made to the database. The tracking and validation of these modifications are detailed in section 7.7.4 track modifications use case.

7.7.1 True-Positive Use-Case

The visualization shown in figure 7.6 displays the digitized part of our data-set. The cleaner Y wants to validate each digitized fluency test. To do so, Y follows a procedure composed of three steps.

He initially decides to correct the tests with a low spelling-based confidence value. To do so, the user selects the lower values in the confidence histogram by brushing them (cursor on figure 7.6). By selecting those values, only the tests with a low confidence value are displayed in the citation panel. The selected fluency tests are ordered by decreasing level of confidence in the citation panel.

For the second step, Y validates the first row of fluency tests. He starts by clicking on the first red rectangle in the fluency test with the lowest confidence (first row of the Citation Panel). This action opens the detail panel to correct this test citation (figure 7.6).

In the detail panel, the cleaner sees that the digitized city name is Valence. This city is located near Lyon in France. However, the user sees that the preceding city is Alicante and the following one is Sevilla, two Spanish cities. The distance between those cities is therefore too large. Moreover, in the similar cities (cities at a small distance from the original one), he sees a Spanish city named Valencia. Thanks to these two insights, the cleaner can infer that the real cited city is not Valence in France but Valencia in Spain. Y enters the corrected city name in the *Corrected City* textbox. Then, specifies the type of error, in this case a digitization error.

This example illustrates how the cleaner can correct the tests that are likely to be more erroneous according to our spelling-based confidence algorithm. Moreover, it

shows how one can help the cleaner to solve a decision problem and gain more insights to our database by the types of errors.

7.7.2 False Positive Use-Case

In a similar manner to the previous use-case, the cleaner Y can use distance-based confidence to enhance his task of data cleaning. In this use-case, he wants to validate each digitized fluency test. Y follows the same steps as in the spelling distance scenario.

By selecting the right criteria, he is able to display the tests to be cleaned, thus allowing him to focus on those tests. Then, the cleaner selects the first red square showing a very low confidence value on a citation.

In the detail panel, he sees the information concerning the citation (shown in figure 7.10). The digitized citation shows the city of Algiers in Algeria. The previous cited city is Pau and the following is Bougue. Thus, the Euclidian distance is relatively high.

FIGURE 7.10 – Detail panel showing a false positive case. Although the cited city (Algiers) is far from its neighbors (Pau and Bougue), we see on the picture of the original test that the patient did cite it (in the orange box).

To confirm the inaccuracy of the digitized citation, the cleaner takes a look at the original citation at the bottom of the detail panel in figure 7.10. However, Y sees that the city that was written down on paper during the test is Algiers. He concludes the low confidence level was a false-positive one. Finally, he validates the correctness of the digital citation by clicking the *validation* button.

Since the confidence based criteria cannot always be true, this use-case highlights the way a cleaner can detect and resolve a false-positive result thanks to the confidence algorithm.

7.7.3 Detect Outliner Subsets

In this use-case the cleaner Y wants to detect a subset of outliners in the data to be cleaned. In this particular use-case, he wants to investigate the relationship between the length of the fluency tests and the year of the test.

Y initially decides to bind a first histogram to the fluency test length (the number of cited cities). By clicking on the tag name of the histoSelector component, he is able to change its binding. Then, he repeats the operation on the following histoSelector component (directly on the right of the previous one) binding it to the year of the test. With this configuration the cleaner is able to visualize the distribution of the test years depending on the selected fluency test lengths (figure 7.7).

Then he selects the fluency test with a low number of listed cities by brushing the histoSelector component. During the brushing sequences, the cleaner can directly see the repartition of test dates. Y discovers that the majority of tests with a low number of listed cities were performed in 2003-2004. Moreover, with the citation panel, the cleaner finds that all these low numbered tests are missing the 30" to 60" second parts. Thus, the cleaner can conclude the presence of an error in the test protocol during the tests in the period from 2003 to 2004.

This use-case illustrates how the cleaner can detect outliner subsets through exploration of data properties. Exploring and interacting with the data-set helped him find unpredicted errors.

7.7.4 Track Modifications

We detail a scenario where an inspector Z wants to validate corrections made by the cleaner Y during the cleaning process.

Once the cleaner has finished the cleaning process, the inspector can validate or invalidate these modifications. To achieve this objective, he starts binding the first histoSelector to the modified attribute and the second to the validated attributes. Then by selecting the modified but invalidated fluency tests (via the two configured histograms) Z displays the list of modifications requiring validation in the citation panel. Within these tests, the inspector can explore and click on each ones. He chooses to click on the first test and check the digitized cited city. Thus, Z opens the detail panel. In the detail panel he sees that the corrected city name is Damascus. By clicking on the button next to the versioning number (left arrow $<$ in figure 7.11), he sees that the initial value of the digitized cited city was a city unknown in the database named *Damas* (the French name for Damascus). Via the original test view in the bottom of the detail panel, Z checks that Damascus is indeed written in the original test. However the expert does not read *Damas* but *Douai* a French city near the preceding

and following cities. Thus, he corrects the cited city a second time to the known city of Douai. Finally, he corrects and validates his change with the *Validate* button.

Through this use-case, we see how the validation of the inspector can spot errors made during the cleaning process and double check the corrections. This enables the correctness of the clean dataset to be improved.

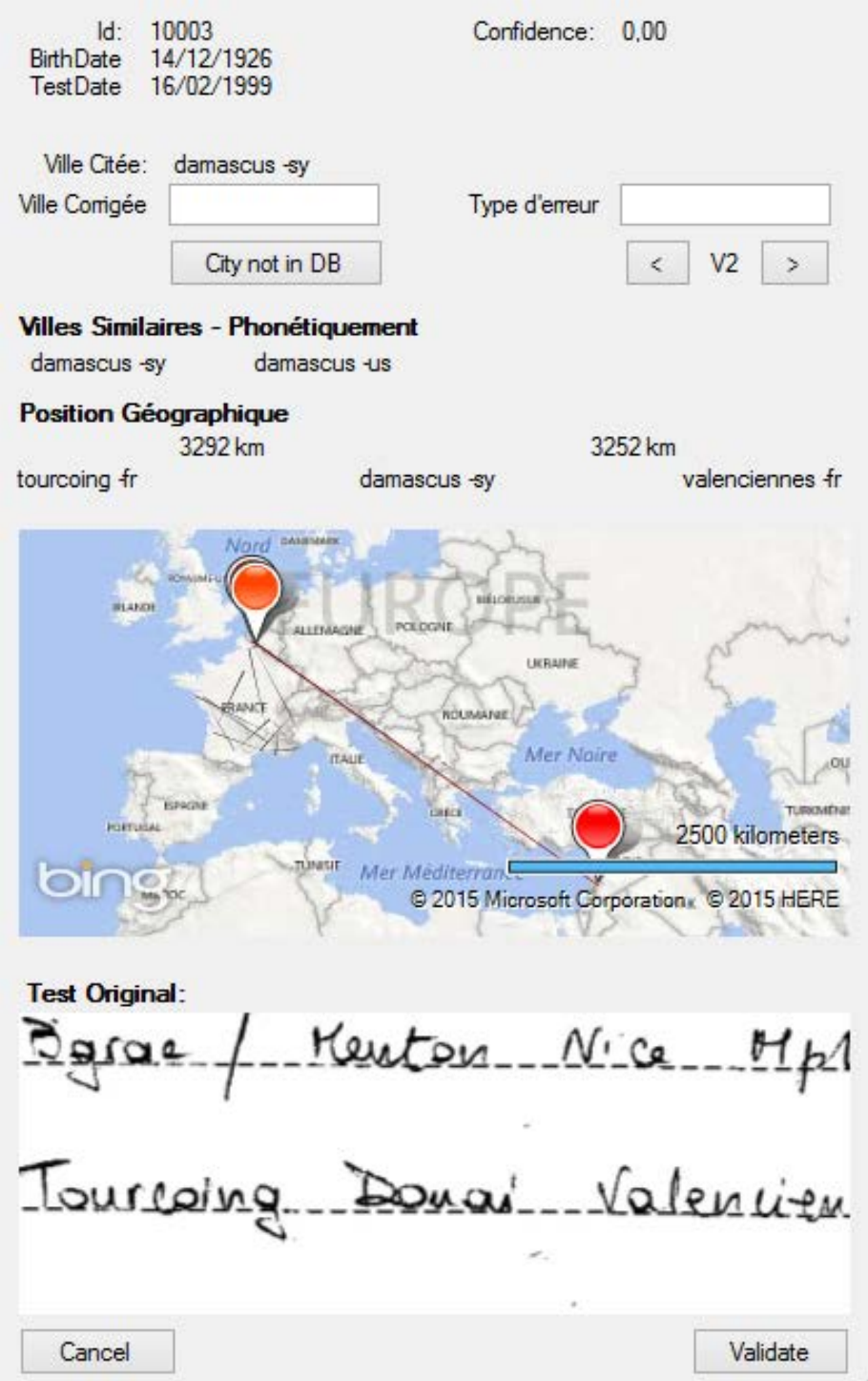


FIGURE 7.11 – Validation of a cleaned test. In this case, the cleaned city is Damascus but the original cited city was Douai. The expert can invalidate the modification and make a new correction to the data.

7.8 Results

In this section, we present the results of our digitization and cleaning process. During the process, we kept track of each modification and errors at each steps and gathered users feedback.

7.8.1 Digitization and Cleaning Results

During our digitization process, we managed to digitize and clean more than 1500 paper-based test with an average of 20 cited cities per test. Overall, we digitized a little less than $45k$ citations. During the construction of our database, each of the tests was digitized, then cleaned and finally inspected. The first two steps allowed us to build a database and correct errors. Conversely, the inspection step, insured us a correct estimation of potentially remaining errors in the database. In the following section, we detail the results of the digitization and cleaning process that led us to build the final database. The overall results of our database is summed up in table 7.4.

Digitization Results

During the digitization process, all the digitizers involved managed to digitize 1745 tests for 394 different subjects. After the digitization, we noted some errors in the input of the patient Id or the date of the test. We did not expect those types of errors in our design. Fortunately, we were able to verify and correct these errors thanks to our collaborators in Bordeaux who had already digitized the correct Ids and test-date for each subject. In total, after the digitization process, we had to correct: 7 wrong *PatientId* representing an error rate of 0.4% per test, and 26 wrong *TestDate* representing an error rate of 1.4% per test. Overall, digitizers enjoyed using our tool. Especially the auto-completion algorithm that helped them to input a city name much faster and with less errors. Moreover, if we did not expect *PatientId* and *TestId* digitization errors, we see that the error rate is correct and we have a rate slightly below 2%.

Cleaning Results

The cleaning step was done by two different users. Overall, the users also enjoyed the two main panels of the cleaning tool (histoSelector and citation panel). The confidence value was thought to be very useful to avoid checking unnecessary citations. However, we noted that the software did not give enough feedback in regards to the loading and saving process of the tool. Especially, in some cases users tried to overwrite the database file on the file containing the modifications, leading to some loss on the modification history. Fortunately, we had a backup modification file that prevented the loss of the complete history.

In regards to errors found in the data-set, from the $45k$ total citations to clean, cleaners found only 404 *wrong cities* and 4 *unknown cities* (see details of errors in

section 7.4). This yield an error rate of 1% for all the database (0.9% *wrong cities* and 0.1% *unknown cities*).

7.8.2 Validation Results

In the validation step, one inspector checked an entire subset of the database to find any residual errors. During this verification, the inspector checked all existing citation without using the confidence indication as opposed to the cleaners. To allow the inspector to complete this task, we used a random subset of 200 subjects (100 dement and 100 non-dement). Overall, the inspector found $10+11$ errors in our random subset. From this, we can infer that our final database yields a potential error rate of less than 5% per subject test and less than 1% per citation.

TABLE 7.4 – Results of digitization, cleaning and validation errors.

Step	**Error**	**Number**	**% (per test)**	**% (per citation)**
Digitization	TestDate	26	1.4%	NA
	PatientId	7	0.4%	NA
Cleaning	Wrong city	404	23%	0.9%
	Unknown city	4	0.2%	$\leq 0.1\%$
Validation	Wrong city	21	$\leq 5\%$	$\leq 1\%$

7.8.3 Final Database

Overall, we have digitized two populations (*i.e.* dement and non-dement) for a total number of 394 tests. Subjects are paired one to one, meaning paired subject (*i.e.* dement and non-dement) share the same values of known Alhzeimer's dependent variables (Alperovitch et al., 2002; *i.e.* age, gender, academic levels...). Thus, we have two databases of 197 subjects paired one to one.

From the 3C study, we know that subjects undertook the fluency tests during various sessions (reaching up to six sessions) over the years. These sessions are numbered from $S1$ to $S6$ and two sessions are generally spaced by two or three years. When at a given session, the patient was declared as dement, clinicians stopped recording their sessions (*i.e.* if a patient was declared as dement at $S2$ we only have the paper-based tests $S0$, $S1$ and $S2$). As we wish to study the evolution of the disease before the dementia, we choose to formalize the database as follows: the sessions Si at which a subject was declared dement is numbered as $T0$. Then we go back in time and label each of the previous sessions as $T-2$ (for $S(i-1)$), $T-4$ and so on until $T-12$ (if the subject had passed 6 sessions). For their paired non-dement subject, we define $T0$ as their session Si (session in which their paired dement subject was declared dement). As such, a pair of subjects have the same number of tests.

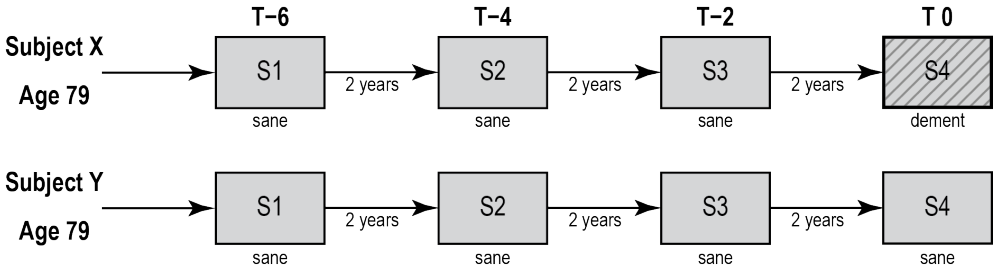


FIGURE 7.12 – Illustration of the shift between T_i (*i.e.* time before dementia) and the session number.

For example in figure 7.12, a subject X developed a dementia during the sixth session and is put in the dement group. A subject Y with the same age, gender and academic level as X follows the same tests process without developing any kind of dementia. Y will be paired with X from which we will define its $T0$ as its paired subject $T0$'s (here X). This shift in the numbering of the tests allows us to compare each population from a point where all dement subject are officially diagnosed with a form of dementia, thus giving us a reference point to start our analysis.

Finally, our database of dement subject contains various type of dementia such as vascular, frontotemporal, Parkinson's or Alzheimer's dementia. In our study, we wish to focus our analysis on Alzheimer's dementia, thus we selected only subjects suffering from Alzheimer's disease and their paired non-dement subject. This yields a final database of 127 dement subjects and 127 paired non-dement subjects.

7.9 Summary

The study of Alzheimer's disease is a unique opportunity to work on an unexploited handwritten data-set. In the digitization and cleaning process the main difficulty is to reduce ambiguities between errors induced by our own cleaning and digitization process and errors made by the patient during the verbal fluency test. Processing and identifying these errors cannot be achieved with an automatic algorithm due to its variability. Thus it must be carried out by users.

To solve this problem we developed two tools that help users during the digitization as well as the cleaning steps. As an example, these tools have helped the people in charge of the digitization process by allowing them to directly input known city names into an existing database. In the cleaning process, our tool helps the user to solve the ambiguity problems inherent to city names (having multiple cities with the same name) as well as solving problems from our digitization process (missing city names).

Our work can be further improved and extended in many aspects. For instance, other confidence tests could be investigated: the homonym test or the toponym test are good candidates to extend the available tools to detect issues in our database. We believe that the tools described in this paper can be useful to provide cleaner data and facilitate the study of Alzheimer's disease progression.

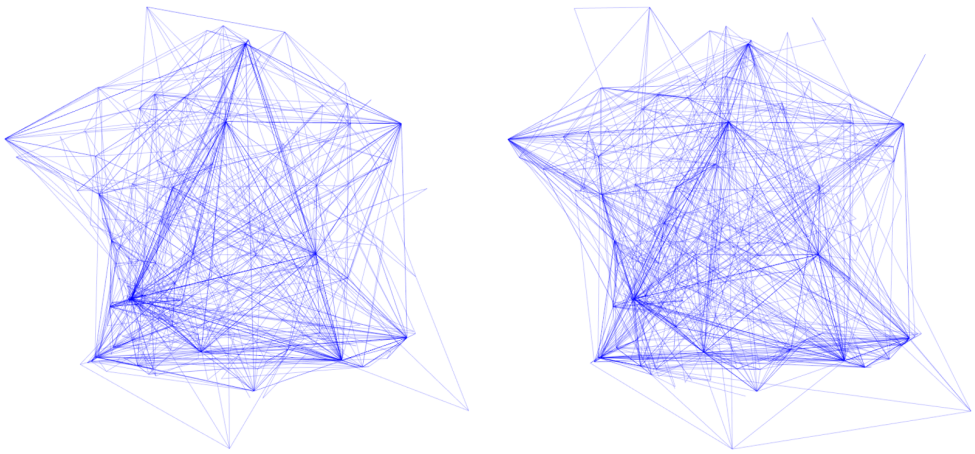
Finally, we believe to have managed to gather a correct database of more than 1500 fluency tests for 394 subjects. Through our process, we avoided as much as possible digitization errors and corrected the remaining ones during the cleaning process. From our validation step, we found a remaining error rate of less than 5% per fluency test with a probability of a citation being erroneous less than 1%. Thus, we believe the cleaned database to be sufficiently correctly for further observations and analysis. In the following chapter, we show advance tools to extract insightful information from our cleaned dataset.

Chapter 8

Bundling Cognitive Maps

In this chapter we illustrate our exploration and analysis process of our cleaned data-set built in chapter 7. Centered around the clutter reduction capabilities of our bundling technique for node-link diagrams, we show how bundling can efficiently be used as a visualization and exploration tool for neuro-psychologist in our study of Alzheimer's disease. Specifically, we show how bundling allowed us to formulate new hypothesis on Alzheimer's disease.

Here, we aimed towards the visual analysis of trail-sets (as defined in chapter 2) representing the citation's trajectory of the various subjects in our two populations. Considering the size of our trail-set (over 3000 trajectories) and our first visual mapping of the trails (see figure 8.1): our trail-set yields clutter and overdraw, making the visual analysis process difficult (if not impossible). As such, to be able to find new geospatially-based insight from our dataset, we need to find a visualization techniques that both reduce clutter and allow us to answer our visualization tasks.



(A) Alzheimer group (B) Control group

FIGURE 8.1 – Raw path-drawing visualization of fluency tests at $T-10$.

From Ellis and Dix, 2007's taxonomy of clutter reduction techniques exposed in chapter 2, section 2.2, we can infer the clutter reduction techniques that best suits our needs. Be a quick process of elimination of existing techniques, we have: First, as we aim at visualizing trajectories, techniques such as *space-filling*, *pixel plotting* or *dimensional reordering* are ruled out. Moreover, we want to have an overview of all

our trajectories, we cannot use *sampling* nor *filtering*. Finally, we want to be able to compare the distances between cities, we cannot use point displacement nor topological distortions techniques. In fine, four clutter reduction techniques could be used in our case: *opacity*, *clustering*, *line displacement* and *bundling*. Furthermore, in terms of low-level visualization tasks (as proposed in Lee et al., 2006's taxonomy – see section 5.1.2), our goals stands in: Visualizing an overview of all the citations trajectories and finding potential patterns similarities and differences during the aging process of each population as well, and comparing the flows of citations trajectories between our two population. As detailed in section 5.1.2, we see here that bundling fits well our defined tasks. Overall, all of the above makes the case to the use bundling techniques to reduce clutter in a geospatially-based visualization of our citation trails.

This final chapter is organized as follows: First, we begin by analyzing and comparing some preliminary cognitive results with our data-set. Then we explore and analyze the fluency maps through bundling for each of the stages before the diagnosis (from 12 years before until the year of diagnosis) and for each phase ($[0;15]$, $[15;30]$ and $[30;60]$ seconds) of the fluency test. Our analysis is first focused on understanding the effects of aging on our two populations (Alzheimer (AD) and control groups (CG)), then we directly compare them in order to find potential tendencies, clusters, correlations and outliers. Finally, we formulate new hypothesis and statistically verify them from the results of our bundled maps.

8.0 Résumé (Français) Application du Bundling aux Cartes Cognitives

Au cours de ce chapitre, nous illustrons notre processus d'exploration et d'analyse de notre base de données des tests de fluence verbale (construite en chapitre 7). A l'aide de notre technique de bundling (exposée en chapitre 6), nous montrons que le bundling est un outil de visualisation et d'exploration efficace dans le cadre de notre étude sur la maladie d'Alzheimer. Plus spécifiquement, nous montrons que l'utilisation du bundling nous a permis de formuler de nouvelles hypothèses sur la maladie d'Alzheimer.

Au cours de notre exploration des tests de fluence verbale, notre travail s'est focalisé sur l'analyse visuelle des jeux de trajectoires (comme défini au chapitre 2) représentant les trajectoires de citation de villes des sujets et ce pour nos deux populations. Compte tenu de la taille de notre jeu de trajectoires (contenant plus de 3000 trajectoires) et de nos premières visualisations de ces dernières (voir figure 8.7), il apparaît nécessaire de réduire l'occultation et le sur-dessin engendrés par le grand nombre de trajectoires à représenter. Ainsi, afin de nous concentrer sur la découverte de nouvelles caractéristiques géospatiales de nos populations, il est nécessaire de trouver une technique de visualisation capable de réduire l'occultation du dessin tout en nous permettant de garder une représentation des trajectoires conforme à la topologie géographique.

Pour répondre à cette problématique, nous fondons notre analyse sur la taxonomie des techniques de réduction d'occultation établie par Ellis et Dix, 2007, et évoquée en chapitre 2, section 2.2. En partant de cette taxonomie, nous pouvons définir la liste des techniques existantes répondant à notre problématique et nos besoins. En premier lieu, comme nous nous focalisons sur la visualisation de trajectoires, nous pouvons éliminer les techniques de réduction d'occultation telles que le *space-filling*, le *pixel plotting* ou le *réagencements de dimensions*. De plus, considérant que notre but est d'avoir un aperçu de l'ensemble des trajectoires de nos populations, nous ne pouvons utiliser des techniques telles que l'*échantillonnage* ou le *filtrage*. Enfin, comme nous souhaitons pouvoir comparer les distances entre les villes, nous ne pouvons utiliser de techniques de déplacement de point ou de distorsions topologiques. In fine, quatre techniques de réduction d'occultation semblent répondre à notre cas d'application : l'*opacité*, le *clustering*, le *déplacement de lignes* et le *bundling*.

De plus, en considérant les tâches de visualisation bas niveau (comme proposé par la taxonomie de Lee et al., 2006 – voir section 5.1.2), le but de notre exploration consiste à visualiser une vue d'ensemble de toutes les trajectoires de citation ainsi qu'à trouver des similitudes ou des différences au cours du processus de vieillissement de nos populations et enfin de comparer les flux de citations de trajectoires entre nos deux populations. Ainsi et comme détaillé en section 5.1.2, nous voyons que le bundling nous permet de répondre à l'ensemble de ces tâches.

De ce fait, une fois l'ensemble des tâches et besoins de visualisation de nos données défini, nous voyons bien que le bundling est une technique adaptée pour réduire l'occultation de visualisation géographique de nos trajectoires.

Dans ce cours résumé en français, nous détaillerons les étapes importantes de notre analyse qui nous ont amenés à formuler et valider statistiquement une potentiel nouveau marqueur de la maladie d'Alzheimer.

Analyses Préliminaires

Avant de débuter notre exploration visuelle des cartes de fluences verbales, nous avons procédé à des analyses statistiques préliminaires de notre base de données. Ces analyses préliminaires ont montré trois points. Premièrement, nos analyses ont à la fois corroboré des résultats précèdent de la maladie ainsi que la validité de notre base de données. Ainsi, nous avons pu valider certaines variables dépendantes de la maladie (*i.e.* le genre et le niveau académique, voir figures 8.4 et 8.5). Ensuite, ces premières analyses ont montré que les tests de fluence verbale semblaient bien liés aux capacités de navigation spatiale de nos sujets. Enfin, elles ont aussi montré que les premiers tests statistiques uniquement fondés sur des relations géospatiales n'étaient pas statistiquement significatifs. En d'autres termes, sans exploration visuelle préalable, nous n'avons pu formuler une hypothèse fondée sur la distribution géographique des fluences verbales qui soit statistiquement valide.

Vers la Visualisation et le Bundling des Cartes de Fluences Verbales

Suite à nos analyses statistiques préliminaires, nous avons visualisé les trajectoires formées par les citations de villes de nos deux populations. Pour ce faire, nous avons représenté géographiquement chaque citation et avons réduit les problèmes d'occultations des données à l'aide du bundling (*c.f.* figures 8.7 et 8.9).

Ainsi, nous avons étudié et comparé le bundling de chaque intervalle à chaque date avant diagnostique pour nos deux populations (*c.f.* section 8.3 et figures 8.10, 8.10, 8.11, 8.15 et 8.16).

Au final, notre exploration systématique de nos deux populations a fait émerger des similitudes et des différences notables au cours du processus de vieillissement de nos deux populations (le groupe contrôle et le groupe Alzheimer), résumées comme suit :

- Les cartes de fluences verbales de nos deux populations partagent un faisceau triangulaire commun cité dans les 15 premières secondes du test.
- Ce principal faisceau triangulaire commence à être affecté par la maladie d'Alzheimer deux à quatre années avant la date du diagnostic.
- Pour le groupe Alzheimer, nous notons une réduction de la distribution géospatiale ainsi que de la densité des faisceaux (*i.e.* trajectoires agrégées) près de 12 années avant le diagnostic de la maladie. Cette tendance est principalement visible dans les 30 dernières secondes du test.
- En parallèle, nous notons que dès 12 ans avant le diagnostic, les citations des sujets du groupe Alzheimer semblent se focaliser et s'intensifier autour de la ville de résidence des sujets (ici Bordeaux).

Toutes ces observations visuelles suggèrent une diminution avec le temps de la capacité des sujets du groupe Alzheimer à explorer la globalité du territoire français au cours des tests de fluence verbale. De plus, cet effet est visible près de 12 ans avant le diagnostic.

Une Nouvelle Hypothèse sur la Maladie d'Alzheimer

L'ensemble des explorations visuelles et observations de nos deux populations (que ce soit l'étude indépendante du processus de vieillissement ou la comparaison de nos populations) suggère que les patients du groupe Alzheimer ont tendance à citer des villes qui se situent près de leur ville de résidence. Cette nouvelle hypothèse concernant la distribution géographique des citations de villes a ainsi pu être testée statistiquement en comparant le ratio de villes citées dans le département de la Gironde entre nos deux populations. Ce test statistique s'est avéré significatif. Ainsi, nos premiers résultats semblent valider l'hypothèse suivante :

«Les patients qui vont développer la maladie d'Alzheimer ont tendance à déplacer leur processus de navigation spatiale depuis une approche allocentrée vers une approche égocentrée. De plus, cet impact de la maladie d'Alzheimer est visible près d'une décennie avant le diagnostic de la maladie par des psychologues »

Pour conclure, au cours de ce chapitre final, nous avons pu montrer comment l'exploration et l'analyse visuelle des tests de fluence verbale à l'aide du bundling nous a permis de formuler une nouvelle hypothèse sur la maladie d'Alzheimer. Ici, l'exploration visuelle grâce au bundling a joué un rôle majeur pour trouver un nouveau marqueur significatif de la maladie. De manière plus générale, nous pensons fermement que notre analyse ainsi que nos résultats ont ouvert la voie vers de nouvelles recherches approfondies sur les tests de fluences verbales et leurs relations avec les cartes cognitives à travers plusieurs approches novatrices. Notamment, nous pourrions envisager de vérifier ces résultats en utilisant des tests de fluences verbales provenant d'une autre ville de la cohorte 3C (Dijon ou Montpellier). De plus, si nos résultats initiaux sont validés par d'autres études, alors nous pourrions potentiellement envisager l'utilisation des tests de fluence verbale comme un outil de diagnostic de la maladie réalisable simplement par un médecin généraliste. Permettant ainsi un diagnostic moins couteux et plus précoce de cette maladie.

8.1 Preliminary Analysis

As argued in the introduction of part III, we already expect some known results to be present in our data-set (such as finding a difference in the citation count for subject developing Alzheimer's Disease (AD), Amieva et al., 2008). This section aims at cross-checking the results of our digitization and cleaning process as well as validating known results. Next, we take some simple geographical analysis based on values derived from the data-set's attributes (such as the means distance between cited cities).

8.1.1 Socio-Demographic Distribution

The state of our data-set and some associated socio-demographic information are summed up in table 8.1. Overall, we confirm that our two groups (*i.e.* AD and CG) are paired one to one, meaning that for any given subject of the Alzheimer group exists one and only one subject in the control group sharing the same age, gender and academic level.

TABLE 8.1 – Socio-demographic data of our two groups (Alzheimer and control). Subjects are perfectly paired in age, gender and education level.

	Alzheimer Group	**Control Group**
Subject count	127	127
Tests count	390	390
Mean age & standard deviation	79.6 ± 5.5	79.6 ± 5.6
Male count	31	31
Female count	96	96
Right-handed count	119	119
Left-handed count	4	4
Ambidextrous	4	4
Academic level 1	22	22
Academic level 2	33	33
Academic level 3	26	26
Academic level 4	26	26
Academic level 5	20	20

Figure 8.2 shows the distribution of the number of test per time before diagnosis (Ti, see section 7.8) in our database. Here, we see that our database has a low number of tests at six years before the diagnostic ($T-6$). This should be taken into account during our analysis, especially to compare $T-6$ with other Ti. The low number of

tests at this specific period is caused by our choice to remove fluency-tests where too much data was missing. In particular, those test were missing any citation after the 30 second mark (see section 7.5.3).

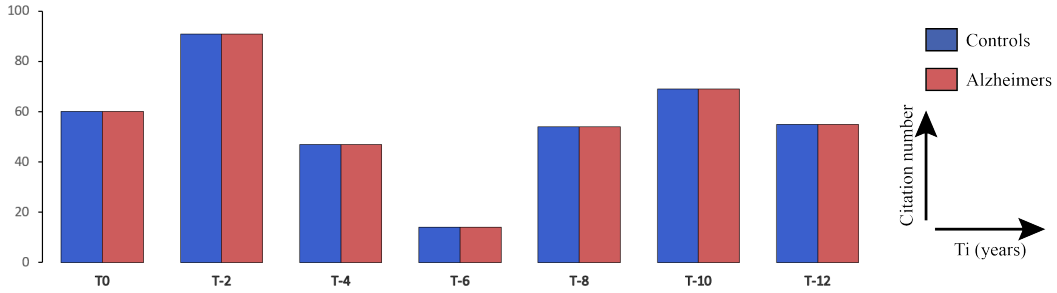


FIGURE 8.2 – Distribution of the number of tests per time before for the two populations.

8.1.2 Ground Truths of the Data-set

In a previous work on Alzheimer in the framework of the 3C study, Amieva et al., 2008 studied the evolution of the Issac Set Test (IST) of future Alhzeimer's disease (AD) subjects and their matched control (Amieva et al., 2008, figure 1.a). They showed that the number of citations in the IST for the Alzheimer group declined faster than for the control group. According to their work, we can see "a sharp and regular decline [of future AD subjects] can be observed until time of diagnosis".

Thus as a first analysis of our fluency test database, we study the evolution of the number of citations for the two populations. Figure 8.3 shows this evolution. For the control group, the number of citations seems to be constant with no sign of improvements caused by a test/re-test effect, neither a decline caused by the aging process. Another possibility is that the two aforementioned effects may cancel one-another. Conversely, the number of citations for the Alzheimer group shows a constant decline in regards to the control group's score. In particular, the future AD subjects shows a significant difference after $T-6$.

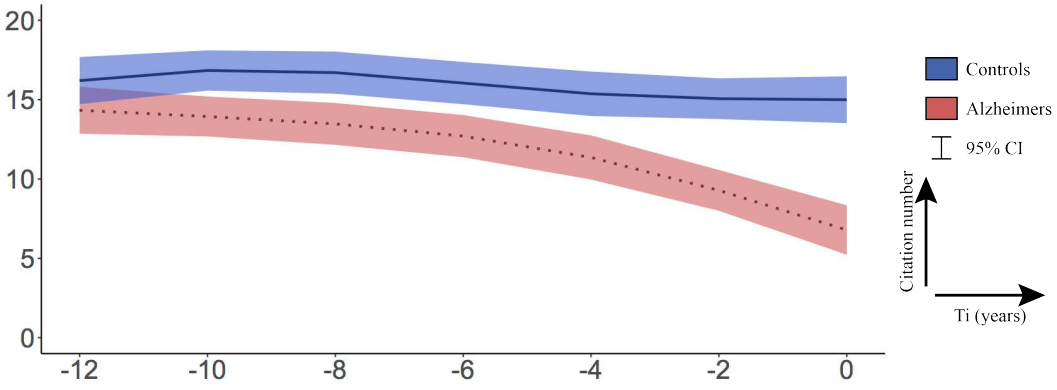


FIGURE 8.3 – Evolution of the citations distribution during the 12 years preceding the diagnosis of Alzheimer's disease.

In comparison with the results exposed by Amieva et al., 2008, we show similar effects in terms of citation quantities between the two population. Thus, corroborating the results of Amieva et al., 2008 and the correctness of our database.

Next, we studied the distribution of the number of citation by gender. Figure 8.4 shows this distribution. We denoted a wide variability in the amount of cited cities between men and women. As such, women seem to cite fewer cities than men in our two populations (Alzheimer and controls). Moreover, our analysis shows that between women, the ones who will develop Alzheimer's disease seem to have a deficit in the number of citation which is significant 10 years before the diagnosis (confidence interval of 95%, $[-4.42; -0.66]$ citations at $T-10$). For men, the difference between AD and controls starts to be significant 4 years before the diagnosis (confidence interval of 95%, $[-7.47; -0.96]$ citations at $T-4$). However, we should point that our statistical test for the male population has a substantial uncertainty. In particular at $T-6$ with only 4 men in each group.

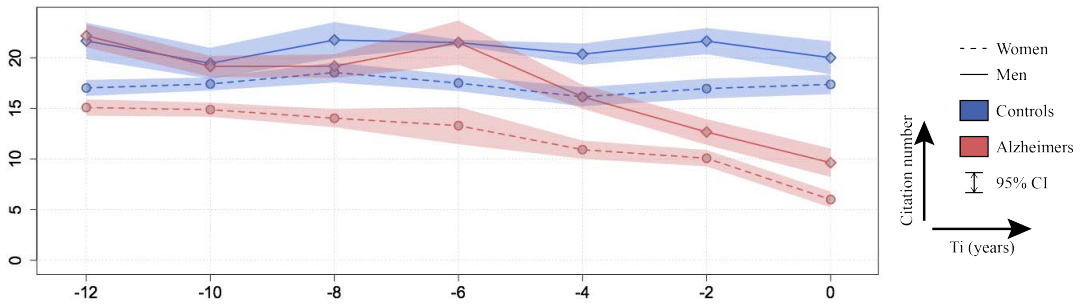


FIGURE 8.4 – Evolution of the distribution of citations by gender.

Previous studies on the full Isaac set test (colors, animals, fruits and cities) did not show any significant differences between men and women. Conversely, neuro-psychology studies (Beatty and Tröster, 1987) showed existing differences in the spatial navigation capabilities between men and women. Here, our results tend to demonstrate that the fluency tests do rely on the spatial navigational capabilities. Thus inferring our first hypothesis stating that the study of the fluency test will give us the opportunity to study the cognitive map of the subjects.

Similarly, we compared the impact of the subject's academic levels independently from the time before the diagnosis. We find that a higher academic level lead to higher scores in the number of cited cities (see figure 8.5). Our models shows a relatively constant difference between the Alzheimer and control populations.

Overall, the impact of gender and academic levels demonstrate the dependency between them and the probability to develop Alzheimer's disease, thus validating our choice to build a database paired one to one in regards to these factors between the two populations.

In addition, we studied the distribution of citations in each interval of the verbal fluency tasks in figure 8.6 (*i.e.* $[0;15]$,$[15;30]$ and $[30;60]$ seconds, see section 7.5.3). 8 years before the diagnosis the Alzheimer's groups has an inferior citation speed between

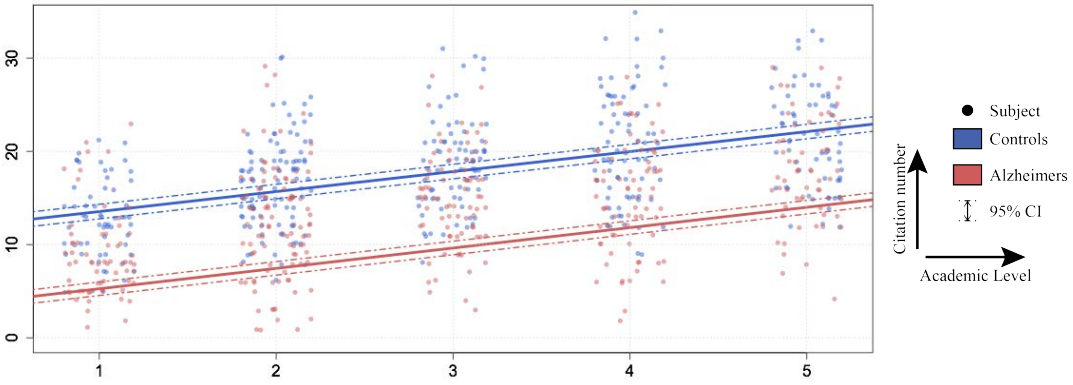
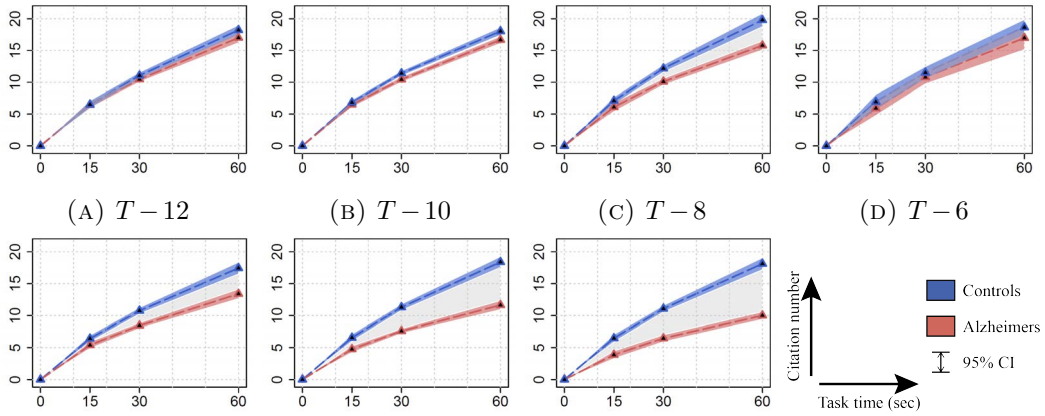


FIGURE 8.5 – Evolution of the citations distribution by academic levels.

30 and 60 seconds although the speed was similar in the interval $[0;15]$. This effect is less pronounced at $T-6$, probably caused by the higher uncertainties in the measure caused by the low number of subjects (only 14 in each groups). This analysis confirms one of the hypotheses of the study, stating that the subjects start using a spatial navigational citation strategies as time passes during the test. The performances at the beginning of the tasks would be linked to semantic categorization processes with city names known by heart and without the need for spatial navigation capabilities. As the subject ages we see that the citations speed between $[30;60]$ seconds decreases as well as for the speed between $[0;15]$ and $[15;30]$ seconds.



(A) $T-12$ (B) $T-10$ (C) $T-8$ (D) $T-6$

(E) $T-4$ (F) $T-2$ (G) $T-0$

FIGURE 8.6 – Temporal distribution of citations during the fluency tests at each Ti for the two populations.

Finally, for repetitions errors (as defined in section 7.4) our analysis show a significant difference between the ratios of repetitions/citations for the two groups as earlier as 8 years before diagnosis and constantly increasing with time until $T0$. At $T-8$ we have a confidence interval of 95% for ratio in $[1\%;7\%]$ and at $T-0$ we have ratio in $[3.4\%;13\%]$. Similarly, the number of intrusions defined as a word not being a city (section 7.4) stay relatively low for the two groups without any significant differences. We noted some intrusions as a pollution from the previous verbal tasks of the Isaac set test asking for names of fruits (*e.g.* a subject citing *Banana* instead of a city).

Partial Conclusion: Our first analysis corroborates both previous known results of the 3C database as well as the correctness of our database. Moreover, they show and validates known AD dependent factors such as gender and academic levels. Last but not least, this first analysis tend to demonstrate that fluency tests rely on spatial navigational capabilities.

8.1.3 Geographical Attribute-Derived Analysis

Focusing on our hypothesis to study the spatial navigation process of our two groups, we studied some spatially based factors derived from the attributes of our database. We studied the following factors for the two populations:

- the mean distance between two cited cities,
- the number of cited foreign cities,
- the population of cited cities to evaluate their importance,
- the frequency of cities citation throughout the fluency task,
- the starting city,
- the area covered by the subject during the citation (using a simple gift wrapping algorithm).

Most of these derived factors did not yield significant results to allow the emergence of a clear distinction between the cognitive maps of the two groups. These observations may be answered by various hypotheses:

- Our results shows a high inter-individual variability between subjects suggesting wide differences between the subject cognitive maps. In turn, this could invalidate the idea of a shared cognitive maps between subjects. These high variabilities inherent to our data-set could also impact on the quality of classic statistical analysis.
- On the other hand, the studied geo-spatial parameters may not be the correct ones or the fluency tests may not be the correct tasks to evaluate them.
- Finally, we could hypothesize the diminution in the number of citations as dysexecutive troubles rather than alterations of the subject's cognitive map. In this case, these dysexecutive troubles would make the task harder for subjects, thus leading to a diminution of the number of citations.

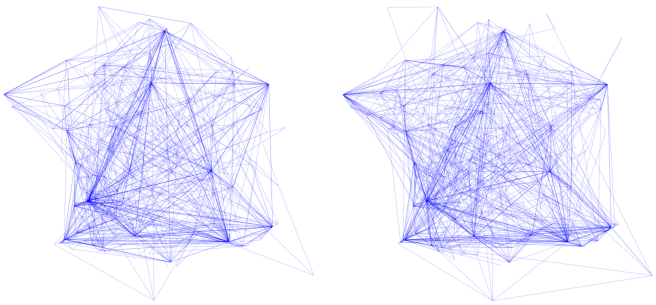
However, some results (such as the ones from figure 8.6) support corroborating evidences towards a deeper study of the geo-spatial navigation of subjects during the fluency test. To allow the definition and evaluation of new derived factors, we choose to visually explore the geographical patterns of the fluency tests.

8.2 Visualizing and Bundling Fluency-tests

Per construction, our data-set is spatially embedded in a 3-dimensional sphere: the earth. As such, we chose to visualize our dataset through a map representation type. However, different types of map projection and representation exist (see MacEachren, 2004) with pros and cons. In this section, we detail our rational to layout, visualize and simplify the drawing of the fluency tests.

8.2.1 Choosing the Right Projection

To answer this question, we used the results of our preliminary analysis. Our initial spatial analysis shows that the mean distance between cited cities is around 1000 kilometers and the mean distance from the subjects' home (*Bordeaux*) is around 1200 kilometers. Moreover, if we study the distribution of foreign cited cities, we see that more than 90% of the cited cities are French cities. From these results, we can confidently estimate that we can project our cited cities in a standard geographical projection of France. We choose to use the well-known stereo-graphic projection of WGS84 centered in $(47.0; 0.0)$ on a plane since in distances can be visually compared. Limiting our visualization to cities in France ensures us that the used projection yields minimal deviation errors. Thus, in our case we can confidently approximate distances and positions of cities in the projected space as their real coordinates in the sphere. In practice, we choose to only visualize cities with latitudes in $[41.0, 52.0]$ and longitudes in $[-5.0; 9.0]$. This includes some foreign cities from bordering countries such as *London* or *Barcelona*. From there, we can confidently visualize and compare cited cities in a locally-based projection (see figure 8.7).



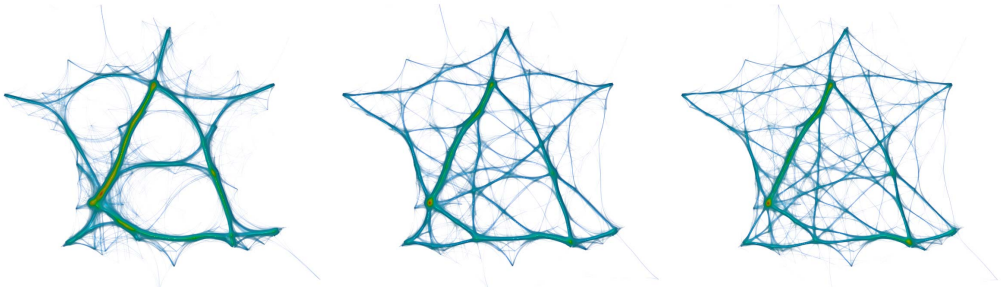
(A) Alzheimer group (B) Control group

FIGURE 8.7 – Raw path-drawing visualization of fluency tests at $T-10$.

8.2.2 Bundling Parametrization

Figures 8.7a and 8.7b show the visualization of our two groups at $T-12$ in our projection system. At first glance, the two visualizations yield clutter and overdraw, making the comparison difficult. To simplify the visualization and allow the emergence of underlying structure, we chose to use our bundling technique (chapter 6). However, prior

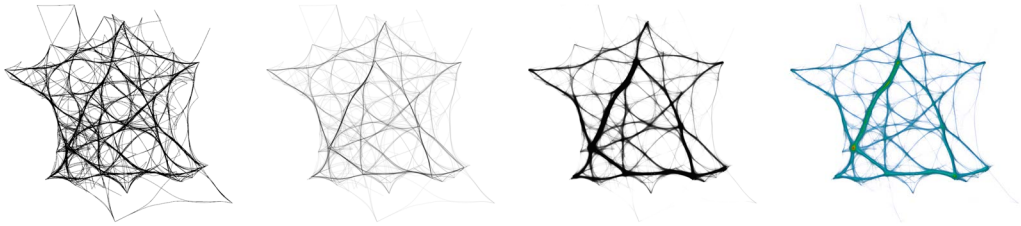
to the analysis of a bundled fluency test, we need to specify the level of desired aggregation in the path drawing (as described in section 4.2). For KDE-based bundling techniques (section 6.1, Lhuillier, Hurter, and Telea, 2017a), the level of aggregation is defined by the kernel radius R expressed as a percentage of the image size I. Here bundling our fluency tests at different scales (R values) yields vastly different visualization (figure 8.8).



(A) $R = 19\% \simeq 200km$ (B) $R = 7\% \simeq 100km$ (C) $R = 3\% \simeq 50km$

FIGURE 8.8 – Multiscale bundling of the fluency test for the control group 12 years before diagnosis of their paired subjects.

In our specific case, data is spatially embedded in a geographical space roughly the size of a square with sides of length 1200 kilometers. Hence, bundling with a radius of 10% is equivalent to bundling edges that are in a 120 kilometers radius of each other. To explore the correct kernel radius to bundle fluency tests, we can use the administrative levels of France: *regions* ($\simeq 200km$, figure 8.8a), *departements* ($\simeq 100km$, figure 8.8b) and *arrondissements* ($\simeq 50km$, figure 8.8c). Using administrative levels follows the hierarchy levels of vista space (Meilinger, 2008), thus bundling at this levels aggregates edges that are similar to a given vista space level. From our results, we choose to use an aggregation level of 7% for comparing the fluency tests.



(A) Raw (B) & Opacity (C) & Width (D) & Color Map

FIGURE 8.9 – Multiscale bundling of the fluency test for the Alzheimer group at $T-12$.

Finally, to visually enhance the bundled results we used opacity, width and color as a function of the density of a bundle (see figure 8.9). As density can largely vary from bundle to bundle, we choose to implement the width parameter as a logarithmic function of the bundles density normalized by the maximum bundle density. This makes denser bundles appear more clearly than lower ones. Conversely, we used a color map (figure 8.9d) to differentiate high densities allowing an easier comparison of denser bundles.

8.3 Exploration of Bundled Cognitive Maps

Having defined our approach to visualize and declutter the fluency tests, we can explore and visually analyze the temporal evolution of our two populations in order to find new insights (*i.e.* tendencies, patterns, outliers). In this section, we present the results of our systematic exploration of bundled fluency tests. First, we visualize the effect of aging on our two populations separately, both on the full length of the fluencies and for each interval. Then we compare the two populations. To be able to compare bundled maps between each other in terms of bundle geospatial distribution and densities, we performed a global normalization between them (*e.g.* to share the same density values).

8.3.1 Visualization of the Aging Process

Prior to comparing the populations, we need to understand and explore the effect of aging on each of our two populations. In this regard, we compare the bundled fluency maps for each population following two axis: First, we compare the entire fluency test of a population between each Ti (time before diagnostic). Then we compare the temporal evolution of the citations during the three phases of the test ([0;15], [15;30] and [30;60] seconds).

Global Fluency Maps

Comparison constraints; As the number of subjects tested can greatly vary between each Ti (see figure 8.2), we cannot compare accurately the distribution nor the densities of bundles between fluency maps without further processing. To answer this problem, we choose to normalize the densities between Tis. As such, we define the maximum density of all the bundled maps as;

$$Dmax = max\left\{\frac{d_{max}(Ti)}{Card(Test_{Ti})} | i \in [0;12]\right\} \tag{8.1}$$

From this definition, we normalize each Ti bundled maps by defining the maximum density as $||d_{max}(Ti)|| = Dmax * Card(Test_{Ti})$. This allows a better comparison of the maps throughout the aging process of each population.

Control group: We compare the bundled fluency maps of a control subject over the twelve years before diagnosis in figure 8.10. At first glance, throughout the aging process, the bundled maps stay pretty much the same. In these maps, we distinguish one main circular pattern (*a main bundle*) highlighted in figure 8.10a. This main bundle follows a route linking the largest French cities. Clockwise it follows Paris, Lyon, Marseille, Montpellier, Toulouse, Bordeaux, Poitier, Orleans and back to Paris. Alongside the main *triangular* bundle, we see secondary bundles covering the main French cities throughout the country. These secondary bundles are mainly in the center of the *triangular* bundle and at the North-East and North-West of it.
Overall, the citations cover all the geospatial space of the French territory with no

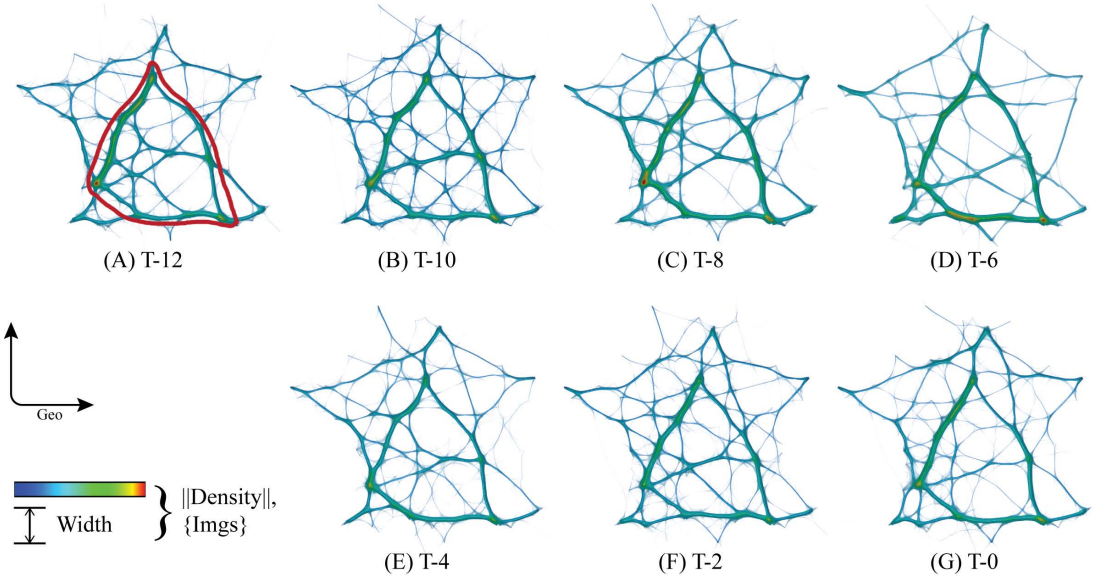


FIGURE 8.10 – Bundling of non-dement group over the twelve years before diagnosis.

noticeable effect of the aging process on the control group. However, we do note some simplification of the overall bundled drawing at $T6$ (figure 8.10d). If the density of the *triangular bundle* stays the same, the secondary bundles do not cover the same amount of space as in the other Ti. This simplification is mainly caused by the low number of subjects at this date (see figure 8.2).

Alzheimer group: The bundled fluency maps of AD subjects from $T12$ to $T0$ is compared in figure 8.11. Here, the bundled maps drastically simplify over time both in terms of bundle density and geospatial distribution. Overall, the main *triangular* bundle stay visible throughout the aging process and its density starts decreasing at $T2$ and $T0$. Conversely, the secondary bundles density and geospatial distribution starts to vanish after $T8$ (*i.e.* 8 years before diagnosis). This process is especially visible for cities in north of France (circle in figure 8.11c) at $T8$. It then expands towards other secondary bundle sites at $T6$, $T4$ (figures 8.11d and 8.11e) and is almost not visible at $T2$ and $T0$ (figures 8.11f and 8.11g).
Finally, we note that the density of bundles near the city of Bordeaux (south west of France, circle in figure 8.11a) stays almost identical in regards to the variation of density in the other parts of France.

Temporal Fluency Maps

In the continuity of the exploration of the aging process for our two populations, we visualize the temporal maps of each of the phases of the fluency tests ($[0;15]$, $[15;30]$ and $[30;60]$ seconds). By splitting the bundled maps for each phase of the test, we aim at visualizing differences in the geospatial behavior of the two populations at each phase.

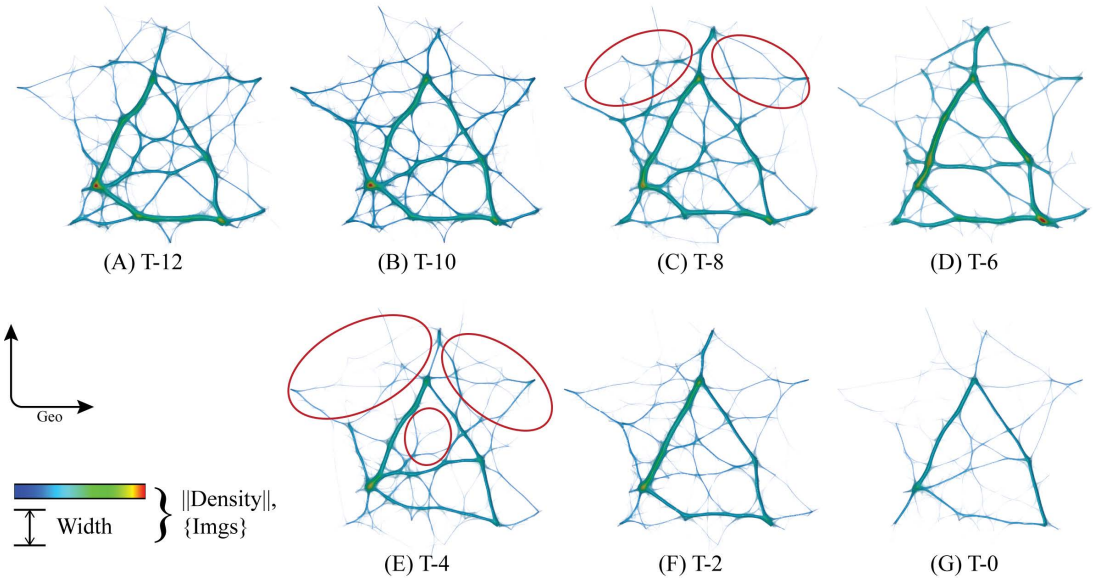


FIGURE 8.11 – Bundling of dement group over the twelves years before diagnosis.

Comparison constraints: Similarly to the global exploration of the fluency tests, we normalize the densities between each bundled drawing by defining the maximum density of all the bundled test at each Ti and each interval as:

$$Dmax = max\left\{\frac{d_{max}(Ti)}{Card(Test_{Ti}) * t_{interval}} | i \in [0;12]\right\}, \tag{8.2}$$

where $t_{interval}$ is the elapsed time for a phase (15 seconds for the two first and 30 seconds for the last one). Then, we modulate the density of each drawing by defining the maximum density as $||d_{max}(Ti)|| = Dmax * Card(Test_{Ti}) * t_{interval}$.

Control group: Figure 8.12 shows the bundling of each phases of the fluency tests over the twelve years before diagnosis for the control group.
For the first interval $[0;15]$ seconds, we can distinguish the main *triangular* patterns described previously. During the aging process of the control group in this first part of the fluency test, we do not note any particular simplification in this main *triangular* bundle (expect maybe at $T4$ but not at $T0$). However, we note a simplification of the secondary bundles starting at $T8$ (figure 8.12b) and regularly decreasing at $T2$ and $T0$ (figures 8.12c and 8.12d).
In the two last intervals of the fluency test ($[15;30]$ and $[30;60]$ seconds) we see a variation of bundle density between the first fifteen seconds and the rest of the test. Despite this density difference, the last two intervals do not show any particular, constant or recurrent pattern during the aging process of the control population. In these intervals, we see that the geospatial distribution of citations are spread wider over the French territory. However, the low densities makes it hard to find any potential pattern or difference during the aging process.

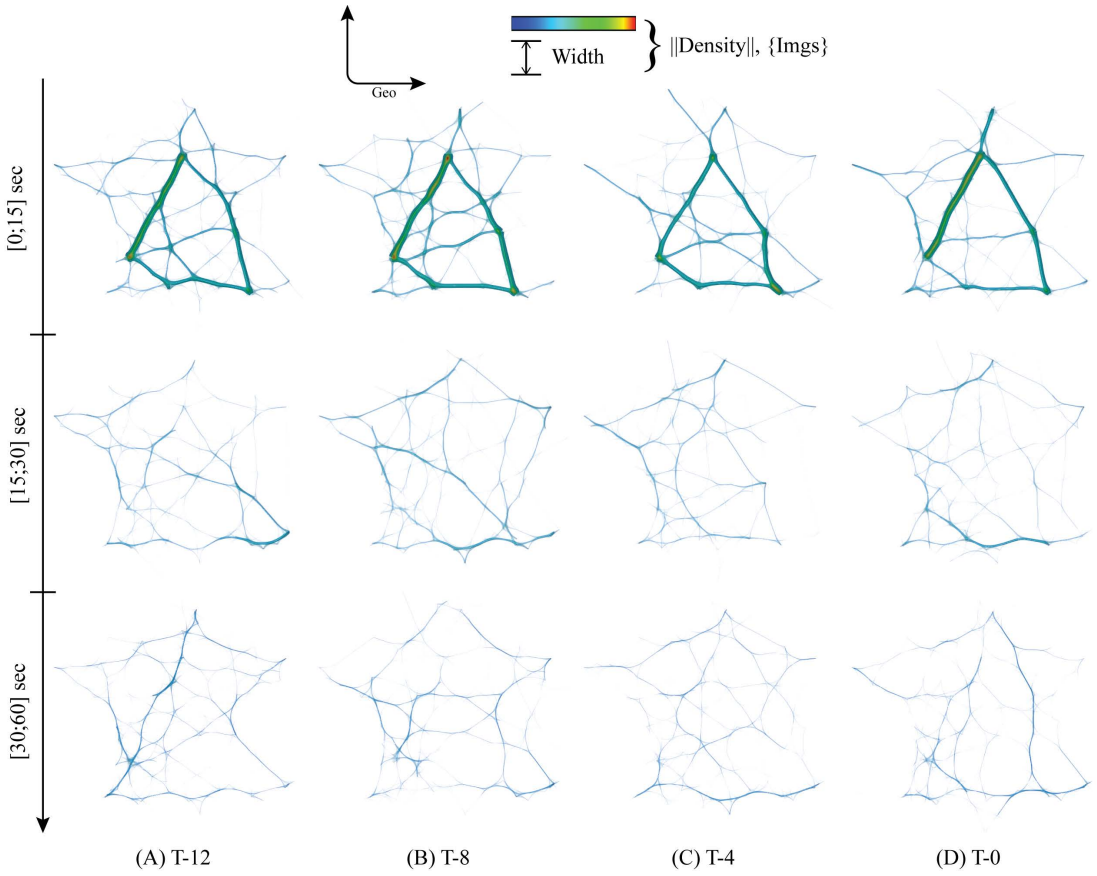


FIGURE 8.12 – Bundling of the fluency tests temporal evolution for the control group.

Alzheimer group: Figure 8.13 compares the bundled drawings of each phase of the fluency test at various times before diagnosis for the Alzheimer group. As with figure 8.12, we note a sensible difference in the bundle densities between the first fifteen seconds and the rest of the test.

In the first fifteen seconds (figure 8.13 top row), the main *triangular* pattern stays visible from $T12$ to $T0$. However, we note a drastic decreasing of the density of this main *triangular* pattern as well as the secondary bundles (depicting the contour of the french territory). This decrease in the bundles density is such that at $T0$ only the main bundle stays visible.

Despite the low densities, the second part of the fluency test ($[15;30]$, figure 8.13 middle row) shows no noticeable patterns. We simply note a constant simplification of the maps with the aging process starting at $T4$. For the $[15;30]$ seconds, $T12$ and $T8$ seems to be similar in terms of geospatial distribution over France.

Finally, the last 30 seconds seconds of the tests seem to simplify as well during the aging process. At first glance, we note that the bundles in the last phase of the test focuses on the South-West area of France near Bordeaux with aging (*i.e.* the bundles are more and more focused around the region of Bordeaux, see circles in bottom row of figure 8.13).

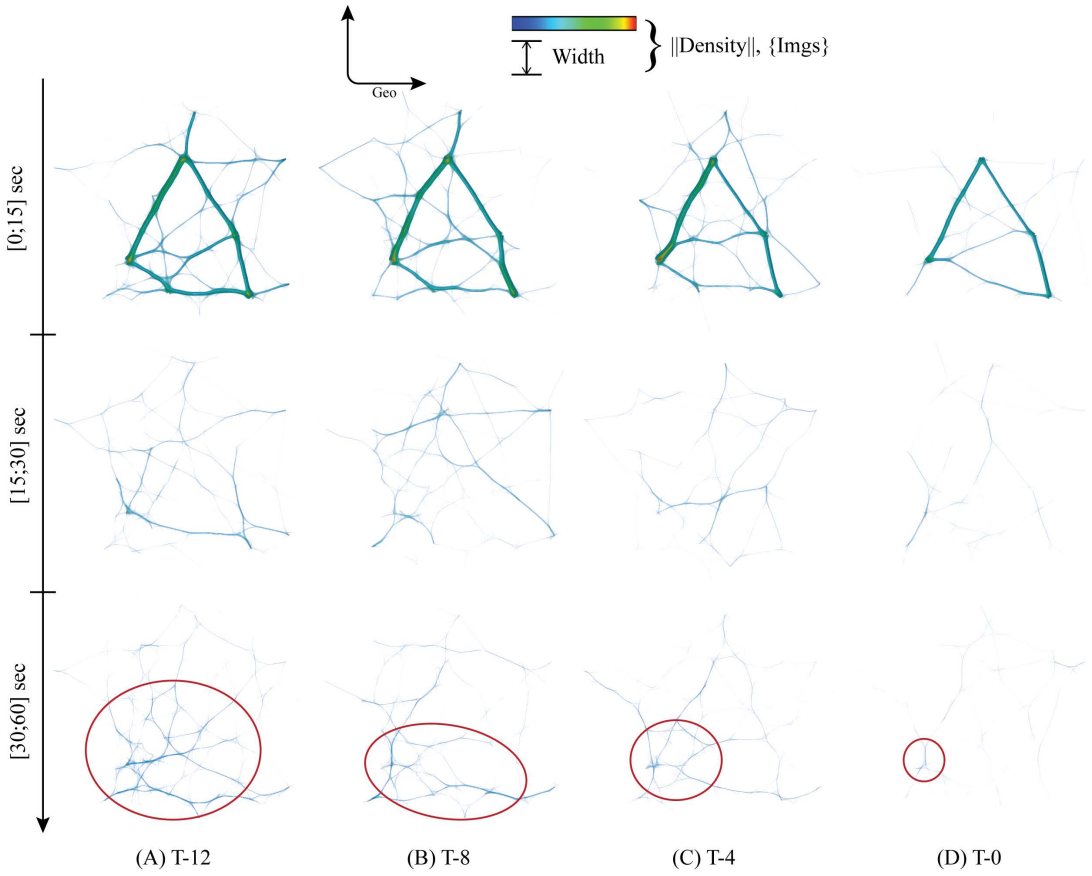


FIGURE 8.13 – Bundling of the fluency tests temporal evolution for the Alzheimer group.

Partial Conclusion: In this sub-section, we explored the bundled fluency maps of our two population independently during the aging process. The exploration of the global exploration of the fluency tests shows that the bundled fluency maps of the control population seemed to be stable over the years, whereas the maps of the Alzheimer groups decreased both in terms of bundle density and geospatial distribution. The further study of each interval of the fluency tests revealed a huge inequality in the bundle density between the first 15 seconds and the rest of the fluency tests for our two groups. This thorough exploration showed some minimal simplification in the control group. For the Alzheimer group, this showed a major decrease of performance in each phase of the tests during the aging process. Specifically, it showed a focus of the citations in the region of Bordeaux: the place of residence of our two populations. This first analysis of our population suggest a further comparison between the two to emphasize and confirm the observed differences.

8.3.2 Visual Comparison of the Two Groups

To further qualify the differences between our two groups, we need to compare the two populations at each time of diagnosis to potentially highlight our first analysis.

Global Fluency Maps

We start by exploring the comparison of the full fluency maps. We know that the number of subjects is the same in each group at each time before diagnosis. As such, to be able to correctly qualify the comparison between them, we normalize the bundles density by the maximum of the density of the two groups at a given Ti. This allows us to compare the groups one by one at a given time before diagnosis.

The one-to-one bundled density maps of our two groups are shown in figure 8.14. Overall, the figure confirms a lower density and geospatial distribution of the bundles for the Alzheimer group compared to the control group.
Specifically, starting at $T8$ we see a reduction in the secondary bundle density in the North-West and North-East of the French territory (circled in figure 8.14c). With aging, this difference is emphasized for the impaired areas and spread in the inner part of the main *triangular* bundle (figure 8.14e and 8.14f). At $T2$ and $T0$ the difference in

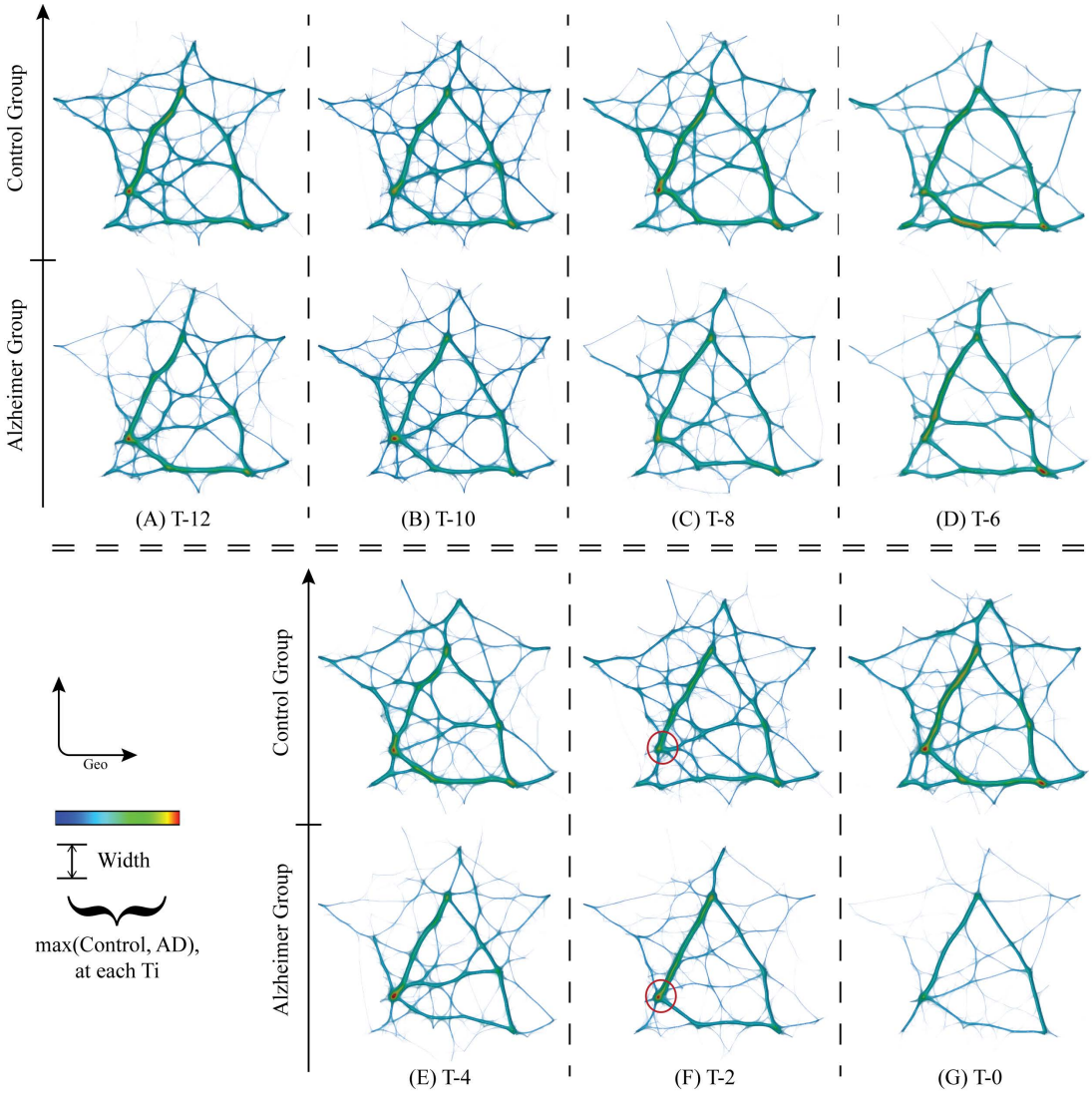


FIGURE 8.14 – Comparison of the bundled maps for the two groups at each Ti.

distribution and density of the secondary bundle is clearly marked in comparison with the control group.
Conversely, the distribution and density of bundles in the area near Bordeaux (south-west of France, circled in red in figure 8.14f), appear to be denser for the Alzheimer group than for the control group for all Ti expect at $T0$.

Temporal Fluency Maps

The comparison of the full fluency maps showed and confirmed existing densities and geospatial differences between our two population and our previous analysis of each phases of the fluency tests showed wide variation of the bundle density between the phases in each population. Hence, we can further compare the fluencies of the two groups by comparing each phase at each Ti. To properly compare them once more, we normalized the densities between the two groups for each phase and each T. The results are shown in figures 8.15 and 8.16.

Starting at $T12$ (figure 8.15a), the comparison shows no significant differences in the first interval ($[0;15]$). For the main *triangular* bundle, the densities are comparable. We note a lack of geospatial distribution for the Alzheimer group in the West of France

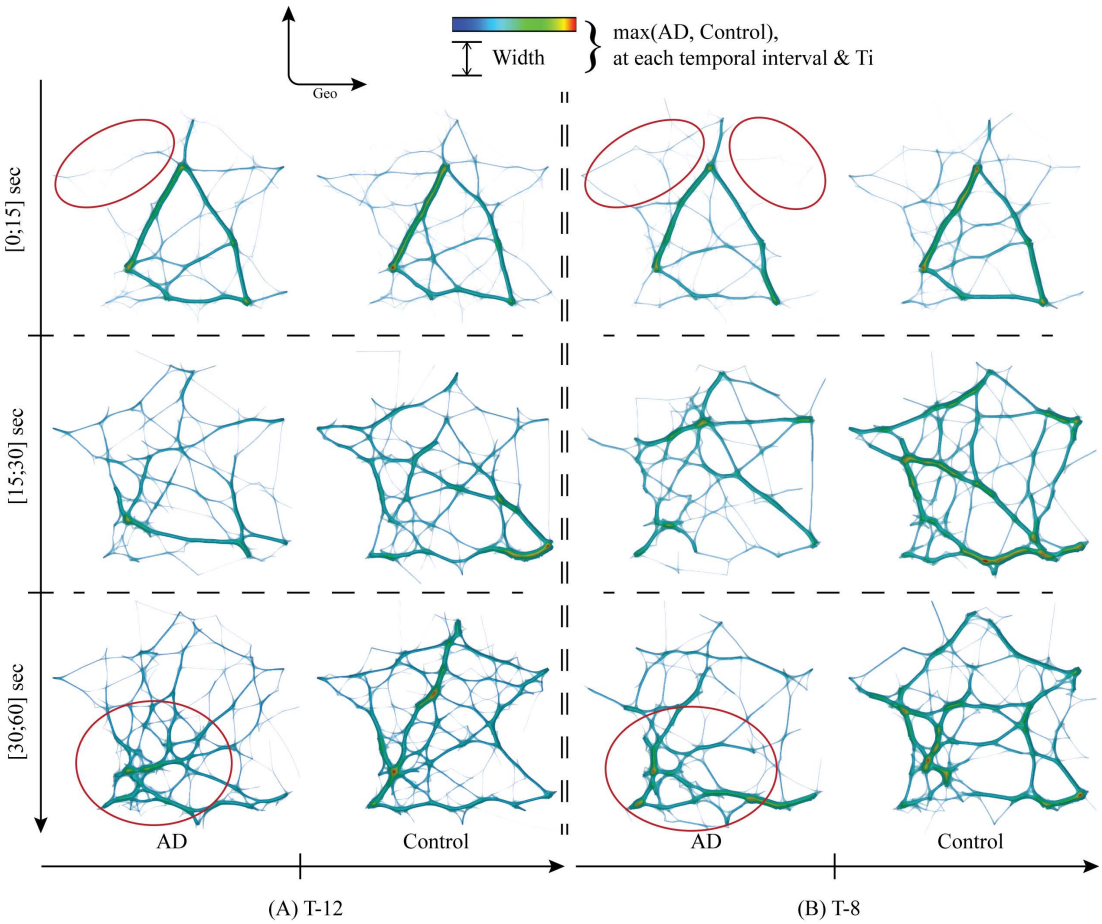


FIGURE 8.15 – Comparison of temporal tests at $T12$ and $T8$ between Alzheimer and control groups.

(circled in the top left image of figure 8.15a). The [15;30] seconds interval shows a higher density of citations in the area near Bordeaux for the AD group. The [30;60] seconds interval clearly shows more bundle in the (wide) region around Bordeaux.

At $T8$ (figure 8.15b), the first 15 seconds are quite similar for the two groups with the main pattern having similar densities. With a small deficit for the AD group in the east of the French territory. The second phase shows an overall decrease in the geospatial distribution of the bundles. However it does not show any pattern or evident focalization on the Bordeaux area. Finally, the last 30 seconds shows again a focalization of the citations of the Alzheimer groups in the area near Bordeaux and the south of France.

The comparison 4 years before the diagnosis (see figure 8.16a) shows noticeable difference in the [0;15] seconds interval. With the south part of the main *triangular* bundle fading away as well as less dense secondary bundles. The second part of the test does not show any recognizable pattern despite a lower density and geo-distribution. In the last phase, the AD subject seems to be more focused on the Bordeaux area compared to the control group (which seem to have a wider distribution over the French territory).

Finally, at $T0$ we see a large difference in each part of the fluency tests (see figure 8.16b). Here, the densities of bundles are extremely low for the AD group compared

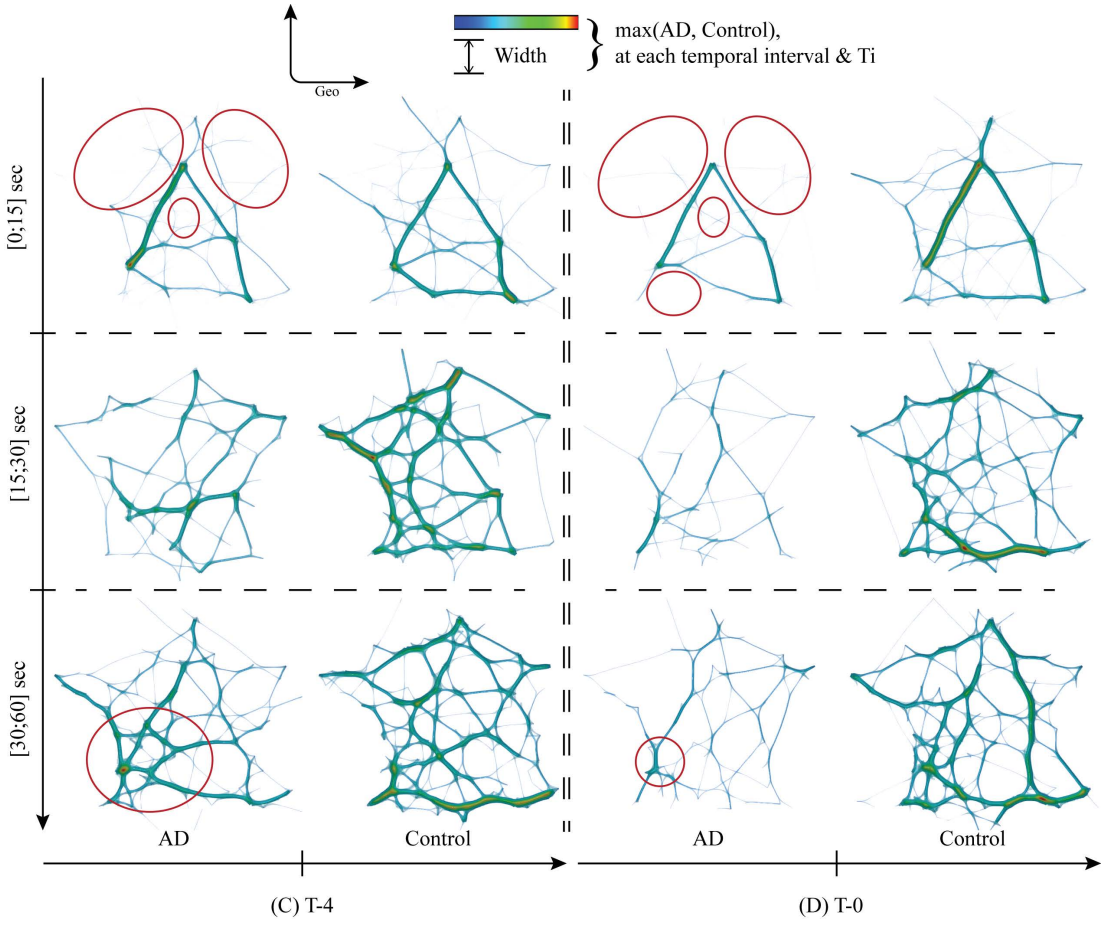


FIGURE 8.16 – Comparison of temporal tests at $T4$ and $T0$ between Control and AD.

to the control group. The first phase shows only the main *triangular* pattern. Conversely, the last two intervals are clearly focused in the Bordeaux area (despite having lower densities than the control group).

Partial Conclusion: After this thorough exploration of the different aspects of the fluency tests with the help of bundling, we have pointed out the following geospatial aspects:

- The fluency maps of our two populations share a same *triangular* bundle that is cited in the first fifteen seconds of the tests.
- This main *triangular* bundle starts to be affected by Alzheimer's disease 2 to 4 years prior to the actual diagnosis of the disease.
- For the AD group, we see a reduction in the geospatial distribution and the density of bundles as early as 12 years before the diagnosis. This is especially visible for the last 30 seconds of the fluency tests.
- In parallel, starting at $T-12$, the citations of the AD subjects seem to focus and intensify in an area near their city of residence, Bordeaux.

All these visual observations suggest a decrease over time in the capacity of AD subjects to geospatial exploration of the French territory that is visible before the diagnosis of the subjects.

8.4 New Hypothesis on Alzheimer's Disease

All of our observations of the two populations both independently through aging, and compared to one another suggest a focalisation on a specific area where AD subject lives. This new hypothesis revealed by the exploration of bundled fluency maps allowed us to formalize new statistical tests to potentially unveil significant differences between our two groups in the years prior to diagnosis.

To statistically test our observations, we quantified the amount of citations in the subjects living state; *Gironde*. Similarly, we measured the amount of citations in French states (*departement*) along the French border (next to another country or the ocean). We chose to split these states and focus on two categories: western and eastern states (detailed in figure 8.17 right row). The western states are defined without the *Gironde* and its two bordering states. Figure 8.17 shows the distribution of citations in each of the defined categories. In addition to the total of citations in a given geographical category (figure 8.17 left row), we computed the ratio of citation for each category. This ratio is simply the number of citations in the concerned category normalized by the total amount of citations (figure 8.17 middle row). This allowed our analysis to be less dependent upon the lack of citations' for the AD group compared to the control one.

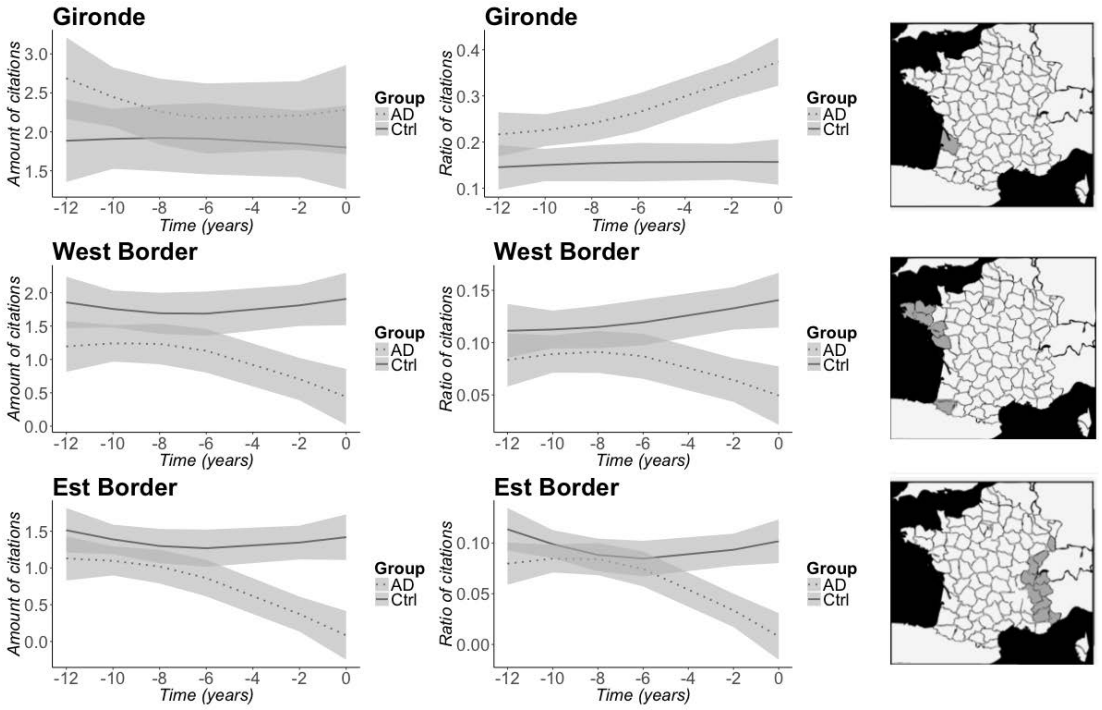


FIGURE 8.17 – Comparison between the AD and control groups of the estimated evolution of the amount and the ratio of cited cities for three geographic areas of the French territory.

Inferring the observations suggested by the bundled fluency maps, our analysis seems to show a focalization of Alzheimer subjects in their state of residence (*Gironde*). Figure 8.17a shows a higher citation ratio starting as soon as $T12$ and increasing through with subjects aging (95% confidence interval $[0.008; 0.15]$). Interestingly enough, the absolute amount of citations for the two populations stays the same at $T2$ and $T0$ despite a drastic drop in the total number of citations for the AD group compared to the control one (as showed in figures 8.6f and 8.6g). This suggest that the decrease in the total amount of citation for the AD population *is* geospatially correlated. Conversely, the western and eastern borders (figures 8.17 and 8.17) appear to be less cited by the AD group as soon as $T12$. The ratio of citations in the western area yield a constant decrease starting at $T10$ as such that it is statistically significant at $T-6$ until $T0$ (95% CI $[-0.07; -0.0014]$). The eastern border see a constant decrease in its ratio of citation starting at $T-6$. The difference is significant for the eastern border at $T-2$ (95% CI $[-0.08; -0.0025]$). These inflection points are consistent with the one observed for the *Gironde* area.

Overall, our first statistical results tend to show that Alzheimer subjects have a tendency to shift their navigational process from an *allocentered* to an *egocentered* one. Morevoer, the impact of the Alzheimer disease is visible before the actual time of diagnostic and is statistically significant considering the areas near the subject residence state. We believe it is a step towards a better understanding of Alzheimer disease with a possible new marker.

8.5 Summary

In this chapter, we have presented how bundling is an efficient tool to support the exploration and analysis of geospatially embedded fluency tests thanks to its capability to reduce clutter in node-link diagrams.

In the context of the study of Alzheimer's disease, our first preliminary analysis validated the database we built from the paper-based fluency test. Moreover, these analysis prove to yield results confirming previous work on the fluency test and more generally on the cognitive loss of Alzheimer's subjects.

However, our first analysis based on attribute derived geographic data did not yield any significant differences between Alzheimer's subjects and control ones. But our extensive analyses of the geospatialization of the fluency maps through bundling proved to bring new insights into the aging process of our two populations. The bundled maps showed a decrease in the geospatial distribution of the city's citations as well as a decrease in their density. These observations suggested the focalization of the citations in the area of residence of the Alzheimer's subject. In other terms, it showed a tendency of AD subjects to shift from an allocentered navigation to an egocentered one.

Here, bundling allowed us to hypothesize a new marker of Alzheimer's disease: the ratio between the citations in the area of residence of a subject and the total amount of citations. Hypothesizing that AD subjects in ratio tend to cite more cities in their states of residence proved to be statistically significant.

To conclude, we strongly believe that our analysis and results paved the way for further extended research on fluency tests and their relationships with cognitive maps. As such, several leads can be followed to confirm of infirm our results. A first potential lead of improvements stands in finding the exact area around subjects city of residence that maximizes the statistical difference between our two groups. Finding this area could give us essential insights into geospatial cognitive levels for our two populations. Secondly, as part of the 3C study (Antoniak et al., 2003), we could digitize and analyze the fluency tests of other subjects that live in other cities (*e.g. Dijon* – north-east of France or *Montpellier* – south-west). The study of other city fluency tests could infer (or not) the validity of our results and especially the relevance of our potential marker of Alzheimer's disease. Overall, the vast possibility of extension and improvement of our work could yield some major societal benefits. As such, if our initial results on fluency tests are confirmed by new study, we could potentially allow general medical practitioners to easily tests their patient.

Chapter 9

Conclusion

This dissertation set out aiming to bridge the gap between the technical complexity of bundling methods and the end-point users. To answer this question, we explored the place of bundling as a information visualization technique following the InfoVis pipeline (Card, Mackinlay, and Shneiderman, 1999, figure 9.1);

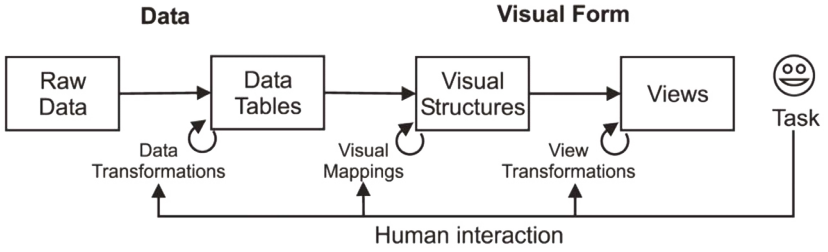


FIGURE 9.1 – InfoVis pipeline according to Card, Mackinlay, and Shneiderman, 1999.

We started by investigating the input *raw data* bundling algorithm works on (part I). Then, we explored the specific techniques and implementation to transform *raw data* into *views* via the *data tables* and *visual structures*. We showed that bundling techniques implementations of the *data tables*, *visual structures* and *views* steps of the InfoVis pipeline can be unified trough our proposed bundling pipeline (introduced in chapter 4). Moreover, we improved the ability for bundling techniques to cope with increasingly large and multi-dimensional data-sets (chapter 6). Finally, we studied both the *tasks* and successful use-cases of bundling (chapter 5).

In parallel, the work presented in this dissertation was set in the context of the study of Alzheimer's disease in a multidisciplinary project called *MEMOIRE*. We reported the application of bundling as a visual tool for cluttered node-link diagram to formalize new hypothesis regarding the disease. Bundling helped neuro-psychologist to find a potential new marker of Alzheimer's disease (chapter 8).

In this final chapter, we summarize the contributions of this dissertation. We conclude this chapter with an outline of future and open research questions regarding bundling as well as our project on the study of Alzheimer's disease.

9.1 Summary and Contributions

Part I: Bundling in the Information Visualization Context

In the first part of this dissertation, we explored the place of bundling techniques in the context of visualization trough two approaches:

Chapter 2: Clutter Reduction for Node-Link Diagrams Chapter 2 gave a first overview of the InfoVis concepts inherent to bundling techniques. We first introduced the various principles and layout methods of the data-sets structures bundling works on. We showed that bundling techniques work on data that are already embedded in a Euclidean space. This can either be a type of graph with its associated drawing, or a trail-set. Next we detailed and expressed existing clutter reduction techniques following the state of the art review by Ellis and Dix, 2007: We have seen the various classes of clutter reduction techniques (*appearance*, *spatial distortion*, or *temporal*), the existing criteria to characterize these techniques, and showed how Ellis and Dix, 2007 compared clutter reduction techniques according to their defined criteria. In the last part of this overview of bundling concepts, we defined principles we deemed necessary to understand them: objectives, requirements and mathematical definition (**Bundling equation**) of bundling techniques.

Chapter 3: A Data-based Taxonomy of Bundling Methods In a second approach, we proposed a taxonomy of bundling methods according to the type of input data-set they work on. This taxonomy was summed up in an *input data to technique* table (see table 3.2) and a visual guide of techniques (see figure 3.15). Following our taxonomy, we showed that bundling methods cover a wide spectrum of data types: graph drawings (trees, compound, DAGs, general oriented or not); 2D trails (vehicle movements, eye tracks, streamlines); and 3D trails (vehicle movements, DTI fibers, streamlines). All these can be either attributed (with several attributes per path) or not, and time dependent or not. As such, we believe that most data types amenable to bundling are covered by existing methods.

Our data-based taxonomy is a step towards understanding on what type of data bundling techniques can work on. And from a user point of view, our taxonomy shows that it is possible to choose between the wide variety of bundling methods depending on the type of input data they work on and without having to dive into the technicalities and algorithmic details of each bundling methods.

Part II: Unifying, Understanding and Improving Bundling

The second part of this dissertation we worked towards solving challenges of bundling techniques through three chapters. The first two chapters solved the challenges of comparing and choosing different bundling algorithms. Then we tackled the computational and scalability challenges of bundling techniques.

Chapter 4: Towards a Unified Bundling Framework In this chapter we proposed a unified bundling framework and showed how existing techniques fit into it. Depicted in figure 9.2, our framework has four main steps: *graph layout*, *similarity definition*, *bundling operator* and *visual exploration*. The first step defines how the data is projected and/or embedded into a Euclidean step prior to bundling. Then, bundling techniques define two functions κ and δ, respectively data-based and drawing-based similarity functions. These functions specify which curved edges are compatible with one another in order to bundle them. Then, the bundling of edges *per se* can be computed with various techniques (*e.g.* explicitly via an intermediary structure or implicitly via an iterative process). In the final step of our pipeline, bundled path-sets are rendered visually, enhanced (*e.g.* with alpha blending, color mapping, shading) and explored (*e.g.* via interactive means).

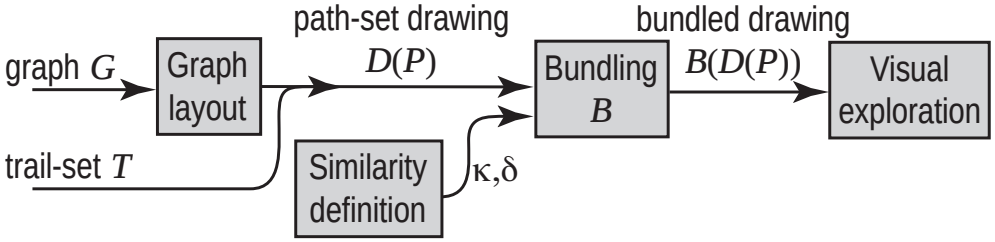


FIGURE 9.2 – Bundling framework steps.

Chapter 5: Bundling Tasks and Applications At the other end of Card, Mackinlay, and Shneiderman, 1999's InfoVis pipeline, we explored the tasks bundling techniques can solve along three axes. First, we defined bundling's clutter reduction capabilities based on the criteria defined by Ellis and Dix, 2007 (presented in chapter 2) and we compared bundling with three clutter reduction techniques (*opacity*, *clustering* and *path displacement*, see table 5.2). Then, we answered the tasks we believe bundling can solve with two well-known task taxonomies: A low-level taxonomy by Lee et al., 2006 and a high-level one (Brehmer and Munzner, 2013). Finally, we demonstrated the adoption by various research field of bundling techniques via a wide set of key applications in various domains: software engineering and data mining, vehicle trajectory analysis, eye-track analysis, multi-dimensional data exploration and vector and tensor fields.

We believe that in conjunction with the two previous chapters, we have systematized and clarified several so-far open points in the bundling literature. This will serve both practitioners in understanding how to choose, parameterize, and use bundling techniques to solve concrete problems and also researchers to compare, discuss, understand, and refine future bundling algorithms.

Chapter 6: Improving Bundling Scalability In a context of ever growing size and multi-dimensionality of data-set coupled with the inherent motive of visualization to be as interactive as possible, we proposed a new bundling method able to cope with

very large attribute-oriented graphs: FFTEB (figure 9.3). FFTEB's key asset is its scalability, both in size of the input graph (number of edges or edge sampling-points) and in the resolution of the final image. FFTEB exploits the properties of the Fast Fourier Transform, and of a GPU streaming design, to bundle graphs which are much larger than what previous state-of-the-art methods could handle, and with less time. Additionally, we showed how FFTEB generalizes attribute-based graph bundling with no computational penalties, something that aforementioned methods cannot do. We demonstrated our approach by comparing FFTEB with nine state-of-the-art methods and using graphs that have up to *billions* of sample points – an order of magnitude that no current method can handle. Additionally, we showed that the high scalability delivered by FFTEB is essential to generate images which can capture fine details of the underlying datasets, which in turn helps their better understanding.

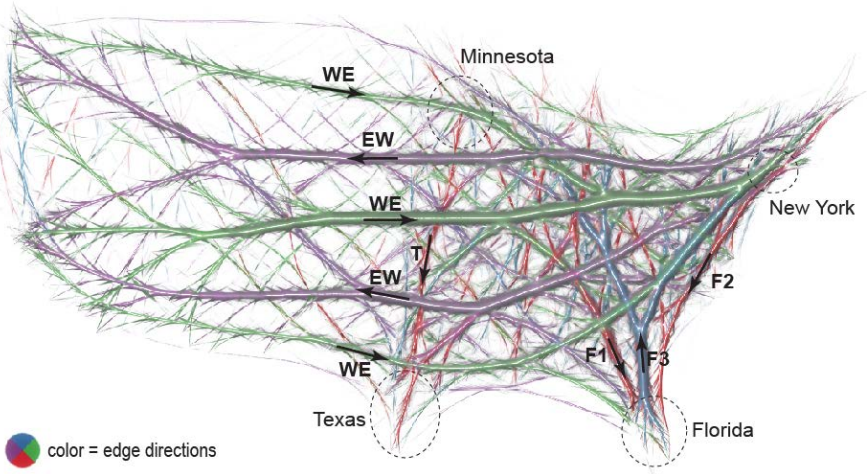


FIGURE 9.3 – FFTEB applied to the large US *migration* graph U.S. Census Bureau, 2003
(600K edges, 63 billion sampling points)

Part III: Application to the Study of Alzheimer Disease

In the last part of the thesis, we showed how bundling was able to contribute to the study of Alzheimer's disease (AD) as an instrument able to produce new perspectives for scientific observations. Using the hand-written results of a French cohort study (3C Antoniak et al., 2003), we studied both the effect of aging and the evolution of Alzheimer's disease on the human brain. We studied how elderly people think their geospatial space (called *cognitive maps* Tolman, 1948) through the analysis of a city fluency tasks (where subjects were asked to cite the maximum number of cities in a minute, Isaacs and Kennie, 1973).

Chapter 7: A Visual Analytics Approach for Digitizing and Cleaning Health Data Prior to the analysis of cognitive maps with bundling, we introduced visual analytics tools to digitize and clean Electronic Health Records (EHR). Our set of tools allowed us to build a correct data-base of our unexploited handwritten data. Thanks to them we gathered a correct database of more 1500 fluency tests for 394 subjects and our

validation showed a reasonable residual error rate of less than 5%. Hence, we believe our proposed set of tools to be one possible answer to the problems of digitization and cleaning of hand-written EHR. Moreover, we believe the cleaned database to be sufficiently correct for further observations and analysis in the following chapter.

Chapter 8: Bundling Cognitive Maps In our last chapter, we presented how bundling was an efficient tool to support the exploration and analysis of geospatially embedded EHR thanks to its clutter reduction capabilities for node-link diagrams. In the context of the study of Alzheimer's disease, our first preliminary analyses validated the database we built from the paper-based fluency test. And these analyses prove to yield results confirming previous work on the fluency test and more generally on the cognitive loss of Alzheimer's subjects. Further explorations and observations of the fluency tests with bundling (figure 9.4) showed a tendency of AD subjects to shift from an allocentered navigation to an egocentered one. Here, bundling allowed us to hypothesize a new marker of Alzheimer's disease: the ratio between the citations in the area of residence of a subject and the total amount of citations. Hypothesizing that AD subject in ratio tend to cite more cities in their states of residence proved to be statistically significant.

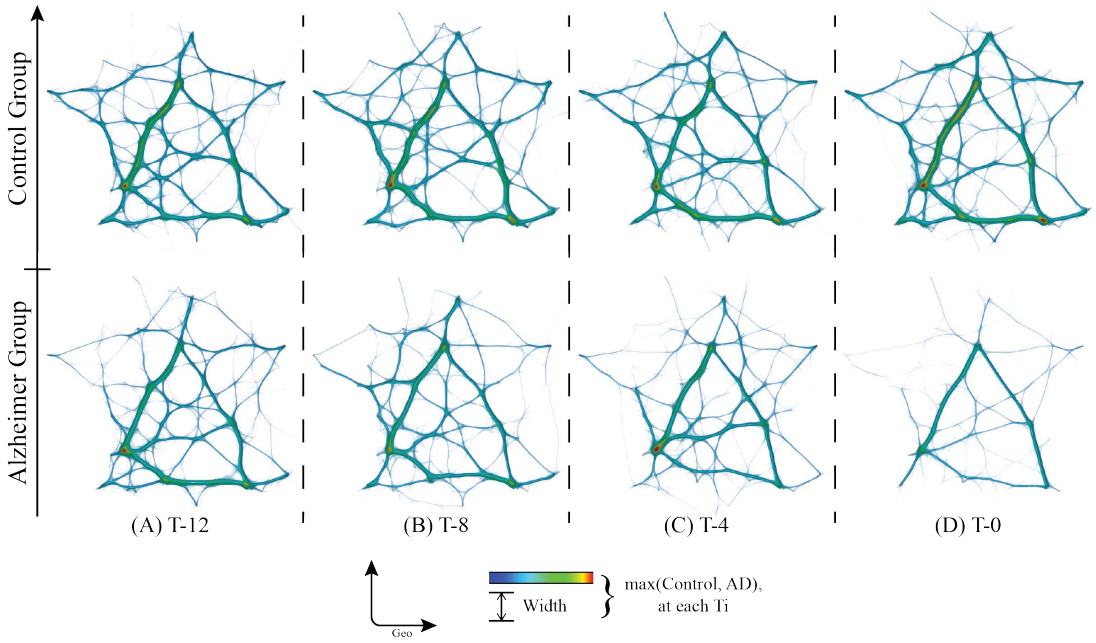


FIGURE 9.4 – Comparison of the bundled fluency tests between a group of subjects that will not develop Alzheimer's disease (control group) and a group of subjects that will develop the disease (Alzheimer group). Subjects in the Alzheimer group are diagnosed as dement (*i.e.* having Alzheimer disease) at $T0$. The bundled images compare the fluency tests starting 12 years before Alzheimer's subjects will develop the disease.

To conclude, we strongly believe that our analysis and results paved the way for further extended InfoVis-oriented research on fluency tests and their relationships with cognitive maps.

9.2 Future Research

Future research areas of the work presented in this dissertation have been described in the last sections of the corresponding chapters. In this section, we sum up the various areas of improvement we identified both related to bundling techniques as well as the study of Alzheimer's disease.

9.2.1 On Bundling Future Challenges

Bundling Control In chapter 4, we paved the way towards a unified technical framework of bundling methods. However, we still lack some design guidelines on how to correctly configure the various parameters of bundling methods. Thus controlling bundling methods is a major technical challenge for the bundling landscape. To solve this, we see the following challenges:

Semantics: Existing parameters have semantics which are typically bound to a specific bundling technique. This makes it hard for users to reproduce results when changing techniques. Our mathematical framework presented in this survey helps this by identifying parameters having identical meanings over different methods, but of course cannot fully solve the problem. A promising direction would be to provide parameters linked to user tasks rather than method technicalities.

Impact: Changing any of the many bundling parameters produces a different result. While reasonable parameter presets exist, these are not always optimal for all datasets and/or visual insights sought. Supporting users to express their interest more effectively, by offering high(er) level parameter control, similar to work done elsewhere in infovis Shrinivasan and Wijk, 2008; Shrinivasan and Wijk, 2009, is a potential improvement direction for bundling. Another solution would be setting parameter values based on the characteristics of the input data $D(P)$ to bundle.

Coupling: The same bundling result can be, usually, obtained by setting different parameters to different values. For example, the same bundle tightness in implicit methods Holten and Van Wijk, 2009; Hurter, Ersoy, and Telea, 2012; Moura, 2015; Zwan, Codreanu, and Telea, 2016 can be obtained by changing either the number of bundling iterations or the advection step size. We call such parameters *coupled*. Few papers discuss coupling, making the parameter space unnecessarily large. This can be alleviated by grouping coupled parameters under a single high-level parameter, similarly to the idea discussed above for impact.

Quality Assessment As discussed in section 5.3, there is no accepted way to measure the quality of a bundling. The core problem is that it is hard to define objective criteria for what a *good* bundling is. Quality metrics have been discussed, and advocated for, since long in information visualization (see the work of Brath, 1997; Miller et al., 1997; Bertini and Santucci, 2006). We believe that quality assessment can be approached by defining what quality is. Following well-established principles in software engineering (Ken, 2003), we can define the quality of a bundled drawing by either measuring its

fitness for purpose (how well it helps solving a certain problem) or by comparing it to a *ground-truth* whose quality is known. Both paths have, however, challenges, outlined in section 5.3 and summed up here after.

Fitness for purpose: This could be solved by large-scale user studies of bundled drawing where the percentage, correctness, and time of completing a given task is measured. However, this encompass the known challenges of generalizing the results of large-scale user studies and the fact that the same bundling methods can generate a wealth of different drawing styles (as further discussed hereafter with the *bundling control* aspect.

Ground truth comparison: This challenge encompass a general challenge of graph drawing; quality metrics. Using quantitative metrics to compare bundled images of different bundling techniques or a raw graph from which we know the ground truth. However, we still need to define such quantitative metrics and/or find a good set of data-set to be used as benchmark.

Bundling Faithfulness A separate issue regards the information alteration or loss produced by bundling, or the so-called faithfulness of the produced drawings (Nguyen, Eades, and Hong, 2013; Nguyen, Eades, and Hong, 2017). Simply put, we need to measure (a) how much of the original information conveyed by the raw image is kept in the bundled image, and (b) how much incorrect information the bundled image adds compared to the original one.

Concerning (a), it is clear that bundling loses some of the information present in the original $D(P)$. To assess this, we can measure the amount of trail distortion created by bundling. Next, depending on the task, we can decide whether this amount is acceptable or not. Further, a ratio of clutter reduction to amount of distortion can be computed, analogously to Tufte's ink-space ratio (Tufte, 1992) to measure the faithfulness of bundling. Another way to increase faithfulness is to animate the image during the bundling (from raw to fully bundled, see Hurter et al., 2014a). This helps users match the input and output of the bundling process, thus reducing the information loss.

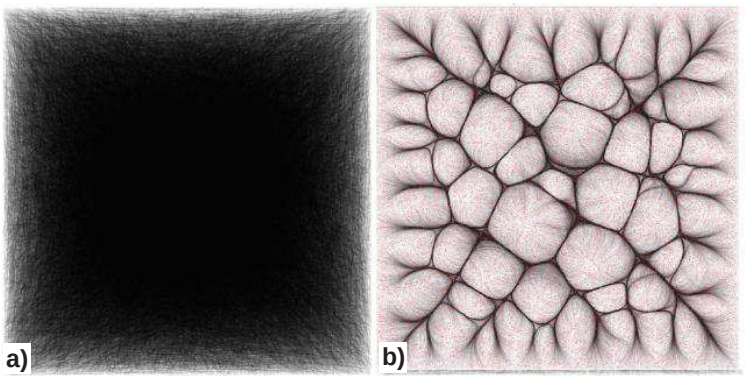


FIGURE 9.5 – Pseudo-random graph (a) and its KDEEB bundling (Hurter, Ersoy, and Telea, 2012) (b). This demonstrate how bundling *can* sometimes yield false insights.

Concerning (b), it has been shown that, for some datasets, bundling *can* yield false insights (as illustrated in figure 9.5 and detailed in section 5.3). In this case, the effect is caused by the fact that bundling methods expect that their input have salient structure, otherwise the bundling process emphasize small-scale noise.

Fast Fourier Transform Edge Bundling Regarding our new bundling method, we believe many possible extensions exist: Using recent results in computation of sparse FFT can accelerate our method one order of magnitude (Wang, Chandrasekaran, and Chapman, 2016), which would make FFTEB cope with most big-data challenges envisable today. Next, FFTEB's efficient attribute-based bundling opens new ways to explore virtually any kinds of edge-compatibility criteria based on any nature and/or number of edge attributes. Finally, the overall generality and scalability of FFTEB makes its extension to interactive 3D bundling a low hanging fruit prospect, with application *e.g.* into medical imaging (Böttger et al., 2014).

9.2.2 On our Application of Bundling to the Study of Alzheimer's Disease

Regarding future works on our application of visualization to the study of Alzheimer, several leads can be followed to confirm or infirm our results.

A first potential lead of improvements stands in finding the exact area around subjects city of residence that maximizes the statistical difference between our two groups. Finding this area could give us essential insights on geospatial cognitive levels for our two populations.

Secondly, as part of the 3C study (Antoniak et al., 2003), we could digitize and analyze the fluency tests of other subjects that live in other cities (*e.g. Dijon* – North-East of France or *Montpellier* – South-East). The study of other city fluency tests could infer (or not) the validity of our results. Especially the relevance of our potential marker of Alzheimer's disease.

Overall, the vast possibility of extension and improvement of our work could yield some major societal benefits. As such, if our initial results on fluency tests are confirmed by new study, we could potentially allow general medical practitioners to easily tests their patient.

On a final and more broader note, our first analysis and results on the study of cognitive maps on Alzheimer's disease raise ethical questions regarding an early diagnosis. Specifically, we can ask ourselves: Should we tell a subject that he will lose his memory decades before it happens? From this ethical question, we may asks ourselves: At what point in his probabilities to develop the disease should we tell a patient? When we are absolutely sure he will develop the disease or when he has a fifty-fifty chance of developing Alzheimer's disease? Here, potential leads for answers could be found in incurable genetic disease that medecine can already diagnose early on such as Huntington's disease (Walker, 2007).

9.3 Conclusion (Français)

Dans cette thèse, nous avons étudié les moyens de combler les lacunes entre la complexité technique des algorithmes de bundling et les utilisateurs finaux. Pour y répondre, nous avons ainsi exploré la place du bundling comme technique de visualisation d'information en suivant la boucle de rendu de l'InfoVis (Card, Mackinlay et Shneiderman, 1999, figure 9.6) ;

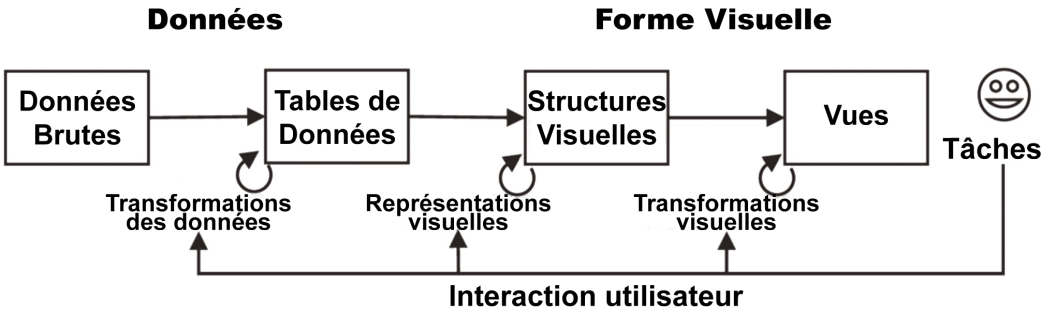


FIGURE 9.6 – Boucle de rendu de la visualisation d'informations d'après Card, Mackinlay et Shneiderman, 1999.

Ce travail a débuté par l'investigation des structures des *données d'entrée* utilisables par le bundling (partie I). Puis, nous avons exploré les spécificités techniques et les implémentations des différentes méthodes de bundling permettant de transformer les données d'entrée en vues à travers différentes *tables de données*. Nous avons montré qu'il était possible d'unifier les différentes implémentations du bundling des étapes du pipeline InfoVis (*tables de données*, *structures visuelles* et *vues*) à travers notre pipeline graphique générique (introduit en chapitre 4). De plus, nous avons amélioré les capacités du bundling à traiter de larges quantités de données multidimensionnelles (chapitre 6). Enfin, nous avons étudié les *tâches* et les cas d'utilisation avérés du bundling (chapitre 5).

En parallèle, les travaux présentés dans cette thèse se sont insérés dans le contexte de l'étude de la maladie d'Alzheimer à travers le projet interdisciplinaire *MÉMOIRE*. Dans cette thèse, nous rapportons l'utilisation du bundling comme outil de visualisation des diagrammes nœuds-liens à forte densité de liens ayant permis aux neuropsychologues de formuler de nouvelles hypothèses sur la maladie. Dans notre cadre, le bundling à aidé les neuropsychologues à trouver un potentiel nouveau marqueur de la maladie d'Alzheimer (chapitre 8).

Dans ce chapitre final, nous résumons l'ensemble des contributions de cette thèse. Nous concluons ce chapitre par une ouverture sur les futures questions de recherche concernant le bundling mais aussi sur l'étude de la maladie d'Alzheimer.

9.3.1 Résumé et Contributions

Partie I : Le Bundling dans le Contexte de la Visualisation d'Informations

Dans la première partie de cette thèse, nous avons exploré la place du bundling dans le contexte de la visualisation d'informaitons à travers deux approches :

Chapitre 2 : Réduire l'Occultation des Diagrammes Nœuds-Liens Le chapitre 2 est une première vue d'ensemble des concepts de la visualisation d'information inhérents aux techniques de bundling. Nous avons d'abord introduit les divers principes et méthodes de dessins liés aux type de données sur lesquelles le bundling s'applique. Nous avons montré que les techniques de bundling s'appliquent à des données qui s'inscrivent dans un espace Euclidien. Il peut s'agir d'un graphe et son dessin associé ou d'un jeu de trajectoires. Puis, nous avons présenté et détaillé les techniques de réductions d'occultation de dessins en suivant l'état de l'art de Ellis et Dix, 2007. Nous avons ainsi abordé plusieurs classes de techniques de réduction d'occultation (*apparence*, *distorsion spatiale*, ou *temporalité*), les critères existants permettant de caractériser ces techniques et avons montré comment Ellis et Dix, 2007 comparent les techniques de réduction d'occultation en fonction des critères définis. Dans la dernière partie de ce chapitre, nous avons définis les principes nécessaires à la compréhension des techniques de bundling : les objectifs, les besoins et notre définition mathématique (**Équation du bundling**) des techniques de bundling.

Chapitre 3 : Taxonomie des Méthodes de Bundling Dans une seconde approche, nous avons proposé une taxonomie des méthodes de bundling fondée sur le type de structure de données d'entrée à traiter. Nous avons résumé cette taxonomie dans un tableau (*données vers technique de bundling* – table 3.1) ainsi qu'un guide visuel des techniques (voir figure 3.1). En suivant notre taxonomie, nous avons montré que les méthodes de bundling couvrent un large spectre de structures de données : dessins de graphes (arbres, cycles, graphes directionnels acycliques, graphes génériques orientés ou non), trajectoires bidimensionnelles (mouvement de véhicules, tracés oculaires, iso-lignes) et tridimensionnelles (mouvements de véhicules, fibres DTI). Toutes ces structures d'entrée peuvent être agrégées selon des compatibilités fondées sur un ou plusieurs attributs ou non, et leurs dépendances temporelles ou non. Ainsi, nous pensons que la plupart des structures potentiellement utilisables pour réaliser de la simplification visuelle par agrégation de liens sont couvertes par les méthodes existantes et détaillées par notre taxonomie.

Notre taxonomie orientée données d'entrée est une étape permettant de comprendre sur quels types de données le bundling s'applique. D'un point de vue utilisateurs, notre taxonomie montre qu'il est possible de choisir une technique parmi la grande diversité des algorithmes de bundling en fonction du type de données à traiter sans avoir à plonger dans les détails de chaque méthode de bundling.

Partie II : Unifier, Comprendre et Améliorer les Techniques de Bundling

La seconde partie de ce manuscrit s'est inscrite dans un travail de recherche visant à résoudre les défis des techniques de bundling à travers trois chapitres. Les deux premiers chapitres ont résolu les défis liés à la comparaison et au choix des différents algorithmes de bundling existants. Puis nous nous sommes attaqués aux défis de la complexité algorithmique et du passage à l'échelle des techniques de bundling.

Chapitre 4 : Un Pipeline Graphique Unifié des Méthodes de Bundling Dans ce chapitre, nous avons proposé un pipeline graphique unifié et avons montré comment les techniques de bundling existantes s'intégraient dans ce pipeline. Représenté en figure 9.7, notre pipeline possède quatre étapes : *tracé d'un graphe*, *définition des similarités*, *opérateur d'agrégation* et *exploration visuelle*. Les deux premières étapes définissent comment les données doivent être projetées ou définies dans un espace Euclidien avant l'étape d'agrégation. Puis, les techniques de bundling définissent deux fonctions de similarité κ (basée données) et δ (basée dessin). Ces fonctions servent à spécifier la compatibilité entre chemins avant de les agréger en faisceau (bundle). Grâce à ses fonctions, l'opérateur d'agrégation peut être appliqué aux différents chemins grâce à diverses approches (*e.g.* explicitement à l'aide d'une structure intermédiaire ou implicitement par un processus d'itérations). Dans l'étape finale de notre pipeline, les chemins agrégés (ou faisceaux) sont visuellement représentés (dessinés sur l'écran), améliorés à l'aide de méthodes de rendu (*e.g.* alpha blending, color mapping, shading) et explorés (*via* des moyens d'interactions).

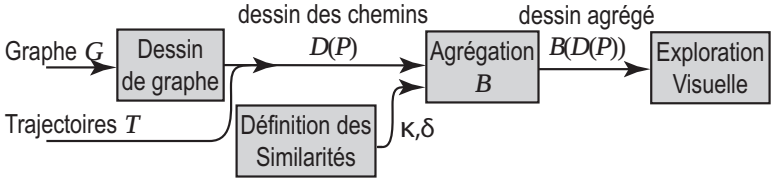


FIGURE 9.7 – Pipeline graphique des méthodes de bundling.

Chapitre 5 : Tâches et Applications du Bundling De l'autre côté du pipeline de l'InfoVis de Card, Mackinlay et Shneiderman, 1999, nous avons exploré les tâches que le bundling peut résoudre selon trois axes. Tout d'abord, nous avons défini les capacités du bundling pour réduire l'occultation des visualisations de type nœuds-liens en fondant notre analyse sur les critères introduits par Ellis et Dix, 2007 et avons comparé le bundling avec trois techniques de réduction d'occultation similaires (*opacité*, *regroupement* et *déplacement de liens*, voir tableau 5.1). Puis, nous avons présenté les tâches de visualisation propres au bundling à travers deux taxonomies de tâches reconnues : une taxonomie bas niveau introduite par Lee et al., 2006 et une taxonomie haut niveau (Brehmer et Munzner, 2013). Enfin, nous avons démontré l'adoption des techniques de bundling par plusieurs champs de recherche à l'aide d'un large éventail de cas d'utilisations avérés dans plusieurs domaines : ingénierie logicielle et la fouille de données,

l'analyse des trajectoires de véhicules, l'analyse de tracés oculaires, l'exploration de données multidimensionnelles et l'agrégation de champs de vecteurs et de tenseurs.

En conjonction avec les deux précédents chapitres, nous pensons que ce travail à systématisé et clarifié plusieurs questions et pistes de recherche de la littérature propre au bundling. Ce travail va ainsi permettre aux utilisateurs de savoir comment choisir, paramétrer et utiliser les techniques de bundling pour résoudre des problèmes concrets et permettre aux chercheurs de comparer, discuter, comprendre et améliorer les futurs algorithmes de bundling.

Chapitre 6 : Une Nouvelle Technique de Bundling : FFTEB Dans un contexte d'augmentation constante de la taille et de la multi-dimensionnalité des jeux de données à traiter couplé avec le désir constant d'avoir des visualisations aussi interactives que possible, nous avons proposé une nouvelle technique de bundling capable d'agréger de très large quantités de données multidimensionnelles : FFTEB (figure 9.8). Le principal atout de notre technique réside dans sa capacité de passage à l'échelle, tant sur le plan de la taille des données d'entrée (nombre de chemins agrégeables) que sur la grande résolution de l'image finale des chemins agrégés. FFTEB utilise les propriétés de la transformée de Fourier rapide (Fast Fourrier Transform – FFT) et sur un algorithme de transfert de données (*streaming*) en mémoire graphique pour agréger des jeux de données qu'aucune autre technique de bundling n'avait encore pu agréger et ce de manière plus rapide.

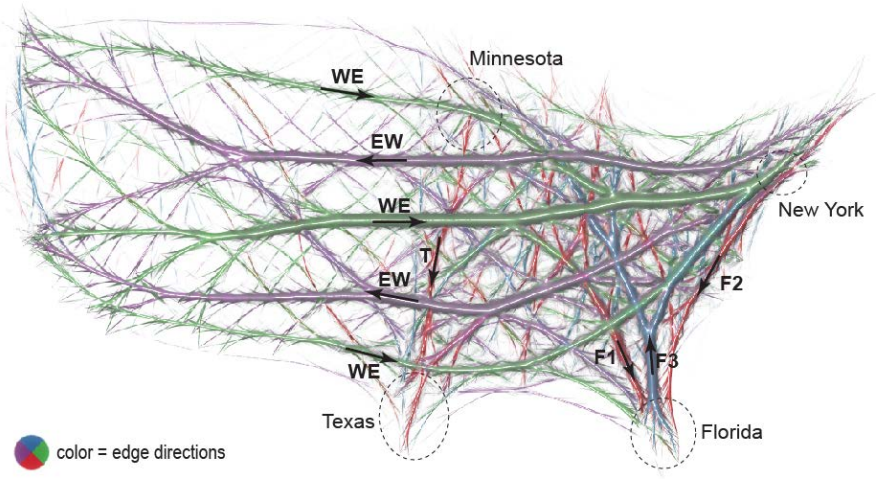


Figure 9.8 – Application de notre technique de bundling, FFTEB, au graphe de migration de populations aux Etats-Unis, U.S. Census Bureau, 2003. (600K liens, 63 millions de points)

De plus, nous avons montré comment FFTEB permet de généraliser l'agrégation de liens en fonction de n'importe quels attributs d'un jeu de chemins sans pour autant augmenter la complexité algorithmique, ce que les précédentes méthodes de bundling ne pouvaient pas faire. Nous avons démontré la validité de notre approche en la comparant avec neuf autres techniques de bundling reconnues et à l'aide de jeux de données contenant jusqu'à 1 million de chemins – un ordre de grande qu'aucune autre méthode

n'a pu atteindre. Enfin, nous avons montré que les capacités de passage à l'échelle de FFTEB étaient essentielles pour générer des images avec une résolution suffisante capable de montrer les détails sous-jacents de certains jeux de données, ce qui en retour a amélioré notre compréhension des données.

Partie III : Application à l'Étude de la Maladie d'Alzheimer

Dans la dernière partie de cette thèse, nous avons montré comment le bundling pouvait contribuer à l'étude de la maladie d'Alzheimer comme outil d'observation pour produire de nouvelles perspectives pour la visualisation scientifique. A l'aide des résultats manuscrits tirés de la cohorte 3C (Antoniak et al., 2003), nous avons étudié à la fois les effets du vieillissement et l'évolution de la maladie d'Alzheimer sur le cerveau humain. Nous avons étudié la manière dont les personnes âgées se représentent mentalement l'espace géographique (aussi appelé cartes cognitives Tolman, 1948) à travers l'analyse d'une tâche de fluence verbale (au cours de laquelle les sujets ont dû citer le plus de villes possibles en une minute, Isaacs et Kennie, 1973).

Chapitre 7 : Une Méthode d'Analyse Visuelle pour la Numérisation de Données de Santé Afin de pouvoir analyser les cartes cognitives à l'aide de bundling, notre première étape a consisté à fournir des outils de visualisation analytique permettant de numériser et nettoyer les données de santé sur papier. Nos outils nous ont ainsi permis de construire une base de données précises des données papier inexploitées auparavant. Grâce à eux, nous avons rassemblé une base de données contenant plus de 1500 tests de fluence verbale pour 394 patients et notre étape de correction et validation a montré un taux d'erreurs résiduel inférieur à 5%. De part ces résultats, nous pensons que les outils proposés créent de nouvelles pistes pour résoudre les problèmes liés à la numérisation et le nettoyage de données de santé manuscrites. De plus, notre faible taux d'erreur résiduel nous permet d'affirmer avoir construit une base de données suffisamment correcte pour faire des observations et des analyses de cette dernière.

Chapitre 8 : Application du Bundling aux Cartes Cognitives Dans le dernier chapitre, nous avons montré en quoi le bundling était un outil capable de supporter l'exploration et l'analyse de données de santé géospatialisées grâce à ses capacités de réduction d'occultation. Dans le contexte de l'étude de la maladie d'Alzheimer, nos premières analyses statistiques ont validé l'exactitude de notre base de données de tests de fluence. Et ces analyses ont aussi montré et confirmé des résultats précédents relatifs aux impacts cognitifs de la maladie d'Alzheimer. De plus amples observations et explorations des tests de fluence à l'aide du bundling (voir figure 9.9) ont montré une tendance des sujets atteints de la maladie à modifier leurs processus de navigation spatiale depuis un point de vue allocentré vers un point de vue égocentré. Dans ce cas, le bundling nous a permis de formuler un nouveau marqueur de la maladie d'Alzheimer : le ratio entre le nombre de villes citées dans les environs de la ville de résidence d'un sujet et le nombre de villes citées au total. Ce marqueur nous a permis de montrer de façon statistiquement significative que les sujets qui vont développer la maladie ont tendance

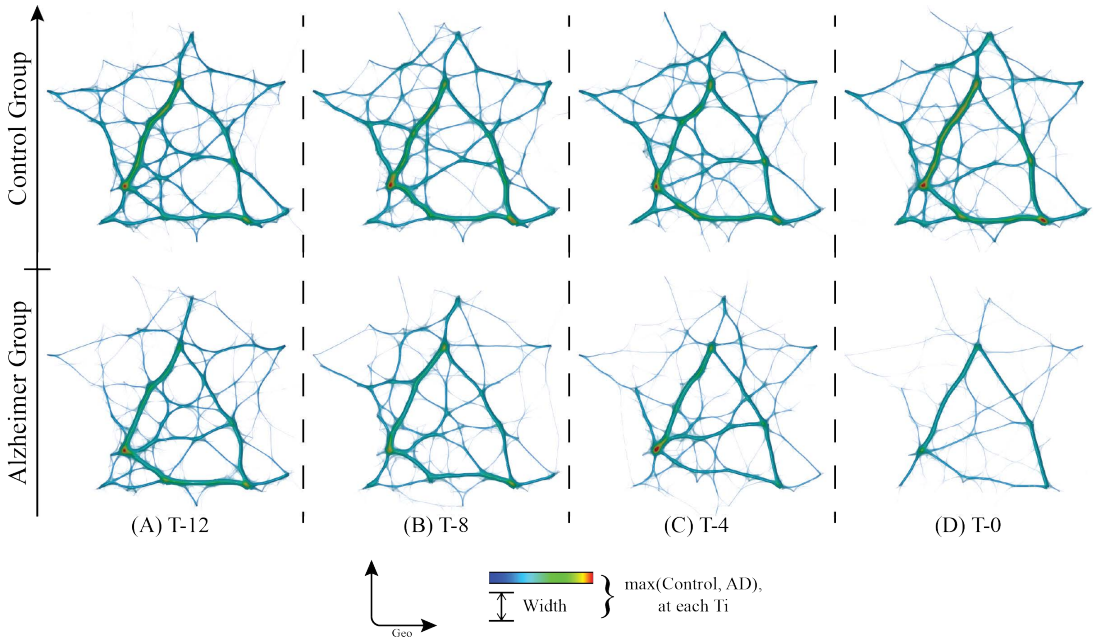


FIGURE 9.9 – Comparaison de l'agrégation des tests de fluence de villes entre un groupe de sujets qui ne seront pas atteints de la maladie d'Alzheimer (groupe contrôle) et un groupe de sujets qui va développer la maladie (groupe Alzheimer). Les sujets du groupe Alzheimer sont diagnostiqués comme déments (*i.e.* officiellement atteints par la maladie) à $T0$. Les images agrégées comparent les tests de fluences verbales près de 12 ans avant l'appariation officielle de la maladie.

à citer plus de villes situées dans la zone de leur ville de résidence que les autres sujets. Ce marqueur est statistiquement significatif près de 12 ans avant le diagnostic officiel.

En conclusion, nous pensons fermement que notre analyse et nos résultats ont ouvert la voie pour étendre la recherche sur les tests de fluences verbales et leurs relations avec les cartes cognitives à l'aide d'approches orientées visualisation d'informations.

9.3.2 Perspectives de Recherche

Les perspectives de recherche des travaux présentés dans cette thèse ont été décrites dans les dernières sections des chapitres correspondant. Néanmoins, nous résumons ici les diverses perspectives identifiées au cours de nos travaux tant sur les techniques de bundling que sur l'étude de la maladie d'Alzheimer.

Perspectives de Recherches relatives au Bundling

Contrôle de l'Agrégation Le chapitre 4 a ouvert la voie vers un cadre d'application unifié des méthodes de bundling. Cependant, le bundling ne dispose pas encore de directives précises quant à la configuration et à la paramétrisation des algorithmes de bundling. Ainsi, savoir contrôler et mesurer la quantité d'agrégation à appliquer sur un jeu de chemins est un des défis majeurs du bundling. Pour répondre à ces défis, nous proposons les pistes suivantes :

La sémantique : Les paramètres d'agrégation sont typiquement liés à un algorithme de bundling spécifique. Ce problème de sémantique des paramètres rend difficile la compréhension et la reproductibilité des résultats par les utilisateurs lors de l'utilisation de différentes techniques. Notre définition formelle du bundling permet jusqu'à une certaine mesure de résoudre ce problème en identifiant les paramètres ayant des sémantiques similaires entre différentes méthodes. Cependant, cela ne résout pas complètement le problème. Une piste intéressante serait de proposer des paramètres d'agrégation ayant une sémantique liée aux tâches de l'utilisateur plutôt que d'être liée aux spécificités techniques de l'algorithme de bundling.

L'impact : Le changement d'un des nombreux paramètres d'un algorithme de bundling produit généralement un résultat agrégé visuellement très différent. S'il existe des réglages prédéfinis pour certains paramètres, ces derniers ne sont pas toujours optimaux pour tous les jeux de données ou la visualisation désirée. Permettre aux utilisateurs d'exprimer leurs objectifs d'agrégation plus efficacement, offrir un meilleur niveau de paramétrisation (de façon similaire à de précédents travaux de l'InfoVis Shrinivasan et Wijk, 2008 ; Shrinivasan et Wijk, 2009) serait un axe d'amélioration potentiel pour le bundling. Une autre solution pourrait résider dans le réglage des paramètres en fonction des caractéristiques du jeu de données ($D(P)$) à agréger.

Le couplage : Le même résultat visuel d'agrégation peut souvent être obtenu en réglant les paramètres de différentes manières. Pour exemple, dans le cas des méthodes implicites (Holten et Van Wijk, 2009 ; Hurter, Ersoy et Telea, 2012 ; Moura, 2015 ; Zwan, Codreanu et Telea, 2016), le même résultat d'agrégation peut être obtenu soit en changeant le nombre d'itérations soit en augmentant la force d'advection des chemins. De tels paramètres sont donc couplés. Peu de travaux discutent des effets du couplage des paramètres dans le bundling, ce qui rend l'espace de paramétrisation des algorithmes de bundling inutilement large. Ceci pourrait être évité en groupant les paramètres couplés à travers un seul paramètre haut-niveau.

Évaluation de la Qualité Comme discuté en section 5.3, il n'y a pas de méthode acceptée pour mesurer la qualité d'une visualisation agrégée. Le principal problème découle de la difficulté à définir des critères objectifs sur ce qu'est une agrégation *correcte*. Les métriques de qualité ont été discutées et défendues depuis longtemps dans le domaine de la visualisation d'information (*c.f.* Brath, 1997 ; Miller et al., 1997 ; Bertini et Santucci, 2006). Pour répondre aux questions d'évaluation de la qualité, il nous faut d'abord définir ce qu'est la qualité d'une visualisation. En suivant des principes reconnus de l'ingénierie logicielle (Ken, 2003), nous pourrions définir la qualité d'un jeu de données agrégées en mesurant son *aptitude à la tâche* (de quelle manière le bundling aide à résoudre certains problèmes) ou en comparant la visualisation avec un jeu de données agrégées de référence dont la qualité est connue. Ces pistes possèdent néanmoins d'autres défis à relever auparavant (explicités en section 5.3).

La Véracité des Informations Visuellement Agrégées Un autre défi du bundling concerne l'altération ou la perte d'informations liées à l'agrégation visuelle des chemins, ou la véracité des informations visuellement agrégées (Nguyen, Eades et Hong, 2013 ; Nguyen, Eades et Hong, 2017). En d'autres termes, il faut mesurer (a) quelle quantité d'information présente dans l'image initiale est encore présente dans l'image agrégée, et (b) quelle quantité d'informations erronées l'agrégation ajoute-t-elle par rapport à l'image initiale.

Concernant le premier point (a), il est clair que l'agrégation visuelle engendre une perte des informations présentes dans le jeu de chemins original ($D(P)$. Pour évaluer cette perte, nous pourrions mesurer la quantité de distorsions des chemins provoquées lors de l'agrégation. Ensuite, en fonction de la tâche utilisateur, il serait envisageable de spécifier une quantité de distorsion maximale acceptable pour la tâche à réaliser. De plus, de manière similaire au ratio encre/fond introduit par Tufte, 1992, la véracité de l'agrégation visuelle pourrait se mesure par un ratio réduction d'occultation par quantité de distorsion. Enfin, un autre moyen pour augmenter la véracité des résultats du bundling aux yeux des utilisateurs consisterait à ajouter de l'animation lors de l'agrégation. Ceci permettrait alors aux utilisateurs de visualiser l'agrégation des liens petit à petit (Hurter et al., 2014a), renforçant ainsi le sentiment de véracité de la visualisation.

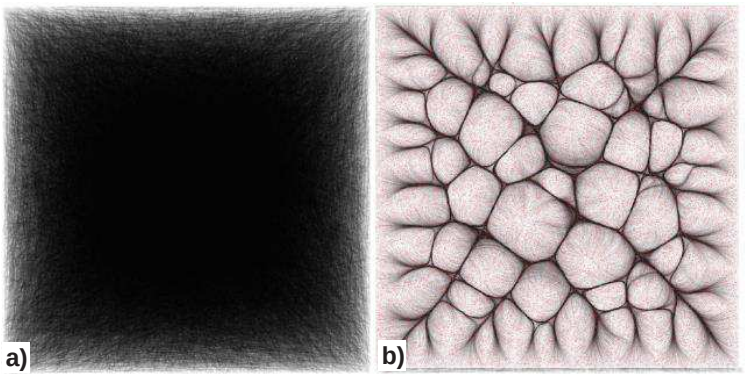


FIGURE 9.10 – Graphe pseudo-aléatoire (a) et sa version agrégée avec KDEEB (Hurter, Ersoy et Telea, 2012) (b). Cette visualisation montre comment le bundling peut *parfois* générer de fausse structures.

Pour le second point (b) des travaux ont montré, que pour certains jeux de données, l'agrégation visuelle *peut* introduire de fausses informations (comme illustré en figure 9.10). Dans ce cas, la création de fausses informations vient du fait que les méthodes de bundling s'attendent à ce que les données d'entrée contiennent des structures suffisamment saillantes pour pouvoir les faire ressortir, dans le cas inverse l'agrégation fait ressortir le bruit parasite.

FFTEB Au sujet de notre nouvelle méthode de bundling, nous pensons que plusieurs pistes d'amélioration existent : par exemples, en utilisant les récents résultats de calcul des transformées de Fourier éparses (*sparse FFT*) nous pourrions accélérer d'un ordre

de grandeur la vitesse de calcul de notre méthode (Wang, Chandrasekaran et Chapman, 2016). Ceci permettrait à notre technique de traiter encore plus rapidement de très grandes quantités de données. Ensuite, notre optimisation de l'agrégation basée attribut ouvre la voie vers l'exploration de n'importe quel type de compatibilité entre chemins basés sur n'importe quel type et nombre d'attributs. Enfin, la généralisation et le passage à l'échelle de notre technique rend son extension à l'agrégation de données en trois dimensions très facile avec de vastes champs d'application et notamment dans l'imagerie médicale (Böttger et al., 2014).

Pistes de Recherches Relatives à notre Étude sur la Maladie d'Alzheimer

Concernant les pistes de recherche relatives à notre application du bundling à l'étude de la maladie d'Alzheimer, plusieurs pistes pourraient être suivies pour confirmer ou informer nos résultats.

Une des premières pistes d'amélioration potentielles consisterait à trouver l'aire exacte autour de la ville de résidence d'un sujet de telle sorte que cette aire maximise la différence statistique entre nos deux groupes de sujets (contrôle et Alzheimer). Trouver cette aire pourrait nous donner plus d'informations quant aux niveaux de géospatialisation cognitive (Meilinger, 2008) de nos populations.

La seconde piste de recherche possible découle de la cohorte sur laquelle notre étude c'est fondée, la cohorte 3C. Ainsi, nous pourrions envisager de numériser et analyser les tests de fluence verbale d'autres sujets vivant dans d'autres villes de France (*e.g.* Dijon ou Montpellier). L'étude de nouveaux tests de fluence verbale pourrait confirmer ou infirmer la validité de nos résultats. Et plus spécifiquement la validité de notre potentiel nouveau marqueur de la maladie d'Alzheimer.

Globalement, le vaste champ de possibilités et d'améliorations de nos premières analyses pourrait apporter des bénéfices majeurs pour la société. Ainsi, si nos résultats initiaux étaient confirmés, nous pourrions potentiellement demander à des médecins généralistes de réaliser ce test simple sur leurs patients.

Pour conclure sur une note plus générale, nos premières analyses et résultats sur l'étude de la maladie d'Alzheimer soulèvent potentiellement des questions éthiques. Plus spécifiquement, cela soulève la questions suivante : devrions nous dire à un sujet qu'il risque de perdre ses facultés cognitives des décennies avant que cela ne se produise ? De cette large question, découle les questions suivantes : à quel stade de probabilité de développer la maladie devrions nous informer le patient ? Quand nous sommes absolument sûrs qu'il va développer la maladie d'Alzheimer ou quand il a une chance sur deux de la développer ? Ici, nous pourrions trouver des pistes de réponses potentielles dans les maladies génétiques incurables mais pouvant être diagnostiquées précocement comme la maladie de Huntington (Walker, 2007).

Bibliography

Abello, J., F. Van Ham, and N. Krishnan (2006). "Ask-graphview: A large scale graph visualization system". In: *IEEE transactions on visualization and computer graphics* 12.5, pp. 669–676.

Ahlberg, C. and B. Shneiderman (1994). "Visual information seeking: Tight coupling of dynamic query filters with starfield displays". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, pp. 313–317.

Ahlberg, C., C. Williamson, and B. Shneiderman (1992). "Dynamic queries for information exploration: An implementation and evaluation". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, pp. 619–626.

Alexander, A. L. et al. (2007). "Diffusion tensor imaging of the brain". In: *Neurotherapeutics* 4.3, pp. 316–329.

Alperovitch, A et al. (2002). "Epidemiological studies on aging in France: from the PAQUID study to the Three-City study". In: *Comptes rendus biologies* 325.6, pp. 665 –672.

Amieva, H. et al. (2008). "Prodromal Alzheimer's disease: successive emergence of the clinical symptoms". In: *Annals of neurology* 64.5, pp. 492–498.

Antoniak, M. et al. (2003). "Vascular factors and risk of dementia: design of the Three-City Study and baseline characteristics of the study population". In: *Neuroepidemiology* 22.6, pp. 316–325.

Archambault, D., T. Munzner, and D. Auber (2008). "GrouseFlocks: Steerable exploration of graph hierarchy space". In: *IEEE transactions on visualization and computer graphics* 14.4, pp. 900–913.

Archambault, D., H. C. Purchase, and B. Pinaud (2010). "The Readability of Path-Preserving Clusterings of Graphs". In: *Computer Graphics Forum*. Vol. 29. 3. Wiley Online Library, pp. 1173–1182.

Assaf, Y. and O. Pasternak (2008). "Diffusion tensor imaging (DTI)-based white matter mapping in brain research: a review". In: *Journal of molecular neuroscience* 34.1, pp. 51–61.

Auber, D. et al. (2017). "TULIP 5". In: *Encyclopedia of Social Network Analysis and Mining*. Ed. by R. Alhajj and J. Rokne. `tulip.labri.fr`. Springer new-York, pp. 1–28.

Auriacombe, S et al. (2010). "Validity of the Free and Cued Selective Reminding Test in predicting dementia The 3C Study". In: *Neurology* 74.22, pp. 1760–1767.

Bach, B. et al. (2017). "Towards unambiguous edge bundling: Investigating confluent drawings for network visualization". In: *IEEE transactions on visualization and computer graphics* 23.1, pp. 541–550.

Barateiro, J. and H. Galhardas (2005). "A survey of data quality tools." In: *Datenbank-Spektrum* 14.15-21, p. 48.

Bastian, M., S. Heymann, M. Jacomy, et al. (2009). "Gephi: an open source software for exploring and manipulating networks." In: *Icwsm* 8, pp. 361–362.

Battista, G. D. et al. (1998). *Graph drawing: algorithms for the visualization of graphs*. Prentice Hall PTR.

Beatty, W. W. and A. I. Tröster (1987). "Gender differences in geographical knowledge". In: *Sex Roles* 16.11, pp. 565–590.

Beck, F. et al. (2014). "The state of the art in visualizing dynamic graphs". In: *EuroVis STAR* 2.

Bederson, B. B., B. Shneiderman, and M. Wattenberg (2002). "Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies". In: *AcM Transactions on Graphics (TOG)* 21.4, pp. 833–854.

Bensalem, I. and M. K. Kholladi (2010). "Toponym disambiguation by arborescent relationships". In: *Journal of Computer Science* 6.6, p. 653.

Berg, M. de et al. (2010). *Computational Geometry: Algorithms and Applications*. Springer.

Bertin, J. (1981). *Graphics and graphic information processing*. Walter de Gruyter.

Bertin, J. (1983). *Semiology of graphics: diagrams, networks, maps*. University of Wisconsin press.

Bertini, E. and G. Santucci (2006). "Visual quality metrics". In: *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*. ACM, pp. 1–5.

Binucci, C. et al. (2009). "Drawing trees in a streaming model". In: *International Symposium on Graph Drawing*. Springer, pp. 292–303.

Blascheck, T. et al. (2014). "State-of-the-art of visualization for eye tracking data". In: *Proceedings of EuroVis*. Vol. 2014.

Bose, R. J. C., R. S. Mans, and W. M. van der Aalst (2013). "Wanna improve process mining results?" In: *Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium on*. IEEE, pp. 127–134.

Bostock, M. (2017). *Hierarchical Edge Bundling Implementation in D3*. `https://gist.github.com/mbostock`.

Bothorel, G., M. Serrurier, and C. Hurter (2013). "Visualization of frequent itemsets with nested circular layout and bundling algorithm". In: *International Symposium on Visual Computing*. Springer, pp. 396–405.

Böttger, J. et al. (2014). "Three-dimensional mean-shift edge bundling for the visualization of functional connectivity in the brain". In: *IEEE transactions on visualization and computer graphics* 20.3, pp. 471–480.

Bourqui, R. et al. (2016). "Multilayer graph edge bundling". In: *Pacific Visualization Symposium (PacificVis), 2016 IEEE*. IEEE, pp. 184–188.

Brandes, U. (2005). *Network analysis: methodological foundations*. Vol. 3418. Springer Science & Business Media.

Brandes, U. and D. Wagner (1998). "Using Graph Layout to Visualize Train Interconnection Data". In: *Graph Drawing*. Springer, pp. 44–56.

Brath, R. (1997). "Metrics for effective information visualization". In: *Information Visualization, 1997. Proceedings., IEEE Symposium on*. IEEE, pp. 108–111.

Brehmer, M. and T. Munzner (2013). "A multi-level typology of abstract visualization tasks". In: *IEEE Transactions on Visualization and Computer Graphics* 19.12, pp. 2376–2385.

Brewer, C., M. Harrower, and T. P. S. University (2015). *Color Brewer*. 5 Class color scheme from red to green. URL: `http://colorbrewer2.org/?type=diverging&scheme=RdYlGn&n=5#type=diverging&scheme=RdYlGn&n=5`.

Buchin, K., B. Speckmann, and K. Verbeek (2011a). "Angle-Restricted Steiner Arborescences for Flow Map Layout." In: *ISAAC*. Springer, pp. 250–259.

Buchin, K., B. Speckmann, and K. Verbeek (2011b). "Flow map layout via spiral trees". In: *IEEE transactions on visualization and computer graphics* 17.12, pp. 2536–2544.

Burch, M. et al. (2011). "Evaluating Partially Drawn Links for Directed Graph Edges." In: *Graph Drawing*. 7034. Springer, pp. 226–237.

Buscaldi, D. and P. Rosso (2008). "A conceptual density-based approach for the disambiguation of toponyms". In: *International Journal of Geographical Information Science* 22.3, pp. 301–313.

Buschmann, S., M. Trapp, and J. Döllner (2012). "Challenges and Approaches for the Visualization of Movement Trajectories in 3D Geovirtual Environments". In: *GIScience Workshop on GeoVisual Analytics – Time to Focus on Time*.

Cao, T.-T. et al. (2010). "Parallel banding algorithm to compute exact distance transform with the GPU". In: *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*. ACM, pp. 83–90.

Card, S. K., J. D. Mackinlay, and B. Shneiderman (1999). *Readings in information visualization: using vision to think*. Morgan Kaufmann.

Carpendale, M. S. T., D. J. Cowperthwaite, and F. D. Fracchia (1995). "3-dimensional pliable surfaces: For the effective presentation of visual information". In: *Proceedings of the 8th annual ACM symposium on User interface and software technology*. ACM, pp. 217–226.

Carpendale, M. S. T. and X Rong (2001). "Examining edge congestion". In: *CHI'01 extended abstracts on Human Factors in Computing Systems*. ACM, pp. 115–116.

Caserta, P., O. Zendra, and D. Bodénes (2011). "3D hierarchical edge bundles to visualize relations in a software city metaphor". In: *Visualizing Software for Understanding and Analysis (VISSOFT), 2011 6th IEEE International Workshop on*. IEEE, pp. 1–8.

Chapin, N. et al. (2001). "Types of software evolution and software maintenance". In: *Journal of Software: Evolution and Process* 13.1, pp. 3–30.

Collins, C. and S. Carpendale (2007). "VisLink: Revealing relationships amongst visualizations". In: *IEEE Transactions on Visualization and Computer Graphics* 13.6, pp. 1192–1199.

Çöltekin, A., S. I. Fabrikant, and M. Lacayo (2010). "Exploring the efficiency of users' visual analytics strategies based on sequence analysis of eye movement recordings". In: *International Journal of Geographical Information Science* 24.10, pp. 1559–1575.

Comaniciu, D. and P. Meer (2002). "Mean shift: A robust approach toward feature space analysis". In: *IEEE Transactions on pattern analysis and machine intelligence* 24.5, pp. 603–619.

Cooley, J. W., P. A. Lewis, and P. D. Welch (1967). "Historical notes on the fast Fourier transform". In: *Proceedings of the IEEE* 55.10, pp. 1675–1677.

Cornelissen, B. et al. (2008). "Execution trace analysis through massive sequence and circular bundle views". In: *Journal of Systems and Software* 81.12, pp. 2252–2268.

Cornelissen, B. et al. (2009). "A systematic survey of program comprehension through dynamic analysis". In: *IEEE Transactions on Software Engineering* 35.5, pp. 684–702.

Cui, W. et al. (2008). "Geometry-based edge clustering for graph visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 14.6, pp. 1277–1284.

Damerau, F. J. (1964). "A technique for computer detection and correction of spelling errors". In: *Communications of the ACM* 7.3, pp. 171–176.

Davis, T. A. and Y. Hu (2011). "The University of Florida sparse matrix collection". In: *ACM Transactions on Mathematical Software (TOMS)* 38.1. `http://www.cise.ufl.edu/research/sparse/matrices`, p. 1.

De Bruijn, O. and R. Spence (2000). "Rapid serial visual presentation: a space-time trade-off in information presentation". In: *Advanced visual interfaces*, pp. 189–192.

Derthick, M. et al. (2003). "Constant density displays using diversity sampling". In: *IEEE Symposium on Information Visualization 2003*, pp. 137–144.

Dickerson, M. et al. (2005). "Confluent Drawings: Visualizing Non-Planar Diagrams in a Planar Way". In: *Journal of Graph Algorithms and Applications* 9.1, pp. 31–52.

Dickerson, M. et al. (2003). "Confluent drawings: Visualizing non-planar diagrams in a planar way". In: *Proc. Graph Drawing*. Springer, pp. 1–12.

Diehl, S. and A. Telea (2014). "Multivariate Networks in Software Engineering". In: *Multivariate Network Visualization*. Ed. by A. Kerren, H. Purchase, and M. Ward. Springer, pp. 13–35.

Diehl, S. (2007). *Software visualization: visualizing the structure, behaviour, and evolution of software*. Springer Science & Business Media.

Diestel, R. (2000). *Graph theory*. Springer-Verlag New York.

Dix, A. and G. Ellis (2002). "by chance enhancing interaction with large data sets through statistical sampling". In: *Proceedings of the Working Conference on Advanced Visual Interfaces*. ACM, pp. 167–176.

Dunne, C. and B. Shneiderman (2009). "Improving graph drawing readability by incorporating readability metrics: A software tool for network analysts". In: *University of Maryland, HCIL Tech Report HCIL-2009-13*.

Dunne, C. and B. Shneiderman (2013). "Motif simplification: improving network visualization readability with fan, connector, and clique glyphs". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, pp. 3247–3256.

Dwyer, T., K. Marriott, and M. Wybrow (2006). "Integrating edge routing into force-directed layout". In: *International Symposium on Graph Drawing*. Springer, pp. 8–19.

Dwyer, T. et al. (2013). "Edge compression techniques for visualization of dense directed graphs". In: *IEEE transactions on visualization and computer graphics* 19.12, pp. 2596–2605.

Dwyer, T. et al. (2014). "Improved optimal and approximate power graph compression for clearer visualisation of dense graphs". In: *Visualization Symposium (PacificVis), 2014 IEEE Pacific*. IEEE, pp. 105–112.

Eades, P. et al. (2015). "Shape-based quality metrics for large graph visualization". In: *International Symposium on Graph Drawing and Network Visualization*. Springer, pp. 502–514.

Ellis, G. and A. Dix (2007). "A taxonomy of clutter reduction for information visualisation". In: *IEEE transactions on visualization and computer graphics* 13.6, pp. 1216–1223.

Emanuelsson, P. and U. Nilsson (2008). "A comparative study of industrial static analysis tools". In: *Electronic notes in theoretical computer science* 217, pp. 5–21.

Epanechnikov, V. A. (1969). "Non-parametric estimation of a multivariate probability density". In: *Theory of Probability & Its Applications* 14.1, pp. 153–158.

Ersoy, O. et al. (2011). "Skeleton-based edge bundling for graph visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 17.12, pp. 2364–2373.

Everts, M. H. et al. (2015). "Exploration of the brain's white matter structure through visual abstraction and multi-scale local fiber tract contraction". In: *IEEE transactions on visualization and computer graphics* 21.7, pp. 808–821.

Fabbri, R. et al. (2008). "2D Euclidean distance transform algorithms: A comparative survey". In: *ACM Computing Surveys (CSUR)* 40.1.

Fekete, J.-D. and C. Plaisant (2002). "Interactive information visualization of a million items". In: *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002.* IEEE, pp. 117–124.

Fua, Y.-H., M. O. Ward, and E. A. Rundensteiner (1999). "Hierarchical parallel coordinates for exploration of large datasets". In: *Proceedings of the conference on Visualization'99: celebrating ten years*. IEEE Computer Society Press, pp. 43–50.

Galhardas, H. et al. (2000). "AJAX: an extensible data cleaning tool". In: *ACM Sigmod Record*. Vol. 29. 2. ACM, p. 590.

Gansner, E. et al. (2017). *GraphViz graph drawing framework*. `www.graphviz.org`.

Gansner, E. R. and Y. Koren (2006). "Improved circular layouts". In: *International Symposium on Graph Drawing*. Springer, pp. 386–398.

Gansner, E. R., Y. Koren, and S. C. North (2005). "Topological fisheye views for visualizing large graphs". In: vol. 11. 4. IEEE, pp. 457–468.

Gansner, E. R. and S. C. North (2000). "An open graph visualization system and its applications to software engineering". In: *Software – Practice and Eperience* 30.11, pp. 1203–1233.

Gansner, E. R. et al. (2011). "Multilevel agglomerative edge bundling for visualizing large graphs". In: *Pacific Visualization Symposium (PacificVis), 2011 IEEE*. IEEE, pp. 187–194.

GeoNames (2015a). *GeoNames French cities database*. `http://download.geonames.org/export/dump/`. URL: `http://download.geonames.org/export/dump/FR.zip` (visited on 05/31/2017).

GeoNames (2015b). *GeoNames world cities over 15000 citizens database*. `http://download.geonames.org/export/dump/`. URL: `http://download.geonames.org/export/dump/cities15000.zip` (visited on 05/31/2017).

Giereth, M., H. Bosch, and T. Ertl (2008). "A 3D treemap approach for analyzing the classificatory distribution in patent portfolios". In: *2008 IEEE Symposium on Visual Analytics Science and Technology*, pp. 189–193.

Graham, M. and J. Kennedy (2003). "Using curves to enhance parallel coordinate visualisations". In: *Proceedings on Seventh International Conference on Information Visualization, 2003. IV 2003.* Pp. 10–16.

Gschwandtner, T. et al. (2014). "TimeCleanser: A visual analytics approach for data cleansing of time-oriented data". In: *Proceedings of the 14th international conference on knowledge technologies and data-driven business*. ACM, p. 18.

Hanjalic, A. (2013). "ClonEvol: Visualizing software evolution with code clones". In: *2013 First IEEE Working Conference on Software Visualization (VISSOFT)*. IEEE, pp. 1–4.

Hansen, G. A., R. W. Douglass, and A. Zardecki (2005). *Mesh enhancement: selected elliptic methods, foundations and applications*. World Scientific.

Haralick, R. M. (1994). *Mathematical Morphology: Theory and Hardware*. Oxford University Press, Inc.

Harris, J., J. L. Hirst, and M. Mossinghoff (2008). *Combinatorics and Graph Theory*. 2^{nd} edition. Springer.

Healey, C. G., K. S. Booth, and J. T. Enns (1995). "Visualizing real-time multivariate data using preattentive processing". In: *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 5.3, pp. 190–221.

Heinrich, J. et al. (2012). "Evaluation of a bundling technique for parallel coordinates". In: *Proc. Information Visualization*, pp. 240–248.

Henry, N., J.-D. Fekete, and M. J. McGuffin (2007). "NodeTrix: a hybrid visualization of social networks". In: *IEEE TVCG* 13.6, pp. 1302–1309.

Herman, I., G. Melançon, and M. S. Marshall (2000). "Graph visualization and navigation in information visualization: A survey". In: *IEEE Transactions on visualization and computer graphics* 6.1, pp. 24–43.

Holten, D. (2006). "Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data". In: *IEEE Transactions on visualization and computer graphics* 12.5, pp. 741–748.

Holten, D. and J. J. Van Wijk (2008). "Visual comparison of hierarchically organized data". In: 27.3, pp. 759–766.

Holten, D. and J. J. Van Wijk (2009). "Force-Directed Edge Bundling for Graph Visualization". In: 28.3, pp. 983–990.

Holten, D. and J. J. Van Wijk (2010). "Evaluation of cluster identification performance for different PCP variants". In: 29.3, pp. 793–802.

Holten, D. et al. (2011). "An extended evaluation of the readability of tapered, animated, and textured directed-edge representations in node-link graphs". In: *Pacific Visualization Symposium (PacificVis), 2011 IEEE*. IEEE, pp. 195–202.

Hoon, M. J. de et al. (2004). "Open source clustering software". In: *Bioinformatics* 20.9, pp. 1453–1454.

Hop, W. et al. (2012). "Using Hierarchical Edge Bundles to visualize complex ontologies in GLOW". In: *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. ACM, pp. 304–311.

Huang, M. L., P. Eades, and J. Wang (1998). "On-line animated visualization of huge graphs using a modified spring algorithm". In: *Journal of Visual Languages & Computing* 9.6, pp. 623–645.

Hurter, C. (2015). "Image-Based Visualization: Interactive Multidimensional Data Exploration". In: *Synthesis Lectures on Visualization* 3.2, pp. 1–127.

Hurter, C., O. Ersoy, and A. Telea (2012). "Graph bundling by kernel density estimation". In: *Computer Graphics Forum*. Vol. 31. 3pt1. Wiley Online Library, pp. 865–874.

Hurter, C., O. Ersoy, and A. Telea (2013). "Smooth bundling of large streaming and sequence graphs". In: *Visualization Symposium (PacificVis), 2013 IEEE Pacific*. IEEE, pp. 41–48.

Hurter, C., A. Telea, and O. Ersoy (2011). "Moleview: An attribute and structure-based semantic lens for large element-based plots". In: *IEEE Transactions on Visualization and Computer Graphics* 17.12, pp. 2600–2609.

Hurter, C., B. Tissoires, and S. Conversy (2009). "FromDaDy: Spreading data across views to support iterative exploration of aircraft trajectories". In: *IEEE TVCG* 15.6, pp. 1017–1024.

Hurter, C. et al. (2014a). "Bundled visualization of dynamicgraph and trail data". In: *IEEE transactions on visualization and computer graphics* 20.8, pp. 1141–1157.

Hurter, C. et al. (2014b). "Color tunneling: Interactive exploration and selection in volumetric datasets". In: *Visualization Symposium (PacificVis), 2014 IEEE Pacific*. IEEE, pp. 225–232.

Hurter, C. et al. (2014c). “Interactive image-based information visualization for aircraft trajectory analysis”. In: *Transportation Research Part C: Emerging Technologies* 47, pp. 207–227.

Inselberg, A. (2009). *Parallel Coordinates: Visual Multidimensional Geometry and its Applications*. Springer.

Isaacs, B and A. J. Akhtar (1972). “The set test: a rapid test of mental function in old people”. In: *Age and ageing* 1.4, pp. 222–226.

Isaacs, B. and A. T. Kennie (1973). “The Set test as an aid to the detection of dementia in old people”. In: *The British Journal of Psychiatry* 123.575, pp. 467–470.

Jain, A. K., M. N. Murty, and P. J. Flynn (1999). “Data clustering: a review”. In: *ACM computing surveys (CSUR)* 31.3, pp. 264–323.

Jia, Y., M. Garland, and J. C. Hart (2011). “Social network clustering and visualization using hierarchical edge bundles”. In: 30.8, pp. 2314–2327.

Johansson, J. et al. (2006). “Revealing structure in visualizations of dense 2D and 3D parallel coordinates”. In: *Information Visualization* 5.2, pp. 125–136.

Kamada, T. and S. Kawai (1989). “An algorithm for drawing general undirected graphs”. In: *Information processing letters* 31.1, pp. 7–15.

Karran, B., J. Trumper, and J. Dollner (2013). “Synctrace: Visual thread-interplay analysis”. In: *Software Visualization (VISSOFT), 2013 First IEEE Working Conference on*. IEEE, pp. 1–10.

Keim, D. et al. (2008). “Visual analytics: Definition, process, and challenges”. In: *Lecture notes in computer science* 4950, pp. 154–176.

Keim, D. A. (2000). “Designing pixel-oriented visualization techniques: Theory and applications”. In: *IEEE Transactions on visualization and computer graphics* 6.1, pp. 59–78.

Ken, S. H. (2003). *Metrics and Models in Software Quality Engineering*. Addison-Wesley.

Kienreich, W. and C. Seifert (2010). “An application of edge bundling techniques to the visualization of media analysis results”. In: *Information Visualisation (IV), 2010 14th International Conference*. IEEE, pp. 375–380.

Kim, J. et al. (2010). “Pilot gaze and glideslope control”. In: *ACM Transactions on Applied Perception (TAP)* 7.3, p. 18.

Kim, W. et al. (2003). “A taxonomy of dirty data”. In: *Data mining and knowledge discovery* 7.1, pp. 81–99.

Klein, T., M. Van Der Zwan, and A. Telea (2014). “Dynamic multiscale visualization of flight data”. In: *Computer Vision Theory and Applications (VISAPP), 2014 International Conference on*. Vol. 1. IEEE, pp. 104–114.

Kobourov, S. G., S. Pupyrev, and B. Saket (2014). “Are crossings important for drawing large graphs?” In: *International Symposium on Graph Drawing*. Springer, pp. 234–245.

Koenderink, J. J. (1984). “The structure of images”. In: *Biological cybernetics* 50.5, pp. 363–370.

Koschke, R. (2003). "Software visualization in software maintenance, reverse engineering, and re-engineering: a research survey". In: *Journal of Software: Evolution and Process* 15.2, pp. 87–109.

Kreuseler, M. and H. Schumann (1999). "Information visualization using a new focus+ context technique in combination with dynamic clustering of information space". In: *Proceedings of the 1999 workshop on new paradigms in information visualization and manipulation in conjunction with the eighth ACM internation conference on Information and knowledge management*. ACM, pp. 1–5.

Kruskal, J. B. and J. M. Landwehr (1983). "Icicle plots: Better displays for hierarchical clustering". In: *The American Statistician* 37.2, pp. 162–168.

Krygier, J. B. et al. (1997). "Multimedia in geographic education: Design, implementation, and evaluation". In: *J. Geogr. Higher Educ.* 21.1, pp. 17–39.

Lambert, A., D. Auber, and G. Melançon (2010). "Living flows: Enhanced exploration of edge-bundled graphs based on gpu-intensive edge rendering". In: *Information Visualisation (IV), 2010 14th International Conference*. IEEE, pp. 523–530.

Lambert, A., R. Bourqui, and D. Auber (2010a). "3D edge bundling for geographical data visualization". In: *Information Visualisation (IV), 2010 14th International Conference*. IEEE, pp. 329–335.

Lambert, A., R. Bourqui, and D. Auber (2010b). "Winding roads: Routing edges into bundles". In: *Computer Graphics Forum*. Vol. 29. 3. Wiley Online Library, pp. 853–862.

Lambert, A., J. Dubois, and R. Bourqui (2011). "Pathway preserving representation of metabolic networks". In: *Computer Graphics Forum*. Vol. 30. 3. Wiley Online Library, pp. 1021–1030.

Lampe, O. D. and H. Hauser (2011). "Interactive visualization of streaming data with kernel density estimation". In: *Pacific Visualization Symposium (PacificVis), 2011 IEEE*. IEEE, pp. 171–178.

Larkin, J. H. and H. A. Simon (1987). "Why a diagram is (sometimes) worth ten thousand words". In: *Cognitive science* 11.1, pp. 65–100.

Lee, B. et al. (2006). "Task taxonomy for graph visualization". In: *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*. ACM, pp. 1–5.

Levenshtein, V. I. (1966). "Binary codes capable of correcting deletions, insertions, and reversals". In: *Soviet physics doklady*. Vol. 10. 8, pp. 707–710.

Lhuillier, A., C. Hurter, and A. Telea (2016). *FFTEB implementation*. Visual Studio C# source code, `http://recherche.enac.fr/~hurter/FFTEB/FFTEB.html`.

Lhuillier, A. and C. Hurter (2015). "Bundling, graph simplification trough visual aggregation: existing techniques and challenges". In: *Proceedings of the 27th Conference on l'Interaction Homme-Machine*. ACM, 10:1–10:10.

Lhuillier, A., C. Hurter, and A. Telea (2017a). "FFTEB: Edge bundling of huge graphs by the Fast Fourier Transform". In: *Pacific Visualization Symposium (PacificVis), 2017 IEEE*. IEEE, pp. 190–199.

Lhuillier, A., C. Hurter, and A. Telea (2017b). "State of the Art in Edge and Trail Bundling Techniques". In: *Computer Graphics Forum*. Vol. 36. 3. Wiley Online Library, pp. 619–645.

Lhuillier, A. et al. (2014). "Cognitive Maps Exploration trough Kernel Density Estimation". In: *EHRVis-Visualizing Electronic Health Record Data. IEEE VIS Workshop 2014*.

Lhuillier, A. et al. (2015). "Visual analytics for the interpretation of fluency tests during Alzheimer evaluation". In: *Proceedings of the 2015 Workshop on Visual Analytics in Healthcare*. ACM, 3:1–3:8.

Li, Q. and C. North (2003). "Empirical comparison of dynamic query sliders and brushing histograms". In: *IEEE Symposium on Information Visualization 2003*, pp. 147–153.

Luo, S.-J. et al. (2012). "Ambiguity-free edge-bundling for interactive graph visualization". In: *IEEE transactions on visualization and computer graphics* 18.5, pp. 810–821.

Maaten, L. v. d. and G. Hinton (2008). "Visualizing data using t-SNE". In: *Journal of Machine Learning Research* 9, pp. 2579–2605.

MacEachren, A. M. (2004). *How maps work: representation, visualization, and design*. Guilford Press.

Martins, R. M., R. Minghim, A. C. Telea, et al. (2015). "Explaining neighborhood preservation for multidimensional projections". In: University College London.

Martins, R. M. et al. (2014). "Visual analysis of dimensionality reduction quality for parameterized projections". In: *Computers & Graphics* 41, pp. 26–42.

McDonnell, K. T. and K. Mueller (2008). "Illustrative parallel coordinates". In: 27.3, pp. 1031–1038.

McGee, F. and J. Dingliana (2012). "An empirical study on the impact of edge bundling on user comprehension of graphs". In: *Proceedings of the International Working Conference on Advanced Visual Interfaces*. ACM, pp. 620–627.

Meilinger, T. (2008). "The network of reference frames theory: A synthesis of graphs and cognitive maps". In: Springer, pp. 344–360.

Menasri, F. et al. (2012). "The A2iA French handwriting recognition system at the Rimes-ICDAR2011 competition." In: *DRR*, 82970Y.

Miller, G. A. (1995). "WordNet: a lexical database for English". In: *Communications of the ACM* 38.11, pp. 39–41.

Miller, N. et al. (1997). "The need for metrics in visual information analysis". In: *Proceedings of the 1997 workshop on New paradigms in information visualization and manipulation*. ACM, pp. 24–28.

Moberts, B., A. Vilanova, and J. J. van Wijk (2005). "Evaluation of fiber clustering methods for diffusion tensor imaging". In: *VIS 05. IEEE Visualization, 2005.* 65–72.

Moura, D. C. (2015). "3D Density Histograms for Criteria-driven Edge Bundling". In: *arXiv preprint arXiv:1504.02687*.

Moustafa, R and E. J. Wegman (2002). "On some generalizations of parallel coordinate plots". In: *Seeing a Million–A Data Visualization Workshop*, pp. 41–48.

Mühlbacher, T. et al. (2014). "Opening the black box: Strategies for increased user involvement in existing algorithm implementations". In: *IEEE transactions on visualization and computer graphics* 20.12, pp. 1643–1652.

Müller, H. and J.-C. Freytag (2005). *Problems, methods, and challenges in comprehensive data cleansing*. Professoren des Inst. Für Informatik.

Munzner, T. (2014). *Visualization analysis and design*. CRC press.

Nagel, T., C. Pietsch, and M. Dörk (2016). "Staged Analysis: From Evocative to Comparative Visualizations of Urban Mobility". In: *Proc. IEEE Visualizaton (Arts Program)*. `https://uclab.fh-potsdam.de/cf`, pp. 23–30.

Newbery, F. J. (1989). "Edge concentration: A method for clustering directed graphs". In: 14.7, pp. 76–85.

Nguyen, Q., P. Eades, and S.-H. Hong (2012). "StreamEB: Stream Edge Bundling." In: *Graph Drawing*. Springer, pp. 400–413.

Nguyen, Q., P. Eades, and S.-H. Hong (2013). "On the faithfulness of graph visualizations". In: *Visualization Symposium (PacificVis), 2013 IEEE Pacific*. IEEE, pp. 209–216.

Nguyen, Q. H., P. Eades, and S.-H. Hong (2017). "Towards Faithful Graph Visualizations". In: *arXiv preprint arXiv:1701.00921*.

Nguyen, Q. H., S.-H. Hong, and P. Eades (2011). "TGI-EB: A New Framework for Edge Bundling Integrating Topology, Geometry and Importance." In: *Graph Drawing*. Springer, pp. 123–135.

Nocaj, A. and U. Brandes (2013). "Stub bundling and confluent spirals for geographic networks". In: *International Symposium on Graph Drawing*. Springer, pp. 388–399.

Nocke, T et al. (2015). "visual analytics of climate networks". In: *Nonlinear Processes in Geophysics* 22.5, p. 545.

Novotny, M. and H. Hauser (2006). "Outlier-preserving focus+ context visualization in parallel coordinates". In: *IEEE Transactions on Visualization and Computer Graphics* 12.5, pp. 893–900.

O'keefe, J. and L. Nadel (1978). *The hippocampus as a cognitive map*. Oxford: Clarendon Press.

Oliveira, P., F. Rodrigues, and P. R. Henriques (2005). "A Formal Definition of Data Quality Problems." In: *IQ*.

Palmas, G. and T. Weinkauf (2016). "Space bundling for continuous parallel coordinates". In: *Proceedings of the Eurographics/IEEE VGTC Conference on Visualization: Short Papers*. Eurographics Association, pp. 61–65.

Palmas, G. et al. (2014). "An edge-bundling layout for interactive parallel coordinates". In: *Visualization Symposium (PacificVis), 2014 IEEE Pacific*. IEEE, pp. 57–64.

Peng, W., M. O. Ward, and E. A. Rundensteiner (2004). "Clutter reduction in multidimensional data visualization using dimension reordering". In: *IEEE Symposium on Information Visualization*. IEEE, pp. 89–96.

Peysakhovich, V., C. Hurter, and A. Telea (2015). "Attribute-driven edge bundling for general graphs with applications in trail analysis". In: *2015 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 39–46.

Phan, D. et al. (2005). "Flow map layout". In: *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.* IEEE, pp. 219–224.

Philips, L. (1990). "Hanging on the metaphone". In: *Computer Language* 7.12 (December).

Philips, L. (2000). "The double metaphone search algorithm". In: *C/C++ users journal* 18.6, pp. 38–43.

Piegl, L. and W. Tiller (1997). *The NURBS Book*. 2^{nd} edition. Springer.

Podlozhnyuk, V. (2007). *FFT-based 2D convolution*. NVidia white papers, `http://developer.download.nvidia.com/compute/cuda/2_2/sdk/website/projects/convolutionFFT2D/doc/convolutionFFT2D.pdf`.

Post, F. H. et al. (2003). "The state of the art in flow visualisation: Feature extraction and tracking". In: 22.4, pp. 775–792.

Press, W. H. et al. (1989). "Numerical recipes". In: 3.

Pupyrev, S., L. Nachmanson, and M. Kaufmann (2010). "Improving Layered Graph Layouts with Edge Bundling." In: *Graph Drawing*. Vol. 6502. Springer, pp. 329–340.

Purchase, H. C., R. F. Cohen, and M. James (1995). "Validating graph drawing aesthetics". In: *International Symposium on Graph Drawing*. Springer, pp. 435–446.

Qu, H., H. Zhou, and Y. Wu (2006). "Controllable and progressive edge clustering for large networks". In: *International Symposium on Graph Drawing*. Springer, pp. 399–404.

Rafiei, D. (2005). "Effectively visualizing large networks through sampling". In: *VIS 05. IEEE Visualization, 2005.* IEEE, pp. 375–382.

Rahm, E. and H. H. Do (2000). "Data cleaning: Problems and current approaches". In: *IEEE Data Eng. Bull.* 23.4, pp. 3–13.

Rao, R. and S. K. Card (1994). "The table lens: merging graphical and symbolic representations in an interactive focus+ context visualization for tabular information". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, pp. 318–322.

Rauber, P. E. et al. (2017). "Visualizing the hidden activity of artificial neural networks". In: *IEEE transactions on visualization and computer graphics* 23.1, pp. 101–110.

Reniers, D., L. Voinea, and A. Telea (2011). "Visual exploration of program structure, dependencies and metrics with solidsx". In: *Visualizing Software for Understanding and Analysis (VISSOFT), 2011 6th IEEE International Workshop on*. IEEE, pp. 1–4.

Reniers, D. et al. (2014). "The Solid* toolset for software visual analytics of program structure and metrics comprehension: From research prototype to product". In: *Science of Computer Programming* 79, pp. 224–240.

Riche, N. H. et al. (2012). "Exploring the design space of interactive link curvature in network diagrams". In: *Proceedings of the International Working Conference on Advanced Visual Interfaces*. ACM, pp. 506–513.

Royer, L. et al. (2008). "Unraveling Protein Networks with Power Graph Analysis". In: *PLOS Computational Biology* 4.7, pp. 1–17.

Sarkar, M. et al. (1993). "Stretching the rubber sheet: a metaphor for viewing large layouts on small screens". In: *Proceedings of the 6th annual ACM symposium on User interface software and technology*. ACM, pp. 81–91.

Schaeffer, S. (2007). "Survey: Graph clustering". In: *Computer Science Review* 1.1, pp. 27–64.

Scheepens, R. et al. (2011a). "Composite density maps for multivariate trajectories". In: *IEEE Transactions on Visualization and Computer Graphics* 17.12, pp. 2518–2527.

Scheepens, R. et al. (2011b). "Interactive visualization of multivariate trajectory data with density maps". In: *Pacific Visualization Symposium (PacificVis), 2011 IEEE*. IEEE, pp. 147–154.

Scheepens, R. et al. (2016). "Visualization, selection, and analysis of traffic flows". In: *IEEE transactions on visualization and computer graphics* 22.1, pp. 379–388.

Schuler, D. and A. Namioka (1993). *Participatory design: Principles and practices*. CRC Press.

Schulz, H.-J. and C. Hurter (2013). "Grooming the hairball-how to tidy up network visualizations?" In: *INFOVIS 2013, IEEE Information Visualization Conference*.

Selassie, D., B. Heller, and J. Heer (2011). "Divided edge bundling for directional network data". In: *IEEE Transactions on Visualization and Computer Graphics* 17.12, pp. 2354–2363.

Sethian, J. A. et al. (2003). *Level set methods and fast marching methods*. Vol. 11. 1, pp. 1–2.

Shneiderman, B. (1996). "The eyes have it: A task by data type taxonomy for information visualizations". In: *Visual Languages, 1996. Proceedings., IEEE Symposium on*. IEEE, pp. 336–343.

Shrinivasan, Y. B. and J. van Wijk (2009). "Supporting exploration awareness in information visualization". In: *IEEE Computer Graphics and Applications* 29.5, pp. 34–43.

Shrinivasan, Y. B. and J. J. van Wijk (2008). "Supporting the analytical reasoning process in information visualization". In: *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, pp. 1237–1246.

Siddiqi, K. and S. Pizer (2008). *Medial representations: mathematics, algorithms and applications*. Vol. 37. Springer Science & Business Media.

Sorzano, C. O. S., J. Vargas, and A. P. Montano (2014). *A survey of dimensionality reduction techniques*.

Stasko, J. and E. Zhang (2000). "Focus+ context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations". In: *IEEE Symposium on Information Visualization 2000. INFOVIS 2000.* IEEE, pp. 57–65.

Stone, M. C., K. Fishkin, and E. A. Bier (1994). "The movable filter as a user interface tool". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* ACM, pp. 306–312.

Sugiyama, K., S. Tagawa, and M. Toda (1981). "Methods for visual understanding of hierarchical system structures". In: *IEEE Transactions on Systems, Man, and Cybernetics* 11.2, pp. 109–125.

Sun, J. et al. (2007). "Graphscope: parameter-free mining of large time-evolving graphs". In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, pp. 687–696.

Szeliski, R. (2010). *Computer vision: algorithms and applications.* Springer Science & Business Media.

Tatler, B. et al. (2010). "Yarbus, eye movements, and vision". In: *i-Perception* 1, pp. 7–27.

Telea, A. and J. J. van Wijk (2002). "An augmented fast marching method for computing skeletons and centerlines". In: *In Proceedings of the symposium on Data Visualisation 2002, VISSYM'02*, pp. 251–259.

Telea, A. and D. Auber (2008). "Code flows: Visualizing structural evolution of source code". In: 27.3, pp. 831–838.

Telea, A. and O. Ersoy (2010). "Image-Based Edge Bundles: Simplified Visualization of Large Graphs". In: *Computer Graphics Forum.* Vol. 29. 3, pp. 843–852.

Telea, A. and J. J. van Wijk (1999). "Simplified Representation of Vector Fields". In: *Proceedings of the Conference on Visualization '99: Celebrating Ten Years.* IEEE, pp. 35–42.

Telea, A. et al. (2009). "Comparison of node-link and hierarchical edge bundling layouts: A user study". In: *Dagstuhl Seminar Proceedings.* Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Telea, A. C. (2015). *Data Visualization: Principles and Practice.* 2^{nd} edition. CRC Press.

Thöny, M. and R. Pajarola (2015). "Vector Map Constrained Path Bundling in 3D Environments". In: *Proceedings of the 6th ACM SIGSPATIAL International Workshop on GeoStreaming.* ACM, pp. 33–42.

Tobler, W. (1981). "Depicting federal fiscal transfers". In: *The Professional Geographer* 33.4, pp. 419–422.

Tolman, E. C. (1948). *Cognitive maps in rats and men.*

Tominski, C. et al. (2006). "Fisheye tree views and lenses for graph visualization". In: *Tenth International Conference on Information Visualisation (IV'06)*, pp. 17–24.

Trümper, J., J. Döllner, and A. Telea (2013). "Multiscale Visual Comparison of Execution Traces". In: *21st International Conference on Program Comprehension (ICPC).*

Tsai, A. et al. (2007). "Fiber tract clustering on manifolds with dual rooted-graphs". In: *IEEE Conference on Computer Vision and Pattern Recognition*.

Tu, S. et al. (2017). *The Protégé ontology editor*. `http://protege.stanford.edu`.

Tufte, E. R. (1992). *The Visual Display of Quantitative Information*. Graphics Press.

U.S. Census Bureau (2003). *County-to-County Migration Flow Files*. `www.census.gov/population/www/cen2000/ctytoctyflow.html`.

Van Ham, F. and A. Perer (2009). ""Search, show context, expand on demand": supporting large graph exploration with degree-of-interest". In: *IEEE Transactions on Visualization and Computer Graphics* 15.6.

Van Ham, F. and M. Wattenberg (2008). "Centrality based visualization of small world graphs". In: 27.3, pp. 975–982.

Van Liere, R. and W. De Leeuw (2003). "GraphSplatting: Visualizing graphs as continuous fields". In: *IEEE Transactions on Visualization and Computer Graphics* 9.2, pp. 206–212.

Van Wijk, J. J. (2002). "Image based flow visualization". In: *ACM Transactions on Graphics (ToG)* 21.3, pp. 745–754.

Van Wijk, J. J. and H. van de Wetering (1999). "Cushion Treemaps: Visualization of Hierarchical Information". In: *Information Visualization, 1999. (Info Vis '99) Proceedings*, pp. 73–82.

Voinea, L. and A. C. Telea (2014). "Visual clone analysis with SolidSDD". In: *Software Visualization (VISSOFT), 2014 Second IEEE Working Conference on*. IEEE, pp. 79–82.

Von Landesberger, T. et al. (2011). "Visual analysis of large graphs: state-of-the-art and future research challenges". In: *Computer graphics forum*. Vol. 30. 6, pp. 1719–1749.

Waldeck, C. and D. Balfanz (2004). "Mobile liquid 2d scatter space (ml2dss)". In: *Eighth International Conference on Information Visualisation, 2004. IV 2004.* IEEE, pp. 494–498.

Walker, F. O. (2007). "Huntington's disease". In: *The Lancet* 369.9557, pp. 218–228.

Wang, C., S. Chandrasekaran, and B. Chapman (2016). "cusFFT: A High-Performance Sparse Fast Fourier Transform Algorithm on GPUs". In: *Parallel and Distributed Processing Symposium, 2016 IEEE International*. IEEE, pp. 963–972.

Ware, C. et al. (2002). "Cognitive measurements of graph aesthetics". In: *Information visualization* 1.2, pp. 103–110.

Wasserman, S. and K. Faust (1994). *Social network analysis: Methods and applications*. Vol. 8. Cambridge university press.

Wegman, E. J. and Q. Luo (1997). "High dimensional clustering using parallel coordinates and the grand tour". In: *Classification and Knowledge Organization*, pp. 93–102.

Wettel, R. and M. Lanza (2007). "Program Comprehension through Software Habitability". In: *15th IEEE International Conference on Program Comprehension (ICPC '07)*.

Wise, J. A., V. D. Hopkin, and M. L. Smith (1990). "Automation and Systems Issues in Air Traffic Control". In: *Proc. NATO Advanced Study Institute on Automation and Systems*. Springer.

Wolff, A. (2007). "Drawing subway maps: A survey". In: *Informatik-Forschung und Entwicklung* 22.1, pp. 23–44.

Wong, N. and S. Carpendale (2007). "Supporting interactive graph exploration using edge plucking." In: *Visualization and Data Analysis*. Vol. 6495. 2.

Wong, N., S. Carpendale, and S. Greenberg (2003). "Edgelens: An interactive method for managing edge congestion in graphs". In: *IEEE Symposium on Information Visualization 2003*. IEEE, pp. 51–58.

Wood, J., J. Dykes, and A. Slingsby (2010). "Visualisation of Origins, Destinations and Flows with OD Maps". In: *The Cartographic Journal* 47.2, pp. 117–129.

Woodruff, A., J. Landay, and M. Stonebraker (1998). "Constant density visualizations of non-uniform distributions of data". In: *Proceedings of the 11th annual ACM symposium on User interface software and technology*. ACM, pp. 19–28.

Wu, J., L. Yu, and H. Yu (2015). "Texture-based edge bundling: A web-based approach for interactively visualizing large graphs". In: *IEEE International Conference on Big Data*, pp. 1230–1237.

Yang, Y. et al. (2017). "Many-to-Many Geographically-Embedded Flow Visualisation: An Evaluation". In: *IEEE transactions on visualization and computer graphics* 23.1, pp. 411–420.

Yi, J. et al. (2006). "Dust & magnet: Multivariate information visualization using a magnet metaphor". In: *Information Visualization* 4.4, pp. 542–551.

Yu, H. et al. (2012). "Hierarchical streamline bundles". In: *IEEE Transactions on Visualization and Computer Graphics* 18.8, pp. 1353–1367.

Zhou, H. et al. (2008a). "Energy-based hierarchical edge clustering of graphs". In: *2008 IEEE Pacific Visualization Symposium*, pp. 55–61.

Zhou, H. et al. (2008b). "Visual clustering in parallel coordinates". In: *Computer Graphics Forum*. Vol. 27. 3, pp. 1047–1054.

Zhou, H. et al. (2013). "Edge bundling in information visualization". In: *Tsinghua Science and Technology* 18.2, pp. 145–156.

Zielasko, D. et al. (2016). "Interactive 3D Force-Directed Edge Bundling". In: *Computer Graphics Forum*. Vol. 35. 3, pp. 51–60.

Zwan, M. van der, V. Codreanu, and A. Telea (2016). "CUBu: Universal real-time bundling for large graphs". In: *IEEE transactions on visualization and computer graphics* 22.12, pp. 2550–2563.

List of Figures

List of Tables

Résumé

Le big data est un challenge majeur de la visualisation ; l'augmentation du nombre de données à visualiser augmente la densité et l'occultation des graphes et il devient difficile de distinguer les éléments qui le compose. Pour résoudre ce challenge, plusieurs techniques de visualisation se focalisent sur la simplification visuelle ; parmi elles, l'agrégation visuelle (bundling) permet l'agrégation des liens pour créer des zones de fortes densités au profit d'espaces plus clairsemés faisant ainsi émerger des structures visuelles. Cette thèse s'efforce à faire le trait d'union entre la complexité technique des algorithmes de bundling et les utilisateurs finaux.

Dans un premier temps, nous avons formalisé l'espace de design des techniques de bundling afin d'améliorer la compréhension des chercheurs et des utilisateurs. Notre formalisation se fonde sur une taxonomie centrée utilisateur organisant l'ensemble des techniques d'agrégation en fonction des données d'entrée. Ensuite, à partir d'une définition formelle du bundling, nous proposons un modèle générique décrivant l'ensemble des étapes usuelles des algorithmes de bundling et montrons comment les techniques existantes implémentent chaque étape. Enfin, à travers une analyse des tâches, nous exposons des cas d'utilisation avérés.

Notre analyse de l'espace des techniques de bundling nous a montré les limites actuelles du bundling quant au traitement de grande quantité de données tant en terme de rapidité de calcul qu'en terme de taille des jeux de données. Ainsi, nous avons résolu ces limites en introduisant une nouvelle technique plus rapide et sans limitation de taille : FFTEB (Fast Fourier Transform Edge Bundling Technique). Notre technique déplace le processus d'agrégation de l'espace pixelaire vers l'espace spectral. Enfin, grâce à un processus de transfert des données, FFTEB résout les problèmes de taille de jeux de données.

En dernier lieu, dans le cadre d'une application à la maladie d'Alzheimer, cette thèse démontre l'efficacité des techniques de bundling comme outil d'exploration visuelle. Dans le contexte d'une étude nationale sur la maladie d'Alzheimer, nous avons focalisé notre recherche sur l'analyse de la représentation mentale de l'espace géographique chez les personnes âgées. Nous montrons que l'utilisation du bundling pour comparer les cartes mentales des populations démentes et non-démentes a permis à des neuropsychologues de formuler de nouvelles hypothèses sur l'évolution de la maladie d'Alzheimer. Ces nouvelles hypothèses nous ont permis de montrer l'émergence d'un potentiel marqueur de la maladie près de douze ans avant que les patients ne soient diagnostiqués comme atteints de cette maladie.

Mots clés : Bundling, Visualisation, Exploration Visuelle, Big Data, IHM, Alzheimer, Cartes Cognitives, Vieillissement.

Abstract

Dense and complex data visualizations suffer from occluded items, which hinders insight retrieval. This is especially the case for very large graph or trails set. To address cluttering issues, several techniques propose to visually simplify the representation, often meeting scalability and computational speed limits. Among them, bundling techniques provide a visual simplification of node-link diagrams by spatially grouping similar items. This thesis strives to bridge the gap between the technical complexity of bundling techniques and the end-point user.

The first aim of this thesis was to improve the understanding of graph and trail bundling techniques as a clutter reduction method for node-link diagrams of large data-set. To do so, we created a data-based taxonomy that organizes bundling methods on the type of data they work on. From this thorough review and based on a formal definition of path bundling, we propose a unified framework that describes the typical steps of bundling algorithms in terms of high-level operations and show how existing methods classes implement these steps. In addition, we propose a description of tasks that bundling aims to address and demonstrate them through a wide set of applications.

Although many techniques exist, handling large data-sets and selectively bundling paths based on attributes is still a challenge. To answer the scalability and computational speed issues of bundling techniques, we propose a new technique which improves both. For this, we shift the bundling process from the image to the spectral space, thereby increasing computational limits. We address the later by proposing a streaming scheme allowing bundling of extremely large data-sets.

Finally, as an application domain, we studied how bundling can be used as an efficient visualization technique for societal health challenges. In the context of a national study on Alzheimer disease, we focused our research on the analysis of the mental representation of geographical space for elderly people. We show that using bundling to compare the cognitive maps of dement and non-dement subjects helped neuro-psychologist to formulate new hypotheses on the evolution of Alzheimer disease. These new hypotheses led us to discover a potential marker of the disease years before the actual diagnosis.

Keywords: Bundling, Visualization, Visual Analysis, Big Data, HCI, Alzheimer, Cognitive Maps, Aging Process.