UNIVERSITÉ DU QUÉBEC À CHICOUTIMI

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À MONTRÉAL
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE
OFFERTE À
L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI
EN VERTU D'UN PROTOCOLE D'ENTENTE
AVEC L'UNIVERSITÉ DU QUÉBEC À MONTRÉAL

PAR
JÉRÔME LAMBERT
B. SC. A.

ÉTUDE D'ALGORITHMES EN VISUALISATION ET EXPLORATION DE GRANDS GRAPHES À L'AIDE DE VALUATIONS

NOVEMBRE 2006

Résumé

La visualisation est un domaine de recherche encore jeune, étudiant les représentations graphiques produite à partir d'ensembles de données afin de les analyser. Ce domaine de recherche devient particulièrement intéressant lorsque l'on considère la quantité d'information de plus en plus importante générée de nos jours dans divers domaines de gestion et de recherche.

Ce mémoire vise deux objectifs principaux. Tout d'abord, nous effectuerons une étude sommaire des algorithmes de visualisation scientifique et d'information, ainsi que des méthodes d'interaction y étant rattachées. Par la suite, nous porterons notre attention sur l'exploration de graphes de grande taille à l'aide de valuations. À partir du graphe étudié, nous produirons un volume correspondant à un histogramme tridimensionnel généré par le calcul d'un triplet de valuations. Chacun des sommets ou arêtes du graphe est associé à un point à l'intérieur du volume. On applique ensuite la convolution d'un noyau Gaussien à ce volume et celui-ci est présenté à l'utilisateur à l'aide d'un algorithme de rendu de volume. L'opération de convolution transforme le nuage de points formant l'histogramme à trois dimensions en un ensemble de nuages diffus et permet de faciliter la sélection de zones d'intérêt à l'intérieur du volume.

De plus, nous introduirons un contrôleur graphique permettant de naviguer à l'intérieur du volume et d'en extraire des régions d'intérêt. Ces régions représentent des groupes de sommets ou d'arêtes et sont dessinées à l'aide d'un algorithme de rendu de surface. Finalement, nous démontrerons qu'il est possible d'extraire des agglomérats structurels d'un graphe en utilisant notre application et d'associer ces regroupements structurels à une interprétation sémantique reliée au domaine d'application du graphe.

Remerciments

J'aimerais tout d'abord remercier mon directeur, Yves Chiricota, pour le temps qu'il m'a consacré, pour son enseignement ainsi que pour son esprit critique. Son enthousiasme et sa passion pour les mathématiques, l'informatique et plus spécialement pour l'infographie, m'ont servi de motivation tout au long de ce projet. C'est en grande partie grâce à ses recherches et à son soutien que j'ai pu me lancer dans cette aventure.

Je tiens également à remercier l'ensemble du corps professoral de l'Université du Québec À Chicoutimi pour leur disponibilité et le partage de leur savoir. Aussi, j'aimerais remercier l'ensemble des membres du Groupe de Recherche en Informatique de l'UQAC que j'ai côtoyés durant ces deux dernières années pour leur aide et leur camaraderie. Plus spécifiquement, j'adresse un merci particulièrement aléatoire à Pierre Delisle. De plus, j'aimerais remercier spécialement mon collègue et ami Éric Lavoie pour ses nombreux conseils au plan professionnel et personnel, ainsi que pour les moments de détente que m'ont apportés ses délires et son humour burlesque.

J'aimerais aussi remercier mes amis qui m'ont toujours soutenu au plan personnel et avec qui j'ai pu me divertir durant mes temps libres ces deux dernières années. Finalement, un merci tout spécial à ma famille pour leur support inconditionnel, et ce, jusqu'au tout dernier moment. Vos encouragements, votre soutien et votre écoute tout au long de mes études m'ont été d'une grande aide et je vous en serai éternellement reconnaissant.

Table des matières

Table des figures

Introduction

Durant les dernières années, la capacité de calcul et de stockage des ordinateurs a considérablement augmenté. L'utilisation de ces machines, dans une multitude de domaines scientifiques et d'ingénierie, permet aux chercheurs d'augmenter continuellement la quantité de données recueillies et de produire des modèles extrêmement détaillés des phénomènes et sujets qu'ils observent. Parallèlement, la représentation efficace de ces données est une problématique relativement nouvelle occupant de plus en plus d'importance dans la communauté scientifique. En informatique, la visualisation est un domaine de recherche visant l'étude des méthodes d'affichage permettant d'analyser des données. On peut diviser la visualisation en deux sous-domaines de recherche : la visualisation d'information et la visualisation scientifique. On associe généralement la visualisation d'information aux algorithmes permettant de représenter des données statistiques, nominales ou abstraites, alors que la visualisation scientifique permet de représenter des données pour lesquelles il existe une relation spatio-temporelle. Celles-ci proviennent généralement de modalités d'imagerie ou encore de simulations.

Un graphe est une structure de données permettant de représenter des relations entre les éléments d'un groupe d'entités distinctes. Les algorithmes classiques de dessinage de graphes sont tirés du domaine de la visualisation d'information. Cependant, ceux-ci sont peu efficaces lorsque le nombre d'éléments et de relations à représenter est

très élevé : il en résulte alors une représentation encombrée. Cet ouvrage vise donc deux objectifs principaux. Dans un premier temps, nous effectuerons l'étude d'une gamme étendue d'algorithmes de visualisation scientifique et d'information. En second lieu, nous introduirons un système de visualisation permettant d'explorer un graphe à partir d'un volume généré à l'aide de valuations calculées sur celui-ci.

Notre approche consiste à associer les sommets et arêtes d'un graphe à des mesures numériques calculées sur ceux-ci afin de les distribuer le plus possible de façon dispersée dans l'espace. Ces mesures sont appelées *valuations* et permettent d'associer une valeur numérique à des caractéristiques structurelles du graphe. Le résultat du calcul de ces valuations peut être représenté sous la forme d'un histogramme à plusieurs dimensions. Celui-ci peut être visualisé à l'aide d'un algorithme de rendu de volume. L'utilisation de tels outils nous semble donc être une approche innovatrice pour explorer des données de type relationnel.

Le premier chapitre de cet ouvrage offre tout d'abord un portrait général du domaine de recherche qu'est la visualisation. Nous effectuerons ensuite une classification des algorithmes de visualisation d'information. Ces classes seront toutes représentées par quelques exemples. La seconde partie de ce chapitre portera sur la visualisation scientifique, plus particulièrement sur les algorithmes de rendu d'ensembles de données tridimensionnels. Ceux-ci sont divisés en deux classes : les algorithmes de rendu de surfaces et de rendu direct de volume. Pour chacune d'elles, nous présenterons quelques exemples.

Un système de visualisation ne peut être complet sans une interface graphique permettant à l'utilisateur d'interagir avec les données lui étant présentées. Le second chapitre vise donc la présentation de quelques concepts généraux d'interface. Pour chacun de ces concepts, un exemple est énoncé. On y introduira de plus le concept de

"contrôleur".

Suite à la revue de littérature présentée dans la première partie du mémoire, nous introduirons dans le troisième chapitre un système de visualisation permettant d'explorer un graphe à partir de valuations calculées sur celui-ci. Nous présenterons la notion de valuation de façon formelle ainsi que les techniques d'exploration de graphe y étant généralement rattachées. Nous décrirons ensuite le fonctionnement de notre système de visualisation en commençant par le processus permettant de générer un volume à partir d'un triplet de valuations calculées sur un graphe, ainsi que les divers mécanismes permettant de le présenter à l'utilisateur. Nous introduirons de plus un contrôleur graphique permettant de naviguer à l'intérieur du volume de données et d'en extraire des régions d'intérêt. Celles-ci sont utilisées afin de filtrer les sommets et arêtes du graphe étudié. Nous conclurons ce chapitre par la présentation de l'interface graphique utilisée dans notre application.

Finalement, le dernier chapitre présente trois exemples d'application concrètes de notre système. Ceux-ci consistent essentiellement en l'extraction de composantes fortement connexes à l'intérieur des graphes étudiés. Comme nous le verrons, les cas étudiés sont tous facilement interprétables et offrent une bonne idée des possibilités de notre application.

CHAPITRE 1

VISUALISATION

1.1 Introduction

La visualisation [19] est un domaine de recherche encore jeune qui étudie les représentations graphiques, souvent interactives, que l'on peut produire à partir de données afin de les analyser. L'objectif visé par cette approche est de permettre une analyse plus complète, que ce soit en mettant en valeur les propriétés statistiques des données ou en les manipulant de façon interactive.

À ce jour, la littérature scientifique distingue deux types de visualisation : la *visualisation scientifique* et la *visualisation d'information.* La distinction entre ces deux branches n'est pas toujours très claire et on peut se demander si cette séparation est justifiée ou si elles devraient plutôt être considérées comme équivalentes. Cette disparité, provenant principalement d'un problème de nomenclature, est énoncée dans [55] :

> *Scientific visualization isn't uninformative, and information visualization isn't unscientific.*

De façon générale, on peut définir la *visualisation d'information* [12][56] comme étant appliquée à des données statistiques ou abstraites (nominales, financières, ...) pouvant comporter un grand nombre de variables tandis que la *visualisation scientifique* [50] est appliquée à des données physiques pour lesquelles il existe généralement

une relation spatio-temporelle intrinsèque (volume de données, visualisation de flux, simulations en ingénierie, ...).

Un des objectifs de ce mémoire est de faire l'étude de méthodes de visualisation de données et d'interaction. Un second vise l'introduction d'une méthode exploitant certains aspects de la visualisation scientifique dans le contexte de la visualisation d'information. Notre approche consiste essentiellement à utiliser des méthodes de visualisation de volumes, généralement associées au domaine de la visualisation scientifique, pour explorer des données statistiques calculées sur des graphes afin d'en extraire de l'information.

La section (1.2) présente une série d'algorithmes de visualisation multidimensionnelle utilisés en visualisation d'information. La section (1.3) traite d'algorithmes de visualisation scientifique, plus particulièrement d'algorithmes de visualisation de données volumétriques et de rendu de surface permettant de produire des représentations graphiques d'ensembles de données tridimensionnels.

1.2 Visualisation d'information

Les algorithmes présentés dans cette section ont pour but la visualisation d'ensembles de données constitués de plusieurs attributs. Nous utiliserons la classification proposée par Keim dans [41]. Une technique de visualisation est caractérisée par trois critères : le *type de données* à visualiser, la *technique de visualisation* et la *technique d'interaction.* En visualisation d'information, un ensemble de données est caractérisé en partie par le nombre de dimensions (attributs) qui le définit. On distingue six types de données à visualiser :

Unidimensionnel Une seule variable est associée à une valeur. On peut représenter ces ensembles de données en utilisant des courbes. Un ensemble de données représentant des mesures par rapport au temps en est un exemple : chaque point

sur la ligne de temps est affecté à une valeur.

Bidimensionnel L'ensemble de données est formé de points définis par deux variables. Ceux-ci sont facilement représentables par des images ou des graphiques à deux axes (plan cartésien). La position des villes sur une carte géographique est un exemple d'ensemble de données bidimensionnel. Celle-ci est déterminée par deux paramètres, soit la longitude et la latitude.

Multidimensionnel Tout ensemble de données étant défini par trois attributs ou plus est dit multidimensionnel. Ces données peuvent provenir de statistiques, de systèmes bancaires, de bases de données, etc. On définit un ensemble de données à n dimensions comme un vecteur $< x_1, x_2, x_3, \ldots, x_n >$ où chaque x_i représente un attribut.

Texte Livres, documents hypertextes ou autres formes de documents textuels. On ne peut caractériser ce type de données par son nombre d'attributs. Celles-ci sont difficilement représentables par des techniques conventionnelles car elles ne sont pas de nature numérique.

Graphes et hiérarchies Ensemble de données représentant des relations. Un graphe est en fait un ensemble de sommets reliés par des arêtes représentant les relations entre ces éléments. La section (3.2) offre une description détaillée de cette structure de données. Les documents [8] et [33] présentent plusieurs algorithmes reliés aux dessinage de graphes. En génie logiciel, les graphes sont aussi utilisés pour supporter la conception de gros logiciels en modélisant les relations entre modules ou le flot d'information dans un algorithme. Dans ce mémoire, nous décrirons une méthode permettant d'explorer ce type de données.

Un des plus grands défis en visualisation d'information est de présenter les données de manière à ce que l'utilisateur du système puisse interpréter plus facilement

l'information qu'il veut en retirer. Cette tâche est relativement triviale lorsqu'on considère une fonction à une ou deux dimensions, mais devient plus complexe pour les fonctions à trois dimensions et plus. En effet, le cerveau humain est incapable d'interpréter facilement plus de trois dimensions. Il n'est cependant pas rare, en visualisation d'information, de travailler avec des ensembles de données à plus de trois attributs. Dans la majorité des cas, ces données sont recueillies en raison de leur éventuelle utilité, mais il n'est généralement pas bénéfique pour l'utilisateur de les considérer dans leur ensemble pour une étude donnée. La visualisation d'information permet de détecter les traits particuliers (ex. propriétés statistiques ou données irrégulières) d'un ensemble de données et par le fait même en facilite la compréhension. De plus, il est impératif qu'un système de visualisation d'information permette une certaine interaction avec la représentation graphique afin que l'utilisateur puisse naviguer à travers ces données. De tels systèmes sont utilisés dans divers domaines, allant de la détection de fraude à l'analyse de données de simulation en ingénierie. Keim effectue une classification dans [41] et regroupe les méthodes de visualisation en six classes distinctes. En voici un résumé.

1.2.1 Affichages 2D/3D standard

Ces affichages sont utilisés pour représenter des ensembles de données à une, deux ou trois variables. Parmis ceux-ci, on retrouve les diagrammes à barres, les graphiques à courbes et les images. Ces méthodes d'affichage sont très répendues et sont souvent utilisées coinjointement à d'autres méthodes plus sophistiquées afin de créer des systèmes de visualisation complets.

1.2.2 Affichages géométriquement transformés

Le principe des affichages géométriquement transformés est d'appliquer des transformations géométriques aux données de façon à en faire ressortir certains aspects. Les

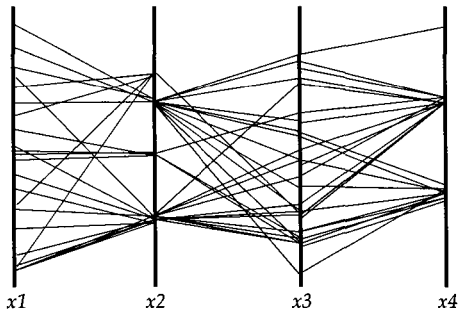


Figure 1.1: Coordonnées parallèles.

méthodes d'analyse statistique exploratoire (voir [28]) comme les *matrices de nuages de points* sont une forme d'affichage géométriquement transformé. Parmis les autres, on retrouve les *coordonnées parallèles*, les *matrices de nuages de points* ainsi que les *hyper coupes*. Les sections suivantes présentent ces trois techniques d'affichage.

1.2.2.1 Coordonnées parallèles

En 1987, Inserlberg et Dimsdale introduisent la visualisation par coordonnées parallèles [36]. Cette méthode consiste à représenter chacune des n coordonnées d'un espace donné par un ensemble d'axes parallèles. Les points à n dimensions sont ensuite représentés comme une ligne polygonale parcourant chacun des n axes (Fig 1.1). Chaque segment de la ligne polygonale relie deux axes et le point d'intersection de la ligne sur un axe représente la valeur du point à n dimensions pour la coordonnée correspondante. Il est à noter que l'ordre dans lequel les axes sont disposés joue un rôle important quant aux résultats observés. Les coordonnées parallèles permettent de déceler facilement des propriétés géométriques multidimensionnelles à l'intérieur d'un affichage à deux dimensions. Par exemple, il existe une dualité intéressante entre les coordonnées parallèles et le plan cartésien : un point d'intersection de plusieurs lignes polygonales entre deux axes signifie que les points correspondants sont colinéaires dans le plan cartésien définit par ces deux dimensions. Cette caractéristique particulière des

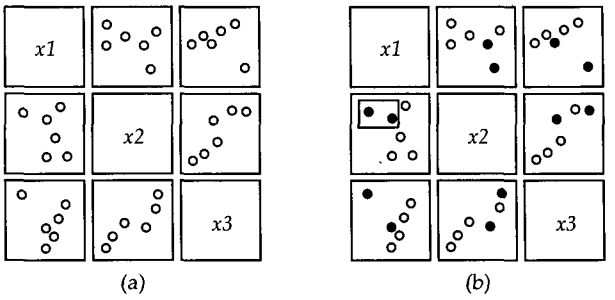


Figure 1.2: Matrices de nuages de points : Une opération de sélection est effectuée en (b) sur un nuage (*brushing*) et les points correspondant sont sélectionnés dans les autres nuages (*linking*) [74].

coordonnées parallèles nous permet entre autre de détecter facilement la corrélations entre deux variables d'un ensemble de données multidimensionnel.

1.2.2.2 Matrices de nuages de points

Une autre méthode permettant de visualiser une fonction à plusieurs variables est de considérer l'ensemble des nuages de points bidimensionnels formés par chaque paire de variables. Les nuages de points sont disposés dans une matrice de taille n où n est le nombre de variables (attributs) de l'ensemble de données. Les matrices de nuages de points (*scatterplots matrices*) [13] sont largement utilisées en statistiques. La figure 1.2a présente un exemple de matrices de nuages de points d'un ensemble de données tridimensionnel. Étant donné un ensemble de variables $x_1, x_2, \ldots, x_n$, l'élément ij de la matrice est le nuage de points formé par les variables x_i et x_j. Les cases de la diagonale principale de la matrice sont généralement utilisées pour identifier les variables. De plus, le nuage de points formé par les variables x_i et x_j est équivalent au nuage de points formé par x_j et x_i pour lequel ont aurrait inversé les axes.

Les matrices de nuages de points nous donnent l'opportunité d'introduire deux nouvelles notions : le *brushing* et le *linking*. Le *brushing* [49] consiste essentiellement en la spécification d'une zone de sélection généralement de forme rectangulaire à l'intérieur

de l'espace d'affichage. Les données à l'intérieur de cette zone sont généralement mises en évidence, par exemple en utilisant des ronds pleins pour les points sélectionnés et des ronds vides pour les points non sélectionnés. Le *linking* consiste à étendre la sélection effective dans d'un nuage de points aux autres nuages de points de la matrice (Fig 1.2b). Ceci permet entre autre d'analyser les relations entre deux variables x_i et x_j pour une région donnée du nuage de points de deux autres variables x_k et x_l. Ces techniques d'interaction seront présentées de façon plus détaillée dans le chapitre 2.

1.2.2.3 Hyper coupes

Les hyper coupes (*hyper slices*) s'apparentent aux matrices de nuages de points. Introduites en 1993 par van Wijk et van Liere [67], elles permettent de visualiser une fonction scalaire f à n variables aussi à l'aide d'une matrice de taille n. Cette fois-ci, les nuages de points sont remplacés par des images S_{ij} (où i et $j \in \{1, \ldots, n\}$), en tons de gris ou en couleur, représentant des coupes de la fonction f pour laquelle ont aurait fixé les variables autres que i et j (Fig 1.3). Les coupes diagonales S_{ii} sont utilisées pour présenter les fonctions à une variable. Au lieu de présenter les données dans leur ensemble, les hyper coupes nous permettent de visualiser une région d'intérêt autour d'un point focal donné dans l'espace multidimensionnel. Plus formellement, soit $\mathbf{c} = (c_1, c_2, \ldots, c_n)$, un point focal, et w_i, la taille de la région d'intérêt le long de la variable i, la région d'intérêt pour la variable i est définie par $R_i = [c_i - w_i/2, c_i + w_i/2]$. Les données non visibles dans une région spécifique peuvent être visualisées en redimensionnant celle-ci ou en déplaçant le point focal $\mathbf{c}$ vers une autre position. Notons trois relations intéressantes entre les coupes composant cette matrice :

- Toute coupe S_{ij} est exactement la même image que la coupe S_{ji} pour laquelle on aurrait inversé les axes.
- La valeur de la fonction $f(\mathbf{c})$ au point focal est la même au centre de chaque

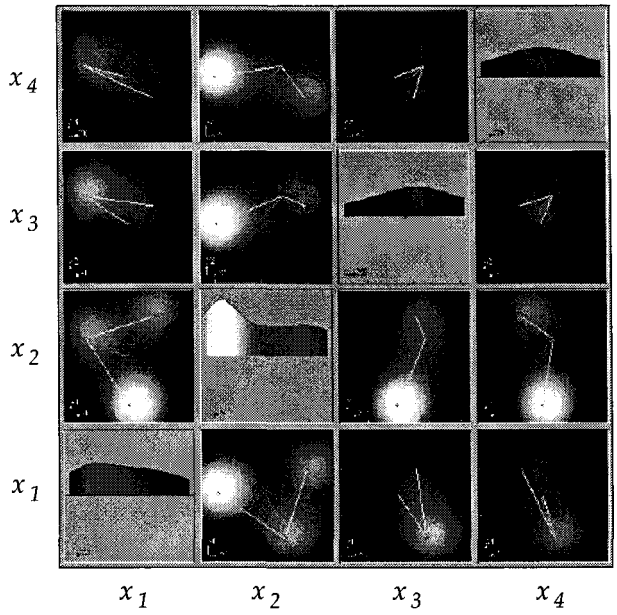


Figure 1.3: Hyper coupe d'une fonction à quatre variables [67].

coupe (autrement dit, chaque coupe est centrée au point focal **c**).

- Les valeurs situées au long d'une droite horizontale passant à travers le centre de chaque coupe sur une rangée sont identiques et il en est de même pour les valeurs situées le long d'une droite verticale passant à travers le centre de chaque coupe le long d'une colonne.

Le point focal peut être déplacé par l'utilisateur en cliquant sur une coupe et en glissant le curseur à l'aide de la souris. De cette façon, la région d'intérêt est déplacée dans la même direction et l'effet de cette translation est appliqué respectivement à l'ensemble des autres coupes. De plus, van Wijk et van Liere proposent un outil permettant de créer des points de repère dans l'espace multidimensionnel. Ces points sont créés à partir de la position du point focal et sont reliés à l'aide de chemins comme on le voit dans la figure 1.3. L'utilisateur peut ainsi garder une trace de son parcours à l'intérieur de l'ensemble de données et revenir à tout moment dans une région d'intérêt déjà visitée.

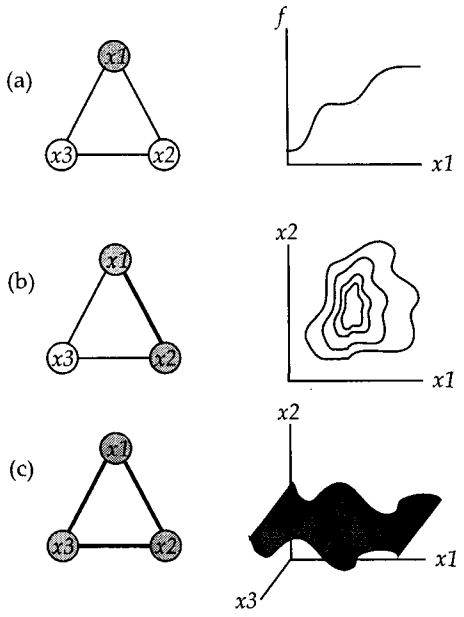


Figure 1.4: Hyper cellule d'une fonction à trois variables [22]. Construction itérative en (a), (b) et (c) d'une cellule à trois dimensions.

1.2.2.4 Hyper cellules

En 2002, dos Santos et Brodlie [22] introduisent la méthode des hyper cellules (*hyper cells*) qui reprend l'idée des hyper coupes en présentant des régions d'intérêt d'un espace multidimensionnel. Cette fois-ci, l'ensemble de données n'est pas présenté sous forme de matrice de coupes bidimensionnelles. On utilise plutôt des boîtes (ou cellules) correspondant à des sous-espaces bornés à une, deux ou trois dimension(s) du domaine de la fonction scalaire f et contenant une évaluation de celle-ci. Ces cellules correspondent à des régions d'intérêt et sont définies essentiellement de la même manière que celles utilisées dans les travaux de van Wijk et van Liere [67]. Plus précisément, la région d'intérêt pour une variable i est définie par $R_i = [c_i - w_i/2, c_i + w_i/2]$, où $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ est un point focal à n coordonnées et w_i est la taille de la région d'intérêt le long de la variable i.

Les cellules sont construites dynamiquement à partir d'une interface basée sur un

graphe complet dont les sommets représentent les attributs de l'ensemble de données. Ce graphe est appelé graphe d'interaction. La sélection des attributs formant le sous-espace est effectuée progressivement en cliquant sur les sommets du graphe. Lorsqu'un sommet est sélectionné, on considère le sous-espace formé de l'attribut correspondant et des autres attributs sélectionnés jusqu'à un maximum de trois. La représentation graphique de cette cellule est alors mise à jour (Fig 1.4).

Lorsqu'une seule variable est sélectionnée, la fonction f évaluée dans le sous-espace correspondant est présentée à l'aide d'un graphique de courbe. Lorsque deux variables sont sélectionnées, elle est présentée sous forme d'image et lorsque trois variables sont sélectionnées, on obtient un volume qui peut être présenté à l'aide d'un algorithme de rendu de volume (voir section 1.3). Le graphe d'interaction permet aussi d'effectuer des rotations sur les cellules et de modifier leur taille le long de chaque variable en utilisant des curseurs disposés sur les arêtes du graphe. Chaque cellule instanciée par l'utilisateur est stockée dans un espace de travail. Celles-ci peuvent ensuite y être modifiées ou supprimées. Contrairement aux hyper couches, ce système présente à l'utilisateur des cellules formées d'au plus trois variables, ce qui permet de déceler des caractéristiques sur l'ensemble de données qu'il ne serait possible d'observer qu'en visualisant plusieurs couches bidimensionnelles. De plus, ce système permet à l'utilisateur d'étudier des régions d'intérêt dont le point focal est différent à un même moment en instanciant plusieurs cellules simultanément.

1.2.3 Affichages emboîtés

L'idée générale de la visualisation par affichages emboîtés est de séparer les variables d'un ensemble de données en plusieurs sous-ensembles de façon hiérarchique (les variables sont alors regroupées selon leurs similarités). De cette façon, on peut visualiser les données en "emboîtant" le système de coordonnées d'un groupe de variables

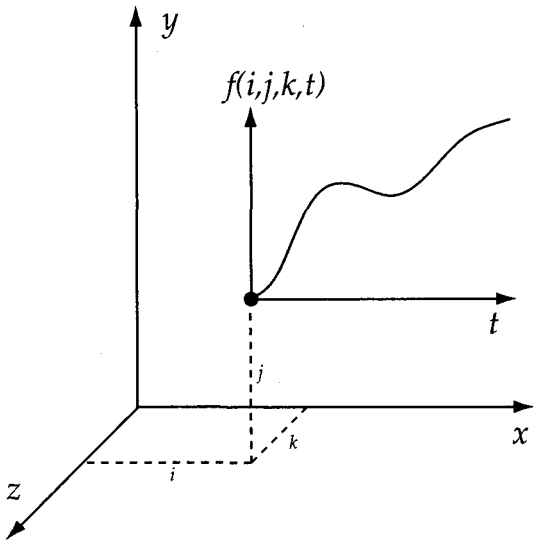


Figure 1.5: Visualisation d'une fonction scalaire à quatre variables à l'aide de *Worlds within worlds.*

à l'intérieur de celui d'un autre groupe hiérarchiquement plus haut. Deux exemples de méthodes utilisant ce principe sont présentés dans les sections suivantes.

1.2.3.1 "Worlds within worlds"

En 1990, Feiner et Beshers introduisent une méthode de visualisation nommée *Worlds within worlds* [25]. Elle consiste à séparer les variables constituant un ensemble de données en groupes de une, deux ou trois variables et à inclure la représentation graphique du sous-espace formé par ces groupes à l'intérieur de celle formée par d'autres groupes. Ces représentations sont appelées *mondes* et peuvent correspondre à des graphiques à une dimension (une variable), des images (deux variables) ou des volumes (trois variables).

Par exemple, considérons une fonction scalaire f ayant pour paramètres quatre variables (x, y, z, t) où x, y et z correspondent aux coordonnées d'un point en trois dimensions et t est une variable représentant le temps. Étant donné que les variables x, y et z définissent l'emplacement d'un point dans un espace tridimensionnel, il nous est

naturel de les regrouper. Considérons maintenant un point $\mathbf{p} = (i, j, k)$. L'évaluation de la fonction f dont les paramètres x, y et z ont été fixés au point $\mathbf{p}$ peut être représentée par un graphique à une dimension, emboîté dans le système de coordonnées des dimensions x, y et z. Le point d'origine de ce graphique correspond au point $\mathbf{p}$ de l'espace tridimensionnel dans lequel il a été inclu. (Fig 1.5). La variable t reste alors la seule n'étant pas fixée. L'utilisateur du système peut déplacer le point $\mathbf{p}$ pour explorer d'autres régions de l'ensemble de données et la représentation du graphique courbe est alors mise à jour.

1.2.3.2 Dimensions emboîtées

Une autre technique basée sur ce principe est introduite en 1990 par LeBlanc, Ward et Wittels [44]. L'affichage à dimensions emboîtées consiste à présenter un ensemble de données à n variables à l'intérieur d'un espace bidimensionnel. Pour ce faire, la représentation d'une paire de variables est incluse à l'intérieur de celle d'une autre paire de variables, et ce, de façon itérative. Considérons un ensemble de données à n variables $(x_1, x_2, \ldots, x_n)$. On considère que la plage de valeurs que peut prendre chacune de ces variables est bornée. La première étape de cette technique consiste à discrétiser ces intervalles en un certain nombre de sous-intervalles de taille identique. On appelle *cardinalité* d'une coordonnée le nombre de sous-intervalles lui étant assigné. Ce nombre est affecté de façon arbitraire. Plus la cardinalité d'une coordonnée est élevée, plus sa résolution sera élevée et s'approchera de la continuité. Inversement, une cardinalité faible correspond à la forte discrétisation d'une variable. Ces sections discrètes serviront de zones d'affichage virtuelles pour les autres coordonnées. On regroupe ensuite ces variables en paires selon leur cardinalité et on leur assigne une orientation (horizontale ou verticale). À la première itération, les deux variables dont la cardinalité est la plus faible seront utilisées pour diviser l'espace d'affichage en section discrètes. Celle avec la

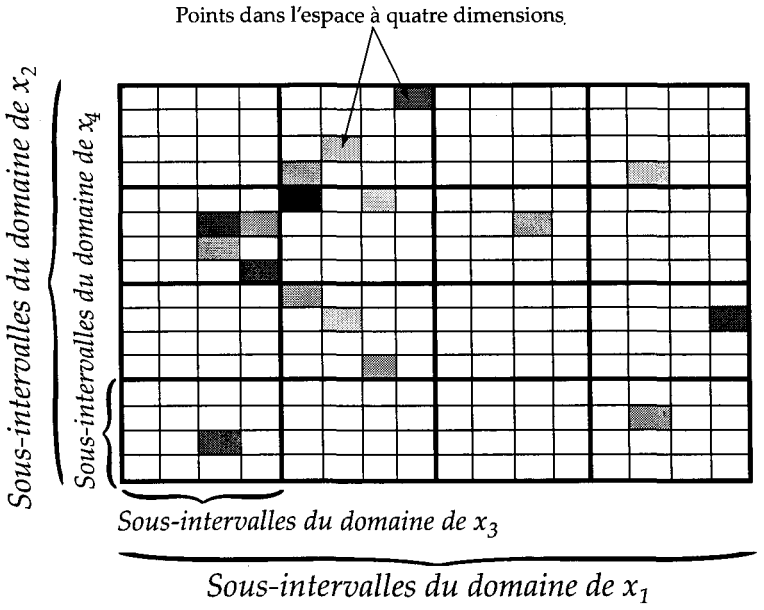


Figure 1.6: Ensemble de données à quatre variables rendus à l'aide de la technique des dimensions emboîtées. Les sous-intervalles des variables x_1 et x_2 sont utilisés pour diviser l'espace d'affichage tandis que les sous-intervalles des variables x_3 et x_4 sont utilisés pour diviser les sous-intervalles des variables x_1 et x_2.

cardinalité la plus élevée servira d'axe horizontal et l'autre d'axe vertical. Le processus est ensuite répété avec les deux autres variables restantes dont la cardinalité est la plus basse (Fig 1.6). Une fois toutes les coordonnées emboîtées les unes aux autres, il suffit d'assigner les valeurs des points à n dimensions aux intensités des pixels correspondants.

1.2.4 Affichages basés sur icônes

Les affichages basés sur icônes sont constitués de petits objets graphiques changeant d'apparence selon la valeur des données qu'ils représentent. Chaque icône (ou *glyph*) est indépendant et représente un point dans l'espace multidimensionnel. Deux exemples typiques d'affichages basés sur icônes sont les *starplots* et les *visages de Chernoff*.

1.2.4.1 Starplots

Les *starplots* [38] consistent en un ensemble de n axes tous reliés à un point d'origine et disposés de façon uniforme autour de celui-ci afin de former une étoile. Chaque axe représente une des n variables. Un point dans l'espace à n dimensions est représenté par des segments de droite reliant chaque paire d'axes adjacents et traversant l'ensemble des axes. Le point d'intersection d'un segment de droite sur un axe correspond à la valeur de la variable associée pour le point représenté par ce *starplot.*

1.2.4.2 Visages de Chernoff

Les visages de Chernoff [15] consistent en une manière un peu plus sophistiquée de représenter un point dans un espace multidimensionnel. On associe chacune des n variables à une propriété d'un petit visage. Ces propriétés peuvent être la longueur de la bouche, l'inclinaison des yeux, la largeur des yeux, la grandeur des sourcils, etc. La combinaison de l'ensemble de ces caractéristiques produit des visages présentant diverses émotions pouvant être interprétés facilement par l'être humain. Par exemple dans le cas de données financières, un visage ayant l'air heureux pourrait représenter des données prospères tandis qu'un visage ayant l'air mécontent pourrait être associé à des données préocupantes.

1.2.5 Affichages denses en pixels

L'objectif des affichages denses en pixels [40][42] est de présenter le plus de données possible sur un dispositif d'affichage à deux dimensions (e.g. écran) en utilisant les plus petits éléments d'image disponibles pour ce dispositif (e.g. pixels). L'approche générale consiste à diviser l'espace d'affichage en n sections de même taille, appelées fenêtres, et d'associer à chacun des n attributs d'une donnée, un pixel coloré selon sa valeur à l'intérieur de ces sections. L'avantage de cette technique est qu'elle permet

l'affichage d'un très grand nombre de données (autant de données qu'il y a de pixels disponibles dans l'espace d'affichage). Les données sont généralement ajoutées aux fenêtres selon un certain ordre en suivant des patrons [42] paramétrables par l'utilisateur. La figure 1.7a est un schéma démontrant le fonctionnement de cette technique d'affichage. S'il existe une relation d'ordre inhérente aux éléments de l'ensemble de données

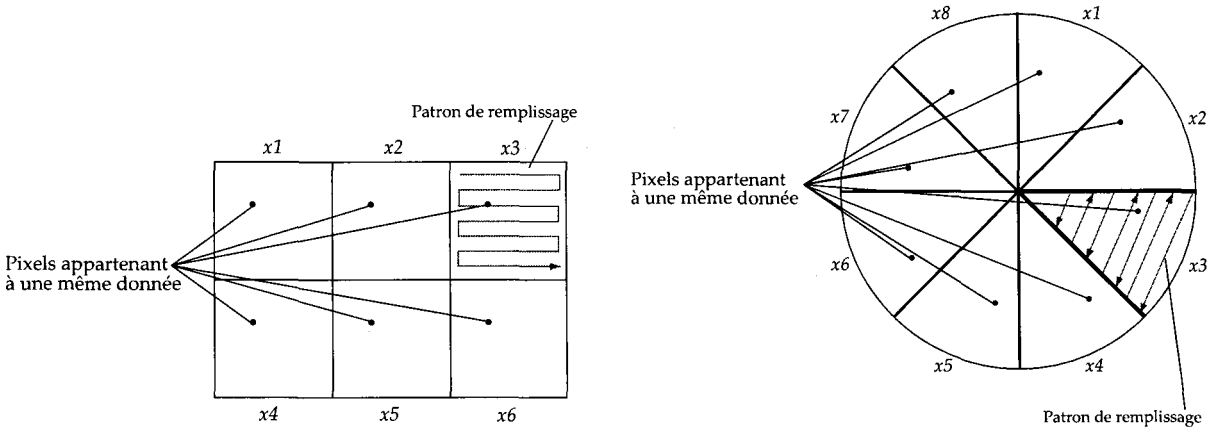


Figure 1.7: Affichages denses en pixels : (a) Fenêtrage de l'espace d'affichage [40], (b) Affichage d'un ensemble de données à huit attributs par la méthodes du cercle [5].

à visualiser (ex. données ordonnées par rapport au temps), on utilise cette dernière pour disposer les données dans l'espace d'affichage. Sinon, on utilise des requêtes. Les techniques utilisant des requêtes permettent de distribuer les données dans l'espace d'affichage selon leur pertinence par rapport aux critères de la requête en faisant généralement ressortir les données plus significatives au centre de l'espace d'affichage. De plus, plutôt que de colorer un pixel selon la valeur de l'attribut correspondant, on lui associe une couleur selon sa pertinence par rapport à cette requête.

Le *cercle segmenté* est un exemple de technique de visualisation dense en pixels basée sur des requêtes [5]. On restreint l'espace d'affichage à un cercle et celui-ci est divisé en n segments de cercles de taille équivalente. Le patron de remplissage utilisé

pour cette technique consiste à ajouter des données en suivant des trajectoires perpendiculaires à la bissectrice de ce segment du centre vers l'extérieur. La figure 1.7b illustre cette technique.

1.3 Visualisation scientifique

Cette section traitera plus particulièrement de la visualisation d'ensembles de données tridimensionnels (ou volumes), car c'est cette branche de la visualisation scientifique qui nous intéresse. Un ensemble de données tridimensionnel peut être vu comme une fonction scalaire f, continue dans $\mathbb{R}^3$. Même si ces données proviennent de sources dont le domaine est continu (ie. photographie, tomographies, ...), l'informatique nous permet uniquement de les manipuler dans le domaine discret. L'échantillonnage est un processus nous permettant de remédier à ce problème. En effet, en rééchantillonnant une fonction scalaire f sur une grille tridimensionnelle régulière, on obtient un volume discret. Les sommets de cette grille sont appelés *voxels* et correspondent à des éléments de volume. Dans la prochaine section, nous décrirons les algorithmes de rendu de surface permettant de visualiser des surfaces extraites de volumes à l'aide de primitives graphiques (triangles). Nous terminerons ce chapitre en présentant quelques algorithmes de rendu de volume, permettant de visualiser directement un ensemble de données tridimensionnel.

1.3.1 Rendu de surface

Introduit dans les années 1970, le rendu de surface est la première approche développée pour la visualisation de volumes. Ces algorithmes ne permettent pas de dessiner directement un volume, mais permettent plutôt de visualiser une *iso-surface* extraite de celui-ci. Une iso-surface correspond à une surface de valeur constante (valeur seuil) à l'intérieur d'un volume. Plus formellement, soit f la fonction scalaire représentant le

volume et c la valeur définissant la surface à extraire, l'iso-surface correspondant à la valeur c est définie par $\{(x, y, z) \in \mathbb{R}^3 | f(x, y, z) = c\}$. Le choix de c permet à l'utilisateur de sélectionner une zone d'intérêt ou un objet particulier à l'intérieur du volume. Ces surfaces sont rendues à l'écran à l'aide de maillages polygonaux.

Les algorithmes de rendu de surface ont été développés à l'origine pour visualiser des volumes de données provenant du domaine médical (angiographie, imagerie par résonance magnétique, tomographie assistée par ordinateur, ...). Par exemple, en imagerie médicale, les données recueillies à partir d'une tomographie assistée par ordinateur correspondent à un volume pour lequel la valeur de chaque voxel représente une densité de matériau. En effet, un tel volume est constitué d'un certain nombre de matériaux de densité différente (muscle, os, air, ...). En sélectionnant une valeur c proche de la densité moyenne d'un certain matériau, il nous est possible d'isoler une surface correspondant aux contours décrits par celui-ci dans le volume. De cette façon, si nous étudions un volume correspondant à la tomographie d'un patient, il nous est possible de sélectionner la valeur de densité moyenne de l'os afin d'en isoler le contour correspondant au squelette. Les sections suivantes présentent deux algorithmes de rendu de surface.

1.3.1.1 Triangulation de courbes de contour

En 1975, Keppel introduit un algorithme [24] permettant de construire des surfaces sous forme de maillages polygonaux à partir de courbes de contour délimitant des organes ou autres zones de densité hétérogènes à l'intérieur d'un sujet en imagerie médicale. Ces courbes peuvent être déterminées à l'aide de patrons (e.g. comparaison entre un atlas de référence médical et une radiographie) et correspondent à l'intersection de la surface d'un objet 3D et d'une coupe 2D du volume. En effet, on peut interpréter un volume comme une série de coupes (images 2D) ordonnées le long de

l'axe z. En gros, l'algorithme consiste à relier des points appartenant à la courbe de contour d'une coupe à ceux de coupes voisines afin de former des triangles qui constitueront un maillage polygonal correspondant à une approximation de la surface à extraire. Ces points sont distribués arbitrairement le long des courbes de contour. Évidemment, pour un ensemble de points donné, il existe plusieurs triangulations possibles et plus le nombre de points choisis est grand, plus la précision du modèle est élevée. L'arrangement de triangles optimal est celui produisant le polyèdre dont le volume est maximal. Ce dernier peut-être obtenu en divisant les ensembles de points formant les contours en sous-ensembles contenant des points consécutifs et en considérant la triangulation optimale pour chacun de ces sous-ensembles. Ces divisions sont effectuées en plusieurs itérations, en effectuant à chaque fois un parcours des points de la courbe de contour dans le sens anti-horaire, et consistent à regrouper les points d'une courbe de contour en sous-ensembles convexes ou concaves.

Plus formellement, considérons une paire de courbes de contour (A, B). On leur associe arbitrairement un ensemble de points $S^A = \{A_1, A_2, \ldots, A_m)$ et $S^B = \{B_1, B_2, \ldots, B_n\}$. Ces ensembles de points sont ordonnés sur chaque courbe dans le sens anti-horaire. Chaque sous-ensemble de points trouvée sur la courbe A est nommée $S^A_{(i,j)}$ où $i = 1, 2, \ldots$ est le nombre d'itération courante et $j = 1, 2, \ldots$ numérote les sous-ensembles. La première itération consiste donc à trouver le premier sous-ensemble convexe $S^A_{(1,1)}$. Celui-ci est déterminée en éliminant tout point A_i pour lequel la courbure passant par $A_{i-1}A_iA_{i+1}$ est concave (négative). La deuxième itération consiste ensuite à extraire les sous-ensembles convexes des sous-ensembles concaves $S^A_{(2,j)}$. Ce processus est ensuite répété récursivement pour chaque sous-séquence. On effectue la même division en sous-ensembles $S^B_{(i,j)}$ pour la courbe de contour B et son ensemble de points S^B. La figure 1.8 illustre cette décomposition.

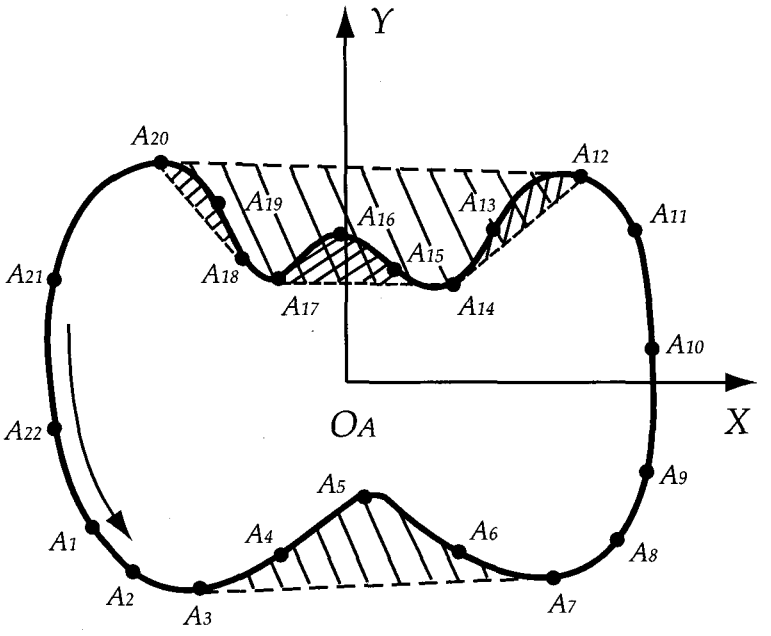


Figure 1.8: Décomposition itérative d'un ensemble de points sur une courbe contour en sous-ensembles convexes et concaves [24] :
- Sous-ensemble convexe (première itération)
  $S^A_{(1,1)} = \{A_1, A_2, A_3, A_7, A_8, A_9, A_{10}, A_{11}, A_{12}, A_{20}, A_{21}, A_{22}\}$
- Sous-ensembles concaves (deuxième itération)
  $S^A_{(2,1)} = \{A_3, A_4, A_5, A_6, A_7\}$
  $S^A_{(2,2)} = \{A_{12}, A_{14}, A_{17}, A_{18}, A_{20}\}$
- Sous ensembles convexes (troisième itération)
  $S^A_{(3,1)} = \{A_{12}, A_{13}, A_{14}\}$
  $S^A_{(3,2)} = \{A_{14}, A_{15}, A_{16}, A_{17}\}$
  $S^A_{(3,3)} = \{A_{18}, A_{19}, A_{20}\}$.

Tel que nous l'avons mentionné, la triangulation optimale de deux courbes de contour, celle approximant le plus fidèlement la surface à extraire, correspond au polyèdre dont le volume est maximal. Afin de déterminer cette configuration, on divise le polyèdre formé des points des deux courbes de contour en une série de polyèdres à six faces. En maximisant le volume de ces derniers pour chaque paire de coupes, nous obtiendrons l'agencement optimal. Soit $P^A = A_i A_{i+1} B_j O_A O_B$ un polyèdre formée de deux points sur la courbe de contour A et d'un point sur la courbe de contour B et

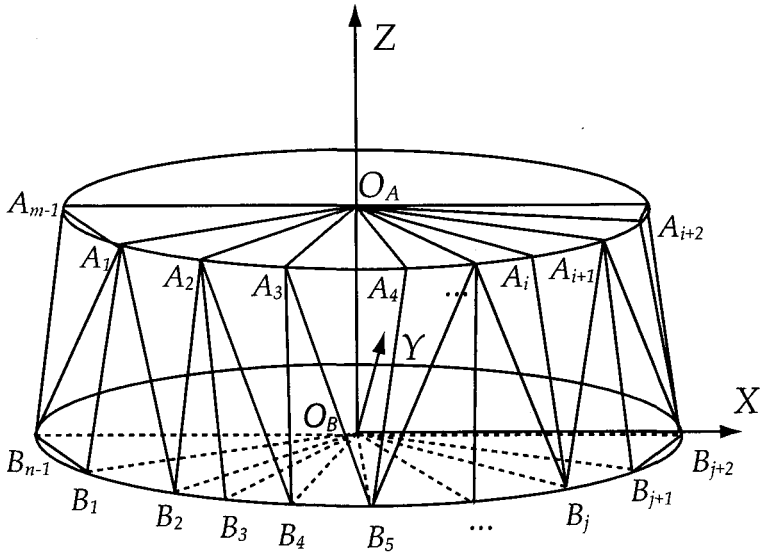


Figure 1.9: Triangulation d'une paire de courbes de contour [24].

$P^B = A_i B_j B_{j+1} O_A O_B$, un polyèdre formée d'un point sur A et de deux points sur B. O_A et O_B sont des points d'origine placés arbitrairement sur leur coupe respective (Fig. 1.9). Le volume du polyèdre formé par les points de la paire de courbes de contour (A, B) correspond à la somme des volumes de l'ensemble des polyèdres P^A et P^B :

$$vol_{total} = \sum_A vol(P^A) + \sum_B vol(P^B) \ . \tag{1.1}$$

On énumère ensuite l'ensemble des triangulations possibles pour les courbes de contour A et B. Considérons une matrice binaire M de taille $n \times m$ pour laquelle la valeur de l'élément a_{ij} est 1 si le i-ème point de S^A est relié au j-ème point de S^B et 0 sinon. Les triangles doivent se conformer aux règles suivantes :

1. Un triangle est composé de deux points adjacents sur une même courbe de contour et d'un autre point sur la courbe de la coupe voisine. Ce qui signifie qu'un triangle est représenté dans la matrice M par deux valeurs 1 adjacentes.

2. Les jonctions entre les tranches ne peuvent s'entrecroiser.

3. Chaque point d'une courbe de contour doit faire partie d'au moins une jonction avec la courbe de la coupe voisine.

La triangulation complète d'une paire de courbes de contour correspond donc à un chemin de valeur 1 joignant les éléments a_{11} à a_{mn} de M. Considérons le cas de base où les deux courbes de contour A et B sont constituées d'un ensemble de points entièrement convexe. On associe un graphe $G = (P, N)$ à la matrice M tel que le sommet P_{ij} du graphe correspond à l'élément a_{ij} de la matrice M. Une arête joignant les sommets P_{ij} et $P_{i,j+1}$ correspond donc au triangle $A_iB_jB_{j+1}$ et, similairement, une arête joignant les sommets P_{ij} et $P_{i+1,j}$ correspond au triangle $A_iA_{i+1}B_j$. De cette façon, un chemin reliant les sommets P_{11} et P_{mn} dans G correspond à une triangulation complète des courbes de contour des coupes A et B. L'ensemble de ces chemins représente l'ensemble des triangulations possibles. On doit ensuite déterminer lequel de ces chemins maximise le volume du polyèdre engendré par la triangulation correspondante. Pour ce faire, pondérons le graphe G tel que la valeur d'un arc correspond au volume du tétraèdre associé. Plus formellement, on associe une valeur v_{ij}, correspondant au volume du tétraèdre formé par les points $A_iA_{i+1}B_jO_B$ ou $A_iB_jB_{j+1}O_A$, aux arcs N_{ij} joignant les sommets $(P_{ij}P_{i,j+1})$ ou $(P_{ij}P_{i+1,j})$ du graphe. La triangulation optimale est alors obtenue en trouvant le chemin de poids maximal entre P_{11} et P_{mn} (Fig 1.10a).

Dans un cas plus général, où les courbes de contour sont formés de sous-ensembles convexes et concaves, les sous-ensembles $S^A_{(1,1)}$ et $S^B_{(1,1)}$ obtenues lors de la première itération du processus de division sont traitées en premier. Les sommets et arcs du graphe correspondant à des points exclus des sous-ensembles $S^A_{(1,1)}$ et $S^B_{(1,1)}$ sont alors ignorés. Une fois le chemin maximal reliant les sommets P_{11} et P_{mn} du graphe G trouvé pour ces ensembles de points convexes, on répète le même processus pour la totalité des sous-ensembles trouvées lors de la deuxième itération du processus de division en

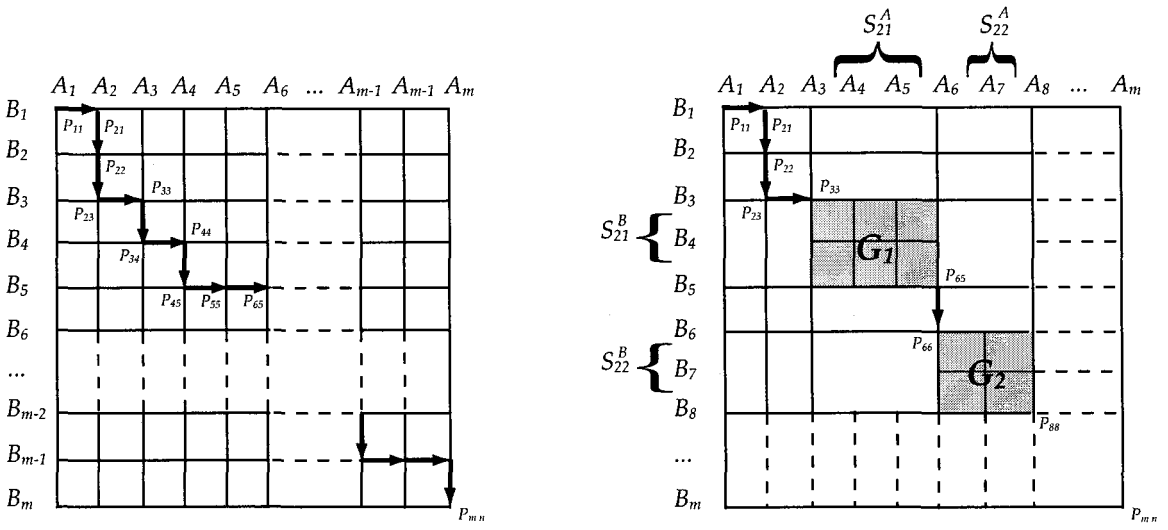


Figure 1.10: Le graphe $G = (P_{ij}, N)$ est associé aux ensembles de points S^A et S^B et permet de déterminer la triangulation maximisant le volume du polyèdre engendré par celle-ci [24]. En (a), la triangulation produisant le polyèdre de volume maximal pour un ensemble de points convexe correspond au chemin de poids maximal (en gras) reliant le sommet P_{11} au sommet P_{mn}. En (b), les ensembles A et B contiennent des sous-ensembles concaves. Ceux-ci sont représentés par les sous-graphes G_1 et G_2 et seront traités à la seconde itération.

trouvant le chemin minimal des sous-graphes associés à ces sous-ensembles car ceux-ci sont concaves (Fig 1.10b). Le processus est ensuite répété itérativement pour l'ensemble des sous-ensembles en calculant le chemin maximal de leur sous-graphe associé si ceux-ci sont convexes, ou le chemin minimal si ceux-ci sont concaves.

Le résultat produit par cet algorithme de rendu dépend fortement de l'ensemble de points utilisé pour effectuer la triangulation. Évidemment, plus l'ensemble de points utilisé est de grande taille, plus l'algorithme produira un résultat détaillé, et inversement.

1.3.1.2 Marching Cubes

L'algorithme des *Marching Cubes* est introduit par William et al. en 1987 [47]. L'idée de base de cet algorithme est de diviser le volume en cubes consitués de huit

voxels et de déterminer de quelle façon la surface à extraire traverse ces cubes. Celle-ci est ensuite représentée à l'aide d'un maillage de triangles. Considérons le volume comme étant constitué d'une série de coupes xy ordonnées sur l'axe z. Les cubes sont créés en sélectionnant quatre voxels (i, j, k), $(i+1, j, k)$, $(i, j+1, k)$ et $(i+1, j+1, k)$ sur la couche $z = k$ et les quatre correspondants $(i, j, k+1)$, $(i+1, j, k+1)$, $(i, j+1, k+1)$ et $(i+1, j+1, k+1)$ au niveau suivant $z = k+1$. On compare ensuite la valeur de chacun des voxels formant le cube à la valeur correspondant à la surface à extraire. Le sommet d'un cube est à l'intérieur de la surface si et seulement si la valeur du voxel correspondant est plus élevée ou égale à celle de la surface ciblée. On lui assigne alors la valeur 1 et dans le cas contraire, la valeur 0. La surface entre en intersection avec les arêtes des cubes pour lesquels les deux sommets correspondants ne sont pas de valeur binaire identique. Étant donné qu'un cube est constitué de huit sommets et que chacun de ces sommets peut être affecté de deux états possibles, il existe un total de 256 (2^8) configurations possibles d'intersection entre l'iso-surface et les segments d'un cube. En tenant compte des symétries, rotations et cas de complémentarité, on obtient quinze configurations distinctes (Fig 1.11). Par exemple, dans le cas où un seul sommet du cube est à l'intérieur de l'iso-surface, les trois arêtes y étant rattachées auront une intersection avec la surface. Un triangle sera alors défini par ces intersections (*cas 1* dans la Fig 1.11). L'ensemble de ces configurations sont ensuite énumérées dans une table de référence. Un index de huit bits est créé pour chacun des cubes (chaque voxel définissant le cube est associé à une valeur binaire). Cet index est utilisé comme pointeur dans une table déterminant quelles arêtes sont en intersection avec la surface selon la configuration donnée. Les points de contacts entre la surface et les arêtes peuvent ensuite être déterminés par interpolation linéaire. Pour chaque triangle, on calcule ensuite un vecteur normal en utilisant le gradient afin de pouvoir y appliquer des effets d'ombrage

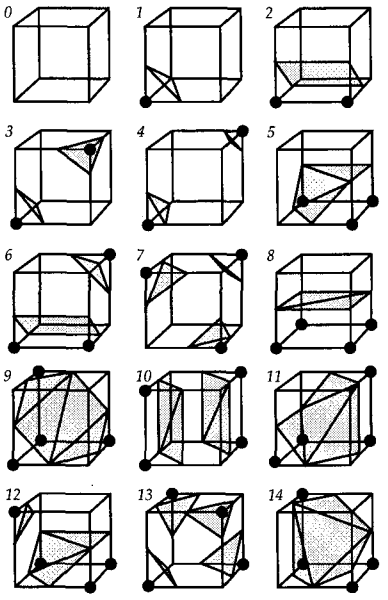


Figure 1.11: Les quinze configurations possibles qu'une surface peut prendre lorsqu'elle entre en intersection avec un cube [47].

et de lumière.

Le vecteur gradient $\overrightarrow{G}$ est essentiellement la dérivée première de la fonction de densité f (définissant le volume). Le gradient d'un sommet (i, j, k) est ainsi approximé par :

$$\begin{aligned} G_x(i,j,k) &= \frac{f(i+1,j,k) - f(i-1,j,k)}{\triangle x} \\ G_y(i,j,k) &= \frac{f(i,j+1,k) - f(i,j-1,k)}{\triangle y} \\ G_z(i,j,k) &= \frac{f(i,j,k+1) - f(i,j,k-1)}{\triangle z} \end{aligned} , \tag{1.2}$$

où $f(i, j, k)$ est la valeur du voxel (i, j, k) et $\triangle x$, $\triangle y$ et $\triangle z$ correspondent à la longueur d'une arête du cube sur l'axe x, y ou z.

Cet algorithme donne généralement des résultats assez réalistes lorsqu'on y intégre des effets d'ombrage et de lumière. Il est à noter que plus le nombre de voxels

composant le volume à étudier est grand, plus la surface résultante sera composée d'un grand nombre de triangles.

1.3.1.3 Autres algorithmes de rendu de surfaces

Les algorithmes de rendu de surface plus récents utilisent généralement les principes de base de l'algorithme des Marching Cubes. Lin et al. [46] présentent une alternative aux Marching Cubes, les Marching Voxels, permettant de mieux estimer l'emplacement de surfaces ainsi que la structure de celles-ci. Plutôt que de générer les surfaces à l'aide de cubes, on génère des triangles pour chaque voxel étant situé sur l'iso-surface à représenter. Ces triangles sont générés à l'aide des voxels voisins. Pour chaque voxel, un total de huit triangles sont générés (on divise l'espace centré au voxel en octants et on crée un triangle pour chacune de ces régions). Ceux-ci sont ensuite combinés afin de créer un maillage. En comparaison avec les Marching Cubes, l'algorithme de Marching Voxels est plus rapide et offre un résultat plus fidèle.

Le rendu de surface pose toutefois quelques désavantages. Pour une valeur seuil donnée, la plupart de ces algorithmes donnent une bonne approximation de l'iso-surface associée, mais ceux-ci effectuent une décision binaire quant à l'appartenance de chaque voxel aux sous-volumes délimités par cette surface. Considérons un volume consitué de deux objets, ou deux régions d'intérêt dans notre cas, se chevauchant. L'utilisation d'une classification binaire est moins appropriée si la transition entre ces deux objets est plutôt longue car il devient difficile de déterminer avec exactitude la valeur de surface exacte permettant de séparer ces deux objets. De plus, les algorithmes de rendu de surfaces présentent essentiellement un volume comme une série de surfaces "flottantes" dans un environnement vide. Ces particularités peuvent créer des artéfacts ou omissions de détails importants lors de la création de rendus.

1.3.2 Rendu de volume

Le rendu direct de volume [23] est introduit vers la fin des années 1980 notamment par les travaux de Marc Levoy et ceux des studios Pixar. Contrairement au rendu de surface où seulement des iso-surfaces représentant des régions d'intérêt extraites du volume sont présentées, le rendu de volume permet de visualiser une projection bidimensionnelle de celui-ci. À l'origine, ces algorithmes ont principalement été développés pour visualiser des données médicales. Ils peuvent également être utilisés pour produire des représentations graphiques de tout autre ensemble de données tridimensionnel réechantillonné sur une grille de points régulière (ie. volume). Dans notre cas, l'ensemble de données à visualiser est un nuage de points auquel on aurrait préalablement appliqué une convolution (voir chapitre 3).

Ces algorithmes consistent généralement en l'association d'une couleur et d'une opacité à chaque voxel et au calcul de la contribution de ces voxels sur un plan de projection bidimensionnel (plan image). Dans le cas de données médicales, la coloration des voxels est effectuée à l'aide d'un processus permettant de déterminer l'apport de chaque matériau (os, air, tissus, ...) pour un voxel donné et de lui associer une couleur et une opacité correspondant à un mélange entre les valeurs typiques définies préalablement pour chacun de ces matériaux. Ce processus est nommé *classification* et est décrit plus exhaustivement pour les algorithmes présentés dans cette section.

Les ensembles de données que nous visualiserons dans ce projet sont de nature statistique. Le processus de classification devient donc aberrent et est remplacé par le choix d'une fonction permettant d'associer directement la valeur d'un voxel à une couleur ou une opacité. Ces fonctions sont appelées *fonctions de transfert.* Plus précisément, pour une fonction de transfert d'opacité, considérons les deux ensembles $U = \{0, 1, 2, \ldots, max\}$ l'image de la fonction f représentant le volume et $V = [0, 1]$

l'intervalle des valeurs d'opacité. Un voxel dont la valeur d'opacité est égale à 0 est complètement transparent tandis que la valeur 1 correspond à une opacité complète. La fonction $t : U \rightarrow V$ est appelée fonction de transfert et associe tout élément de l'ensemble U à une valeur d'opacité dans V. Une fonction de transfert est généralement déterminée par un ensemble de points clés, les valeurs intermédiaires entre ces points étant approximés linéairement. Le rendu de volume amène certes de nombreux avantages, mais aussi quelques inconvénients. La complexité spatiale et temporelle de ces algorithmes est généralement assez élevée. De plus, étant donnée sa tolérance envers les zones de transition plus floues, il est sensible aux bruits et artéfacts provenant des données sources.

On peut regrouper ces algorithmes en deux grandes classes [72] selon leur fonctionnement : le *Backward* et *Forward Mapping*. Les algorithmes utilisant le *Backward Mapping* projettent le plan image sur l'ensemble de données tandis que les algorithmes utilisant le *Forward Mapping* projettent le volume sur le plan image. Ces classes, ainsi que quelques algorithmes, sont présentés dans les sections suivantes.

1.3.2.1 Backward Mapping

L'idée de base des algorithmes de *Backward Mapping* est de projeter les pixels du plan image (plan de projection) à travers l'ensemble de données. Les axes de projections sont perpendiculaire au plan de projection. Ceux-ci s'apparentent à des rayons que l'on lancerait à partir de chaque pixel du plan image, c'est pourquoi cette méthode de rendu est aussi appelée lancer de rayons ou *raycasting*. Chaque voxel rencontré par ces rayons apporte une contribution à l'apparence finale du pixel à partir duquel le rayon est projeté. Étant donnée la nature discrète du volume, il est très peu probable que le rayon traverse directement les voxels. Ceux-ci doivent donc être rééchantillonnés le long du rayon (Fig 1.12).

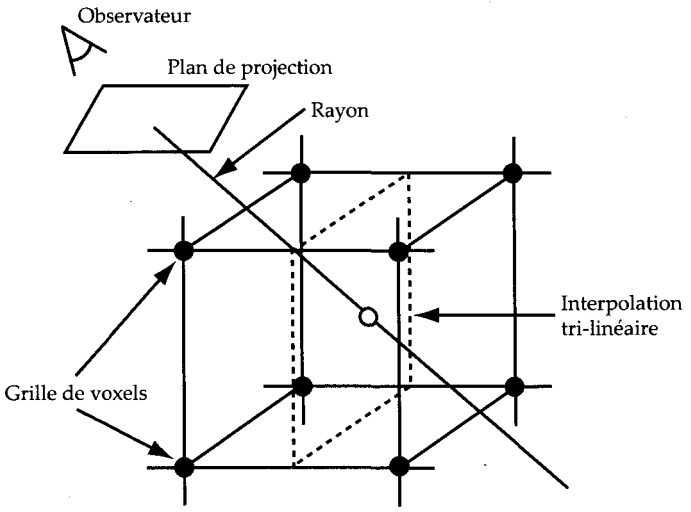


Figure 1.12: Rééchantillonnage le long d'un rayon à l'aide de l'interpolation linéaire.

Considérons un cube formé de huit voxels, l'interpolation tri-linéaire permet d'approximer la valeur $\rho(x, y, z)$ d'un point situé à l'intérieur de ce cube en effectuant une moyenne pondérée des valeurs $\rho_{i,j,k}$ de ses sommets. La valeur $\rho(x, y, z)$ est donnée par l'expression suivante :

$$\rho(x, y, z) = \sum_{i=1}^{2} \sum_{j=1}^{2} \sum_{k=1}^{2} (D - T_i)(D - U_j)(D - V_k)\rho_{ijk} \ , \tag{1.3}$$

où D correspond à la longueur d'une arête du cube et T_i, U_j et V_k sont les distances entre le point xyz et les sommets du cube :

$$\begin{aligned} T_i &= |x - x_i| \\ U_j &= |y - y_j| \\ V_k &= |z - z_k| \ . \end{aligned} \tag{1.4}$$

Ce modèle suppose que l'intensité entre deux voxels voisins varie linéairement. Les deux

sections suivantes présentent des algorithmes basés sur les principes mentionnés ci-haut.

Projection par intensité maximale

La projection par intensité maximale (*MIP*) consiste essentiellement à lancer des rayons perpendiculaires au plan de projection à travers le volume à visualiser. Ceux-ci traversent alors une série de voxels de différentes intensités. Pour chacun de ces rayons, on extrait la valeur maximale des voxels traversés et on la projette sur le plan image. Plusieurs versions de cet algorithme produisent différents résultats (projection par intensité minimale, projection par intensité moyenne, ...). La projection par intensité maximale comporte certaines limitations quant à sa flexibilité et aux résultats produits mais s'est avérée très utile, notamment dans le domaine de l'angiographie (l'étude des vaisseaux sanguins non visibles à partir de radiographies standards). De plus, cette méthode de rendu ne requiert que peu de ressources.

Algorithme de Levoy

En 1988, Marc Levoy introduit un algorithme [45] basé lui aussi sur le principe du lancer de rayons. À l'origine, il a été développé dans l'optique de visualiser des données provenant du domaine de l'imagerie médicale. L'algorithme consiste en un pipeline formé de cinq étapes (Fig. 1.13). Après la première étape, celle de l'acquisition, les données sont envoyées vers deux modules. Un premier effectue le calcul d'effets de lumière et d'ombrage et un deuxième est composé de procédures de classification. Dans le cas où les données ne sont pas de nature médicale ou si les procédures de classification sont aberrantes, celles-ci sont remplacées par la spécification d'une fonction de transfert. Une fois la classification terminée, on obtient deux nouveaux volumes. Le premier contient les couleurs des voxels et l'autre contient leur valeur d'opacité. Ces volumes sont ensuite rééchantillonnés le long de rayons projetés sur le plan de projection et l'intensité des pixels de l'image finale est ensuite déterminée à l'aide d'une opération de composition.

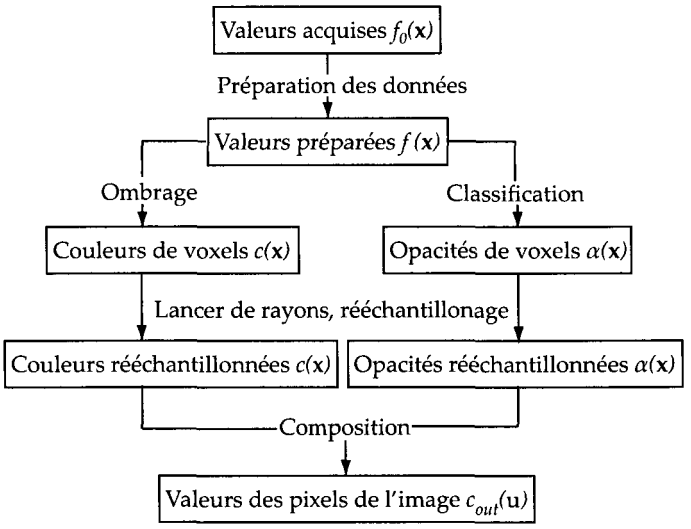


Figure 1.13: Pipeline de l'algorithme de Levoy [45].

Nous allons maintenant détailler ces procédures dans les sections suivantes.

Classification La procédure de classification décrite par Levoy permet de mettre en évidence les contours des régions d'intérêt d'un volume en affectant, à chaque voxel, un niveau d'opacité propre à sa composition. Cette procédure permet de classifier des données provenant de modalités d'imagerie médicale (tomographie, imagerie par résonnance magnétique, ...). Supposons qu'un volume est constitué d'un nombre N de matériaux (os, tissus, air, ...) et considérons f_n, la valeur de densité typique pour le matériau de type n $(1 < n < N)$. On peut ordonner ces matériaux selon leur valeur de densité typique de telle sorte que $f_n < f_{n+1}$. Étant donné que les transitions entre les divers matériaux composant un volume sont plus ou moins floues, on remarquera qu'une fois les types de matériaux ordonnés, un matériau de type n_1 ne peut entrer en contact avec un matériau de type n_2 si la différence $|n_2 - n_1| > 1$. Ceci nous permet d'assigner à chaque matériau de type n une opacité α_n. Ces valeurs d'opacité sont préalablement déterminées par l'utilisateur. On assigne des opacités intermédiaires aux voxels dont la

densité se situe entre deux valeurs typiques. Si notre objectif est de mettre en évidence les contours des régions d'intérêt d'un volume (frontières entre les matériaux dans ce cas-ci), on peut calculer l'opacité d'un voxel relativement à la valeur de son gradient car celui-ci est plus fort dans les zones de transition entre deux matériaux. Étant donné un voxel à la position $\mathbf{x} = (x_i, y_j, z_k)$ et sa densité $f(\mathbf{x})$, sa valeur d'opacité est calculée en utilisant une approximation du gradient :

$$\begin{aligned}
&\nabla f(\mathbf{x}) = \nabla f(x_i, y_j, z_k) = \\
&\frac{1}{2}\Big(f(x_{i+1}, y_j, z_k) - f(x_{i-1}, y_j, z_k)\Big) \\
&\frac{1}{2}\Big(f(x_i, y_{j+1}, z_k) - f(x_i, y_{j-1}, z_k)\Big) \\
&\frac{1}{2}\Big(f(x_i, y_j, z_{k+1}) - f(x_i, y_j, z_{k-1})\Big) \ .
\end{aligned} \tag{1.5}$$

La valeur d'opacité pour un voxel d'intensité $f(\mathbf{x})$ ($f_n \leq f(\mathbf{x}) \leq f_{n+1}$) peut ensuite être calculée en utilisant une estimation par interpolation linéaire entre les valeurs d'opacité typiques des matériaux n et $n+1$, pondérée par la norme du vecteur gradient à la position $\mathbf{x}$:

$$\alpha(\mathbf{x}) = |\nabla f(\mathbf{x})| \left(\alpha_{n+1}\left(\frac{f(\mathbf{x}) - f_n}{f_{n+1} - f_n}\right) + \alpha_n\left(\frac{f_{n+1} - f(\mathbf{x})}{f_{n+1} - f_n}\right)\right) \ . \tag{1.6}$$

Le résultat de cette phase est un tableau tridimensionnel (volume) composé de valeurs d'opacité.

Effets d'ombrage (*shading*) Cette étape du pipeline permet de colorer l'image finale et de créer une illusion de profondeur au rendu. Le modèle d'ombrage utilisé est celui de Phong [53]. Ce modèle utilise quatre vecteurs pour calculer la couleur c d'un voxel à la position $\mathbf{x}$ (Fig. 1.14). Le vecteur $\vec{L}$ représente la direction de la source lumi-

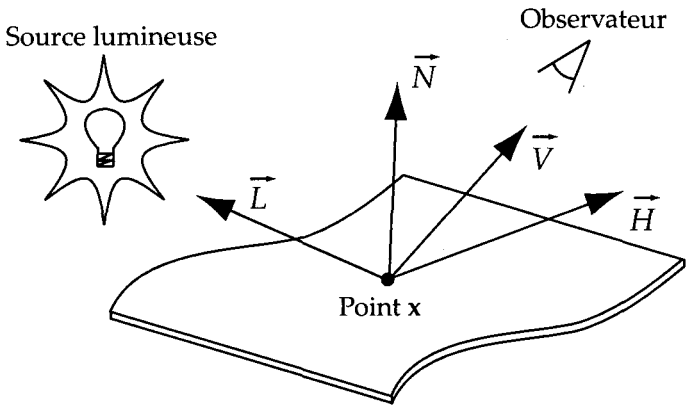


Figure 1.14: Modèle d'illumination de Phong. Les vecteurs utilisés sont le vecteur normal à la surface ($\vec{N}$), la direction de l'observateur ($\vec{V}$), un vecteur de réflection ($\vec{H}$), et la direction de la source lumineuse ($\vec{L}$).

neuse et $\vec{V}$ représente la direction du plan de projection et est perpendiculaire à celui-ci. Le vecteur $\vec{H}$ représente la direction que prendrait un rayon parfaitement réfléchi provenant de la source lumineuse. Étant donné qu'on utilise une projection orthographique (les rayons sont parallèles), les vecteurs $\vec{H}$, $\vec{V}$ et $\vec{L}$ restent constant. On obtient $\vec{H}$ à l'aide de l'équation suivante :

$$\overrightarrow{H} = \frac{\overrightarrow{V} + \overrightarrow{L}}{|\overrightarrow{V} + \overrightarrow{L}|} . \tag{1.7}$$

$\vec{N}$ est le vecteur normal de la surface au voxel de position $\mathbf{x}$. Celui-ci est obtenu à l'aide du gradient (équation 1.5) de la façon suivante :

$$\overrightarrow{N}(\mathbf{x}) = \frac{\nabla(f(\mathbf{x})}{|\nabla(f(\mathbf{x})|} . \tag{1.8}$$

L'ensemble de ces vecteurs sont normalisés. On calcule la couleur c d'un voxel à la position $\mathbf{x}$ comme suit :

$$c(\mathbf{x}) = c_p k_a + \frac{c_p}{k_1 + k_2 d(\mathbf{x})} \left[(k_d(\overrightarrow{N}(\mathbf{x}) \cdot \overrightarrow{L}) + k_s(\overrightarrow{N}(\mathbf{x}) \cdot \overrightarrow{H})^n) \right]$$

où

c_p	=	couleur de la source de lumière parallèle
k_a	=	coefficient de réflexion ambiant
k_d	=	coefficient de diffusion
k_s	=	coefficient de réflexion spéculaire
k_1,k_2	=	constantes d'atténuation
$d(\mathbf{x})$	=	distance du voxel au plan de projection
$\overrightarrow{L}$	=	vecteur normalisé en direction de la source lumineuse
$\overrightarrow{H}$	=	vecteur normalisé en direction de la réflexion d'intensité maximale
$\overrightarrow{N}$	=	vecteur normal à la surface au voxel $\mathbf{x}$

(1.9)

À noter que la réflexion spéculaire ne modifie pas la direction des rayons les uns par rapport aux autres. Les rayons parallèles avant réflexion sont parallèles après réflexion, comme pour un miroir. Le résultat de cette phase est un tableau tridimensionnel (volume) composé de valeurs de couleurs $c(\mathbf{x}) = (R, G, B)$ où R,G et B représentent les taux de saturation en rouge, vert et bleu du mélange de couleurs.

Lancer de rayons Après avoir produit les tableaux d'opacité et de couleurs, on peut procéder aux lancer de rayons et au processus de composition permettant de produire l'image finale. Pour chaque pixel du plan de projection, on lance un rayon perpendiculaire à celui-ci traversant chacun de ces tableaux. Pour chaque rayon lancé, on calcule les couleurs $c(\mathbf{x})$ et opacités $\alpha(\mathbf{x})$ en rééchantillonnant les voxels à K positions $\mathbf{x}' = (x'_i, y'_j, z'_k)$ équidistants le long du rayon. Les couleurs et opacités sont calculées en utilisant l'interpolation tri-linéaire (1.3) à partir des huit voxels à proximité (Fig. 1.15).

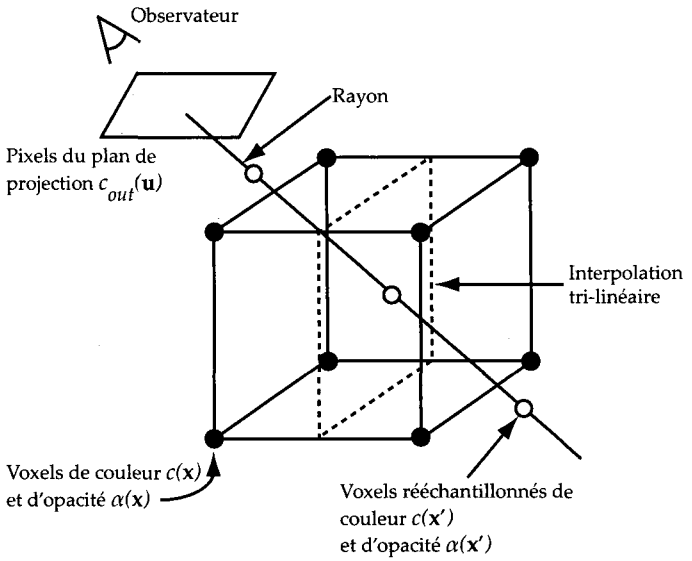


Figure 1.15: Réchantillonnage / Lancer de rayons pour l'algorithme de Levoy[45].

L'espace correspondant à l'arrière du volume, coté opposé au plan de projection, est ensuite coloré par une couleur opaque d'arrière-plan c_{bkg}. La couleur $c_{out}(\mathbf{u})$ d'un pixel à la position $\mathbf{u} = (u_i, v_j)$ sur le plan image est obtenue en combinant la couleur initiale $c_{in}(\mathbf{u})$ du rayon projeté avec les couleurs des voxels rééchantillonnés rencontrés sur son passage :

$$c_{out}(\mathbf{u}) = c_{out}(u_i, v_j) = \sum_{k=0}^{K} \left[c(x_i, y_j, k)\alpha(x_i, y_j, k) \prod_{m=k+1}^{K} (1 - \alpha(x_i, y_j, m)) \right] . \quad (1.10)$$

La qualité des images produites par le lancer de rayons est directement reliée à la quantité de rayons lancés et à la résolution de l'ensemble de données. Cette méthode de rendu permet de générer des images plus complètes et plus fidèles à partir d'un volume que les méthodes de rendu de surface et ce principalement lorsque le volume contient des zones de transitions larges entre deux matériaux. Par contre, le lancer de rayons est plus sensible que le rendu de surface aux artéfacts et bruits présents dans le volume

d'origine.

1.3.2.2 Forward Mapping

Les algorithmes utilisant le *Forward Mapping* projettent l'ensemble de données sur le plan de projection. De façon générale, le volume est divisé en une série de coupes et chacun des voxels de ces coupes est projeté et "écrasé" sur un plan temporaire parallèle au plan de projection. Ce dernier est ensuite combiné au plan image à l'aide d'un opérateur de composition. L'image est complète lorsque l'ensemble des voxels ont été projetés ou que celle-ci est complètement saturée et que l'ajout d'autres voxels n'aurrait plus aucun effet sur l'image résultante. Les sections suivantes présentent deux algorithmes basés sur le principe du *Forward Mapping* : l'algorithme de Drebin, Hanrahan, Carpenter et l'algorithme de Lee Westover.

Algorithme de Drebin, Hanrahan, Carpenter

Une des équipes de recherche et développement des studios Pixar, composée de Robert Drebin, Pat Hanrahan et Loren Carpenter, introduit en 1988 un algorithme [23][52] permettant d'effectuer un rendu de volumes de données. Encore une fois, même si celui-ci a été développé à des fins de visualisation de données médicales, il peut être utilisé pour visualiser d'autres types de volumes. Cet algorithme peut être décrit sommairement sous la forme d'un pipeline présenté dans la Figure 1.16. Nous détaillerons ces différentes étapes dans les sous-sections suivantes.

Classification La première étape de l'algorithme consiste à effectuer une classification des voxels composant le volume. Étant donné que l'algorithme a été développé à l'origine pour visualiser des données médicales, ce processus consiste à déterminer la composition en matériaux (os, tissus, air, ...) de chaque voxel selon sa valeur. On considère qu'un voxel est composé d'un certain pourcentage de chaque type de matériau.

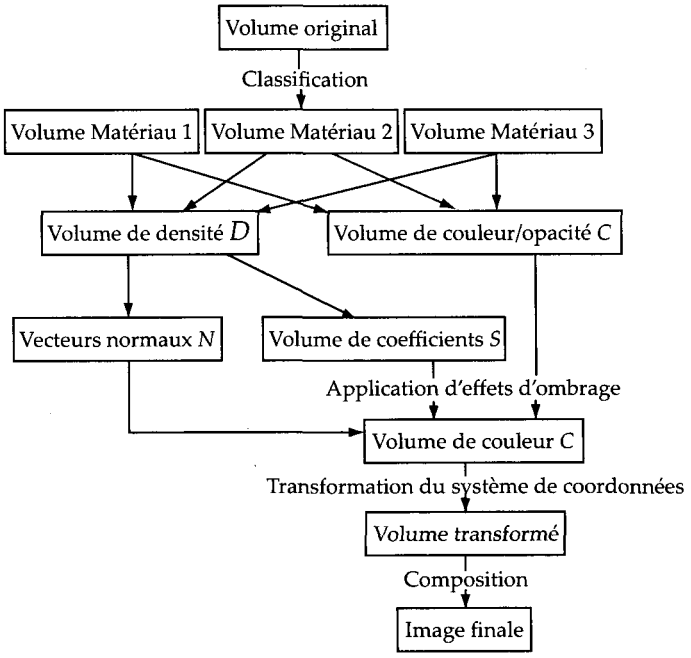


Figure 1.16: Pipeline de l'algorithme de Drebin [23].

On peut représenter les valeurs de densité des voxels composant le volume à l'aide d'un histogramme (Fig 1.17a). On considère aussi que chaque matériau composant le volume correspond à une plage distincte de valeurs de densité différentes. En effet, la plage de valeurs de densité correspondant aux os est plus élevée que celle correspondant aux tissus. Étant donné ces informations, on peut calculer les distributions individuelles relatives de chaque matériau à l'intérieur du volume (Fig. 1.17b). Ceci nous permettra de déterminer la composition en matériaux d'un voxel étant donné sa valeur de densité. La probabilité qu'un voxel soit d'intensité I est donnée par :

$$P(I) = \sum_{i=1}^{n} p_i P_i(I) \ , \tag{1.11}$$

où $P_i(I)$ est la probabilité qu'un matériau i soit d'intensité I et p_i est le pourcentage de matériau i composant le voxel donné. Dans certains cas, pour une tomographie par

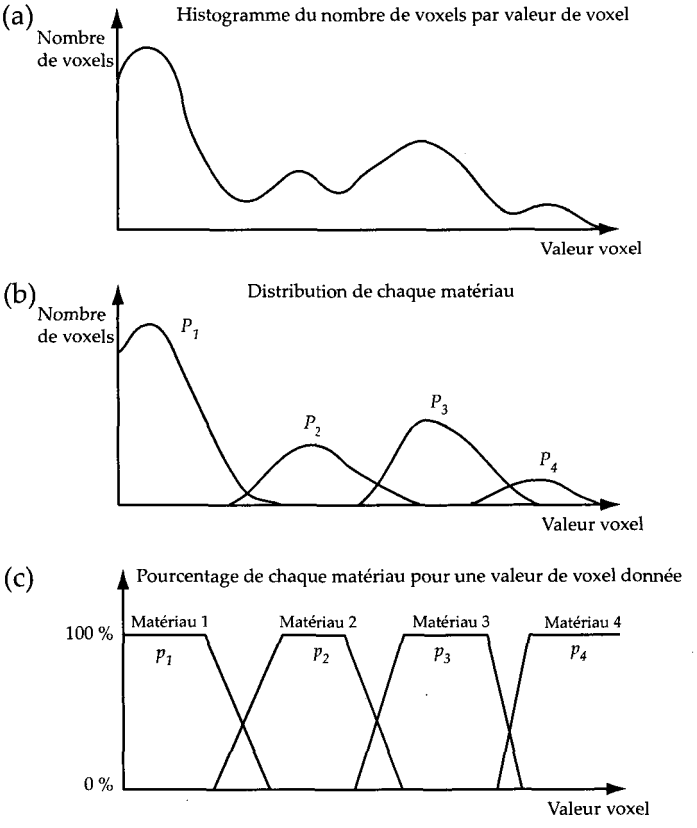


Figure 1.17: Classification pour l'algorithme de Drebin : À l'aide de l'histogramme d'origine (a), on obtient les distributions P_i de chaque matériau (b). En utilisant ces distributions, on peut déduire le pourcentage de composition de chaque matériau pour une valeur donnée (c) [23].

exemple, les distributions P_i sont connues à priori. Si chacune de ces distributions est connue, il est possible de calculer le pourcentage $p_i(I)$ de chaque type de matériau pour un voxel d'intensité I (Fig. 1.17c), à l'aide de l'estimation Bayésienne (voir [29]) :

$$p_i(I) = \frac{P_i(I)}{\sum_{j=1}^{n} P_j(I)} . \tag{1.12}$$

La somme des pourcentages de matériaux pour une valeur donnée est 100%.

Après avoir calculé le pourcentage d'appartenance de chaque voxel à chacun des matériaux, on obtient n nouveaux volumes (i.e. un pour chaque type de matériau). Les voxels de ces nouveaux volumes ont pour valeur les pourcentages de composition de leurs matériaux respectifs aux voxels du volume initial. Connaissant la composition de chacun des voxels, on peut créer un volume de couleurs et de valeurs d'opacité. La couleur et l'opacité d'un voxel sont exprimées à l'aide d'un quadruplet (R, G, B, α) où $R, G, B \in [0, 1]$ correspondent respectivement à la saturation en rouge, vert et bleu du mélange de couleurs et $\alpha \in [0, 1]$ est un coefficient d'opacité. Tel que mentionné précédemment, chaque voxel représente une certaine proportion de chacun des matériaux contenus dans le volume. La couleur résultante de ce mélange est donc donnée par :

$$C = \sum_{i=1}^{n} p_i C_i \ , \tag{1.13}$$

où $C_i = (\alpha_i R_i, \alpha_i G_i, \alpha_i B_i, \alpha_i)$ est la couleur associée au matériau i. On constate que les couleurs sont prémultipliées à leur opacité.

Afin d'appliquer des effets d'ombrage et de lumière au rendu, il est nécessaire de recueilllir des informations sur la position des surfaces à l'intérieur du volume, leurs vecteurs normaux ainsi que la netteté de celles-ci. Une surface est détectée lorsque deux ou plusieurs matériaux de densités différentes se rencontrent. Plus la transition entre les valeurs de voxel est brusque, plus la surface est nette. Pour chaque matériau i, on associe arbitrairement une valeur de densité ρ_i. Le volume de densité est obtenu en additionnant le pourcentage de chaque matériau i multiplié par sa valeur de densité ρ_i. Pour chaque voxel, la densité totale D est donnée par :

$$D = \sum_{i=1}^{n} p_i \rho_i \ . \tag{1.14}$$

Les vecteurs normaux sont estimés à l'aide du gradient du volume de densité. Le vecteur normal $\vec{N}$ de la surface présente à la position d'un voxel donné est calculé comme suit :

$$\begin{aligned} N_x &= \nabla_x D = D_{x+1} - D_x \\ N_y &= \nabla_y D = D_{y+1} - D_y \\ N_z &= \nabla_z D = D_{z+1} - D_z \ . \end{aligned} \tag{1.15}$$

Ces vecteurs sont normalisés et peuvent ensuite être stockés dans une matrice tridimensionnelle de vecteurs normaux. De plus, un volume S correspondant à la netteté des surfaces nous sera nécessaire. Celui-ci est obtenu en calculant la norme des vecteurs normaux trouvés précédemment :

$$S = |\vec{N}| \ . \tag{1.16}$$

Le gradient est une opération de dérivation et est donc sensible aux bruits. On peut appliquer un filtre passe-bas au volume d'origine pour "lisser" celui-ci et ainsi éliminer la plupart des artéfacts produits par le bruit.

Modèle d'illumination L'algorithme de Drebin, Hanrahan et Carpenter utilise un modèle d'illumination consistant à lancer un rayon de lumière d'intensité I à travers chaque voxel et d'en calculer son intensité I' à la sortie. Le rayon est lancé de l'arrière du voxel vers l'observateur et son intensité est modifiée par la composition du voxel déterminée lors de l'étape de classification (Fig. 1.18). Pour ce modèle, on suppose qu'un seul rayon traverse chaque voxel, que le rayon n'est pas atténué dans son parcours

et que son intensité est modifiée par l'opérateur de composition **over** [54] défini comme suit :

$$I' = C \textbf{ over } I = C + (1 - \alpha_C)I \ , \tag{1.17}$$

où α_C est la composante d'opacité de la couleur C d'un voxel. Le premier terme correspond à la lumière émise et le second terme à l'absorption de lumière. Afin de simuler la réflexion de lumière et l'ombrage produit par les surfaces, on doit diviser un voxel en deux régions de couleurs différentes : la région à l'avant de la surface est de couleur C_f et celle à l'arrière est de couleur C_b. La couleur C d'un voxel est en fait la composition des couleurs de ces deux sous régions en plus de celle de la couleur C_s surface :

$$C = C_f \textbf{ over } C_s \textbf{ over } C_b \ . \tag{1.18}$$

La couleur C_s d'une surface est obtenue à l'aide de son vecteur normal, son coefficient de netteté, sa couleur de diffusion C_d, la direction $\vec{L}$ ainsi que la couleur C_l de la source lumineuse et le vecteur $\vec{E}$ pointant vers l'observateur. On calcule cette valeur à l'aide de la formule suivante :

$$C_s = (f(\vec{N}, \vec{L})C_d + g(\vec{E}, \vec{N}, \vec{L})C_l) * S \ , \tag{1.19}$$

où f est une fonction de diffusion, g est une fonction de diffusion spéculaire et C_d est la couleur de diffusion de la surface. Les fonctions f et g sont celles présentées dans l'algorithme de Phong [53] :

$$f(\vec{N}, \vec{L}) = \frac{k_d(\vec{N} \cdot \vec{L})}{k+d} \ , \tag{1.20}$$

$$g(\vec{E}, \vec{N}, \vec{L}) = \frac{k_s(\vec{R} \cdot \vec{E})^n}{k+d} \ , \tag{1.21}$$

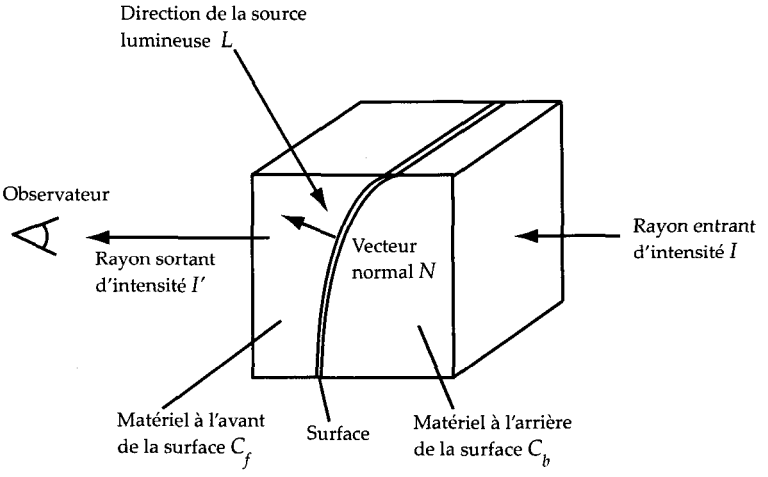


Figure 1.18: Modèle d'illumination pour un voxel dans l'algorithme de Drebin : un rayon d'intensité I entre dans le voxel et en ressort avec une nouvelle intensité I' en direction de l'observateur [23].

où k_d est un coefficient de diffusion, k_s un coefficient de réflexion spéculaire, k est une constante d'atténuation, d est la distance d'un voxel au plan de projection et n est un coefficient déterminant le niveau de luisance de la surface. Plus n est élevé, plus la surface brille. Ici, $\vec{R}$ est un vecteur de réflexion calculé en utilisant $\vec{R} = 2\vec{N}(\vec{N} \cdot \vec{L}) - \vec{L}$. L'ombrage des surfaces est proportionnel au coefficient de netteté de celles-ci, c'est pourquoi une surface plutôt homogène ne produira pas de lumière réfléchie. Généralement, l'algorithme donne de bons résultats lorsque $C_d = C_b$. Le choix du matériau à l'arrière et à l'avant de la surface est déterminé par le signe du gradient du volume de densité par rapport à la direction de la vue de l'observateur. Si le signe est positif, alors le voxel à l'avant a une plus grande densité que celui à l'arrière, et vice-versa. Une fois les matériaux triés, il est facile d'assigner les bonnes couleurs à C_f et C_b à l'aide du volume de couleurs trouvé préalablement.

Création de l'image finale Après avoir appliqué les effets d'ombrage, on obtient un nouveau volume de couleur C. Pour produire l'image finale, on "écrase" ce

volume le long de l'axe z. Chaque coupe C_k du volume le long de cet axe est superposée à la coupe suivante de l'arrière vers l'avant en utilisant l'opérateur **over** défini précédemment. La projection orthographique à la $k^{\text{ième}}$ coupe sur l'axe z est donnée par :

$$I_k = C_k \textbf{ over } I_{k+1} \ , \tag{1.22}$$

où I est l'image composée. L'opération peut aussi être effectuée dans l'ordre inverse avec l'opérateur **under** défini comme suit :

$$A \textbf{ under } B \equiv B \textbf{ over } A \ .$$

Afin de visualiser le volume sous plusieurs angles, on doit appliquer des transformations au système de coordonnées. Les voxels sont ensuite projetés sur le plan image z tel que mentionné précédemment, mais cette fois-ci en utilisant le système de coordonnées transformé. Toute rotation peut être définie en utilisant trois rotations successives à l'aide des angles d'Euler (Ψ,Φ,Θ). La première rotation est d'angle Ψ par rapport à l'axe z, la deuxième est d'angle $\Phi \in [0, \pi]$ par rapport à l'axe y et la troisième est d'angle Θ une fois de plus par rapport à l'axe z (Fig 1.19). Étant donnée une transformation de perspective P_z, la transformation appliquée au système de coordonnées est donnée par :

$$T = P_z(z_e)R_z(\Psi)R_y(\Phi)R_z(\Theta) \ . \tag{1.23}$$

La transformation de perspective correspond généralement à un redimensionnement des coupes sur le plan z par un facteur de $\frac{1}{(z_e - z)}$. Cette tranformation donne l'effet d'une perspective non orthogonale en aggrandissant les objets composant la scène, plus ils sont près de l'observateur. Une fois la transformation appliquée au système de coor-

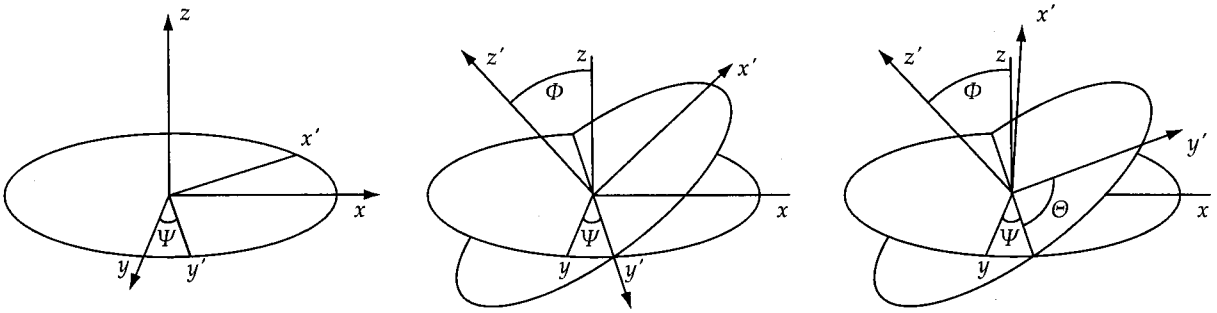


Figure 1.19: Trois rotations successives sont appliquées au système de coordonnées à l'aide des angles d'Euler.

données, le volume est projeté sur le long de l'axe z sur le plan de projection suivant ce nouveau système de coordonnées.

Algorithme de Lee Westover

En 1990, Lee Westover a introduit un algorithme [37][72][73] de rendu direct de volume basé lui aussi sur le *forward mapping*. Rappelons-nous que ce type d'algorithme consiste à produire une image à partir d'un volume en projetant ses voxels sur un plan (voir section 1.3.2.2). L'algorithme de Westover est inspiré de notions de reconstruction de signal. En traitement de signal, la reconstruction consiste essentiellement à approximer le signal continu correspondant à un ensemble d'échantillons distribués uniformément dans l'espace. Le signal est approximé en effectuant la convolution d'un noyau de reconstruction sur l'ensemble d'échantillons. Le noyau de reconstruction est essentiellement une fonction permettant de diffuser les échantillons dans l'espace. Dans notre cas, le signal correspond au volume dans le domaine continu et nous utiliserons un noyau de reconstruction Gaussien. On considère que chaque voxel contient une certaine "énergie" définie par le noyau de recontstruction et que sa contribution à l'image est déterminée par la projection de cette énergie sur le plan image (Fig 1.20). Cette projection est appelée *fonction empreinte*. Les voxels sont projetés un peu à la manière de

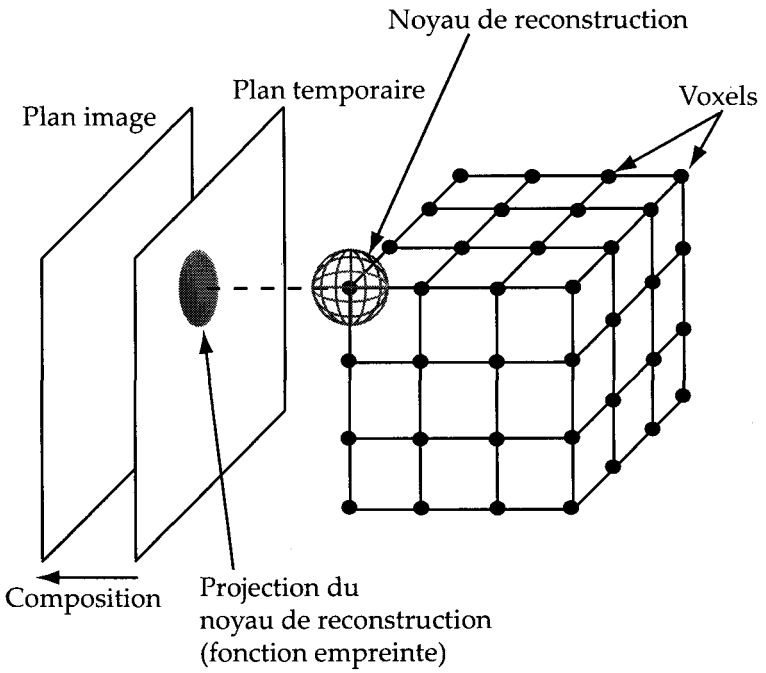


Figure 1.20: La projection de chaque voxel sur le plan image est déterminée par une fonction empreinte. Ce processus s'appel splatting [73].

balles de neige qu'on lancerait sur un mur. Un voxel donné influencera donc l'intensité de plusieurs pixels. On peut s'imaginer le volume comme une série de coupes (images) sur l'axe z. La projection du volume sur le plan image est calculée coupe par coupe en diffusant chaque voxel d'une coupe sur un plan temporaire. Une fois l'ensemble des voxels projetés sur celui-ci, on l'accumule au plan image à l'aide d'un opérateur de composition. Le processus est ensuite répété itérativement pour l'ensemble des coupes restantes. Voyons maintenant les diverses étapes de cet algorithme plus en détail.

Transformations et coloration Préalablement à la projection des voxels sur le plan image, le système de coordonnées doit être transformé selon le point de vue désiré et les voxels doivent être colorés. L'étape de transformation consiste à convertir les coordonnées d'un voxel de l'espace du volume d'origine (i, j, k) vers un espace in-

termédiaire (x, y, z). La projection des voxels sera ensuite effectuée sur le plan image le long de l'axe z de cet espace intermédiaire. La couleur d'un voxel est déterminée à l'aide de trois composantes selon la valeur de celui-ci : l'intensité émise I_{em}, l'intensité diffuse I_{dif} et l'intensité spéculaire I_{spec}. Chacune de ces composantes correspond à un triplet (R, G, B) représentant la saturation de rouge, vert et bleu du mélange de couleurs. L'apport de ces composantes est déterminé à l'aide de tables de référence et de fonctions de transfert. La couleur finale d'un voxel (x, y, z) est donnée par :

$$I = I_{em} + I_{dif} + I_{spec} \ . \tag{1.24}$$

Le résultat correspond à un ensemble d'échantillons (x, y, z) auxquels on associe une couleur (R, G, B, α) où α est un coefficient d'opacité et est lui aussi déterminé à l'aide d'une table de référence ou d'une fonction de transfert.

Reconstruction du volume et contribution d'un voxel Tel que nous l'avons mentionné, cet algorithme consiste à déterminer l'apport de chaque voxel au rendu en approximant le volume à visualiser dans le domaine continu. Le système de coordonnées transformé, ainsi que les voxels colorés précédemment, sont utilisés pour cette opération. Rappelons nous que la génération du rendu est effectuée à l'aide d'une série de coupes parallèles au plan image. La contribution d'un voxel à un plan temporaire est déterminée par la convolution des fonctions représentant le voxel donné ainsi que le noyau de reconstruction. Le résultat de cette opération est appelé fonction empreinte et est dénoté p. Lorsqu'une projection orthogonale est utilisée, p est identique pour chaque voxel projeté. Elle n'a donc besoin d'être calculée qu'une seule fois. Les étapes suivantes nous permettront de déterminer la fonction p.

La reconstruction du volume dans le domaine continu à partir d'un ensemble ρ

d'échantillons distribués uniformément est effectuée à l'aide d'une convolution :

$$g = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(u-x, v-y, w-z)\rho(x,y,z)\text{Ш}(x,y,z)dudvdw \ , \tag{1.25}$$

où h est le noyau de reconstruction déterminant l'énergie de chaque voxel, (u, v, w) définissent l'emplacement du noyau et Ш est l'opérateur de rééchantillonnage Shah, défini comme une somme infinie d'impulsions δ (delta de Dirac) :

$$\text{Ш}(x,y,z) \equiv \sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} .\delta(x-i, y-j, z-k) \tag{1.26}$$

L'auteur propose d'utiliser un noyau de reconstruction Gaussien, par exemple :

$$h(x,y,z) = e^{-(\frac{x^2+y^2+z^2}{\sigma^2})} \ . \tag{1.27}$$

On peut obtenir la valeur du volume reconstruit en un point (x, y, z) à l'aide de :

$$g(x,y,z) = \sum_{D_x}\sum_{D_y}\sum_{D_z} h(x-D_x, y-D_y, z-D_z)\rho(\mathbf{D}) \ , \tag{1.28}$$

où $\mathbf{D} = (D_x, D_y, D_z)$ est la position d'un voxel. Cette équation correspond à la contribution de l'ensemble des voxels à un point (x, y, z) du volume reconstruit. Considérons maintenant la contribution d'un voxel $\mathbf{D}$ à un point (x, y, z) du volume reconstruit :

$$g_{\mathbf{D}}(x,y,z) = h(x-D_x, y-D_y, z-D_z)\rho(\mathbf{D}) \ . \tag{1.29}$$

La contribution totale d'un voxel donné en un point (x, y) du plan image est calculée en sommant ses contributions le long de l'axe z (celui-ci est perpendiculaire au plan

image) :

$$g_{\mathbf{D}}(x,y) = \int_{-\infty}^{\infty} h(x - D_x, y - D_y, w)\rho(\mathbf{D})dw \ . \tag{1.30}$$

La fonction ρ reste toujours constante pour un voxel donné et est indépenante de w, c'est pourquoi on peut écrire :

$$g_{\mathbf{D}}(x,y) = \rho(\mathbf{D}) \int_{-\infty}^{\infty} h(x - D_x, y - D_y, w)dw \ . \tag{1.31}$$

La contribution d'un voxel $\mathbf{D}$ à un point (x,y) du plan de projection est indépendante de la valeur de ce voxel et dépend essentiellement de la position de la projection. La fonction empreinte correspondant à cette contribution est donc définie par :

$$p(x,y) = \int_{-\infty}^{\infty} h(x,y,w)dw \ . \tag{1.32}$$

On remarque que la fonction empreinte correspond essentiellement à l'intégration du noyau de reconstruction sur l'axe z de projection. Celle-ci définit comment un voxel à la position $\mathbf{D}$ diffuse son énergie au point $(D_x + x, D_y + y)$ du plan image :

$$g_D(x,y) = \rho(\mathbf{D})p(x,y) \ . \tag{1.33}$$

Comme mentionné précédemment, la fonction empreinte est identique pour tous les voxels lorsqu'une projection orthogonale est utilisée. Le calcul d'une fonction empreinte pour chaque voxel implique un grand nombre de calculs. C'est pourquoi il est préférable de précalculer l'effet de l'empreinte sous forme de table de référence et d'utiliser celle-ci pour calculer la diffusion d'un voxel sur son entourage. La table de référence est générée en intégrant, le long de l'axe z, le noyau de reconstruction sur une grille discrète. Lorsqu'un noyau de reconstruction sphérique est utilisé, la fonction empreinte prend

la forme d'un cercle (Fig 1.21a). Dans un cas plus général, on utilise un noyau de reconstruction en forme d'ellipsoïde construit à partir d'un noyau sphérique auquel on aurrait appliqué la transformation définissant le point de vue de l'observateur. Le noyau de reconstruction en forme d'ellipsoïde peut être orienté de diverses façons et sa projection prend la forme d'une ellipse (Fig 1.21b). Lorsqu'un tel noyau est utilisé, une correspondance entre la table de référence circulaire du noyau sphérique (cas général) et la table de référence transformée par le point de vue doit être calculée (Fig 1.22). Cette transformation T^{-1} consiste en un redimensionnement S_x sur l'axe x et S_y sur l'axe y ainsi qu'une rotation d'angle θ autour de l'axe de projection de l'empreinte.

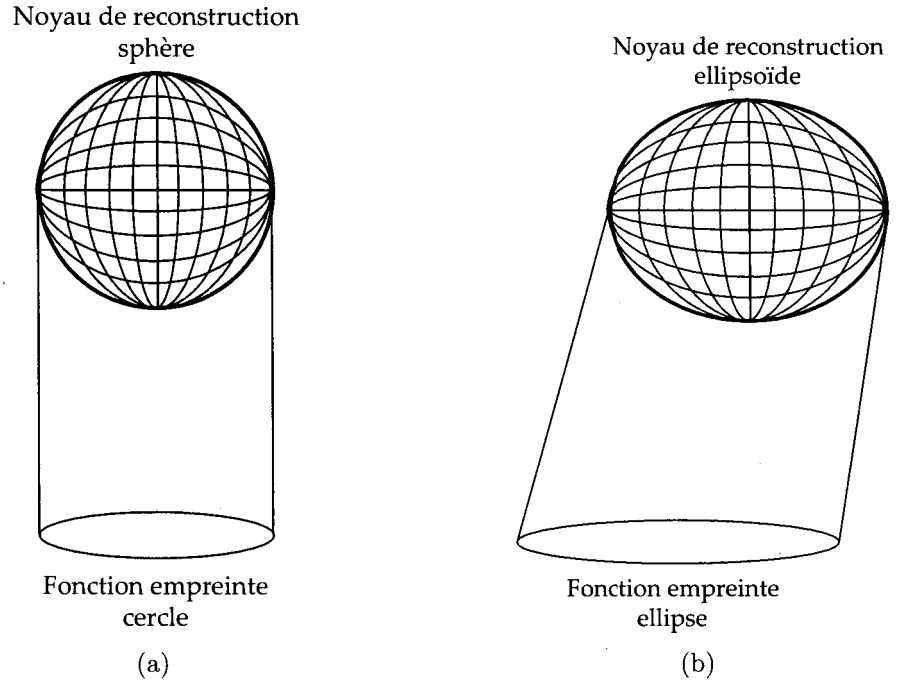


Figure 1.21: Fonction empreinte. En (a), un noyau sphérique est utilisé. La projection de ce noyau correspond à une fonction empreinte circulaire générique. En (b), on utilise un noyau modifié selon le point de vue de l'observateur. Sa projection correspond à une ellipse [73].

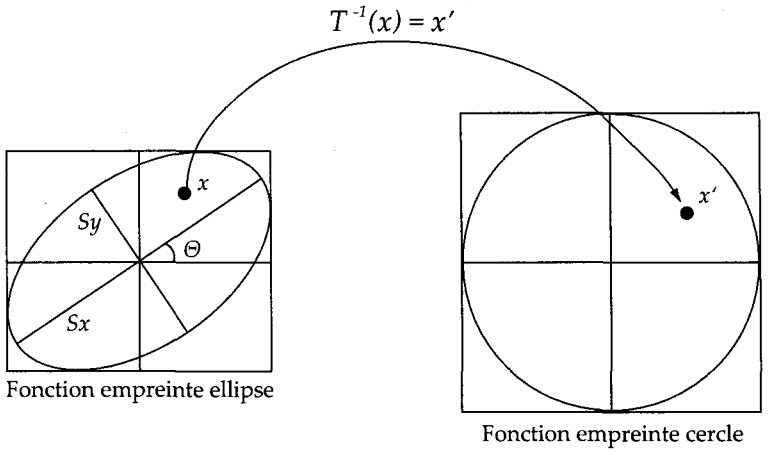


Figure 1.22: La transformation T^{-1} associe chaque point **x** de la fonction empreinte elliptique à un point **x'** de la fonction empreinte circulaire [73].

Composition de l'image finale On définit une feuille (plan temporaire) comme étant un plan parallèle au plan image et qui traverse le volume de données. Pour chaque voxel faisant partie d'une feuille, on détermine sa contribution sur celle-ci à l'aide de la fonction empreinte stockée dans la table de référence. Lorsque tous les voxels faisant partie d'une feuille ont été diffusés sur celle-ci, on la superpose au plan image à l'aide d'une opération de composition, **over** par exemple [54] (voir 1.17). On traite ensuite la feuille suivante de l'arrière du volume vers l'observateur. Lorsque la dernière feuille est composée au plan image, celui-ci correspond au rendu du volume. La qualité du résultat généré par cet algorithme est directement reliée à la fonction d'empreinte utilisée. Une fonction d'empreinte trop précise produit des artéfacts, le contraire produit des images peu précises. Il est important de noter que l'un des principaux avantages de cet algorithme est qu'il est facilement implémentable sur une architecture parallèle, à condition que ses calculs n'utilisent que des informations dans un voisinage rapproché des voxels traités.

1.3.2.3 Sommaire des algorithmes de rendu de volume

Le tableau (1.1) propose une comparaison entre les trois algorithmes (exluant la projection par intensité maximale) présentés dans les sections précédentes. Étant donné

TAB. 1.1: Tableau Comparatif des Algorithmes de Rendu Direct de Volume

	Backward Mapping	Forward Mapping
Exemples d'algorithmes	Levoy [45]	Drebin-Hanrahan-Carpenter [23] et Westover [72]
Qualité du rendu	Les algorithmes de lancer de rayons donnent les meilleurs résultats. La qualité dépend du nombre de rayons utilisés et de la résolution de l'ensemble de données.	Ce type d'algorithme donne généralement de bons résultats, de moins bonne qualité que le lancer de rayons. Dans le cas de l'algorithme de Westover, la qualité dépend principalement du noyau de reconstruction utilisé.
Rapidité d'exécution	Sans optimisation, la génération d'un rendu est lente et il est pratiquement impossible de créer un système de visualisation en temps réel à l'aide du lancer de rayons.	Le forward mapping implique moins de calculs que le backward mapping et est donc généralement plus rapide.
Projection	Une vue en projection parallèle ou perspective peut facilement être calculée.	Le forward mapping permet à l'origine de calculer des projections parallèles. On peut donner un effet de perspective en agrandissant les parties du volume se rapprochant de l'observateur.

un même volume, les images produites par un algorithme de splatting et de lancer de rayons peuvent contenir quelques petites différences structurelles. Ce phénomène est dû au fait que les deux algorithmes n'utilisent pas la même méthode d'interpolation. Meissner et al. présentent une étude comparative [51] du lancer de rayon et du *splatting* au niveau du temps de calcul et de la qualité du rendu. Le lancer de rayons est généralement plus lourd et offre de meilleures performances lorsque le volume est essentiellement formé de surfaces opaques non couvertes de surfaces transparentes. Cette caractéristique du volume permet d'avorter le calcul d'un rayon lorsqu'il touche une surface opaque. Le nombre de calculs augmente donc lorsque plusieurs couches transparentes apparaissent devant une couche opaque. On peut aussi remarquer que le lancer de rayons offre généralement des images plus nettes que le splatting. Le splatting produit des résultats plus rapidement lorsque le nombre de voxels significatifs (ceux influençant le résultat

final) est petit. Dans son étude, l'auteur utilise un algorithme permettant de déterminer quels voxels sont significatifs. La performance du splatting diminue lorsque des voxels opaques sont situés à l'arrière d'autres voxels opaques. Un autre aspect important est la taille de l'image à générer. Pour le lancer de rayons, le nombre de calculs augmente lorsque la résolution de l'image à générer augmente. Pour le splatting, seul le calcul de la fonction empreinte est plus lourd, rendant la diminution de performance moins apparente.

CHAPITRE 2

INTERACTION

2.1 Introduction

Que ce soit pour déplacer le point de vue de la caméra dans une vue 3D, pour effectuer une requête et extraire des données cibles d'un ensemble ou encore pour explorer plus en détail un point donné dans l'espace à n dimensions, les mécanismes d'interaction d'un système de visualisation sont un aspect essentiel de celui-ci. Il est important que l'utilisateur puisse configurer le rendu qui lui est présenté selon ses besoins afin qu'il puisse en retirer le maximum d'information. La section (2.2) présente sommairement différentes techniques d'interaction pour des applications de visualisation scientifique et d'information. Nous conclurons ce chapitre en décrivant le concept de contrôleur (*widget*) dans la section (2.3).

2.2 Méthodes d'interaction et de distortion

Comme dans le cas des algorithmes de visualisation d'information présentés dans la section (1.2), nous utiliserons la classification proposée par Keim dans [41]. Les mécanismes d'interaction présentés dans ce chapitre permettent à l'utilisateur de manipuler les représentations graphiques qui lui sont présentés afin d'en extraire des régions ciblées ou de faire ressortir certaines caractéristiques des données explorées. Les principes généraux énoncés dans cette section sont décrits dans le contexte de la visualisation d'information, mais sont aussi appliquables à la visualisation scientifique. Keim distingue

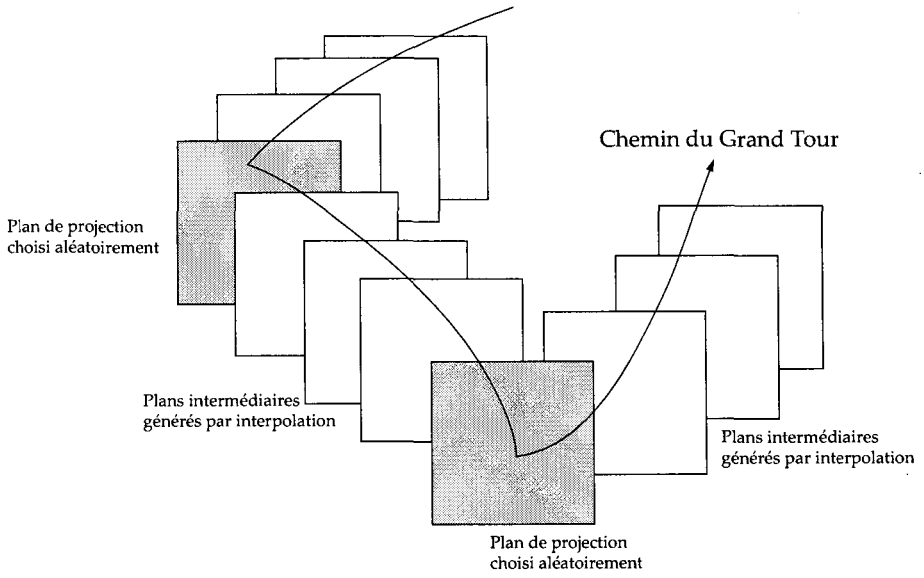


Figure 2.1: Grand Tour : Une séquence visuelle est construite à partir d'une série de plans de projection sélectionnés aléatoirement et de plans générés par interpolation entre ceux-ci. [7].

cinq classes de mécanismes d'interaction : les projections dynamiques, le filtrage interactif, le zoom interactif, les distortions interactives et le liage de vues interactif. Celles-ci sont présentées sommairement dans les sections suivantes.

2.2.1 Projections dynamiques

Les projections dynamiques [21][75] permettent d'explorer un ensemble de données multidimensionnel en parcourant dynamiquement des projections (sous-espaces) de celui-ci. Introduit en 1985 par Asimov [7], le *Grand Tour* est un exemple de système de projections dynamiques. Cette technique d'exploration consiste à présenter à l'utilisateur une séquence visuelle constituée de transitions entre une série de projections bidimensionelles repères. Ces projections peuvent être sélectionnées de façon arbitraire, manuelle, ou selon certains critères calculés sur l'ensemble de données. Les projections intermédiaires composant les transitions entre les projections repères sont générées par interpolation (Fig. 2.1). Cette technique d'interaction est utile pour visualiser un en-

semble de données sous plusieurs angles et peut être vue comme la "généralisation" d'une rotation bidimensionelle, même si elle ne correspond pas à une rotation au sens propre. Wegman propose plus tard dans [71] une version plus versatile du Grand Tour, permettant de présenter à l'utilisateur des projections à k dimensions d'un ensemble de données à n dimensions ($k \leq n$). Ces projections sont présentées à l'utilisateur à l'aide de coordonnées parallèles.

2.2.2 Filtrage interactif

Lorsque l'ensemble de données à visualiser est de taille trop importante, il arrive parfois que sa représentation graphique soit trop encombrée et difficilement interprétable. Une solution à ce problème est de permettre à l'utilisateur de limiter le rendu à seulement une partie de l'ensemble de données pour ainsi en diminuer la complexité de l'affichage. On peut arriver à cette fin en extrayant un certain nombre d'éléments "intéressants" relativement à un critère déterminé par l'utilisateur et en masquant les éléments jugés moins pertinents. Cette opération peut être effectuée en sélectionnant directement les données dans la représentation graphique (*browsing*) ou à l'aide d'une requête (*querying*). Il est important de noter que la rédaction d'une requête efficace et la sélection directe de données à l'intérieur d'un rendu sont des opérations non-triviales. C'est pourquoi elles sont généralement utilisées conjointement à d'autres méthodes d'interaction.

Le concept de lentilles magiques [11] est un exemple de système à filtrage interactif. Ces lentilles sont essentiellement des contrôleurs (voir section 2.3) que l'utilisateur manipule directement sur les données visualisées. Pour chaque contrôleur, on associe une requête. Lorsque ceux-ci sont superposés à l'ensemble de données visualisé, seules les données se conformant aux critères de la requête restent visibles, sans toutefois altérer le reste du rendu. Ces lentilles fonctionnent un peu à la manière de filtres optiques

(seulement certaines fréquences traversent le filtre). Lorsque plusieurs filtres sont superposés, leurs requêtes sont combinées. Ce concept est généralisé en trois dimensions dans [68].

Les requêtes dynamiques [2][3] sont un autre exemple de système à filtrage interactif. L'utilisateur ajuste différents paramètres d'une requête à l'aide de contrôleurs graphiques comme des boutons glissoires (*sliders*) et des cases à cocher. Lorsque l'utilisateur modifie les valeurs des paramètres de la requête, la représentation graphique est mise à jour de façon dynamique. De plus, l'utilisation de contrôleurs graphiques permet de borner les valeurs des paramètres de la requête. Ce système permet aux utilisateurs non initiés d'explorer les données plus facilement en ayant un retour d'information direct sur leurs actions.

2.2.3 Zoom interactif

Le zoom est une opération couramment utilisée dans les applications de nature graphique. Celle-ci permet à l'utilisateur de porter son attention sur une zone précise de l'ensemble de données. Étant donné un ensemble de données de grande taille, celui-ci est présenté à l'origine sous une forme fortement comprimée, à un niveau de détail peu élevé, pour qu'on puisse le visualiser dans son ensemble. L'utilisateur peut cibler une région d'intérêt et explorer celle-ci de façon plus détaillée en l'agrandissant. Lors d'un grossissement, il est parfois avantageux de recourir à une représentation différente, mais plus détaillée, des données observées plutôt que d'utiliser une simple version agrandie de celles-ci. Par exemple, des données représentées par de simples points lorsqu'on les observe en leur ensemble peuvent correspondre, après grossissement, à des icônes ou *glyphs* (voir section 1.2.4). Ce processus est appelé *zoom sémantique* (Fig 2.2). L'outil de visualisation PAD++ [9] est un exemple d'interface basé sur le zoom interactif qui incorpore de tels mécanismes.

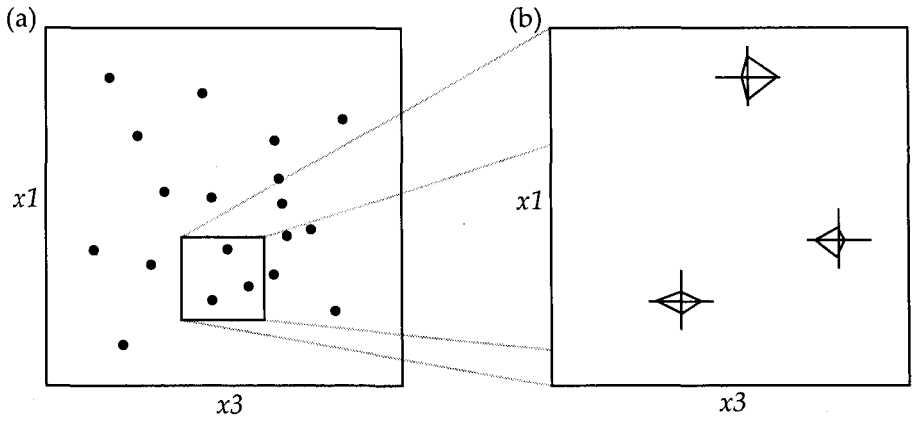


Figure 2.2: Zoom interactif : (a) Nuage de points correspondant à la projection sur le plan x_1x_3 d'un ensemble de données à quatre dimensions. L'encadré représente la zone d'intérêt. (b) La zone d'intérêt sélectionnée en (a) est agrandie et plus détaillée (les points sont maintenant représentés par des icônes).

2.2.4 Distortions interactives

L'idée des distortions interactives est de permettre l'exploration de données ciblées en les présentant sous une forme plus détaillée, tout en gardant un aperçu général de l'ensemble de celles-ci. Similairement au zoom interactif, l'utilisateur se doit de choisir une région d'intérêt, laquelle lui sera présentée sous une forme agrandie et plus détaillée. Plus les données sont rapprochées du centre de la région d'intérêt, plus elles sont détaillées, tandis que les données éloignées sont représentées de façon plus grossière et sont de dimension réduite. La dimension, la position ainsi que le niveau de détail des données visualisées sont donc relatifs au point focal de la zone d'intérêt et du niveau de distortion. L'effet visuel obtenu par l'application de ce concept donne des résultats similaires à ceux obtenus par l'utilisation d'une lentille grand angle en photographie.

La visualisation œil de poisson (*graphical fisheye views*) [59] est un exemple de système utilisant les distortions interactives appliquées notamment à la visualisation de graphes (Fig. 2.3). Dans cet exemple, un graphe de 100 sommets et 124 arêtes est

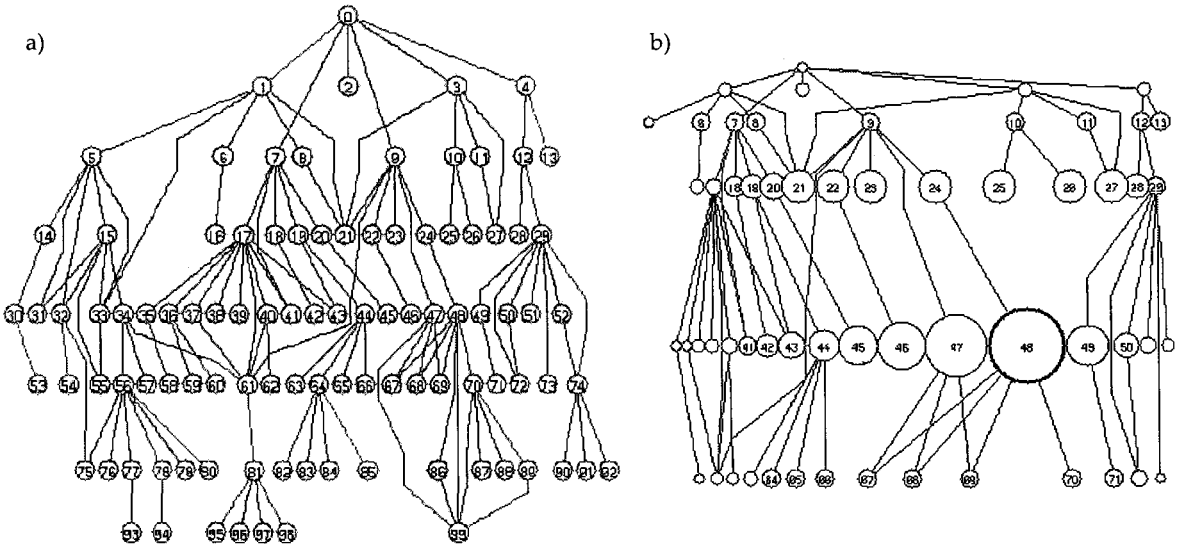


Figure 2.3: Affichage œil de poisson [59] : a) Graphe visualisé à l'aide d'un affichage standard. b) Affichage en œil de poisson du graphe présenté en (a). Le focus est porté sur le sommet (48).

visualisé à l'aide d'un affichage standard et d'un affichage œil de poisson. Dans le cas de l'affichage œil de poisson, le focus est porté sur le sommet (48). Celui-ci est présenté de façon plus détaillée. Plus les autres sommets sont éloignés du point focal, plus ils sont représentés de façon grossière.

2.2.5 Vues liées interactives

Les vues liées interactives permettent essentiellement de lier différentes représentations graphiques d'un même ensemble de données de façon à ce que les actions effectuées sur l'une d'elles se répercutent sur les autres. On utilise généralement cette méthode pour combiner les forces de différentes techniques de visualisation afin d'en surpasser les faiblesses ou pour faire ressortir divers aspects d'un ensemble de données en le visualisant de plusieurs manières. Ce concept est particulièrement intéressant lorsqu'appliqué à une opération de sélection de données. Les points correspondants à une

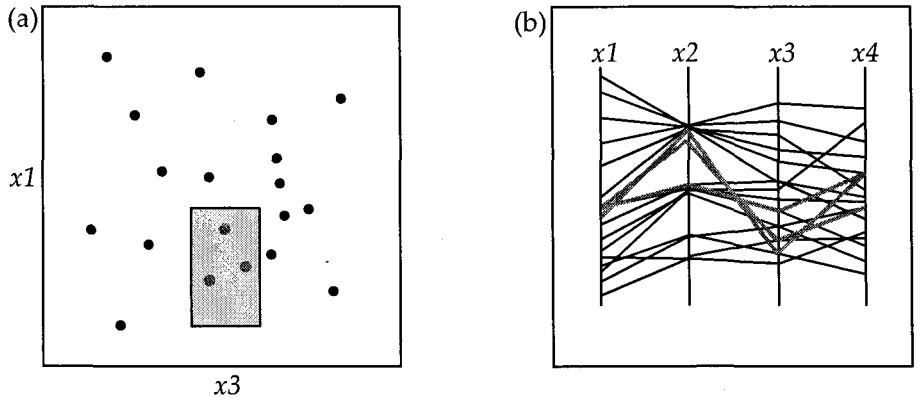


Figure 2.4: Vues liées interactives : (a) Une sélection (*brushing*) est effectuée dans le nuage de points (projection sur le plan x_1x_3) représentant un ensemble de données à quatre dimensions. (b) Les points à n dimensions sélectionnés dans le nuage de point sont mis en évidence dans une vue en coordonnées parallèles (*linking*).

opération de sélection (*brushing*) effectuée à l'intérieur d'une vue sont alors mis en évidence sur l'ensemble des autres vues. Ce concept est couramment utilisé conjointement à la visualisation d'information par matrice de nuages de points (voir section 1.2.2.2). Plusieurs systèmes de visualisation comme Polaris [62] et Xmdv [69] intègrent un grand nombre de méthodes de visualisation et permettent de lier différentes vues.

2.3 Contrôleurs

Étant donné la nature tridimensionnelle des données dans les applications de visualisation scientifique, les manipulations que l'utilisateur peut effectuer sur celles-ci ne sont pas des tâches triviales. En effet, l'utilisateur doit souvent avoir recours à des périphériques d'entrée à deux dimensions (ie. souris) pour effectuer des manipulations en trois dimensions. Les *contrôleurs*, ou *widgets*, sont des composantes d'interface graphique généralement représentées par de petits objets géométriques simples en deux ou trois dimensions, disposés à l'intérieur de l'espace de visualisation [20][34] et pouvant être manipulés. Ceux-ci sont utilisés comme intermédiaire pour manipuler l'ensemble

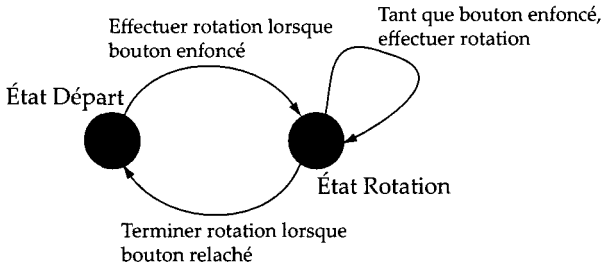


Figure 2.5: Automate (ATN) décrivant le comportement d'une sphère virtuelle[20].

de données ou pour afficher de l'information à l'utilisateur.

Le nombre de *degrés de liberté* d'un objet géométrique correspond essentiellement au nombre de paramètres indépendants définissant sa position et son orientation. Par exemple, un bouton glissoire en deux dimensions n'a qu'un seul degré de liberté. Évidemment, plus le nombre de degrés de liberté d'un contrôleur est restreint, plus le nombre de paramètres qu'il peut représenter est limité. La position et l'orientation d'un contrôleur dans l'espace 2D sont déterminées par trois degrés de liberté. Un contrôleur 3D, quant à lui, est configuré dans l'espace par un total de six degrés de liberté (trois de rotation et trois de positionnement). L'utilisateur manipule ces contrôleurs par des actions simples qui sont interprétées directement par celui-ci. L'apparence visuelle d'un contrôleur doit être représentative de la fonction qu'il exerce. Par exemple, l'utilisation d'une sphère est un choix intéressant pour un contrôleur permettant d'effectuer des rotations en trois dimensions sur un objet géométrique.

Un automate est un outil permettant de décrire efficacement le comportement d'une interface. On peut décrire le comportement d'un contrôleur à l'aide d'un automate d'état fini (FSA) [35][61]. Souvent, ceux-ci ne sont pas adéquats pour une interface trop complexe car il en résulte souvent un accroissement démesuré du nombre d'états et de transitions. On utilise les réseaux de transition augmentés (ATN) pour remédier à ce problème. Ceux-ci permettent l'utilisation de variables, chemins conditionnels et

autres mécanismes près de la programmation. Par exemple, la Figure 2.5 illustre le comportement d'un contrôleur de sphère virtuelle [14] englobant un objet géométrique et permettant d'effectuer une rotation sur celui-ci. Le contrôleur traduit le mouvement de la souris par une rotation sur l'objet en question lorsqu'un bouton est enfoncé. Étant donné la nature tridimensionnelle des données que nous aurons à visualiser, la conception d'un contrôleur permettant de les explorer dans leur ensemble est une étape importante de notre projet et sera traitée dans le chapitre suivant.

CHAPITRE 3

EXPLORATION DE GRAPHES BASÉE SUR LE CALCUL DE VALUATIONS

3.1 Introduction

Suite à la revue de littérature scientifique dans le domaine de la visualisation et de l'interaction des chapitres 1 et 2, nous approfondirons le thème de l'exploration des graphes. Les définitions et notions importantes relatives à cette structure de données sont présentées dans la section 3.2. On appelle *topologie* d'un graphe la description de la relation de voisinage et d'adjacence entre ses sommets. L'exploration d'un graphe, uniquement par sa topologie, peut s'avérer difficile lorsqu'on étudie de grands graphes. Plus la taille d'un graphe augmente, plus la complexité de sa représentation visuelle est importante. Il existe une limite physique et une limite cognitive à l'affichage de données sur un support physique. Ces notions sont présentées dans [48]. La limite physique correspond essentiellement au nombre maximal d'éléments affichables sur un support. Par exemple, le nombre d'éléments affichables sur un écran ne peut être supérieur au nombre de pixels disponibles par sa résolution maximale. La limite cognitive est une notion un peu plus difficile à définir. Elle correspond à la limite de compréhension et d'interaction d'un utilisateur face à une représentation graphique trop complexe. Nécessairement, plus un graphe est grand, plus la surface nécessaire pour le représenter adéquatement sera grande. Étant donné que celle-ci est limitée par le dispositif d'affichage utilisé, la solution de base est de réduire la taille des objets graphiques et de

les regrouper dans l'espace d'affichage. Il devient alors difficile pour l'utilisateur d'en extraire de l'information ou même de manipuler simplement le rendu.

Un des objectifs de ce mémoire est le développement d'un outil permettant l'exploration de graphes de grande taille. Ce chapitre présente l'approche pour laquelle nous avons optée. Elle consiste en gros à associer chaque sommet du graphe à un point d'un volume tridimensionnel, déterminé par le calcul de statistiques sur ses sommets, et d'explorer le graphe à partir de ce dernier. Ce processus est présenté de façon détaillée dans la section (3.3).

3.2 Définitions concernant les graphes

Cette section vise l'introduction de quelques notions concernant les graphes. Notons que les définitions présentées ici peuvent être retrouvées dans [30]. Un graphe G consiste en un ensemble de *sommets* $V = \{v_1, v_2, \ldots, v_n\}$ et un multi-ensemble d'*arêtes* $E = \{e_1, e_2, \ldots, e_n\}$ et permet de représenter des relations entre différentes entités. Une arête e est définie par une paire non-ordonnée de sommets $\{u, v\} \in V$. Étant donnée cette définition, on dit que le sommet u et l'arête e sont mutuellement *incidents*. Il en va de même pour le sommet v et l'arête e. Notons un graphe G par $G = (V, E)$. Introduisons maintenant quelques définitions concernant les sommets. Deux sommets sont *adjacents* s'ils sont reliés par une arête. On définit le *voisinage* d'un sommet comme l'ensemble des sommets y étant adjacents. On nomme *degré* d'un sommet la taille de son voisinage. Un sommet de degré 1 (relié à un seul autre sommet) est appelé *feuille*. La figure 3.1 présente quelques-unes de ces notions.

Notons maintenant quelques définitions concernant les arêtes et les parcours dans un graphe. Une arête est *multiple* s'il existe au moins une autre arête définie par la même paire de sommets à l'intérieur du graphe. On appelle *boucle* une arête reliant un sommet à lui-même. Ces notions sont présentées dans la figure 3.2. Une suite de

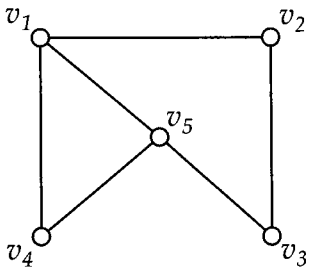


Figure 3.1: Exemple de graphe. Ici, le graphe est défini par un ensemble de sommets $V = \{v_1, v_2, \ldots v_5\}$ et un ensemble d'arêtes $E = \{\{v_1, v_2\}, \{v_1, v_4\}, \ldots \{v_3, v_5\}\}$. On remarque que le sommet v_5 est adjacent aux sommets v_1, v_3, v_4 et que ceux-ci constituent son voisinage. Le sommet v_5 est donc de degré 3.

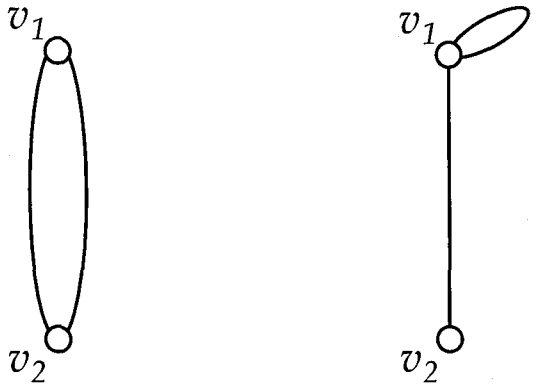


Figure 3.2: Arêtes boucles et arêtes multiples. En (a), les sommets v_1 et v_2 sont reliés par des arêtes multiples. En (b), le sommet v_1 est relié à lui-même par une boucle.

sommets et arêtes $s_0, e_1, s_1, e_2, s_2, \ldots, e_n, s_n$ définit une *chaîne* si et seulement si chaque arête e_i est incidente à s_{i-1} et à s_i. Pour un graphe sans arêtes multiples, on définit une chaîne comme une liste de sommets tels que deux sommets adjacents dans la liste sont adjacents dans le graphe. Une chaîne est appelée *cycle* si son sommet d'origine correspond à son sommet d'arrivée. La *distance* entre deux sommets u et v correspond au nombre d'arêtes constituant la plus courte chaîne reliant ces deux sommets.

Portons maintenant notre attention sur quelques types de graphes particuliers.

Introduisons d'abord la notion d'arc. Un *arc* est une paire ordonnée (couple) $(u, v) \in V$ et possède une direction qui est définie par le sommet *initial* u et le sommet *terminal* v. Un graphe est *orienté* si les relations entre ses sommets sont définies par des arcs plutôt que par des arêtes. Les graphes orientés impliquent certaines nouvelles définitions. Une suite de sommets et d'arcs $s_0, e_1, s_1, e_2, s_2, \ldots, e_n, s_n$ dans un graphe orienté définit un *chemin* si et seulement si chaque arc e_i de cette suite relie le sommet s_{i-1} au sommet s_i. La notion de degré d'un sommet à l'intérieur d'un graphe orienté diffère elle aussi de la notion de degré pour un sommet à l'intérieur d'un graphe non-orienté. On appelle *degré entrant* d'un sommet v le nombre d'arêtes dont le sommet terminal correspond à v et *degré sortant*, le nombre d'arêtes dont le sommet initial correspond à v. Dans un autre ordre d'idée, un graphe est *pondéré* si on associe un poids à chacune de ses arêtes. La longueur d'un chemin à l'intérieur d'un graphe pondéré peut être calculée en sommant le poids des arêtes constituant celui-ci. Lorsque chacun des sommets d'un graphe non-orienté sont adjacents les uns aux autres, on dit que ce graphe est *complet.*

Introduisons maintenant la notion de sous-graphe et de graphe partiel. Un sous-graphe $G_i = (V_i, E_i)$ du graphe $G = (V, E)$ est défini par un ensemble de sommets $V_i \subseteq V$ et par l'ensemble $E_i \subseteq E$ des arêtes incidentes aux sommets dans V_i. Un graphe $G_i = (V, E_i)$ est un graphe partiel de G si $E' \in E$. Autrement dit, le graphe partiel G_i correspond au graphe G duquel on aurrait enlevé un ensemble d'arêtes. Une *clique* est un sous-graphe complet et un *stable* est un sous-graphe sans arêtes. Ces notions sont présentées dans la figure 3.3.

Prenons note que pour la suite de ce mémoire, nous considérerons les graphes non-orientés, non-pondérés et qui ne contiennent pas de boucles ni d'arêtes multiples. Ces graphes sont appelés *graphes simples.* Ce type de graphe est utilisé pour représenter des relations simples entre les entités d'un ensemble de données et sont d'autant plus

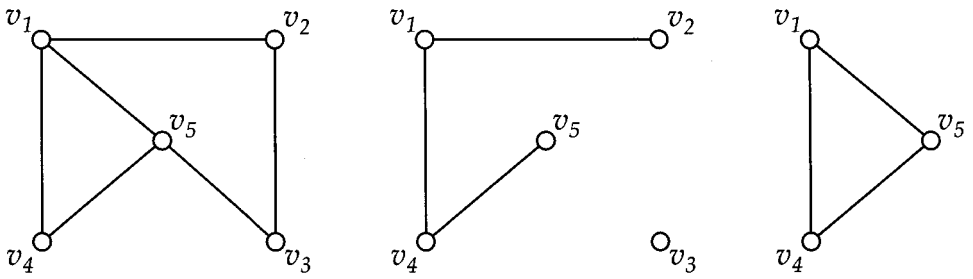


Figure 3.3: Sous-graphes et graphes partiel. Soit le graphe $G = (V, E)$ représenté en (a). Le graphe $G_p = (V, E_p)$ est un graphe partiel de G et est représenté en (b). Le graphe $G_s = (V_s, E_s)$ est un sous-graphe de G et est représenté en (c). On remarque aussi que G_s est une clique.

utiles pour en analyser la structure de ces relations comme nous le verrons dans les sections suivantes.

3.3 Exploration basée sur les valuations

L'exploration de graphes de grande taille peut être facilitée par l'utilisation de principes de réduction ou d'abstraction permettant de réduire la complexité visuelle de leur représentation graphique. Ces principes se basent souvent sur l'exploitation des propriétés structurelles d'un graphe. Ces propriétés sont propres à la topologie d'un graphe et indépendante de l'information qu'il représente. En particulier, nous porterons notre attention sur l'utilisation de mesures numériques associant une valeur aux éléments (arêtes ou sommets) d'un graphe. Ces mesures sont en fait des statistiques et sont nommées *valuations*. Plus précisément, une valuation est une fonction ϕ associant une valeur numérique à un élément (sommet ou arête) d'un graphe $G = (V, E)$. Une valuation peut être *structurelle* si elle donne de l'information sur la topologie d'un graphe, ou *sémantique* si elle donne de l'information sur les données représentées par celui-ci. Par exemple, considérons un graphe G représentant une base de données ci-

nématographique dont les sommets représentent des acteurs et les arêtes relient deux sommets si les deux acteurs associés ont fait partie de la distribution d'un même long métrage. Le nombre de films dans lequel un acteur a joué, l'âge d'un acteur ou toute autre information le concernant sont toutes des valuations sémantiques. Étant donné qu'elles utilisent l'information représentée par le graphe, les valuations sémantiques requièrent une bonne connaissance du domaine d'application du graphe étudié. Dans ce mémoire, nous porterons notre attention sur les valuations structurelles, qui elles, ne requièrent pas de connaissances préalables au domaine relatif à l'information représentée par le graphe. La section (3.3.1) présente un sommaire des valuations utilisées dans notre projet. Par la suite, diverses méthodes d'exploration de graphe seront présentées dans la section (3.3.2).

3.3.1 Valuations structurelles

Introduisons tout d'abord quelques définitions. Un graphe G' est *isomorphe* à G si et seulement s'il existe une fonction bijective ψ de V vers V' telle que pour tout couple de sommets adjacents (v_i, v_j), leurs images $\big(\psi(v_i), \psi(v_j)\big)$ sont adjacentes dans G'. On appelle *invariant* d'un graphe G une propriété restant identique pour tout graphe G' isomorphe à G. Le nombre de sommets, le nombre d'arêtes ainsi que le degré de chacun des sommets sont considérés comme des invariants d'un graphe. Il est à noter que toute valuation structurelle calculée sur un graphe est un invariant. En effet, la valeur calculée pour une valuation structurelle pour les éléments de deux graphes isomorphes donne nécessairement le même résultat car ceux-ci ont la même structure. Il existe un très grand nombre de valuations structurelles et la description d'une majorité de celles-ci dépasse la portée de cet ouvrage. Les sous-sections suivantes présentent certaines valuations choisies et utilisées dans notre application.

3.3.1.1 Propriétés structurelles

Toute propriété du graphe reliée à sa topologie peut être considérée comme une valuation structurelle. Le degré d'un sommet, ainsi que le degré maximal et minimal entre deux sommets définissant une arête, sont tous des valuations structurelles.

3.3.1.2 Indice de Jaccard

Étant donné une arête e formée de deux sommets u et v, l'indice de *Jaccard* [31] mesure le degré de similarité entre les sommets u et v quant à leur voisinage. Cet indice correspond essentiellement au ratio du nombre de sommets connectés à la fois à u et v par rapport au nombre de sommets étant connectés à u ou v. Plus formellement, soit le graphe $G = (V, E)$, $V_u \subset V$ l'ensemble des sommets connectés à u, et $V_v \subset V$ l'ensemble des sommets connectés à v, l'indice de *Jaccard* est dénoté s_{uv} et est défini comme suit :

$$s_{uv} \equiv \frac{|V_u \cap V_v|}{|V_u \cup V_v|} \,. \tag{3.1}$$

Un sous-graphe est dense lorsqu'un grand nombre de relations existe entre ses sommets. Une valeur faible signifie que les sommets u et v ont peu de relations similaires à l'intérieur du graphe et font donc partie de deux sous-graphes denses distincts. L'arête incidente à ces sommets relie ces deux sous-graphes et son élimination aurait alors de fortes chances de dissocier ces deux sous-graphes. Autrement, une valeur forte signifie que les sommets u et v ont des relations similaires et font partie d'un même sous-graphe dense.

3.3.1.3 Strength

Étant donné une arête e formée de deux sommets u et v, la valuation *Strength* [17] détermine si l'arête e fait partie d'un sous-graphe dense formé des voisinages de u et de v ou si elle correspond plutôt à un lien entre deux sous-graphes denses distincts.

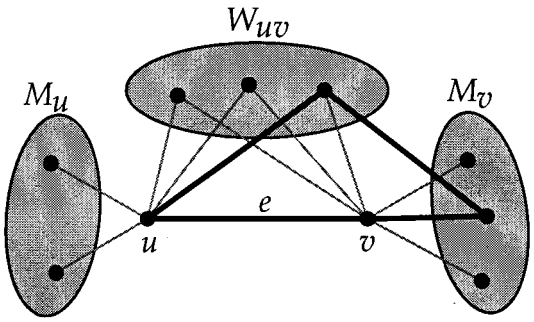


Figure 3.4: Valuation *Strength*. Chemin de longueur quatre passant par l'arête e. [17]

Cette valuation correspond au nombre de chemins effectifs de longueur trois et quatre passant par l'arête e par rapport au nombre total de chemins possibles (Fig 3.4).

Considérons les sous-ensembles $V_u \subset V$ et $V_v \subset V$ les voisinages des sommets u et v respectivement, ainsi que les sous-ensembles M_u, M_v et W_{uv} définis par :

$$M_u = V_u \backslash V_v \ , \tag{3.2}$$

$$M_v = V_v \backslash V_u \ , \tag{3.3}$$

$$W_{uv} = V_u \cap V_v \ . \tag{3.4}$$

Le sous-ensemble M_u correspond au voisinage du sommet u excluant le voisinage du sommet v, M_v correspond au voisinage du sommet v excluant le voisinage du sommet u et W_{uv} correspond à l'intersection des voisinages de u et v. On remarque que l'union de ces trois sous-ensembles correspond au voisinage des sommets u et v. Définissons maintenant la fonction s déterminant le ratio d'arêtes existantes reliant deux sous-ensembles de sommets $A \subset V$ et $B \subset V$ par rapport au nombre d'arêtes possibles étant donné le nombre de sommets présents dans A et B :

$$s(A, B) = \frac{e(A, B)}{|A||B|} \ . \tag{3.5}$$

Ici, $e(A, B)$ est une fonction déterminant le nombre d'arêtes connectant les sous-ensembles A et B. De plus, considérons la fonction $s(A)$ déterminant le ratio entre le nombre d'arêtes existantes reliant les sommets de A entre eux et le nombre d'arêtes possibles :

$$s(A) = \frac{e(A)}{\binom{|A|}{2}} . \tag{3.6}$$

Considérons maintenant les fonctions $\gamma_3(e)$ et $\gamma_4(e)$ déterminant le ratio de chemins existants de longueur 3 et 4 passant par l'arête e relativement au nombre de chemins possibles à l'intérieur du graphe :

$$\gamma_3(e) = \frac{|W_{uv}|}{|W_{uv}| + |M_u| + |M_v|} , \tag{3.7}$$

$$\gamma_4(e) = s(M_u, W_{uv}) + s(M_v, W_{uv}) + s(M_u, M_v) + s(W_u v) . \tag{3.8}$$

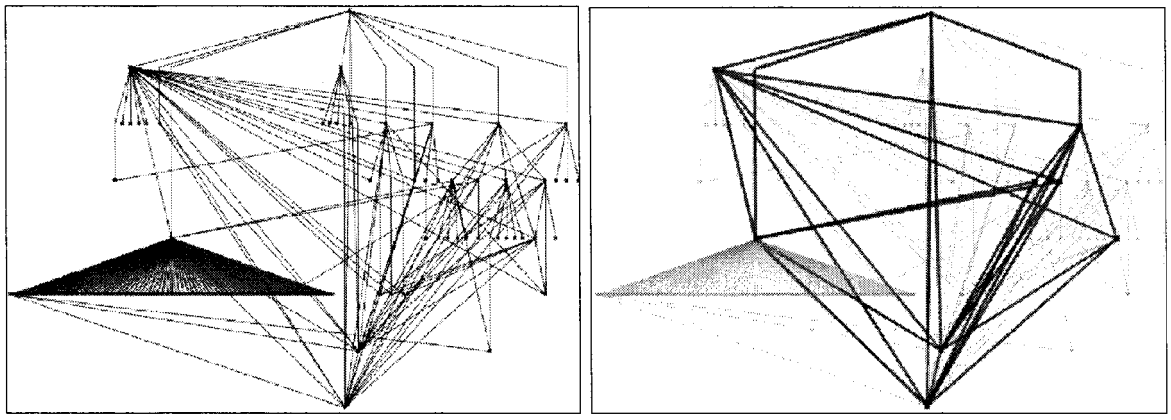
La valuation *Strength* est donnée par la sommation de ces deux fonctions :

$$\Sigma(e) = \gamma_3(e) + \gamma_4(e) . \tag{3.9}$$

Une valeur faible de $\Sigma(e)$ signifie généralement que l'arête e est un lien entre deux sous-graphes denses tandis qu'une valeur forte indique que l'arête e est plutôt située à l'intérieur d'un sous-graphe dense.

3.3.2 Filtrage et exploration

Une approche utilisée pour la visualisation de graphes est l'association de propriétés graphiques à des valuations [32] (Fig 3.5). Par exemple la taille de la représentation d'un sommet pourrait être associée à son degré. L'exploration de grands graphes peut être facilitée par cette technique car elle permet l'identification rapide de relations importantes entre ses sommets.



(a) (b)

Figure 3.5: Association de valuations aux caractéristiques graphiques de la représentation d'un graphe [32] : Les arêtes intéressantes relativement à une valuation structurelle donnée du graphe en (a) sont mises en évidence en (b).

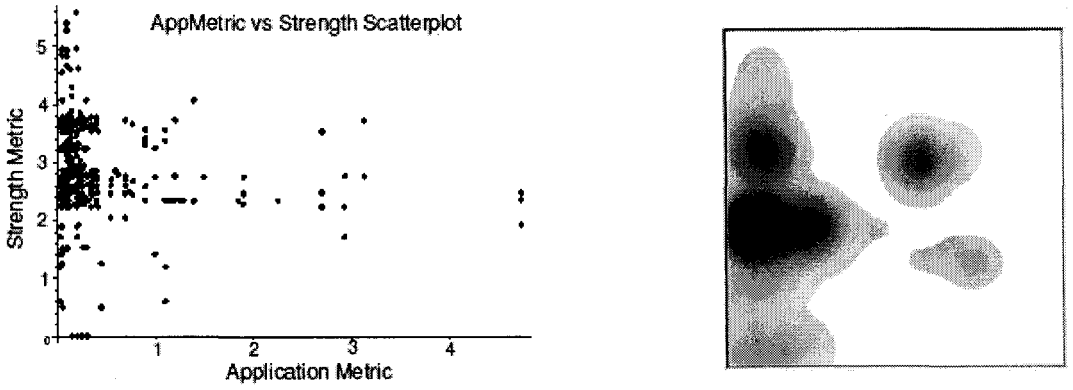
L'agglomération [33] est un autre exemple de processus permettant de réduire la complexité visuelle de la représentation d'un graphe. Les sommets sont groupés en agglomérats distincts lorsque leur valeur calculée par à une valuation est semblable. L'affichage du graphe peut ensuite être simplifié en le restreignant seulement qu'à une fraction des agglomérats ou encore en mettant l'emphase sur un agglomérat ciblé. Il est aussi possible de limiter la représentation graphique aux agglomérats eux-mêmes en représentant ceux-ci par des icônes (*glyphs*) liés par des arêtes plutôt que par l'ensemble de leurs sommets.

Nous nous intéresserons plus particulièrement au filtrage interactif appliqué à l'exploration de graphes. Le filtrage interactif permet à l'utilisateur de réduire le nombre d'éléments apparents dans la représentation graphique selon un critère donné. Dans le cas qui nous intéresse, le filtrage est effectué relativement à une valuation structurelle. L'utilisateur spécifie une valeur seuil et les sommets ou arêtes, dont la valeur calculée par rapport à cette valuation est inférieure à la valeur seuil, sont alors masqués. Reprenons l'exemple d'une valuation structurelle calculant le degré d'un sommet. En utilisant ce

critère, il nous est possible de limiter l'affichage du graphe aux sommets dont le nombre de relations est élevé. Ces opérations de filtrage peuvent être raffinées par l'utilisation combinée de plusieurs valuations. Par exemple, l'utilisation simultanée d'une valuation structurelle et d'une valuation sémantique permet de filtrer les éléments d'un graphe à la fois selon sa structure et selon l'information qu'il représente.

De façon générale la sélection d'une valeur seuil est effectuée à l'aide de boutons glissoires. Toutefois, l'utilisation de ce type d'interface peut s'avérer inefficace dans certains cas. En effet, la distribution statistique d'une valuation calculée sur un graphe est rarement uniforme. Un changement de valeur à une extrémité du bouton glissoire pourrait produire un effet d'envergure complètement différente d'un changement équivalent à l'autre extrémité. Il devient alors difficile pour l'utilisateur d'effectuer un filtrage précis. Cette particularité est aussi présente lors de l'association de propriétés graphiques à des valuations pour l'exploration visuelle d'un graphe. Cette limitation est une des motivations ayant mené au travaux de Chiricota, Jourdan et Melancon dans [18]. Ces travaux présentent un système d'exploration de graphe par filtrage interactif utilisant comme paramètres un couple de valuations. On peut interpréter le résultat du calcul de deux valuations sur les sommets d'un graphe comme un nuage de points à deux dimensions. Chaque axe du nuage de points représente une valuation et chaque point (x, y) représente un ou plusieurs sommet(s) ou arête(s). L'emplacement de ces point sur le premier axe est donné par la valeur calculée par la première valuation sur le sommet ou arête correspondant et l'emplacement sur le second axe est donné par la valeur calculée par la deuxième valuation sur ce même sommet ou arête. De plus, on associe une intensité à chaque point du nuage. En effet, plusieurs éléments du graphe peuvent être associés aux même valeurs de valuation et cette intensité représente le nombre d'éléments dans l'ensemble du graphe ayant pour valeurs calculées ce couple de

valeurs.



(a) (b)

Figure 3.6: Convolution d'un noyau gaussien sur un nuage de points [18] : Les points en (a) se fusionnent en (b) après l'application d'une convolution. Cette nouvelle image facilite la sélection de régions d'intérêt.

Afin de faciliter la sélection de régions d'intérêt à l'intérieur du nuage de points, on effectue la convolution d'un noyau Gaussien sur ce dernier. Cette opération a pour effet de diffuser les points du nuage en régions floues se fusionnant les unes avec les autres. En effet, selon Wattenberg et al. [70], l'utilisateur interprète ce qui lui est présenté à l'écran selon plusieurs "couches perceptuelles" arrangées de façon hiérarchique. Les couches au haut de la hiérarchie représentent la strucutre générale de l'image tandis que les couches plus basses représentent les détails. Prenons l'exemple d'un texte. Celui-ci peut être décomposé en quatre couches perceptuelles (Fig 3.7) : lettres, mots, lignes et paragraphes.

Wattenberg et al. proposent un modèle mathématique permettant d'approximer l'effet d'une couche perceptuelle plus élevée dans la hiérarchie à partir d'une image. Ce calcul est basé sur la convolution d'un noyau Gaussien sur l'image d'origine. Plus formellement, soit une image $f : [0, L] \times [0, L] \rightarrow [0, 1]$ où L est la taille de l'image

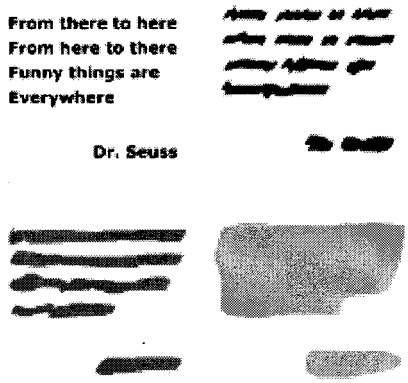


Figure 3.7: Un texte est décomposé en quatre couches perceptuelles [70].

d'origine, on y applique un filtre flou de facteur s de la façon suivante :

$$f_s = f * G_s \ , \tag{3.10}$$

où $*$ dénote la convolution. Le noyau Gaussien G_s est défini par :

$$G_s(x,y) = \frac{1}{2\pi s^2} e^{-(x^2+y^2)/2s^2} \ . \tag{3.11}$$

Dans notre cas, plus on utilise une couche structurelle haute dans la hiérarchie, plus les points adjacents dans le nuage se fusionnent pour former des régions floues uniformes (Fig 3.6). Ce processus permet à l'utilisateur de mieux percevoir la structure générale du nuage de points et facilite ainsi la sélection de zones d'intérêt. Il est souvent intéressant de sélectionner des régions régulières comme des corrélations fortes entre deux valuations et la sélection par *brushing* rectangulaire (en encadrant les données directement à l'intérieur de la représentation graphique) ne permet pas de telles actions aisément. En traitement d'image, la segmentation est un processus consistant à séparer une image en une ou plusieurs régions distinctes selon un critère donné. Plutôt que

d'utiliser une technique de *brushing*, la sélection d'une zone d'intérêt à l'intérieur des nuages diffus est effectuée à l'aide d'un simple clic et d'un algorithme de segmentation. Lorsque l'utilisateur effectue un clic à l'intérieur de l'ensemble de nuages diffus, le pixel sélectionné est utilisé comme point d'origine, et sa valeur, comme valeur seuil pour définir la région d'intérêt correspondante. Celle-ci est construite de façon itérative. Au départ, la région d'intérêt est définie par le pixel d'origine. À chaque itération, tout pixel adjacent à la région d'intérêt actuelle et dont la valeur est supérieur ou égale à la valeur seuil est ajouté à la région d'intérêt. La région finale est obtenue lorsqu'aucun autre pixel ne peut être ajouté à celle-ci. Cet alogrithme est nommé *Seeded Region Growing*[1]. Un sommet ou une arête du graphe fait partie de l'ensemble sélectionné si et seulement si son couple de valeurs de valuations (x, y) est inclus dans la région d'intérêt. La liste des sommets ou arêtes sélectionnés peut ensuite être utilisée afin de filtrer les éléments du graphe à visualiser (Fig 3.8).

Le travail présenté dans ce chapitre peut être vu comme une généralisation des travaux présentés dans [18]. Notre approche consiste essentiellement à explorer un graphe de grande taille à l'aide du calcul d'un triplet de valuations sur ses sommets ou arêtes. Le résultat du calcul de ces valuations peut-être exprimé sous la forme d'un nuage de points en trois dimensions pour lequel l'intensité d'un point représente la fréquence de ce triplet de valeurs. Étant donné la nature tridimensionnelle de cet ensemble de données, l'utilisation d'un algorithme de rendu de volume nous semble la meilleure option pour le visualiser. Ceci implique qu'après avoir été combiné à un noyau Gaussien à l'aide d'une convolution, le nuage de points devra être rééchantillonné sur un grille de voxels régulière.

L'exploration et la manipulation d'un ensemble de données en trois dimensions n'est pas un problème trivial. Certaines régions intéressantes du volumes peuvent être

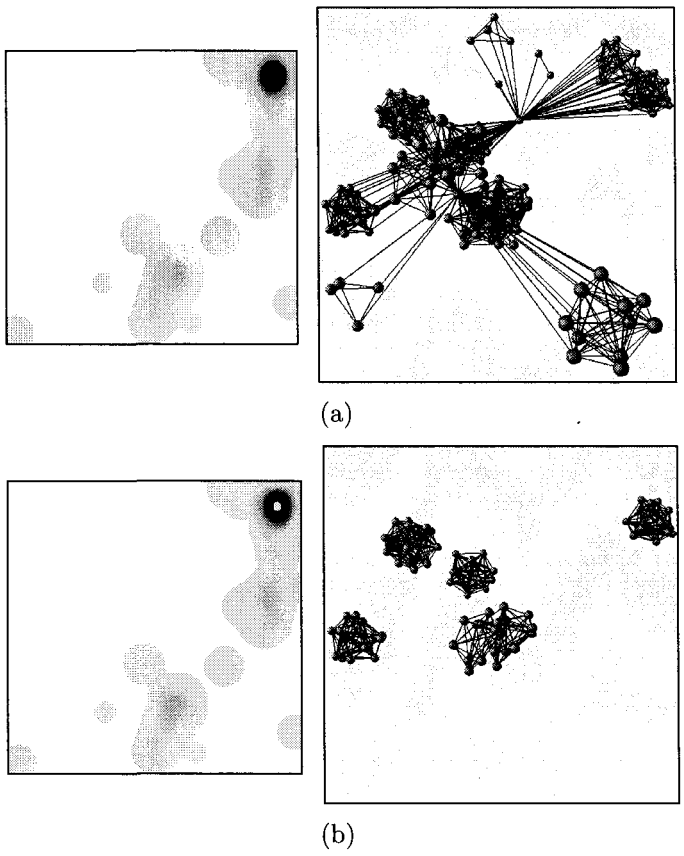


Figure 3.8: La représentation graphique d'un graphe est liée à la sélection d'une région d'intérêt dans un nuage de points à deux dimensions. En (a) et (b), les sommets dont les deux valeurs de valuations se situent dans la région sélectionnée en jaune sont mis en évidence dans la représentation graphique du graphe. [18].

cachées par d'autres ou même incluses l'une dans l'autre. C'est pourquoi nous proposons l'utilisation d'un contrôleur 3D afin d'explorer l'intérieur du volume. Ce dernier prend la forme d'une boîte rectangulaire et fonctionne un peu à la manière d'une sonde que l'on introduirait dans le volume et qui nous permettrait d'en extraire de l'information sous forme de coupes. Ces coupes sont sélectionnées en manipulant un curseur en forme de disque le long de la sonde. L'utilisateur peut ensuite sélectionner des régions d'intérêt en spécifiant une valeur seuil à l'intérieur de ces coupes. Ces régions sont rendues à l'écran à l'aide d'un algorithme de rendu de surface et peuvent être sélectionnées

indépendamment pour obtenir la liste des sommets ou arêtes associés.

La section (3.4) présente l'interface permettant à l'utilisateur d'interagir avec notre application. Les divers éléments constituant cette interface sont ensuite présentés en détail dans les sections suivantes. La section (3.5) décrit le processus utilisé pour créer un volume à partir des valuations calculées sur un graphe et d'une convolution. Ce volume est ensuite présenté à l'utilisateur à l'aide d'un algorithme de rendu de volume. L'utilisateur peut manipuler et configurer l'apparence de ce rendu grâce aux mécanismes décrits dans la section (3.6). Finalement, le contrôleur sonde ainsi que ses fonctionnalités sont ensuite introduits dans la section (3.7).

3.4 Interface utilisateur

Cette section vise à présenter l'interface utilisée dans notre application. Les différentes vues utilisées dans notre système sont présentées simultanément à l'observateur dans trois fenêtres séparées. Outre la vue tridimensionnelle permettant de visualiser le volume généré à partir du calcul de valuations sur le graphe, les autres fenêtres présentent de l'information relative au contrôleur et à sa configuration. Lorsqu'une action est effectuée sur le volume à l'intérieur de la vue 3D, les autres vues sont mises à jour dynamiquement. Par exemple, la fenêtre affichant l'image de la coupe actuellement sélectionnée est mise à jour de façon interactive lorsque l'utilisateur fait glisser le sélecteur de coupe le long du contrôleur. La figure 3.9 présente le fenêtrage utilisé dans notre application. La vue 3D est positionnée au milieu et les deux vues permettant de visualiser l'information recueillie par le contrôleur occuppent la partie droite de la fenêtre d'affichage. Un panneau de contrôle permettant de configurer le contrôleur et le rendu de volume est présenté à la gauche de la vue 3D. Après avoir importé un graphe, l'utilisateur sélectionne trois valuations et spécifie quelques paramètres relatifs à la génération du volume (voir section 3.5) à l'aide du panneau de contrôle. Une fois ces

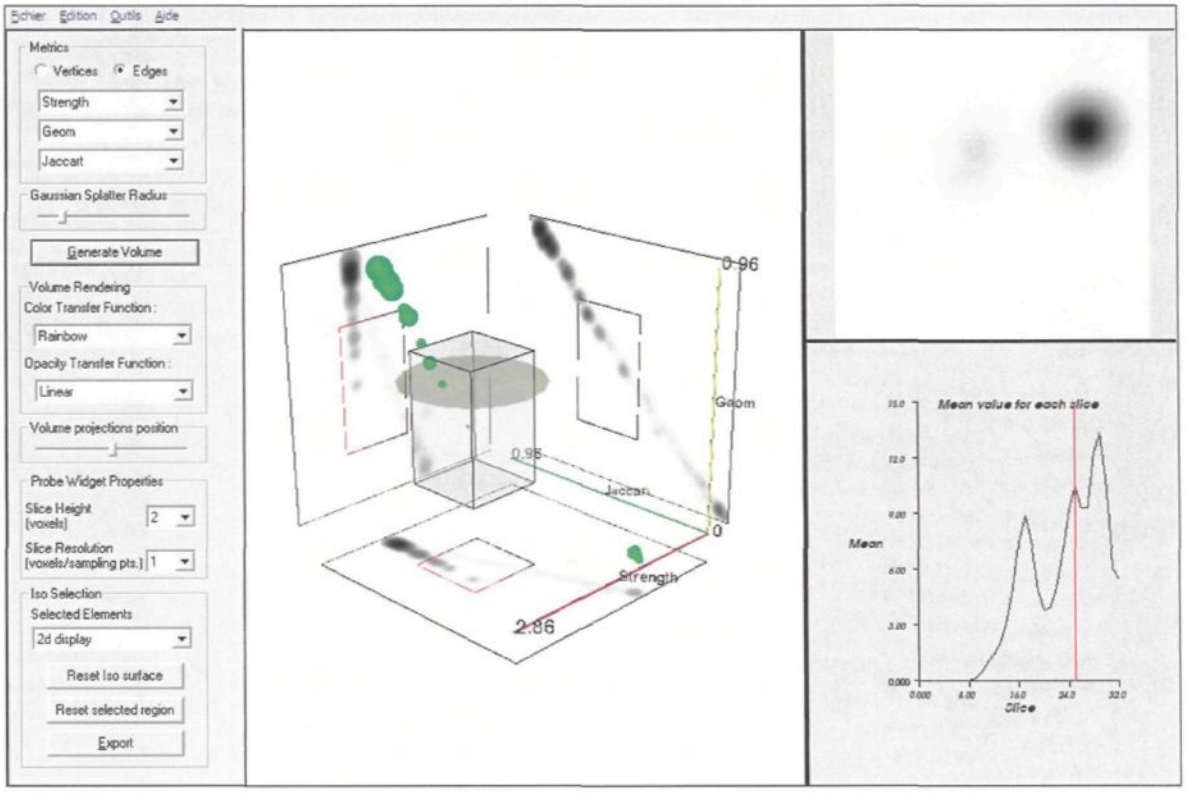
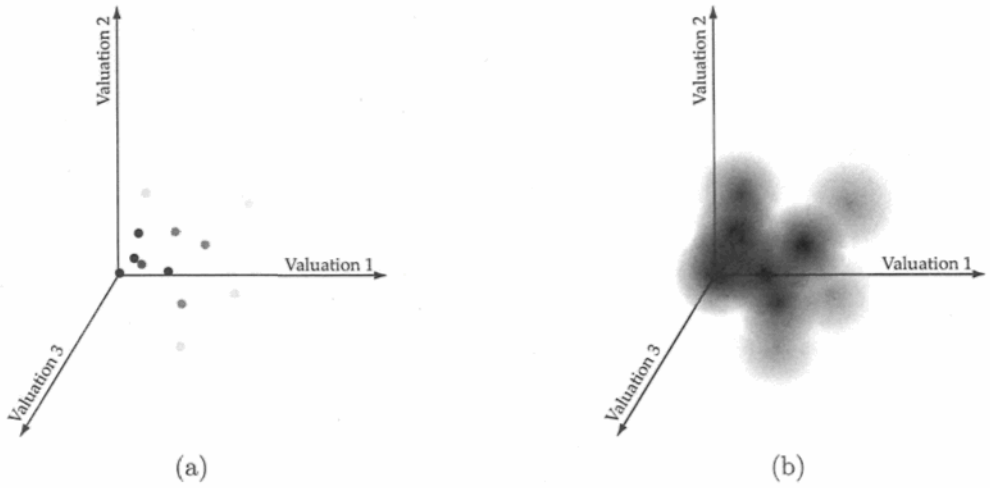


Figure 3.9: Fenêtrage du logiciel de visualisation.

paramètres spécifiés, le volume est généré. Les paramètres déterminant l'apparence du rendu de volume (voir section 3.6) ainsi que la configuration du contrôleur sonde (voir 3.7) peuvent aussi être modifiés à partir du panneau de contrôle. La liste des éléments extraits du volume à l'aide du contrôleur sonde est présentée dans une liste déroulante au bas du panneau de contrôle et peut être exportée vers un fichier texte.

3.5 Construction du volume



(a) (b)

Figure 3.10: Le nuage de points en (a) est diffus dans une grille régulière de voxels en (b) après l'application d'une convolution.

Le calcul d'un triplet de valuations sur chaque élément du graphe produit un histogramme en trois dimensions. Chaque élément est associé à un point (x, y, z) de l'histogramme et l'intensité de chacun de ces points correspond à la fréquence de ce triplet de valeurs pour l'ensemble du graphe (Fig 3.10a). Plus formellement, considérons le graphe $G = (V, E)$ et trois valuations ϕ_1, ϕ_2 et ϕ_3. Soit K_i un ensemble image, chaque valuation est une fonction $\phi_i : E \rightarrow K_i$ si elle est calculée sur les arêtes et $\phi_i : V \rightarrow K_i$ si elle est calculée sur les sommets. Maintenant, considérons les fonctions h_1, h_2 et h_3 telles que h_i compte le nombre d'éléments dans G étant associés à une valeur v

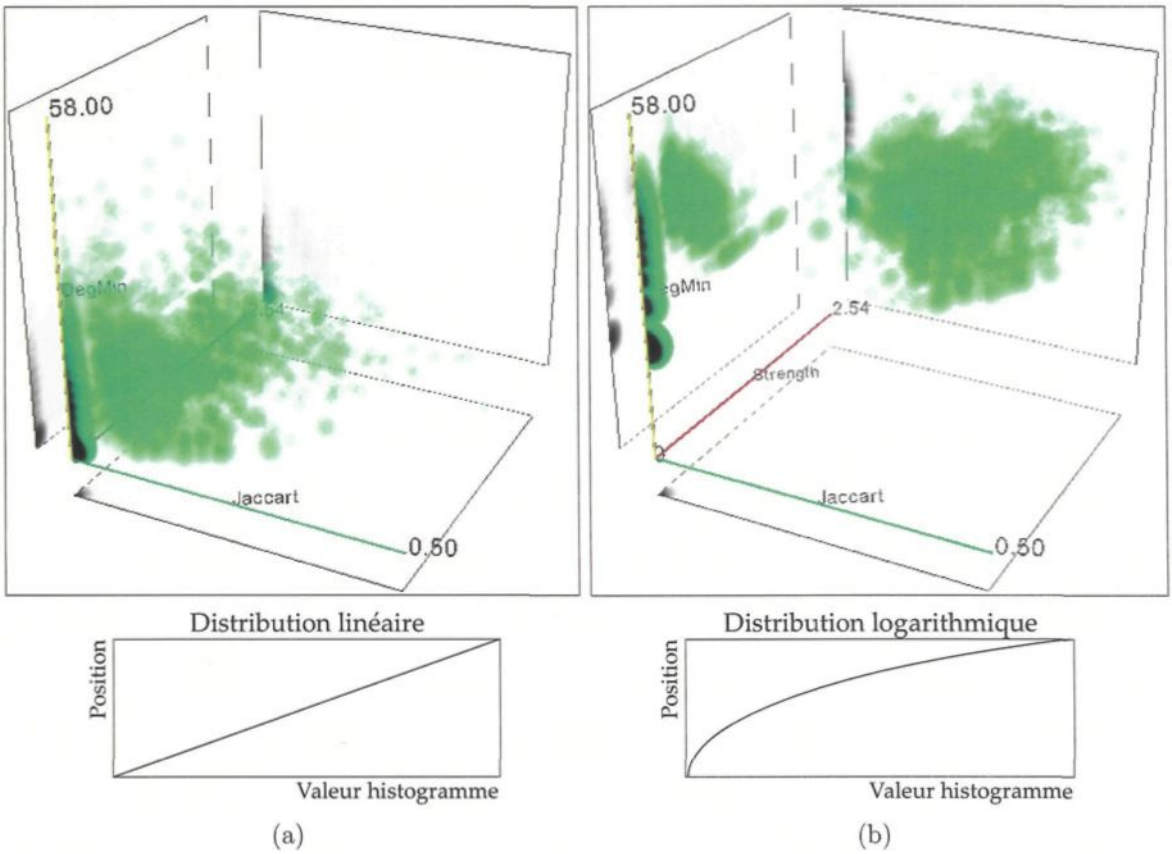
par la valuation ϕ_i. Ces fonctions correspondent essentiellement à l'histogramme de la distribution de chaque valuation. Étant donné n_i, la valeur calculée maximale pour la valuation i pour le graphe étudié et d la dimension du volume à générer, celui-ci est défini par :

$$f_h(x,y,z) = \left(d\frac{h_1}{n_1}, d\frac{h_2}{n_2}, d\frac{h_3}{n_3} \right) . \tag{3.12}$$

Ce processus peut toutefois créer des résultats peu satisfaisant lorsque les valeurs de l'histogramme tridimensionnel sont distribuées de façon non-uniforme. Le cas échéant, les points de fréquence élevée sont généralement regroupés près de l'origine car il arrive courrament d'avoir une grande quantité d'évaluations faibles (Fig 3.11a) pour certaines valuations. L'utilisation d'une distribution logarithmique nous permet alors de mieux disperser les données à l'intérieur du volume (Fig 3.11b). Dans ce cas, ce dernier est donné par :

$$\begin{aligned} f_h(x,y,z) \quad = \Bigg(& \log_d(\frac{d-1}{n_1 h_1 + 1}d), \\ & \log_d(\frac{d-1}{n_2 h_2 + 1}d), \\ & \log_d(\frac{d-1}{n_3 h_3 + 1}d) \Bigg) . \end{aligned} \tag{3.13}$$

Afin de faciliter la sélection de régions d'intérêt à l'intérieur du volume, la présentation d'un ensemble de nuages diffus représentant la structure générale et correspondant à une couche perceptuelle plus abstraite de l'histogramme tridimensionnel nous semble avantageuse. Ces raisons sont les mêmes que celles énoncées dans la section précédente. Pour ce faire, on effectue la convolution d'un noyau Gaussien au nuage de points et on rééchantillonne le résultat sur une grille de voxels régulière afin que celui-ci soit compatible avec un algorithme de rendu de volume. Les points de l'histogramme



(a) (b)

Figure 3.11: Distribution des échantillons de l'histogramme à l'intérieur du volume : On utilise une distribution linéaire en (a) et une distribution logarithmique en (b).

se diffuseront et formeront un ensemble de nuages diffus (Fig 3.10b). La contribution d'un point $\mathbf{p} = (i, j, k)$ de l'histogramme tridimensionnel au voxel de position (x, y, z) du volume convolué est donnée par l'expression suivante :

$$f_p(x, y, z) = S f_h(\mathbf{p}) e^{E(r/R)^2} , \tag{3.14}$$

où r est la distance entre le point $\mathbf{p}$ et le voxel en question. Les paramètres E, R et S permettent de configurer le noyau Gaussien : $E \leq 0$ est un facteur d'exponentiation correspondant au contraste du noyau, R correspond à son rayon et S est un facteur pondérant l'intensité du noyau relativement à la valeur scalaire $f_h(\mathbf{p})$. La valeur finale d'un voxel correspond à la valeur maximale de l'ensemble des contributions des points $\mathbf{p}$ du nuage :

$$f(x, y, z) = max(f_p(x, y, z)) . \tag{3.15}$$

Le volume résultant correspond à un ensemble de nuages diffus plus ou moins opaques. L'algorithme ainsi que les mécanismes utilisés pour présenter le volume à l'utilisateur sont décrits dans la section suivante.

3.6 Présentation du volume

Suite à la revue de littérature présentée dans la section (1.3), l'utilisation d'un algorithme de rendu de volume nous semble la meilleure solution pour visualiser un volume construit à partir du calcul d'un triplet de valuations sur un graphe. En effet, le volume obtenu suite aux opérations effectuées dans la section précédente est constitué d'un ensemble de nuages d'intensité plus ou moins forte. Tel que mentionné précédemment dans la section (1.3), un des avantages de ce type d'algorithme est qu'il nous permet de visualiser le volume en utilisant divers niveaux de transparence. La section (3.6.1) présente l'algorithme de visualisation utilisé dans notre application tandis que

les sections (3.6.2) et (3.6.3) présentent les mécanismes permettant à l'utilisateur de manipuler le volume.

3.6.1 Algorithmes et fonctions de transfert

La nature de nos données (le volume est constitué essentiellement de nuages semi-transparents) est la raison qui nous a fait opter pour un algorithme de lancer de rayons (voir section 1.3.2.1)(Fig 3.12). L'objectif de ce mémoire n'étant pas de développer un tel algorithme, nous utilisons les composantes logicielles de la librairie VTK (voir Annexe A). Celle-ci implémente un algorithme de lancer de rayons similaire à celui de Marc Levoy (voir section 1.3.2.1).

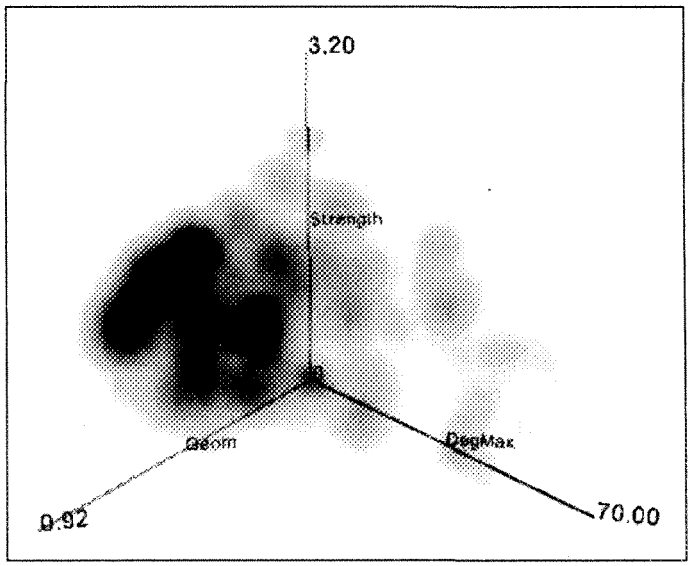


Figure 3.12: Volume rendu à l'aide d'un algorithme de lancer de rayons.

Tel que mentionné dans l'annexe A, l'apparence du rendu est déterminée par trois fonctions de transfert : une déterminant la couleur de la contribution d'un voxel, une pour son opacité et une autre déterminant son opacité selon la valeur du gradient à son emplacement. Cette dernière est facultative et peut être utilisée pour mettre l'emphase sur les zones de transition importantes dans le volume. Ces zones correspondent

généralement aux surfaces d'un objet. Par exemple, une zone de transition brusque entre deux matériaux d'un volume d'imagerie médicale correspond généralement à une surface délimitant un organe. Nous n'utiliserons pas cette fonction de transfert car la notion de zones de transition et d'objets est aberrante dans notre domaine d'application.

Notre application permet à l'utilisateur de chosir lui-même les deux fonctions qui déterminent l'apparence du rendu du volume qu'il étudie parmis une sélection de fonctions de transfert de couleur et d'opacité prédéterminées. Nous proposons trois fonctions de transfert de couleur (plus précisément deux monochromes et une polychrome) ainsi que trois fonctions de transfert d'opacité. L'utilisateur peut déterminer quelles fonctions de transfert il désire utiliser à l'aide de menus déroulants dans le panneau de contrôle.

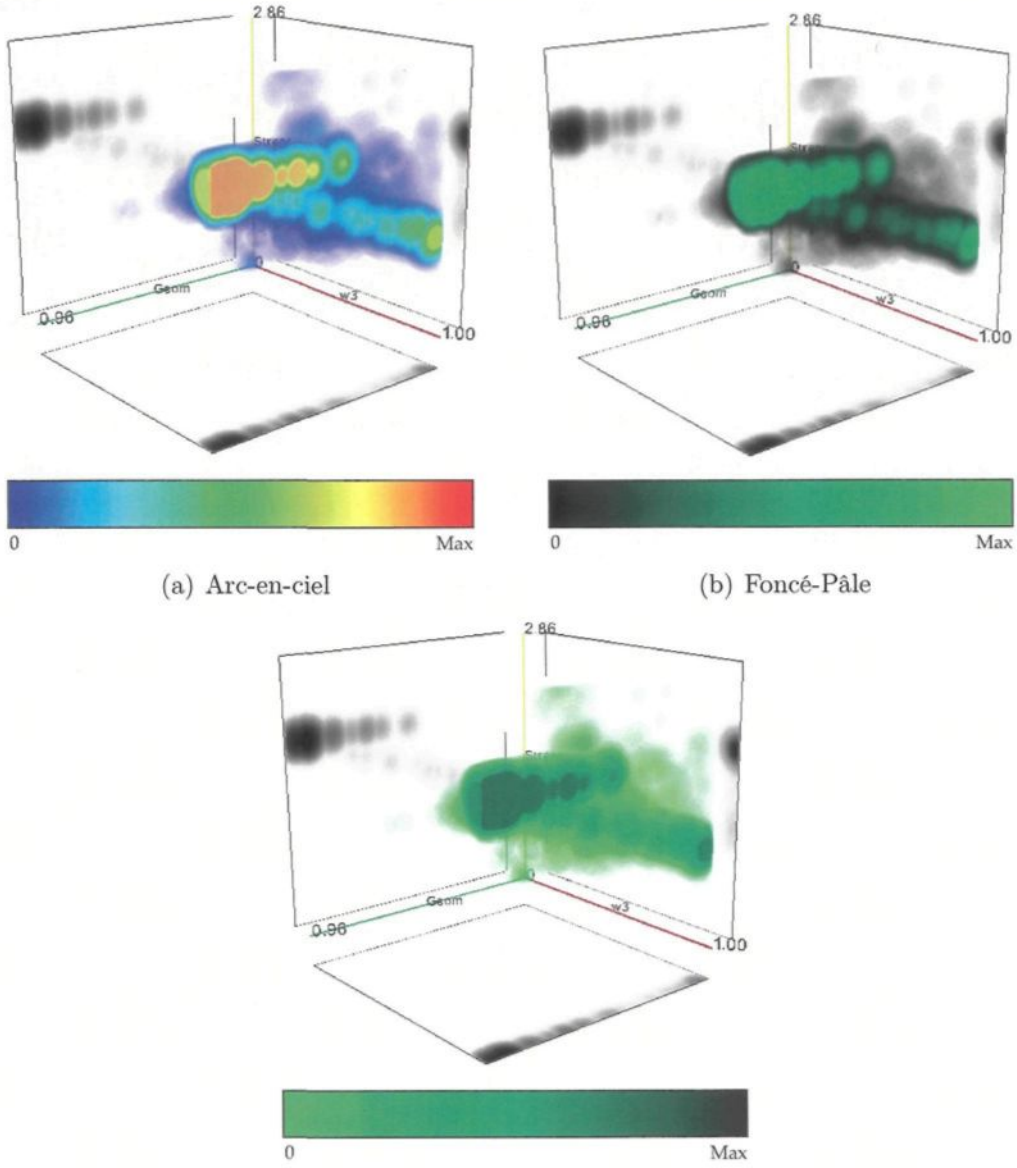
La fonction de transfert polychrome "arc-en-ciel" colore les voxels de faible valeur dans des teintes de bleu et les voxels de valeur plus élevée dans des teintes rougeâtres. Les valeurs intermédiaires sont interpolées linéairement le long du spectre chromatique (Fig 3.13a). Cette fonction de transfert est intéressante car elle utilise une grande gamme de couleurs, mais a toutefois le désavantage de donner l'impression à l'utilisateur que les données sont regroupées en régions distinctes. En effet, une zone de transition linéaire de valeurs scalaires n'apparaît pas comme une transition linéaire de couleurs aux yeux de l'observateur, mais plutôt comme une série de changements plus ou moins aigüs de teintes (voir [10] et [57]). Les deux fonctions de transfert monochromes colorent l'ensemble des voxels d'une teinte unique, mais d'intensité différente selon leur valeur. La première fonction colore un voxel d'intensité faible si sa valeur est basse et d'intensité forte si celle-ci est élevée. La deuxième fonction monochrome effectue essentiellement la coloration inverse. Celles-ci utilisent une gamme de couleurs moins large, mais offrent à l'observateur une meilleure coloration au niveau perceptuel. Pour une fonction mono-

chrome, une variation scalaire linéaire apparaît comme une variation linéaire de couleur aux yeux de l'observateur (Fig 3.13b et Fig 3.13c).

Les trois fonctions de transfert d'opacité utilisées dans notre système de visualisation nous permettent de mettre l'emphase du rendu sur différentes gammes de valeurs scalaires. Celles-ci sont toutes monotones croissantes et permettent de faire ressortir les voxels de valeur scalaire faible ou élevée du volume. Les fonctions de transfert ainsi que les différents résultats produits par celles-ci sont représentés dans la figure 3.14. Dans cet exemple, la fonction d'opacité faisant ressortir les voxels de valeur scalaire faible est adéquate pour observer la structure générale du volume dans son entiertée tandis que celle faisant ressortir les voxels de valeur scalaire élevée peut être utilisée pour cibler les zones à forte concentration. La fonction linéaire offre une combinaison de l'effet visuel des deux fonctions précédentes.

3.6.2 Manipulations

Afin d'observer le volume sous tous ces angles, il est essentiel de permettre à l'observateur de se déplacer librement autour du volume. On peut s'imaginer l'observateur comme étant une caméra visionnant l'environnement dans lequel le volume est présenté (Fig 3.15). La caméra est toujours orientée vers le haut et pointe vers le volume afin de ne pas confondre l'utilisateur. Le rendu du volume à l'aide d'un algorithme de lancer de rayon est une opération lourde. C'est pourquoi nous préservons un niveau acceptable d'interactivité en utilisant un rendu grossier lorsqu'une manipulation est effectuée et en pleine définition lorsque celle-ci est terminée. De cette façon, l'utilisateur reçoit un retour visuel de façon constante et peut évaluer l'effet de ses actions dynamiquement.

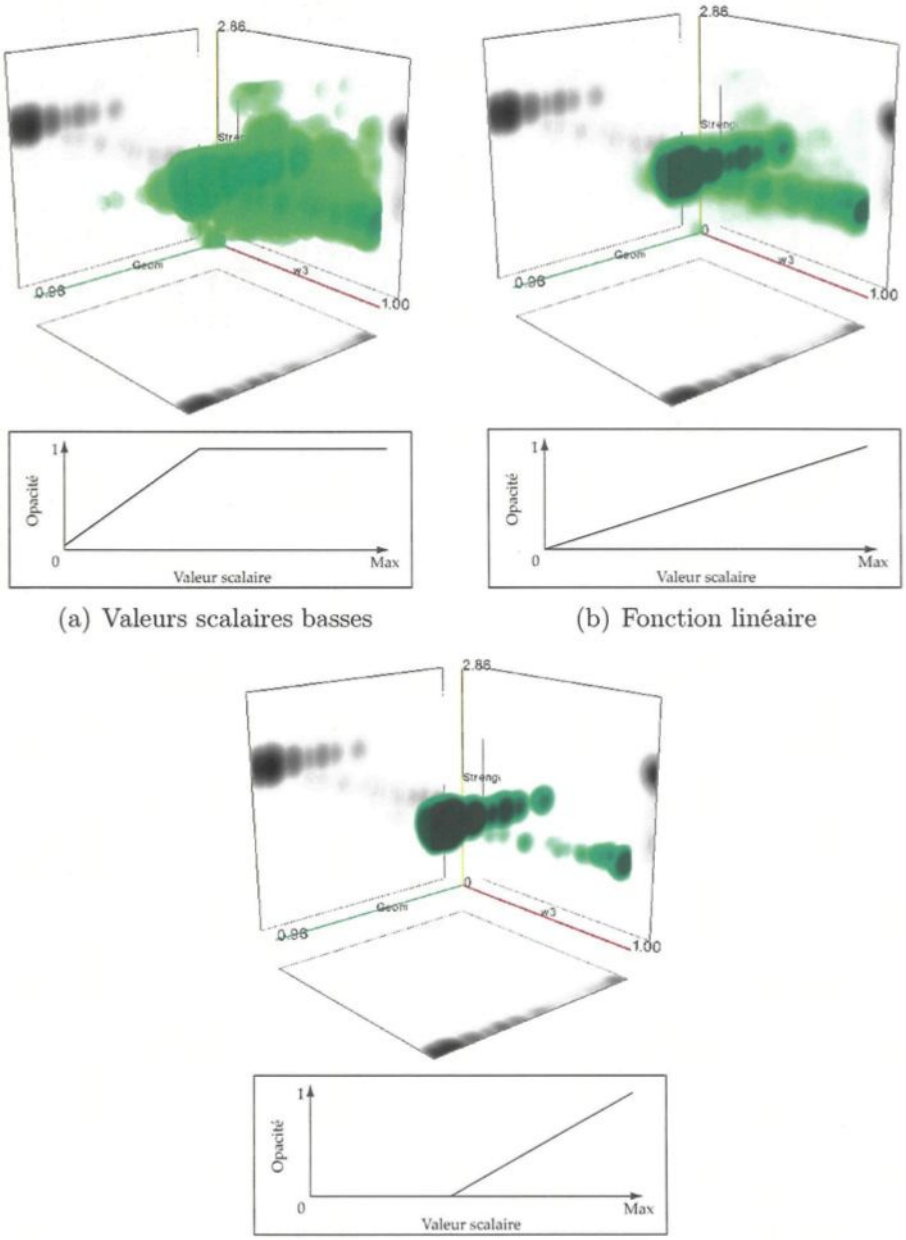


(a) Arc-en-ciel

(b) Foncé-Pâle

(c) Pâle-Foncé

Figure 3.13: Rendu de volume utilisant diverses fonctions de transfert de couleurs. En (a), une plus grande gamme de couleurs est utilisée, mais les zones de transitions linéaires n'apparaissent pas comme étant linéaires étant donné le changement de teinte. En (b) et (c), une fonction monochrome est utilisée et les zones de transitions linéaires paraissent linéaires au yeux de l'observateur.



(a) Valeurs scalaires basses

(b) Fonction linéaire

(c) Valeurs scalaires élevées

Figure 3.14: Rendu de volume utilisant diverses fonctions de transfert d'opacité. En (a), une fonction de transfert faisant ressortir les voxels de valeur scalaire basse est utilisée. Cette fonction permet à l'utilisateur d'analyser la structure générale du volume. En (c), une fonction de transfert faisant ressortir les voxels de valeur scalaire élevée est utilisée. Cette fonction permet à l'utilisateur de repérer les zones de forte concentration. La fonction en (b) est linéaire et permet de combiner l'effet visuel des deux fonctions précédentes.

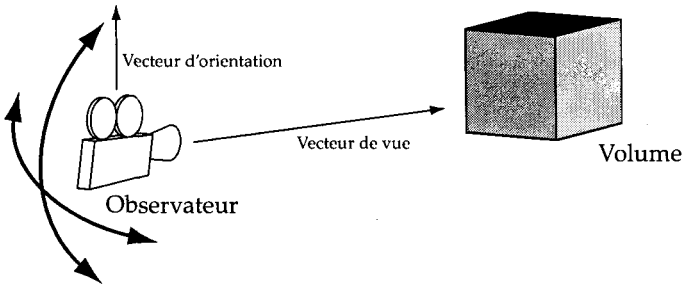


Figure 3.15: Manipulation du rendu. L'observateur (caméra) est toujours orienté vers le haut et pointe vers le volume. La caméra peut être déplacée autour du volume afin d'observer celui-ci sous tous ses angles.

3.6.3 Projections

Les volumes provenant de modalités d'imagerie, du domaine médical par exemple, représentent des organes ou autres formes familières et sont généralement facilement identifiables. À l'opposé, les volumes que nous étudions sont constitués d'un ensemble de nuages diffus dont la structure est abstraite et difficilement reconnaissable par l'utilisateur. L'utilisation unique d'un rendu tridimensionnel n'est donc généralement pas suffisante pour que l'utilisateur se construise un modèle mental fidèle du volume. Selon St John et al. [39], l'utilisation de vues bidimensionnelles conjointement aux vues trdimensionnelles dans un système de visualisation 3D est complémentaire et permet à l'utilisateur de mieux se situer dans l'espace. En effet l'utilisation d'une vue tridimensionnelle permet à l'usager de percevoir la structure générale du volume étudié tandis que l'utilisation de vues bidimensionnelles permet l'observation de détails et des relations spatiales entre le volume et les objets qui le compose. Par conséquent, l'utilisateur peut situer les objets dans l'environnement 3D les uns par rapport aux autres et ainsi les manipuler plus aisément. L'intégration de ces vues dans un même système de visualisation peut être effectuée selon trois approches : les icônes d'orientation (utilisation de fenêtres séparées), les méthodes "en place" (utilisation de plans de coupe à l'intérieur du

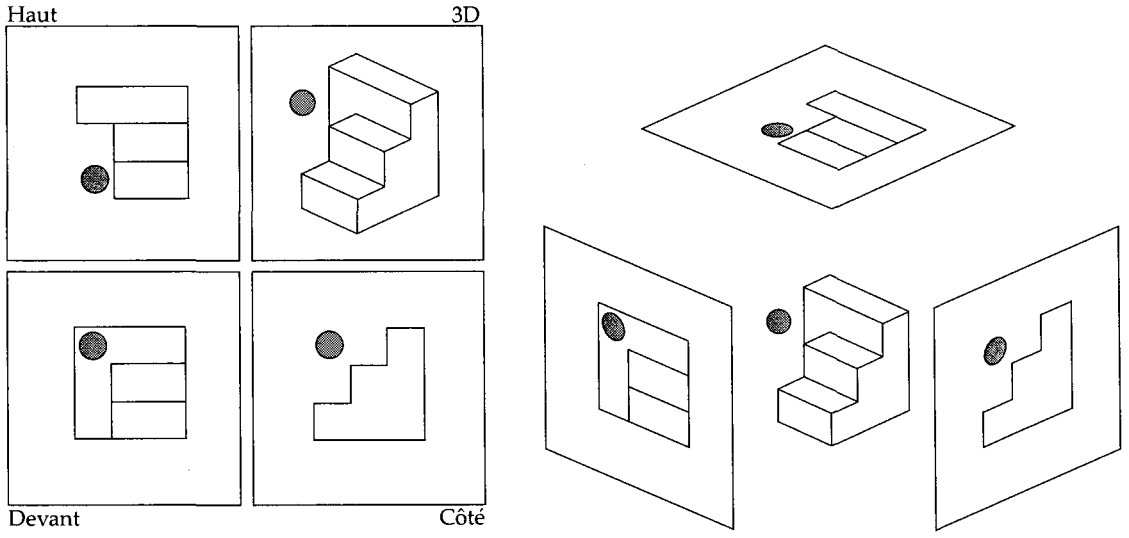
volume) ou la méthode *ExoVis*. Tory et al. comparent ces méthodes dans [64][66][65] :

En place La méthode "en place" permet d'explorer un volume à l'aide d'un plan de coupe. Le plan définit un demi-espace et est disposé à l'intérieur du volume. Ce dernier est alors rééchantillonné sur le plan et la partie du volume à l'extérieur du demi-espace est cachée. La relation entre le volume et la coupe bidimensionnelle est alors évidente car celle-ci est positionnée directement à l'intérieur du volume. Cette méthode est difficilement envisageable pour notre application car elle se prête mal à des vues bidimensionelles autres que des plans de coupes.

Icônes d'orientation La méthode par icônes d'orientation consiste à disposer les vues bidimensionnelles et tridimensionnelles du volume à visualiser côte à côte dans des fenêtres séparées (Fig 3.16a). Melanie Tory présente dans [63] une étude empirique démontrant que certaines difficultés de compréhension surviennent lors de l'étude d'un modèle 3D à l'aide de cette méthode. Ces difficultés sont dûes au fait que les vues sont séparées les unes des autres et que le lien entre chacune d'elles n'est pas évident, l'utilisateur devant effectuer une rotation mentale pour faire correspondre l'orientation des vues 2D à celle du volume.

ExoVis Cette méthode consiste essentiellement à présenter les vues orthogonales 2D sur des plans disposés autour du volume 3D (Fig 3.16b). L'orientation de ces plans respecte l'orientation des vues et celles-ci ne peuvent être déplacées qu'en conservant leur orientation d'origine. La relation entre les vues 2D et la représentation 3D est donc plus évidente avec *ExoVis* qu'avec la méthode par icônes orientés car l'utilisateur n'a pas à effectuer de rotation mentale des vues orthogonales.

L'intégration de trois vues orthgonales bidimensionnelles (une pour chaque axe) au système de visualisation nous semble avantageuse. Ce genre de configuration est courrament utilisé dans les systèmes de design assisté par ordinateur (CAD) pour lesquels



(a) Icônes d'orientation (b) ExoVis

Figure 3.16: L'utilisation conjointe de vues orthogonales 2D et d'une vue 3D à l'aide de la méthode des icônes d'orientation en (a) et d'ExoVis en (b) nous permet de mieux situer la balle relativement aux blocs dans l'environnement.

une bonne perception des rapports spatiaux entre les objets composant une scène est primordiale. Nous utiliserons la méthode *ExoVis* afin d'intégrer ces projections à notre système (Fig 3.17). De plus, cet ajout nous permettra d'étudier les relations statistiques telles que la corrélation entre deux valuations.

Notre application offre deux types de projections. Le premier type consiste à effectuer une sommation le long de l'axe de projection pour chaque pixel du plan. Plus précisément, la somme le long de l'axe z est calculée ainsi :

$$proj_z(x,y) = \sum_{i=0}^{n_z} f(x,y,i) \ , \tag{3.16}$$

où $f(x,y,i)$ est la valeur d'un voxel et n_z le nombre d'échantillons constituant le volume sur l'axe z. Ce résultat est ensuite normalisé en fonction de la valeur maximale de l'ensemble des pixels formant la projection (Fig 3.18a). Le deuxième type consiste à

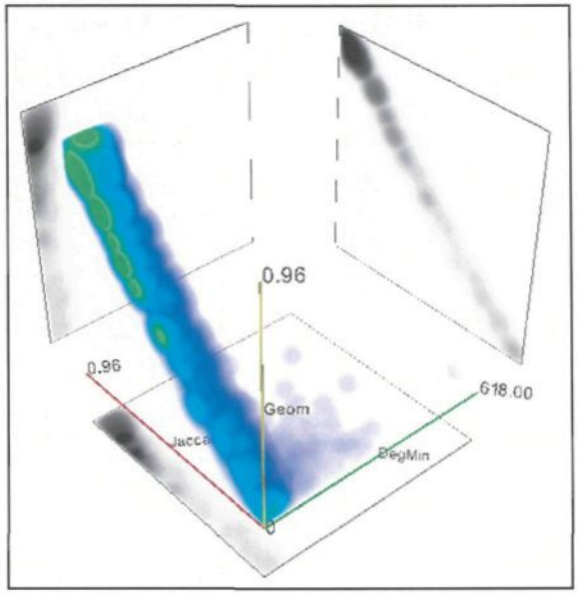
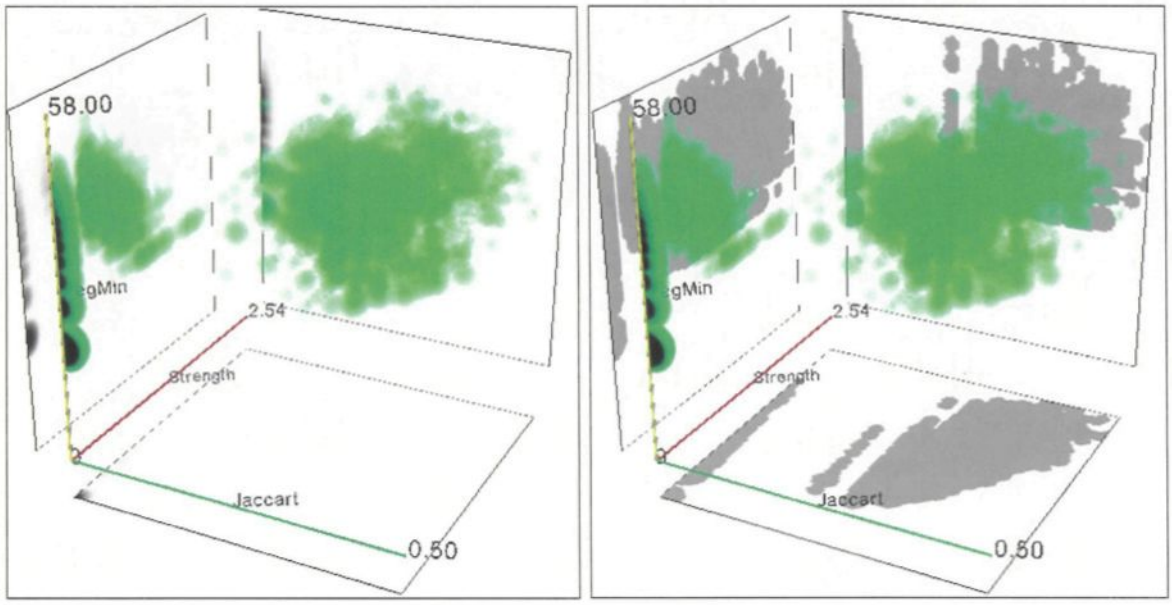


Figure 3.17: Intégration de projections en utilisant la méthode d'ExoVis. Ces projections nous permettent de mieux se représenter mentalement la structure générale du volume étudié. Dans cet exemple, on peut aussi remarquer une forte corrélation entre les valuations *Geometry* et *Jaccard*.

colorer un pixel du plan de projection s'il existe une valeur non nulle le long de l'axe de projection correspondant. Ce type de projection est dit binaire, car il produit un résultat en deux teintes (Fig 3.18b). L'utilisateur peut lui-même déterminer quel type de projection il désire utiliser à l'aide du panneau de contrôle.

3.7 Contrôleur sonde

Un des objectifs de notre application est de permettre à l'utilisateur du système de sélectionner des régions d'intérêt à l'intérieur du volume étudié afin d'en extraire de l'information. Cette tâche n'est pas triviale. En particulier, on ne peut sélectionner une région dans l'espace tridimensionnel par un simple clic de souris car cette opération ne permet de spécifier que deux coordonnées. Il existe a priori peu d'outils permettant d'explorer l'intérieur d'un volume dans la littérature et ceux-ci sont généralement basés



(a) (b)

Figure 3.18: Types de projections utilisées dans notre application. : On utilise des projections normalisées en (a) et des projections binaires en (b).

sur les projections "en place" (voir section 3.6.3). L'approche que nous proposons est basée sur l'utilisation d'un contrôleur tridimensionnel pouvant être positionné à l'intérieur du volume et permettant son exploration sous forme de coupes.

Le nombre de degrés de liberté d'un périphérique d'entrée correspond au nombre de paramètres indépendants qui définissent son orientation et sa position dans l'espace. La souris informatique possède deux degrés de liberté (elle peut se déplacer librement selon deux axes) et ne permet donc pas à l'utilisateur de spécifier trois coordonnées à l'aide d'une seule action. La sélection d'un point ou d'un groupe de points ciblé directement dans un ensemble de données à trois dimension est une opération qui peut difficilement être effectuée à l'aide d'un dispositif de pointage à deux degrés de liberté comme une souris.

L'introduction d'un contrôleur (voir section 2.3) nous permet de passer outre

les limitations mentionnées ci-haut. Ce contrôleur prend l'apparence d'une sonde qui peut être orientée et positionnée à l'intérieur du volume. La sonde est constituée d'une série de grilles de points rectangulaires superposées et le volume est rééchantilonné sur celles-ci. Une fois l'opération de rééchantillonage terminée, ces grilles 2D correspondent à des coupes d'orientation arbitraire du volume. L'utilisateur n'a plus qu'à sélectionner une coupe et celle-ci lui est présentée sous forme d'image dans une fenêtre adjacente (voir figure 3.9). Une région d'intérêt peut ensuite être spécifiée en cliquant sur un point à l'intérieur de la coupe sélectionnée.

La section (3.7.1) décrit les caractéristiques ainsi que le fonctionnement du contrôleur sonde, et les sections (3.7.3) et (3.7.4) présentent les diverses opérations de manipulation et de paramétrage permettant de configurer le contrôleur afin d'extraire des données du volume.

3.7.1 Caractéristiques et fonctionnement

La sonde prend la forme d'un polyhèdre à six faces (boîte rectangulaire). Celle-ci est composée d'une représentation graphique lui permettant d'être repérée et manipulée à l'intérieur du volume ainsi que d'une grille de points servant au rééchantillonnage (Fig 3.19). Lorsque l'utilisateur manipule la sonde, sa représentation graphique est mise à jour interactivement. La grille de points est mise à jour ou regénérée seulement lorsque l'utilisateur a terminé d'interagir avec la sonde. Les sections suivantes présentent une description détaillée de ces deux facettes du contrôleur.

3.7.1.1 Grille de points de rééchantillonnage

Le contrôleur est constitué d'une série de grilles de points rectangulaires et régulières disposées les unes au dessus des autres, de façon à former une grille tridimensionnelle ayant la forme d'une boîte rectangulaire. Chaque grille rectangulaire représente

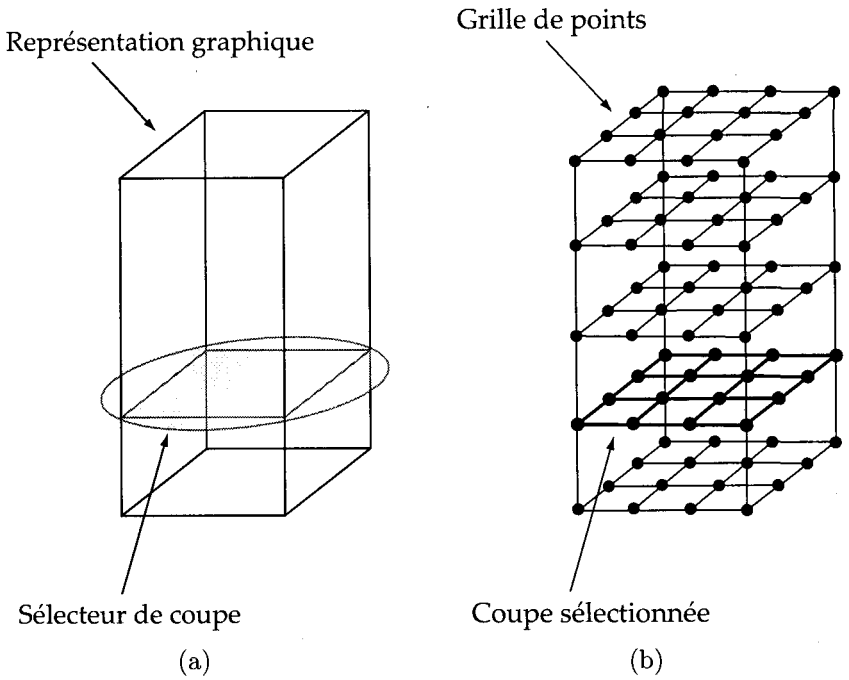


Figure 3.19: Schéma du contrôleur. La représentation graphique du contrôleur est constituée d'un polyhèdre à six faces et d'un disque permettant de sélectionner une coupe. La grille de points permet de rééchantillonner le volume et est constituée d'une série de grilles bidimensionnelles superposées. Dans cet exemple, la coupe sélectionnée en (a) est mise en évidence en (b).

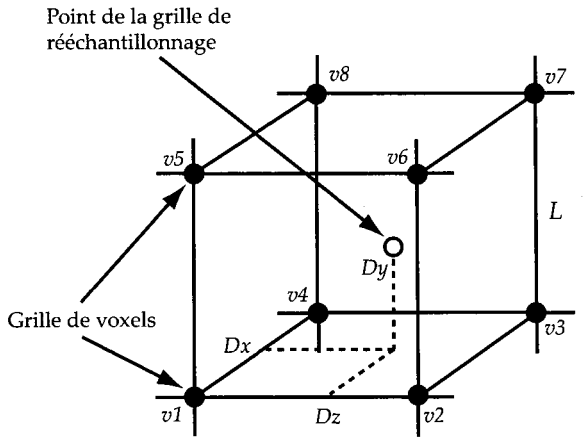


Figure 3.20: Rééchantillonnage du volume sur la grille de rééchantillonnage du contrôleur. On utilise l'interpolation trilinéaire.

une image de coupe d'orientation correspondante à celle du contrôleur. Le nombre de points constituant une grille ainsi que le nombre de grilles formant la sonde sont des paramètres déterminés par l'utilisateur et peuvent être spécifiés dans le panneau de contrôle. La grille de points est générée conformément à l'orientation, la dimension et aux paramètres de résolution du contrôleur lorsque celui-ci est activé ou redimensionné. Une fois la grille générée, le volume est ensuite rééchantillonné sur celle-ci. La valeur de chacun des points de la grille de rééchantillonnage est interpolée à partir des valeurs des voxels voisins du volume (Fig 3.20). Plus formellement, la valeur d'un point (x, y, z) positionné à l'intérieur d'une cellule cubique formée de huit voxels est estimée en effectuant une moyenne pondérée des valeurs de ces voxels :

$$\begin{aligned} f(x,y,z) &= f_{v_1}(1-D_x/L)(1-D_y/L)(1-D_z/L) \\ &+ f_{v_2}(D_x/L)(1-D_y/L)(1-D_z/L) \\ &+ f_{v_3}(D_x/L)(1-D_y/L)(D_z/L) \\ &+ f_{v_4}(1-D_x/L)(1-D_y/L)(D_z/L) \\ &+ f_{v_5}(1-D_x/L)(D_y/L)(1-D_z/L) \\ &+ f_{v_6}(D_x/L)(D_y/L)(1-D_z/L) \\ &+ f_{v_7}(D_x/L)(D_y/L)(D_z/L) \\ &+ f_{v_8}(1-D_x/L)(D_y/L)(D_z/L)\ , \end{aligned} \tag{3.17}$$

où L correspond à la distance entre deux voxels, D_x, D_y et D_z sont les distances entre le point de rééchantillonnage et le voxel v_1 et f_{v_i} est la valeur du voxel v_i. Cette formule est semblable à celle utilisé dans l'algorithme de Levoy lors du rééchantillonnage du volume le long d'un rayon (voir section 1.3.2.1). Les valeurs des points formant la grille de rééchantillonnage sont mis à jour à chaque activation du contrôleur, changement de

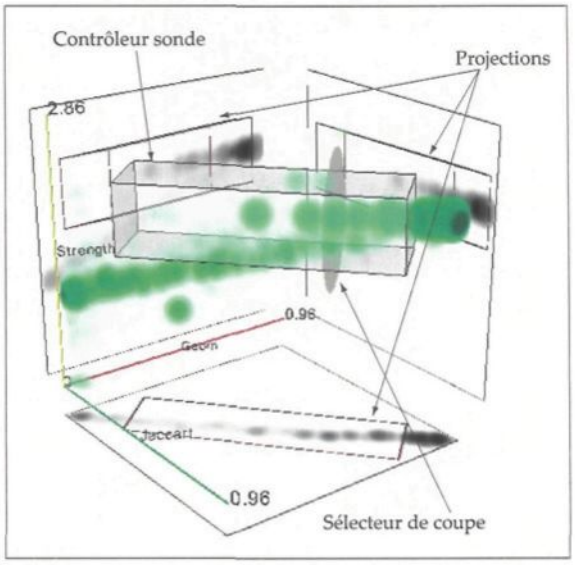


Figure 3.21: Le contrôleur sonde est représenté graphiquement par une boîte rectangulaire. Le disque peut être déplacé le long de la sonde et permet la sélection d'une coupe du volume. Des projections sont utilisées afin d'aider l'utilisateur à modifier la position du contrôleur relativement au volume.

configuration ou opération de manipulation effectuée sur le contrôleur. Une fois l'opération de rééchantillonage terminée, l'utilisateur peut sélectionner une coupe à l'intérieur du contrôleur. Celle-ci lui est alors présentée sous forme d'image dont les valeurs des pixels correspondent au valeurs normalisées des points de la grille rectangulaire composant cette coupe.

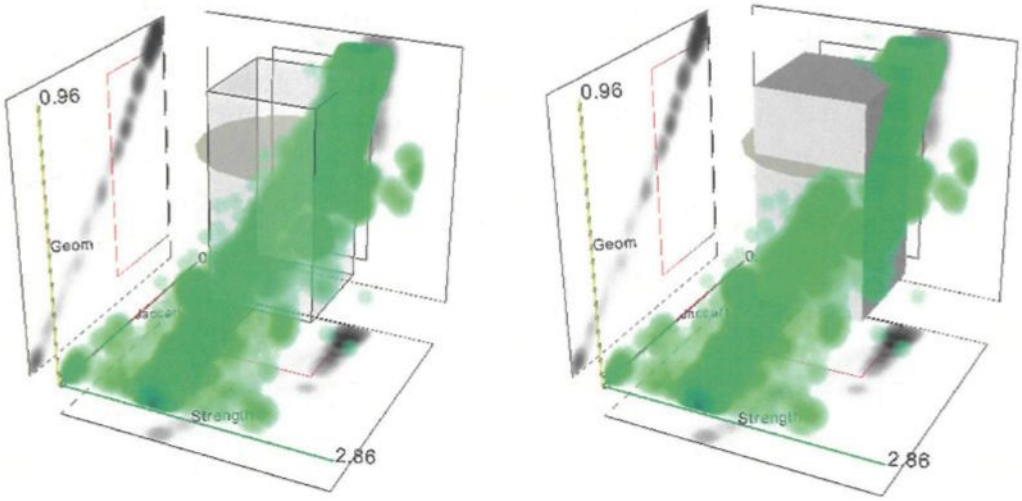
3.7.1.2 Représentation graphique

La représentation graphique de notre contrôleur consiste essentiellement en une boîte rectangulaire construite à partir de polygones (Fig 3.21). À cette boîte rectangulaire, s'ajoute un disque pouvant glisser le long du contrôleur et permettant la sélection d'une coupe. De plus, l'ajout de projections (voir section 3.6.3) du contrôleur au système de visualisation nous permet de mieux percevoir sa position et son orientation par rapport au volume. Ces projections se superposent à celles du volume. Tel que

mentionné dans la section (2.3), un contrôleur est un élément d'interface et par conséquent doit toujours être visible, mais ne devrait jamais cacher les données visualisées ou distraire l'utilisateur. C'est la raison pour laquelle notre outil est doté de trois modes d'affichage : semi-transparent, opaque et affichage restreint au contrôleur.

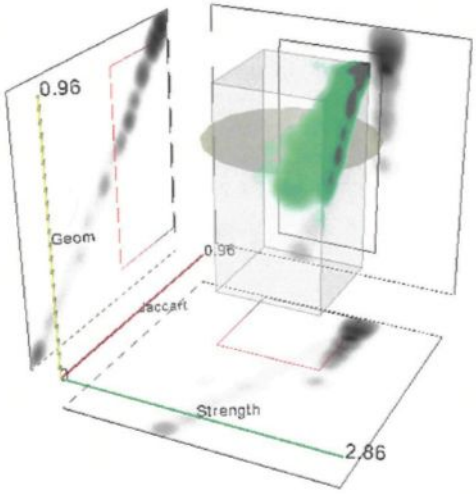
Contrôleur semi-transparent et opaque

Par défaut, le contrôleur est présenté comme une boîte semi-transparente délimitée par un cadre opaque (Fig 3.22a). La librairie graphique que nous utilisons permet le rendu mixte de volume et de polygones lorsqu'un ou plusieurs objets géométriques chevauchent partiellement ou entièrement le volume. L'algorithme utilisé consiste en premier lieu à effectuer le rendu des objets géométriques opaques, puis des objets géométriques semi-transparents. Cette étape est prise en charge par la librairie graphique OpenGL. Celle-ci utilise les fonctionnalités matérielles de la carte graphique de l'ordinateur. Lors de ce processus, OpenGL utilise un algorithme permettant de supprimer l'affichage de tout objet géométrique caché par un autre objet ou par lui-même. L'algorithme du *tampon Z* ou *tampon de profondeur* consiste essentiellement à déterminer, pour chaque pixel du plan image, quel objet géométrique est le plus près de l'observateur et d'afficher celui-ci. Ce processus est présenté dans la figure 3.23. Dans cet exemple, on trace un axe de projection passant par un pixel du plan image et perpendiculaire à ce dernier. L'axe traverse les différents objets composant la scène et l'objet dont la distance relative au plan image est la plus courte est celui dessiné à cet emplacement. La profondeur de la projection pour chaque pixel est emmagasinée dans un tampon dont la résolution spatiale est égale à celle du plan image. À l'origine, les cases mémoire du tampon sont initialisées à une valeur se rapprochant de l'infini. Pour chaque objet géométrique dessiné, on calcule la profondeur de chaque pixel du plan image. On compare ensuite ces valeurs à celles du tampon de pofondeur. Si la nouvelle valeur est plus



(a) Semi-Transparent

(b) Opaque



(c) Affichage restreint au contrôleur

Figure 3.22: Modes d'affichage du contrôleur sonde. En (a), le contrôleur est représenté par une boîte semi-transparente délimitée par un cadre opaque. Le cadre est caché par les régions de volume opaques tandis que la boîte reste toujours visible. En (b), on utilise une boîte opaque ainsi qu'une boîte semi-transparente. La boîte opaque est cachée par les régions de volume opaques tandis que la boîte transparente reste toujours visible. En (c), une boîte semi-transparente est utilisée et le rendu de volume est restreint à l'espace occupé par celui-ci.

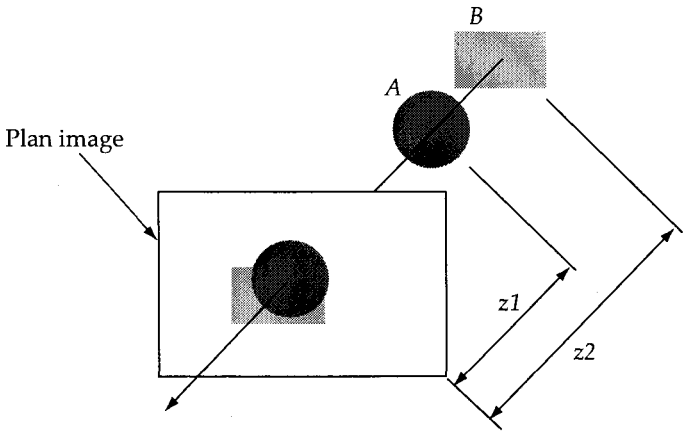


Figure 3.23: Algorithme du *tampon Z* [4].

élevée que celle contenue dans le tampon, cela signifie qu'un autre objet déja dessiné est plus près de l'observateur et par conséquent ce pixel est ignoré. Sinon, l'objet traité est plus près de l'observateur que tout autre objet ayant déja été traité et on colore le pixel selon sa couleur. Les couleurs des pixels sont emmagasinées dans un *tampon de couleurs* sous la forme d'un quadruplet $RGB\alpha$. Les valeurs RGB correspondent aux taux de saturation du mélange de couleurs en rouge, vert et bleu et α est un coefficient d'opacité. Chacune de ces valeurs est comprise dans l'interval $[0, 1]$ où 0 représente l'absence de couleur et 1 représente une pleine saturation.

Une fois le tampon Z et le tampon de couleurs mis à jour, l'algorithme de lancer de rayons est exécuté. Pour chaque pixel du plan image, on simule le lancer d'un rayon qui traverse le volume. La couleur du pixel à partir duquel on a lancé le rayon est obtenue en composant différentes valeurs de voxel échantillonnées le long de ce dernier (voir section 1.3.2.1). Les valeurs de profondeur emmagasinées dans le tampon Z sont converties en distances le long du rayon et sont ensuite utilisées pour borner ce dernier. De cette façon, pour un pixel donné, le volume ne sera pas dessiné si celui-ci est situé à l'arrière d'un objet géométrique. La couleur résultant de l'opération de composition

effectuée par l'algorithme de lancer de rayons est ensuite composée à celle déja obtenue pour le pixel correspondant lors du rendu géométrique. On utilise une opération de mélange alpha (*alpha blending*) pour composer ces deux valeurs. On appelle combinaison convexe une combinaison linéaire dont les coefficients sont non-négatifs et totalisent 1. Le mélange alpha est une combinaison convexe de deux quadruplets $RGB\alpha$ décrivant un mélange de couleurs. Étant donné la couleur de destination $\mathbf{C}_d\alpha_d = (R_d, B_d, G_d, \alpha_d)$ du pixel contenu dans le tampon de couleurs après rendu géométrique et la couleur source $\mathbf{C}_s\alpha_s = (R_s, B_s, G_s, \alpha_s)$ résultant du lancer de rayons, la couleur après composition par mélange alpha est donnée par :

$$\begin{aligned} \mathbf{C}'_d\alpha'_d &= \big(\alpha_s R_s + (1-\alpha_s)R_d, \alpha_s G_s + (1-\alpha_s)G_d, \\ &\qquad \alpha_s B_s + (1-\alpha_s)B_d, \alpha_s\alpha_d + (1-\alpha_s)\alpha_d\big) \ . \end{aligned} \tag{3.18}$$

On remarque que, pour un pixel donné, si la couleur source résultant du lancer de rayon est complètement opaque, la couleur de ce pixel après rendu géométrique n'aura aucune incidence sur le résultat final.

L'algorithme du tampon Z comporte toutefois une limitation importante. Lorsqu'on effectue le rendu de polygones semi-transparents, le tampon Z n'est pas mis à jour. Étant donné cette particularité, les polygones semi-transparents sont superposés au volume lors du rendu mixte de volume et d'objets géométriques. Pour cette raison, l'utilisation conjointe d'objets géométriques semi-transparents (boîte rectangulaire) et opaques (cadre) nous permet de mieux évaluer la position du contrôleur relativement au volume. Les polygones opaques (cadre) sont cachés par les zones opaques du volume tandis que les polygones semi-transparents (constituant la boîte) restent toujours visibles. La version opaque du contrôleur sonde fonctionne similairement. La partie de la sonde à l'avant du volume cache ce dernier tandis que la partie à l'intérieur de celui-ci

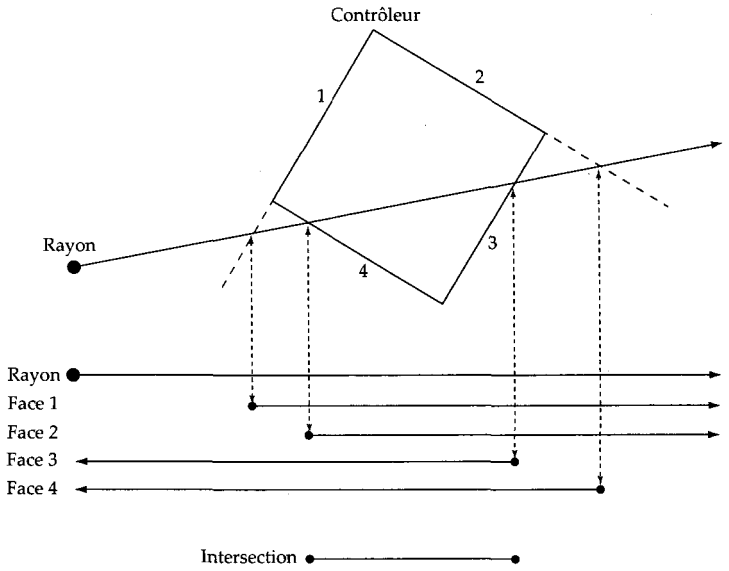


Figure 3.24: Intersection entre un rayon et le contrôleur. Chacune des faces du polyhèdre définit un demi-espace. L'intersection d'un rayon avec ces demi-espaces correspond à une demi-droite. L'intersection de l'ensemble de ces demi-droites correspond à la section du rayon étant incluse à l'intérieur de la sonde [6].

reste visible mais semi-transparente.

Rendu de volume restreint au contrôleur

Les deux modes d'affichage précédents ne nous permettent pas d'explorer l'intérieur du volume directement. C'est pour cette raison que nous avons intégré un autre mode d'affichage permettant de restreindre le rendu de volume à l'espace occupé par le contrôleur. Chaque rayon lancé lors du rendu peut traverser le contrôleur en deux points. Il s'agit donc de déterminer l'emplacement de ces points le long du rayon et d'utiliser ceux-ci pour borner son parcours.

La boîte rectangulaire définissant le contrôleur est un polyhèdre constitué de six faces. Chacune de ces faces correspond en fait à une section de plan. Un plan passant par un point $\mathbf{x_0} = (x_0, y_0, z_0)$ peut être exprimé par quatre paramètres à l'aide de

l'expression suivante :

$$p(x, y, z) = ax + by + cz + d = 0 \ , \tag{3.19}$$

où les paramètres a, b et c définissent le vecteur $\vec{P_n}$ normal au plan et $d = -ax_0 - by_0 - cz_0$. Chacun de ces plans délimite un demi-espace défini par l'expression suivante :

$$D_p = \{(x, y, z) \in \mathbb{R}^3 | p(x, y, z) > 0\} \ . \tag{3.20}$$

L'intersection de l'ensemble de ces demi-espaces correspond au volume occupé par le polyhèdre. On remarque que l'intersection d'un rayon et d'un demi-espace correspond à une demi-droite. L'intersection des demi-droites résultant des intersections du rayon avec l'ensemble des plans formant le polyhèdre correspondant au segment de droite inclus dans le contrôleur (Fig 3.24). Plus formellement, on définit un rayon par un point d'origine $\mathbf{R_o} = (x_o, y_o, z_o)$ et un vecteur de direction normalisé $\vec{R_d} = (x_d, y_d, z_d)$. On peut définir un point se trouvant à une distance t le long du rayon par :

$$\mathbf{R}(t) = \mathbf{R_o} + t\vec{R_d} \ . \tag{3.21}$$

Déterminer les points d'intersection du rayon et de la sonde revient donc à trouver les distances t_{near} d'entrée et t_{far} de sortie du rayon à l'intérieur du polyhèdre représentant le contrôleur. À l'origine, t_{near} et t_{far} sont initialisés respectivement à $-\infty$ et $+\infty$. Pour chaque plan, la distance t le long du rayon correspondant au point d'intersection est définie par :

$$t = -v_n / v_d \ , \tag{3.22}$$

où

$$v_n = \vec{P_n} \cdot \mathbf{R_o} + d \,,$$
$$v_d = \vec{P_n} \cdot \vec{R_d} \,.$$

Il faut maintenant déterminer si le plan est situé devant la sonde (pouvant modifier t_{near}) ou à l'arrière de celle-ci (pouvant modifier t_{far}). Si $v_d = 0$, le rayon est parallèle au plan et n'entre donc pas en intersection avec celui-ci. Si $v_d > 0$, le rayon percute une face à l'arrière de la sonde. Si $t < t_{far}$, t_{far} est alors mis à jour. Autrement, si $v_d < 0$, le rayon percute une face au devant de la sonde et t_{near} est mis à jour si et seulement si $t > t_{near}$. Une fois ces tests effectués pour l'ensemble des plans définissant la sonde, le rayon est borné par l'interval $[t_{near}, t_{far}]$ et le processus est répété pour chaque rayon. Comme le démontre la figure 3.24, ce mode d'affichage permet d'effectuer des coupes d'orientation arbitraire à l'intérieur du volume et de visualiser celles-ci directement dans le rendu, un peu à la manière de la méthode "en place" décrite dans la section (3.6.3). De plus, cacher une certaine partie du rendu permet à l'utilisateur de garder le focus sur la région étudiée et lui évite toute distraction provenant d'autres régions non ciblées.

3.7.2 Visualisation d'une coupe

Tel que mentionné dans les sections précédentes, le contrôleur est constitué d'une grille de points composée de couches rectangulaires. Une fois l'opération de rééchantillonnage terminée, chaque couche représente une coupe du volume. L'utilisateur peut visualiser ces couches en les sélectionnant à l'aide d'un curseur en forme de disque pouvant être déplacé le long de la sonde. Celles-ci sont alors normalisées et affichées dans des fenêtres séparées (Fig 3.25). Pour chaque coupe, la valeur moyenne des points de rééchantillonnage est calculée et ces résultats sont présentés dans un graphique cartésien.

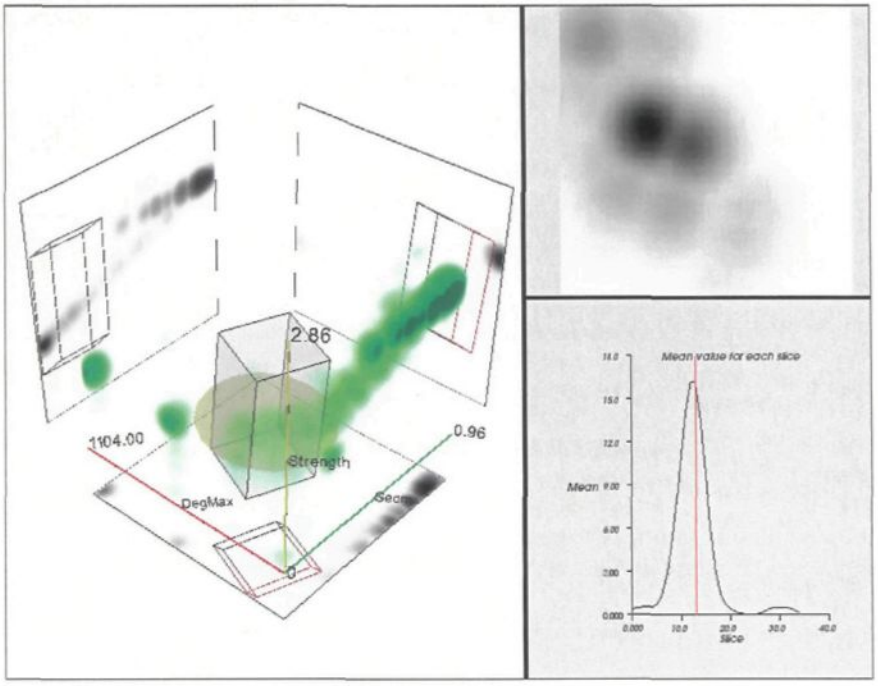
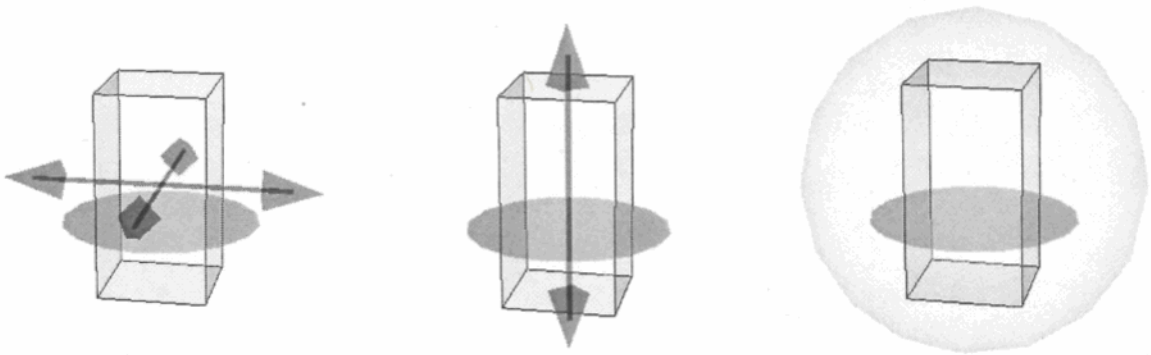


Figure 3.25: Sélection d'une coupe. La coupe sélectionnée est présentée dans une fenêtre séparée.

L'abscisse présente les couches numérotées de 0 à $n-1$, n étant le nombre de couches, tandis que l'ordonnée correspond aux valeurs moyennes. Lorsque l'utilisateur choisit une couche, celle-ci est représentée par un curseur rouge dans le graphique cartésien et est mise à jour de façon interactive.

3.7.3 Utilisation

Le contrôleur sonde est manipulé à l'intérieur du volume à l'aide de combinaisons de touches de clavier et de la souris. L'utilisateur peut le déplacer, le redimensionner et modifier son orientation. Lorsqu'une opération de transformation est effectuée sur le contrôleur, une aide visuelle est présentée à l'utilisateur. Les translations sont représentées par des flèches (Fig 3.26a) et (Fig 3.26b) indiquant l'axe sur lesquel elles sont restreintes et les rotations sont représentées par une sphère virtuelle [14] (Fig 3.26c).



(a) Translation restreinte aux axes x et z

(b) Translation restreinte à l'axe y

(c) Sphère virtuelle de rotation

Figure 3.26: Aides visuels de manipulation du contrôleur. Les flèches rouges en (a) et (b) sont utilisées lors d'une opération de translation pour montrer à l'utilisateur sur quel(s) axe(s) il peut déplacer le contrôleur. En (c), le contrôleur est entouré d'une sphère virtuelle lors d'une rotation. On remarque le curseur de sélection de coupe en (a), (b) et (c).

L'avantage de la sphère virtuelle est qu'elle simule les fonctionalitées d'un périphérique de type *trackball* et qu'elle permet une rotation autour de trois axes simultanément. Lors de ces manipulations, la représentation graphique est déplacée ou réorientée de façon interactive tandis que la grille de points associée au contrôleur n'est mise à jour que lorsque l'opération est terminée. On applique la transformation équivalente à la grille de points après une translation ou une rotation. Dans le cas d'un redimensionnement, la grille doit être regénérée afin de conserver la résolution spécifiée par l'utilisateur. Une fois la grille de points transformée ou regénérée, le volume est rééchantillonné sur celle-ci et l'affichage de la coupe sélectionnée ainsi que le graphique de valeurs moyennes de coupes sont mis à jour. Le comportement de notre contrôleur est décrit à l'aide d'un automate (voir section 2.3) dans la figure 3.27. Tel que mentionné préalablement, la sélection d'une coupe est effectuée en cliquant directement sur un disque de sélection transparent et en le déplaçant le long de la sonde à l'aide de la souris.

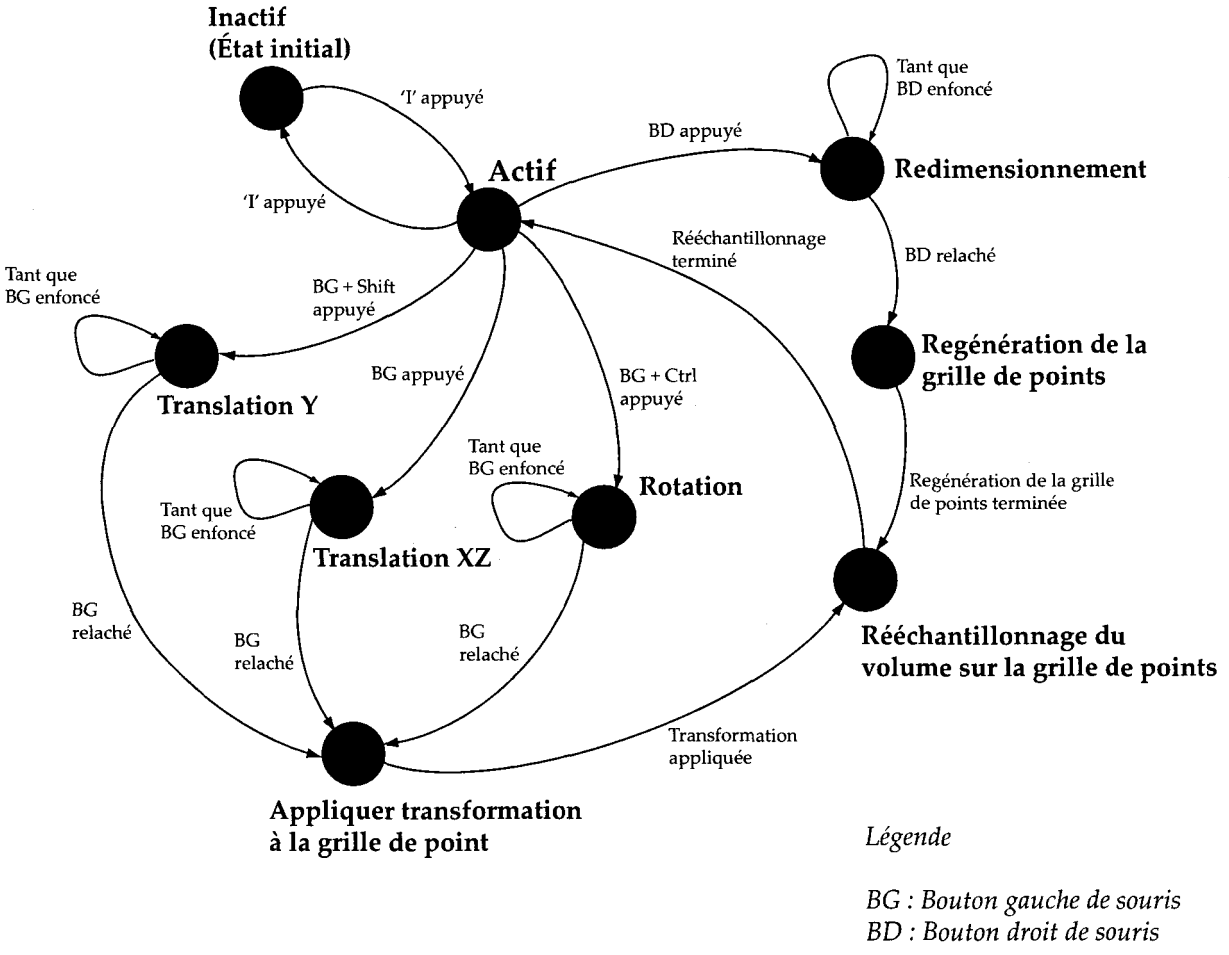


Figure 3.27: Automate décrivant le comportement du contrôleur sonde.

3.7.4 Sélection de régions d'intérêt

L'image de coupe actuellement sélectionnée permet la sélection directe d'un voxel à l'intérieur du volume. En effet la sélection d'un élément dans une grille tridimensionnelle est une opération complexe et peut devenir une opération triviale lorsqu'on ne considère qu'une partie de celle-ci. Dans notre cas, l'utilisateur interagit avec une grille bidimensionnelle qu'il a préalablement extraite du volume à l'aide du contrôleur sonde. Celui-ci n'a qu'à pointer l'élément désiré à l'aide d'une souris et le sélectionner à l'aide d'un clic. Le valeur du voxel correspondant au pixel sélectionné dans la coupe est ensuite utilisée comme iso-valeur pour générer une région d'intérêt par seuillage. Plus formellement, à partir de la valeur t choisie par l'utilisateur dans la coupe, les voxels composant le volume V sont regroupés en deux ensembles :

$$\begin{aligned} R &= \{(x,y,z) \in V \mid f(x,y,z) \geq t\} , \\ S &= \{(x,y,z) \in V \mid f(x,y,z) < t\} , \end{aligned} \tag{3.23}$$

où $f(x,y,z)$ est la valeur du voxel à la position (x,y,z), R correspond à l'ensemble des voxels inclus dans la région d'intérêt et S correspond à l'ensemble des voxels y étant exclus. On associe la valeur 1 à chaque voxel inclus dans R et la valeur 0 à chaque voxel inclus dans S à l'intérieur d'un nouveau volume. Ce dernier est ensuite rendu à l'aide de l'algorithme des marching cubes (voir section 1.3.1.2). (Fig 3.28). L'utilisateur peut basculer entre l'affichage de l'iso-surface représentant la région d'intérêt et l'affichage du volume ou combiner les deux modes. L'ensemble R est ensuite utilisé pour filtrer les sommets ou arêtes du graphe étant inclus dans la région d'intérêt. Soit le graphe $G = (V, E)$, l'ensemble de sommets et V l'ensemble d'arêtes E, l'ensemble $V' \subseteq V$ des

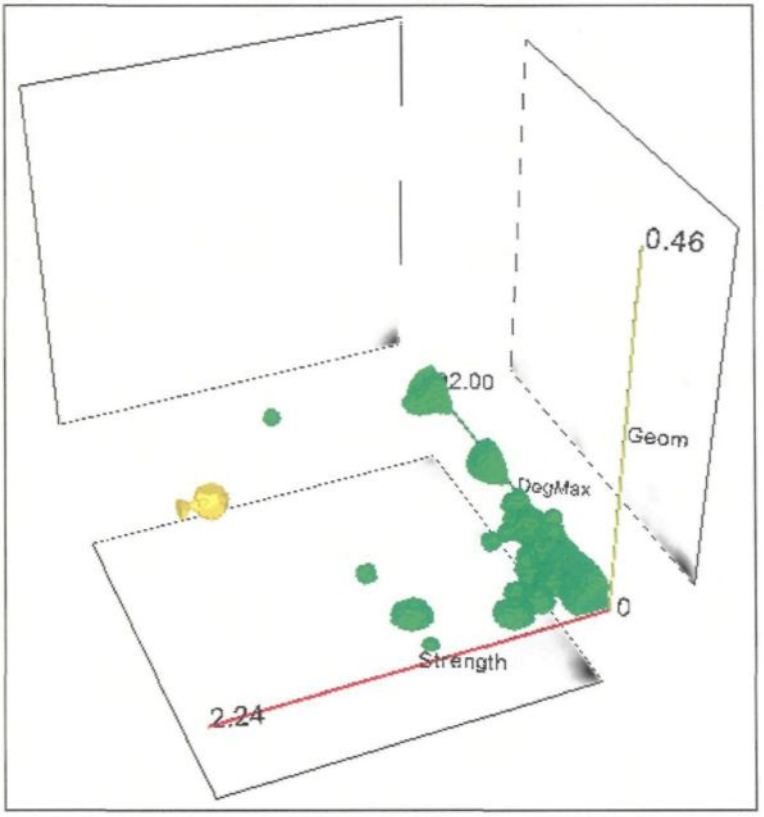


Figure 3.28: Sélection de régions d'intérêt. L'iso-surface générée à partir du volume est rendue à l'aide d'un algorithme de rendu de surface. La région connexe sélectionnée est mise en évidence en jaune.

sommets faisant partie de la région d'intérêt est défini par :

$$V' = \{v_i \in V \mid \big(\phi_1(v_i), \phi_2(v_i), \phi_3(v_i)\big) \in R\} \ , \tag{3.24}$$

où ϕ_1, ϕ_2 et ϕ_3 sont les valuations après normalisation.

De plus, on remarque que la région R est nécessairement constituée de une ou plusieurs régions connexes. Afin de raffiner sa recherche, l'utilisateur peut sélectionner une sous-région connexe $R_i \subseteq R$. L'ensemble des sommets extraits du volume est alors restreint à ceux inclus dans la région R_i. L'utilisateur effectue la sélection en cliquant directement sur une des régions à l'intérieur de la représentation graphique. Un rayon est

lancé perpendiculairement au plan image à l'endroit du clic et la première sous-région connexe percutée par celui-ci est sélectionnée. Cette opération est appelée *picking*. La sous-région sélectionnée est alors mise en évidence, et l'ensemble V' est calculé à nouveau cette fois-ci en considérant seulement la sous-région R_i. Les sommets faisant partie de la région d'intérêt sont présentés à l'utilisateur sous forme de liste et peuvent être exportés vers un fichier texte.

CHAPITRE 4

APPLICATIONS ET RÉSULTATS

4.1 Introduction

Dans le dernier chapitre, nous avons présenté un système permettant de générer un volume de données à partir du calcul d'un triplet de valuations sur un graphe. Ce processus peut être interprété comme l'association de chacun des sommets du graphe à un point dans l'espace $\mathbb{R}^3$, sa position étant déterminée par le calcul du triplet de valuations. Ces points sont généralement distribués de façon dispersée à l'intérieur du volume. De plus, étant donné que nous utilisons des valuations structurelles, leur distribution révèle de l'information sur la topologie du graphe. Par exemple, considérons un volume généré à partir de valuations calculées sur des arêtes, plus précisément l'indice de *Jaccard*, *Strength* et le degré minimal entre les deux sommets définissant une arête. On se rappelle que l'indice de *Jaccard* permet d'évaluer la similarité entre le voisinage de deux sommets définissant une arête, tandis que *Strength* permet d'évaluer si le lien défini par cette arête est situé à l'intérieur d'un sous-graphe dense ou s'il sert plutôt de lien entre deux sous-graphes denses distincts (voir section 3.3.1). Le degré minimal entre deux sommets permet quant à lui d'évaluer le niveau de connectivité de ces sommets aux autres sommets du graphe. On appelle *agglomérats naturels* [58] un groupe de sommets ayant une quantité importante de relations entre eux par rapport au reste des sommets du graphe. Une arête dont les deux sommets la définissant possèdent

des voisinages similaires et dont le degré minimal est élevé fait généralement partie d'un agglomérat structurel. L'outil de visualisation présenté dans la section précédente peut être utilisé pour extraire des regroupements de sommets formant des agglomérats directement à l'intérieur du volume.

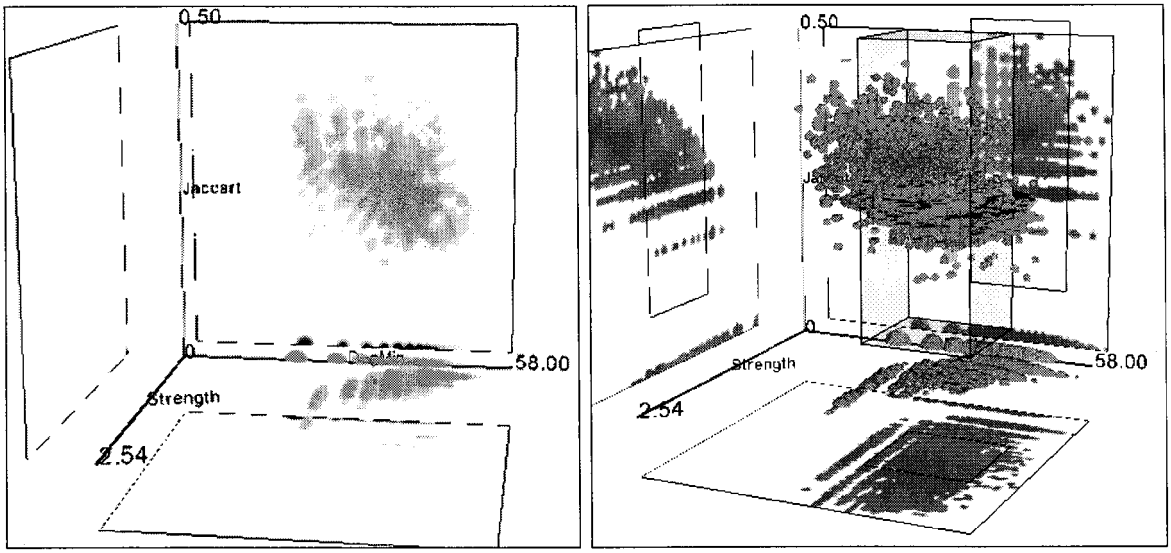
Les sections suivantes portent sur l'application des méthodes présentées dans les chapitres précédents sur des exemples de graphes concrets. Nous débuterons par l'étude d'un graphe d'association de mots. Ensuite nous tenterons d'extraire des composantes logicielles du graphe d'association de classes de la librairie MFC. Finalement, nous terminerons par l'extraction de groupes d'acteurs à l'intérieur d'une base de données cinématographique.

4.2 Graphe d'associations de mots (CIGOGNE)

Les liaisons sémantiques qu'effectue l'être humain entre les différents mots des languages offrent une problématique intéressante. En effet, ces liaisons ne sont pas identiques pour chaque individus et il est intéressant d'étudier la part objective du sens des mots. CIGOGNE [43] est un projet de recherche ayant pour objectif l'évaluation de différentes techniques de visualisation et d'exploration de graphes appliquées à des réseaux sémantiques. Ce programme a été mené conjointement par le LSC[1], le LIRMM[2] ainsi que LIPSI[3]. Dans le cadre de ce projet, les chercheurs ont développé une base de données représentant des associations entre des mots de la langue française au cours d'une étude menée auprès d'environ 350 volontaires. Ces derniers devaient remplir un formulaire en ligne constitué d'une liste de noms commums pour lesquels ils devaient associer un autre mot de façon libre.

Les données recueillies par cette étude peuvent être représentées sous la forme

[1]Laboratoire de Sciences Cognitives de l'université de Bordeaux
[2]Laboratoire d'Informatique, de Robotique et de Microélectrique de Montpellier
[3]Laboratoire d'Ingénierie des Processus et des Services Industriels de Bayonne



(a) (b)

Figure 4.1: Volume généré à partir du graphe CIGOGNE en (a) et iso-surface extraite de ce volume en (b).

d'un graphe pour lequel chaque sommet représente un mot et chaque arête relie deux sommets si ceux-ci représentent des mots ayant été associés par un individu au cours de l'étude. Le graphe est constitué d'environ 3000 sommets et 10000 arêtes. L'utilisation de notre application pour l'étude de ce graphe permet l'extraction de groupes de mots associés par plusieurs individus. En effet, on peut supposer que des sommets dont le voisinage est commun en grande partie sont inclus dans un même agglomérat et correspondent à des mots dont l'association sémantique est forte. Nous utiliserons les valuations *Strength*, *Jaccard* ainsi que le degré minimal entre deux sommets définissant une arête afin de générer ce volume. Rappelons-nous que ces valuations sont calculées sur des arêtes et qu'une valeur faible correspond généralement à une arête dont les sommets font partie de deux agglomérat distincts, tandis qu'une valeur forte correspond à une arête dont les sommets ont des voisinages similaires et regroupés à l'intérieur d'un même agglomérat (voir section 3.3.1). L'extraction d'un groupe de mots revient à

sélectionner une région d'intérêt à l'intérieur du volume. Les groupes de mots à forte association correspondent à des régions d'intérêts constituées de voxels correspondant à des sommets du graphe ayant une forte valeur calculée pour chacune des trois valuations. Une fois le volume généré (Fig 4.1a), le contrôleur sonde a été utilisé afin d'en extraire une iso-surface. Évidemment, plus la valeur correspondant à la surface à générer est grande, plus la taille des groupes de mots associés aux régions d'intérêt correspondant à cette surface est grande. Aussi, il est important de noter que plusieurs groupes de mots peuvent correspondre au même triplet de coordonnées à l'intérieur du volume et se retrouvent donc à l'intérieur de la même région d'intérêt.

Pour cet exemple, nous avons utilisé une valeur seuil faible afin de générer un maximum de régions d'intérêt (Fig 4.1b). Nous avons ensuite procédé empiriquement à l'extraction de sept régions intéressantes distribuées à l'intérieur du volume. Parmi celles-ci, trois correspondent à des valeurs calculées fortes, trois à des valeurs modérées et une à des valeurs faibles. Ces régions correspondent à des sous-ensembles de sommets numérotés de R_1 à R_7. Le tableau (4.1) et la figure (4.2) exposent quelques-uns de ces résultats. Considérons les régions R_1 et R_2 pour lesquelles les valeurs calculées pour les valuations *Strength* et *Jaccart* sont relativement fortes par rapport au reste de l'ensemble de données. Celles-ci contiennent les couples de mots *(baignore, douche)* ainsi que *(datte, figue)*. Le lien sémantique entre les concepts représentés par les mots formant ces couplets est particulièrement évident. Considérons maintenant les sous-graphes formés des voisinages immédiats des sommets *datte* et *figue*. Ceux-ci sont représentés dans les figures 4.3a et 4.3b. On remarque que ces deux sous-graphes sont pratiquement identiques car un seul sommet (*pruneau*) les distingue. Étant donné le fait que ces deux mots sont associés au même groupe de mots et que ce groupe de mots est relativement restreint, on peut affirmer que ceux-ci représentent des concepts plutôt

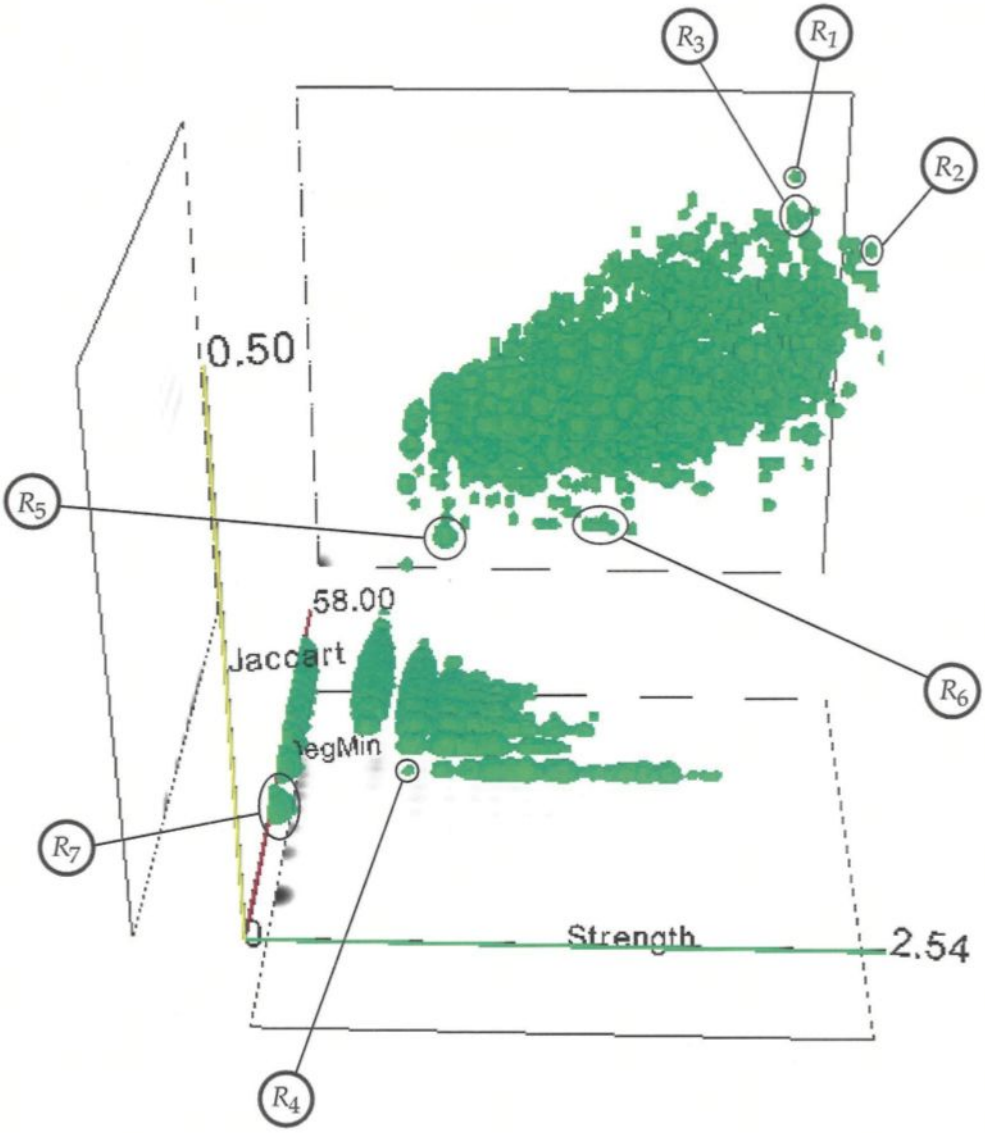


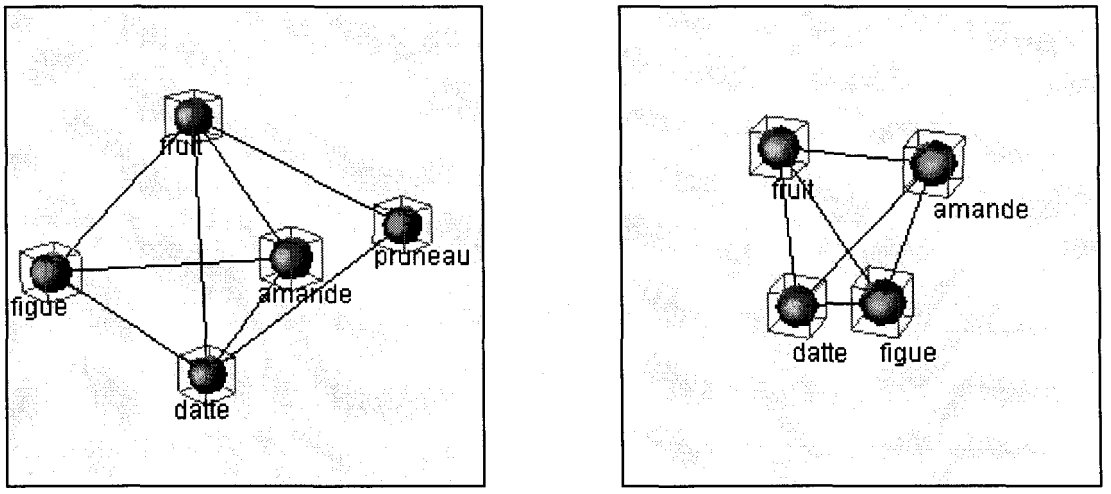
Figure 4.2: Extraction de régions d'intérêt dans le graphe d'associations de mots du projet CIGOGNE.

TAB. 4.1: Associations de mots

Région	Groupe de mots
R_1	baignoire douche
R_2	datte figue
R_3	félin léopard lion guitare harpe
R_4	animal végétal
R_5	animal cerf kangourou loutre
R_6	animal lionne manchot
R_7	abbé abeille ablation abomination abondance abricot ... zoulou zuchette

précis aux yeux des participants et que, par conséquent, l'association entre ces deux mots est très forte.

De la même manière, la région R_3 est constituée de deux groupes de mots à association forte, soit *(guitare, harpe)* et *(félin, léopard, lion)*. Les sous-graphes représentant le voisinage immédiat des sommets du deuxième groupe sont présentés dans les figures 4.4a, 4.4b et 4.4c. On définit l'intersection de deux graphes comme l'intersection de l'ensemble de leurs sommets et de leurs arêtes. La figure 4.4d représente l'intersection des sous-graphes de voisinage des mots *lion*, *léopard* et *félin*. On remarque que ce sous-graphe contient le mot *félin* ainsi que des noms de félins (i.e. *panthère*, *léopard*, *guépard*, *lion* et *tigre*). Encore une fois, on peut affirmer que ces mots représentent des concepts bien définis aux yeux des participants. Les sommets constituant ces sous-ensembles forment assurément un agglomérat structurel.



(a) (b)

Figure 4.3: Voisinage des sommets *datte* et *figue* en (a) et (b). On remarque qu'un seul sommet différencie ces deux sous-graphes.

Considérons maintenant les régions R_5 et R_6 pour lesquelles les valuations en *Jaccard* et *Strength* sont moins élevées. On remarque que ces deux groupes sont constitués du mot animal ainsi que de noms d'animaux. La région R_4 quant à elle contient les mots *animal* et *végétal*. On constate que les valuations en *Jaccard* et *Strength* sont faibles tandis que le degré minimal est élevé pour les sommets de la région R_4. Les figures 4.5a et 4.5b illustrent les sous-graphes représentant le voisinage immédiat des sommets *animal* et *végétal*. On constate que le sous-graphe du voisinage du sommet *animal* contient 53 sommets, pour la plupart représentant des noms d'animaux, et qu'étonnament, le sous-graphe du voisinage du sommet *végétal* ne contient que 3 sommets. Étant donné cette distribution, on peut supposer que ces deux mots ont été associés car ils correspondent tous deux à des classes de vivants. De plus, il semble être plus difficile pour un individu d'associer le mot *végétal* à une espèce que pour le mot *animal*.

La région R_7 contient les arêtes pour lesquelles les valuations *Jaccard* et *Strength* ont une valeur nulle ou faible. Par conséquent elle comporte un grand nombre de som-

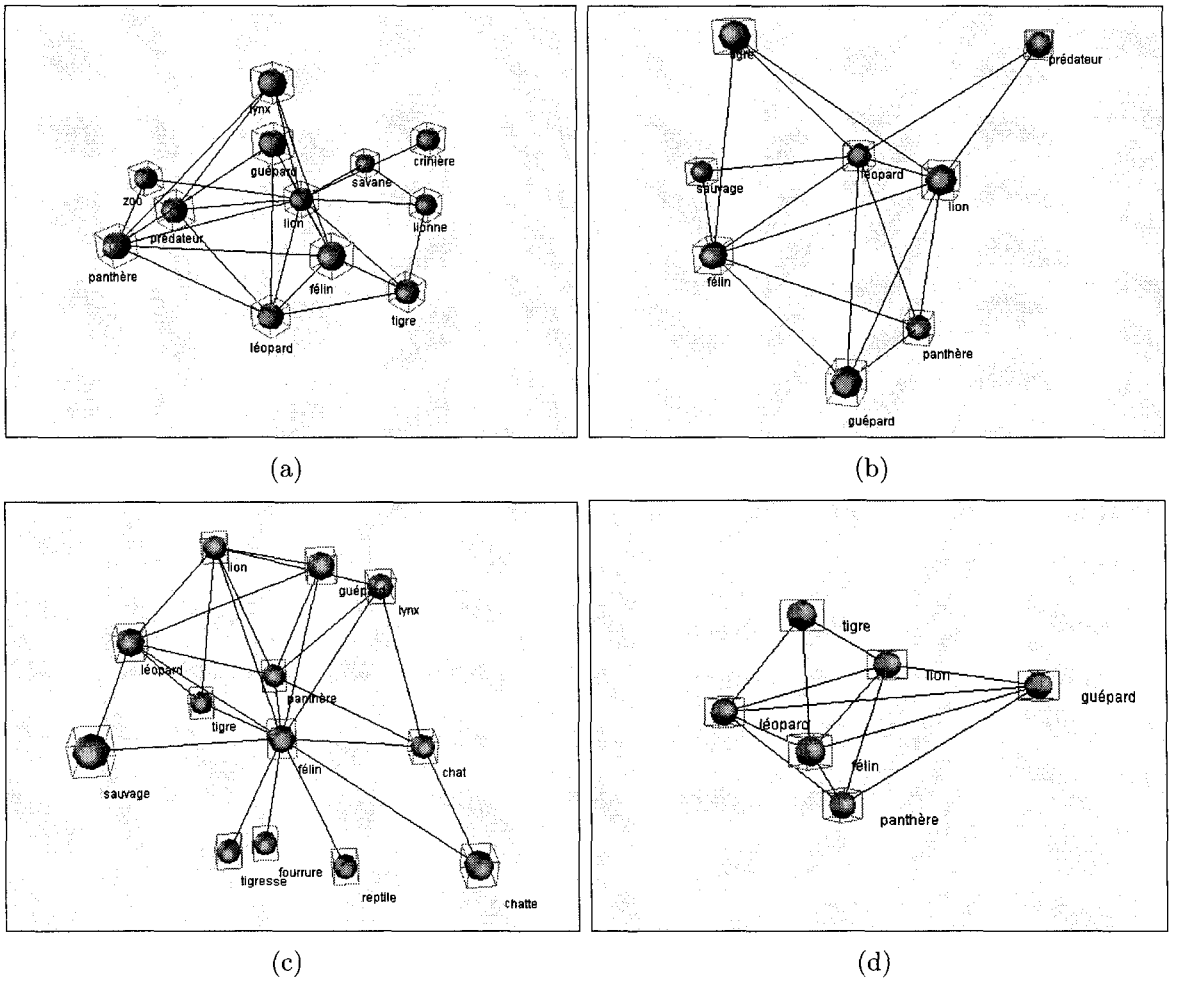
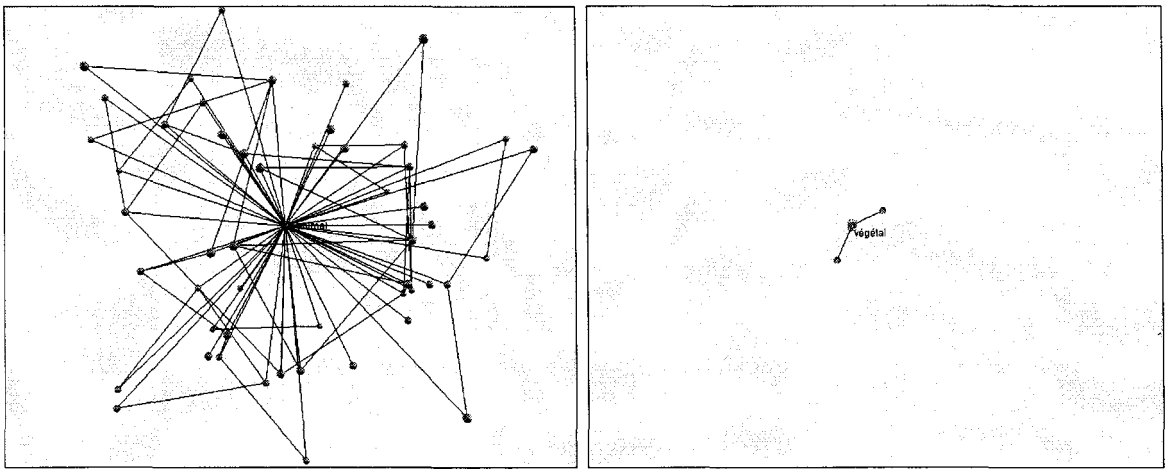


Figure 4.4: Voisinage des sommets *lion*, *léopard* et *félin* en (a), (b) et (c). Intersection des voisinages des sommets *lion*, *léopard* et *félin* en (d).



(a) (b)

Figure 4.5: Voisinage des sommets *animal* en (a) et *végétal* en (b).

mets dont l'association est faible. Ces quelques tests nous démontrent clairement qu'en sélectionnant des régions correspondant à des agglomérats structurels, on obtient des groupes de mots dont le lien sémantique est évident.

4.3 Extraction de composantes logicielles (librairie MFC)

La rétro-ingénierie logicielle (*reverse engineering*) est un domaine du génie logiciel qui porte sur l'analyse *a posteriori* de logiciels afin d'en comprendre la structure et le fonctionnement. Chikofsky et al. définissent la rétro-ingénierie logicielle [16] comme étant le processus permettant d'analyser un système informatique en identifiant les composantes logicielles le constituant ainsi que leurs relations et en créant des représentations abstraites de celui-ci. Le langage de modélisation unifié (UML) est un langage graphique permettant de modéliser de façon abstraite des procédés et systèmes informatiques en programmation orientée objet. Pour plus d'information sur UML, se référer à [27]. Plusieurs outils de rétro-ingénierie, dont Pragsoft UML Studio[4], permettent de

[4]`http://www.pragsoft.com`

créer des modèles UML à partir de code source de programmes ou librairies informatiques. Les relations entre les classes représentées par ces modèles UML peuvent être exprimées à l'aide de graphes. Par exemple, considérons un graphe pour lequel chaque sommet représente une classe et chaque arête représente une relation d'association ou d'héritage entre deux classes (voir [27]). On appelle ce graphe, *graphe d'association de classes.* En programmation orientée objet, une composante logicielle est un regroupement logique de classes offrant une fonctionnalité logicielle qui ne doit pas être dépendante de contexte, mais plutôt réutilisable. Les classes formant une même composante logicielle ont généralement des relations d'association semblables par rapport au reste du graphe et font partie d'un même agglomérat structurel. Nous présentons un exemple d'application de notre outil de visualistion où il est question d'extraire des agglomérats structurels correspondant à des composantes logicielles d'une librairie, à partir d'un graphe d'association de classes.

La librairie MFC (*Microsoft Foundation Classes*) est un ensemble de classes C++ encapsulant l'interface de programmation du système d'exploitation Windows de Microsoft. Le graphe d'association de classes de la librairie MFC contient un total de 1524 sommets et 1314 arêtes. Encore une fois, nous avons généré le volume en utilisant les valuations *Jaccard*, *Strength*, et le degré minimal entre deux sommets définissant une arête. On remarque que la valuation *Jaccard* est faible pour une grande partie des arêtes. Cela signifie qu'une majorité de classes sont associées entres elles sans toutefois avoir de voisinage commun. On remarque de plus que le degré minimal pour l'ensemble des relations est relativement élevé. Nous avons extrait des regroupements de classes ayant des associations similaires à l'aide du contrôleur sonde à l'intérieur du volume, de la même manière que celle présentée dans l'exemple précédent. Bien sûr, il existe un grand nombre de regroupements possibles mais nous nous sommes limités à l'extraction d'un

échantillon de quatre regroupements représentant des classes dont l'association est forte. Ces régions d'intérêt ont été extraites de façon empirique. La figure 4.6 et le tableau 4.2 présentent ces résultats. Considérons d'abord la région R_1 qui regroupe cinq classes (i.e.

TAB. 4.2: Associations de classes

Région	Classes
R_1	CDaoDatabase CDaoQueryDef CDaoRecordset CDaoTableDef CDaoWorkspace
R_2	CDatabase CRecordset
R_3	CDocument CDoctTemplate
R_4	CPreviewView CPreviewDC

CDaoDatabase, *CDaoQueryDef*, *CDaoRecordset*, *CDaoTableDef*, *CDaoWorkspace*). En étudiant la documentation de MFC [26], on constate que ces cinq classes forment une composante permettant de gérer le support des bases de données pour l'interface DAO (*Data Access Object*). Similairement, la région R_2 contient deux classes (*CDatabase, CRecordset*) permettant de gérer le support pour bases de données de l'interface ODBC (*Object DataBase Connectivity*). Cette interface permet la connectivité entre les clients de bases de données fonctionnant sous Windows et les systèmes de gestion de base de données tierce partie. Les régions R_1 et R_2 sont particulièrement intéressantes car elles contiennent la totalité des classes constituant les composantes logicielles mentionnées ci-haut tel que spécifié par la documentation de MFC.

La région R_3, quant à elle, contient deux classes (i.e. *CDocTemplate* et *CDocument*). L'intersection du voisinage de ces deux sommets est illustrée dans la figure 4.7. Après vérification dans la documentation de la librairie MFC, on constate que ces classes font partie d'une composante logicielle permettant de créer des documents MFC à l'aide de modèles (*template*). Finalement, la région R_3 contient elle aussi deux classes

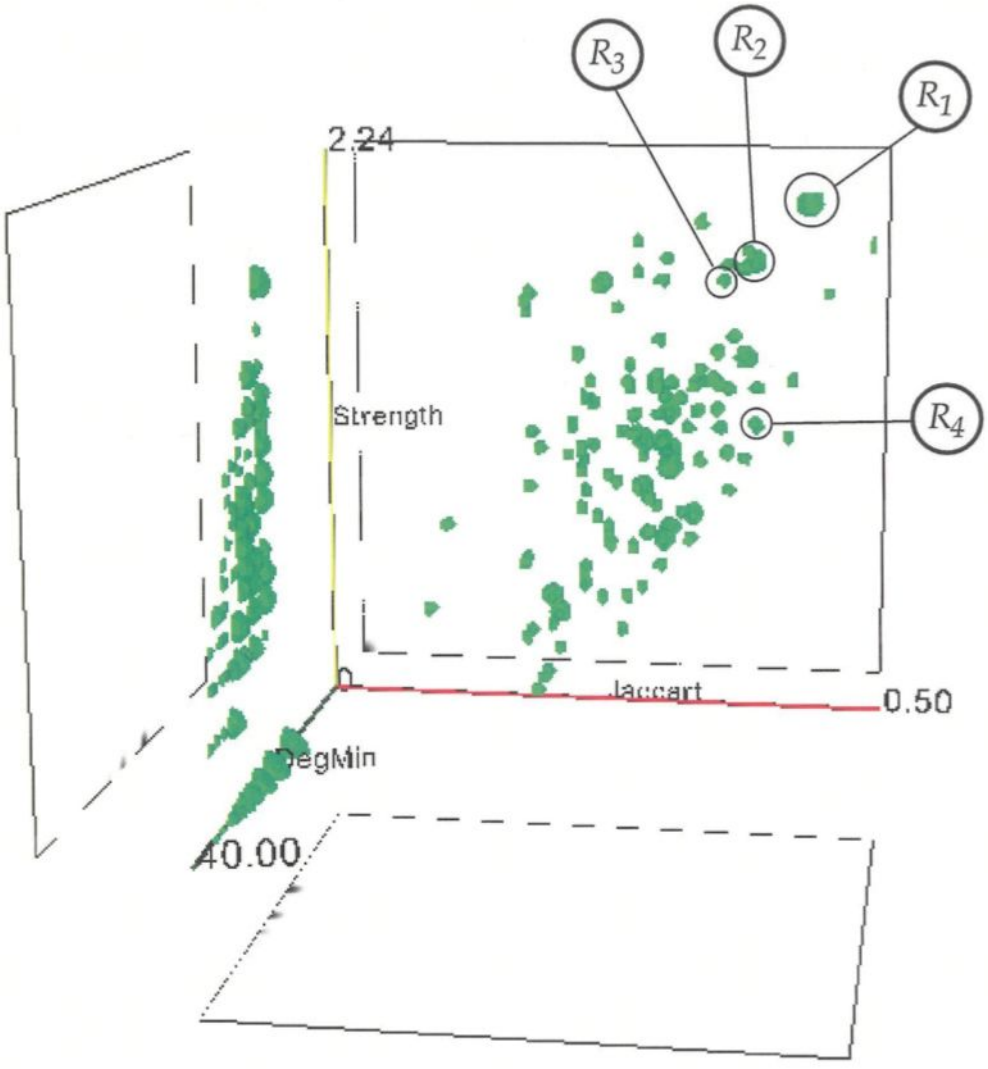


Figure 4.6: Extraction de régions d'intérêt dans le graphe d'association de classes de la librairie MFC.

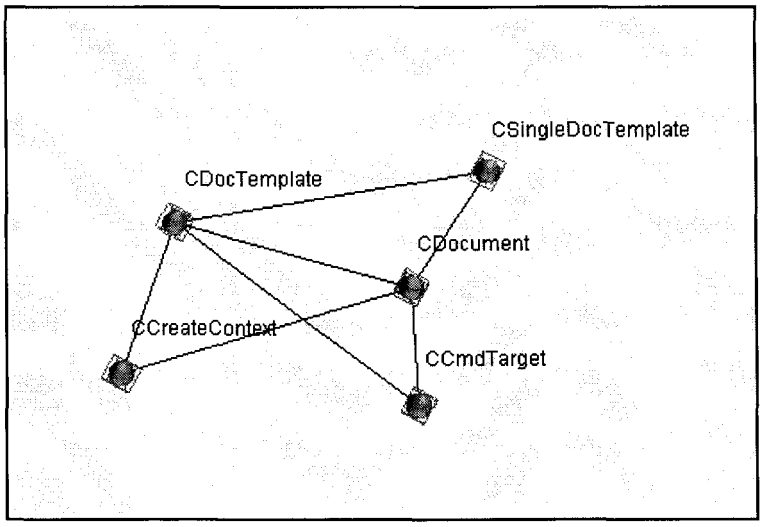


Figure 4.7: Sous-graphe représentant l'intersection des voisinages des sommets inclus dans la région R_3. Les classes représentées par ces sommets sont utilisées lors de la création d'un nouveau document MFC à l'aide d'un modèle.

(i.e. *CPreview* et *CPreviewDC*). La figure 4.8 illustre l'intersection des voisinages des deux sommets représentant ces classes. Les classes représentées par les sommets formant ce sous-graphe sont utilisées lors de la gestion de vues de prévisualisation d'impression.

Suivant ces quelques exemples, nous avons constaté que les régions d'intérêt, dont les valeurs calculées en *Jaccard* et *Strength* sont non nulles, contiennent des sommets représentant des classes faisant généralement partie d'une même composante logicielle. Toutefois, le graphe que nous avons utilisé pour effectuer ce test présentait peu d'agglomérats structurels. En l'étudiant avec le logiciel de manipulation directe de graphes "FW3DGraphes", on a pu remarquer qu'il contenait un seul sous-graphe connexe de grande taille et une grande quantité de sommets isolés et de sous-graphes connexes de petite taille. Le sous-graphe connexe de grande taille était constitué des sommets représentant les classes héritant de la classe *CObject*. Aussi, nous avons remarqué qu'il est relativement difficile d'extraire des composantes logicielles dans une librairie comme MFC, étant donné l'absence de sous-graphe dense. Notamment, l'association entre les

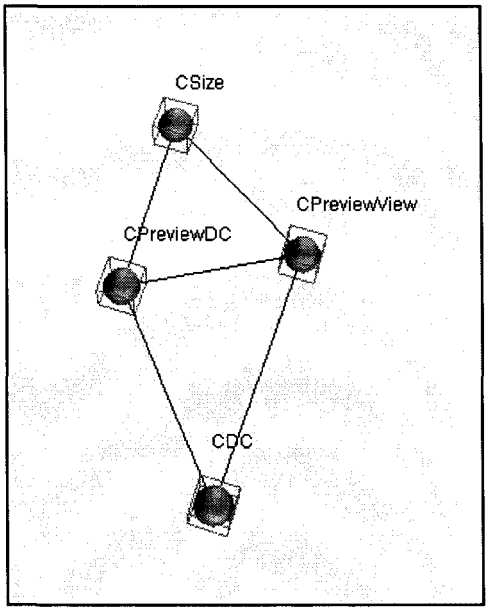


Figure 4.8: Sous-graphe représentant l'intersection des voisinages des sommets inclus dans la région R_4. Les classes représentées par ces sommets sont utilisées lors de la génération d'une vue de prévisualisation pour l'impression d'un document.

classes représentées par les sommets des régions R_1 et R_2 est plus évidente que celle des sommets des régions R_3 et R_4. La technique que nous présentons serait plus efficace pour extraire des composantes logicielles d'une librairie constituée de classes ayant un grand nombre de relations.

4.4 Base de données cinématographique (IMDB)

L'*Internet Movie DataBase*[5] (IMDB) est une base de données accessible en ligne contenant de l'information sur l'industrie du cinéma international, principalement sur les films y étant produits ainsi que les acteurs, compagnies et autres personnes qui y sont rattachés. De cette base de données, on peut extraire un graphe dont les sommets représentent les acteurs et dont les arêtes relient deux acteurs si ceux-ci sont apparus dans le même film ou la même émission de télévision. Étant donné le nombre important

[5] `http://www.imdb.com`

de sommets et relations constituant ce graphe, nous limiterons cet exemple au sous-graphe défini par le voisinage des sommets représentant Cameron Diaz et Tom Cruise. Cette fois-ci, on peut supposer que les agglomérats compris à l'intérieur de ce graphe correspondent à des films car plusieurs acteurs ayant participé à une même production sont nécessairement liés les uns aux autres. Une fois de plus, nous utiliserons les valuations *Jaccard*, *Strength*. Le processus utilisé pour extraire des régions d'intérêt dont la valeur calculée pour les trois valuations est forte est le même que celui utilisé pour l'exemple de la section (4.2). Cette fois-ci nous avons extrait quatre régions d'intérêt de façon empirique. La figure 4.9 présente ces régions et le tableau 4.3 liste les sommets rattachés à celles-ci.

TAB. 4.3: Associations d'acteurs

Région	**Acteurs / Individus faisant partie d'une production**
R_1	Aretos, Guillaume (II) Lithgow, John Murphy, Eddie Myers, Mike Smith, Simon J.
R_2	Connolly, Billy Duffin, Shay Goldwyn, Tony Igawa, Togo Keitel, Harvey Sheffer, Craig Spall, Timothy Zane, Billy
R_3	Cleghorne, Ellen Farley, Chris Meadows, Tim Miller, Dennis (I) Mohr, Jay Rock, Chris Sandler, Adam Schneider, Rob Spade, David
R_4	Affleck, Ben Carrey, Jim Holmes, Katie Kudrow, Lisa Spink, Daniel

Considérons tout d'abord la région R_1. En étudiant la base de données IMDB, on remarque que l'ensemble des acteurs représentés par les sommets compris à l'intérieur

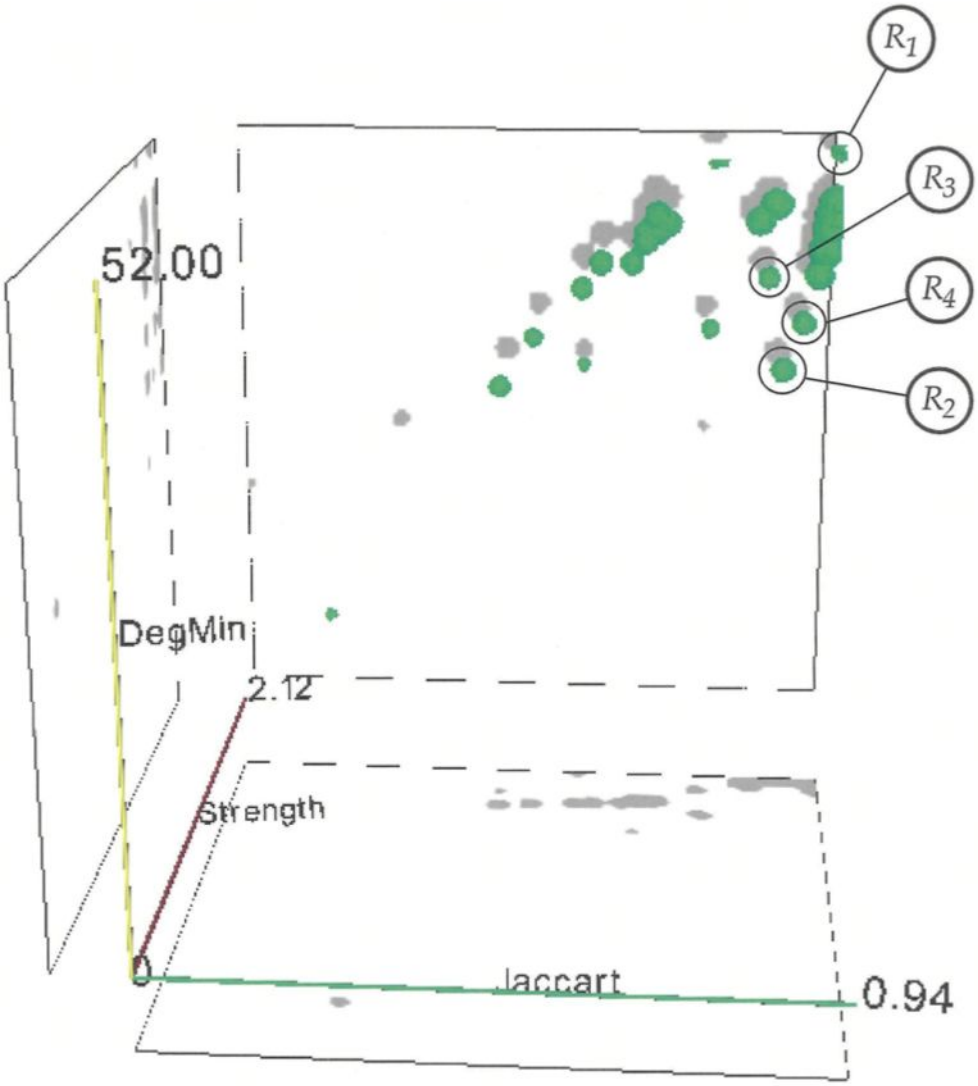


Figure 4.9: Extraction régions dans le graphe créé à partir de la base de données cinématographique IMDB.

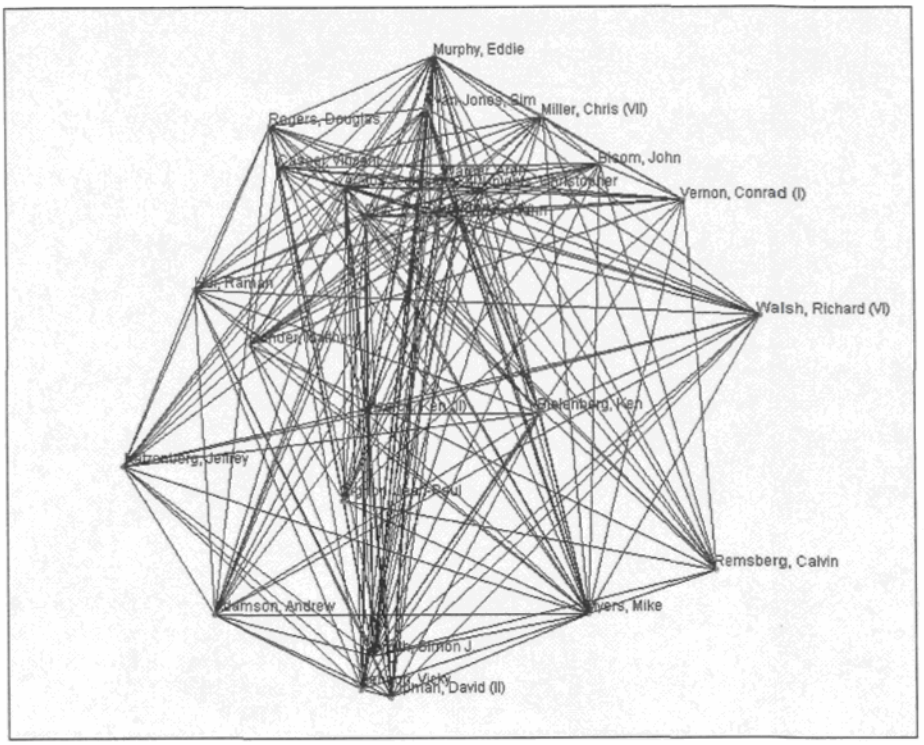


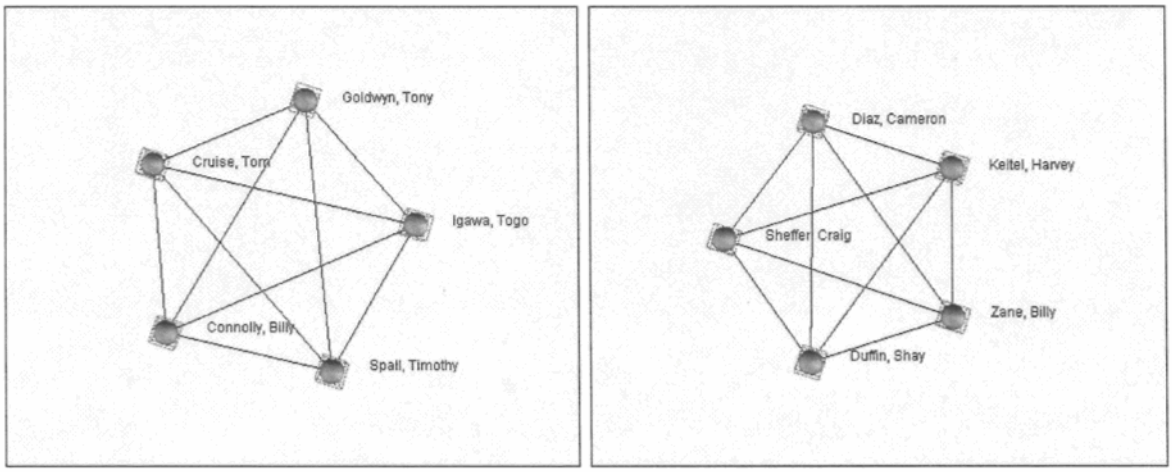
Figure 4.10: Sous-graphe représentant l'intersection des voisinages des sommets inclus dans la région R_1. Les individus représentés par ces sommets ont tous participé à la réalisation du long-métrage *Shrek*.

de cette région ont prêté leur voix à un personnage du film d'animation *Shrek*. On remarque aussi que l'actrice Cameron Diaz a participé à cette réalisation. Soit G_i le sous-graphe représentant le voisinage du sommet $i \in R_1$. Le graphe G_{R_1} représentant l'intersection des sous-graphes de vosinage des sommets inclus dans la région R_1 est présenté dans la figure 4.10 et est donné par :

$$G_{R_1} \equiv \bigcap_{i \in R_1} G_i \tag{4.1}$$

Le sous-graphe G_{R_1} correspond à un agglomérat dont l'ensemble des individus représentés par les sommets inclus dans celui-ci ont participé à la réalisation du long-métrage *Shrek*.

Similairement, la région R_2 est composée d'un groupe d'acteurs qu'on peut diviser en deux sous-ensembles (i.e. R_2^1 ={Billy Connolly, Tony Goldwyn, Togo Igawa, Timothy Spall} et R_2^2 ={ShayDuffin, Harvey Keitel, Craig Sheffer, Billy Zane}). En effet, après une recherche sur IMDB, on remarque que le sous-ensemble R_2^1 contient uniquement des acteurs ayant joué dans le film *Last Samurai* et que le sous-ensemble R_2^2 contient uniquement des acteurs ayant joué dans le film *Head above the water*. On



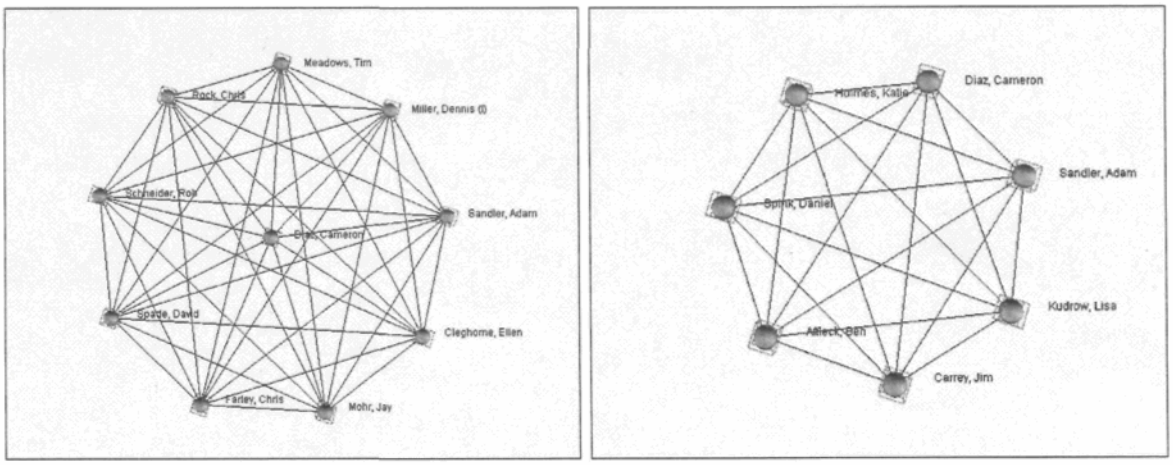
(a) (b)

Figure 4.11: Le sous-graphe en (a) correspond au film *Last samurai* et le sous-graphe en (b) correspond au film *Head above the water*. Ces deux sous-graphes forment des cliques.

remarque aussi que pour les deux régions d'intérêt, les graphes correspondant au voisinage de chacun des cinq sommets sont identiques et correspondent aux graphes formés des sommets de ces régions. Ceux-ci sont présentés dans la figure 4.11. De plus, on constate que ces deux graphes sont des cliques de topologie identique. C'est ce qui explique pourquoi ils sont associés au même triplet de valeurs de valuations.

En plus d'une grande quantité d'information sur le monde du cinéma, la base de données IMDB contient aussi des références à des évènements ou séries télévisuelles. La région R_3 regroupe des artistes ayant tous déjà participé au moins une fois à l'émis-

sion de varitétés *Saturday Night Live*. La région R_4 contient quant à elle un ensemble d'acteurs réunis par le fait qu'ils ont tous participé au gala des *MTV Movie Awards* en 1999. Une fois de plus, l'intersection des sous-graphes représentant le voisinage de l'ensemble des sommets inclus dans ces deux régions correspond à deux cliques : G_{R_3} et G_{R_4}. Celles-ci sont illustrées dans la figure 4.12.



(a) (b)

Figure 4.12: En (a), le sous-graphe G_{R_3} correspondant à l'intersection des voisinages des sommets inclus dans la région R_3. Ces acteurs ont participé à l'émission de variétés *Saturday Night Live*. En (b), le sous-graphe G_{R_4} correspondant à l'intersection des voisinages immédiats des sommets inclus dans la région R_4. Ces acteurs ont participé au gala des *MTV Movie Awards* de 1999.

4.5 Conclusion

Nous avons illustré dans ce chapitre comment il est possible d'extraire aisément des agglomérats structurels d'un graphe de taille importante après avoir associé ces sommets aux voxels d'un volume tridimensionnel. Dans une structure de données relationnelle, les agglomérats correspondent à un groupe de sommets dont la relation est forte et il est souvent possible de trouver un lien concrèt entre ces éléments. Notons que nous avons obtenu un plus grand succès pour les graphes comprenant un grand nombre

de relations (i.e. graphe d'association de mots et IMDB) que pour les graphes contenant peu de sous-graphes denses (i.e. graphe d'association de classes de MFC). Notons aussi que les graphes étudiés dans ce chapitre ont été choisis parce qu'ils sont facilement interprétables et que les résultats sont vérifiables sans difficulté. Ces techniques pourraient toutefois être utilsées dans un contexte plus abstrait.

Conclusion

Les deux objectifs principaux de cet ouvrage ont été l'étude générale du domaine de recherche qu'est la visualisation et le développement d'un système permettant d'explorer des graphes de grande taille. Nous avons tenté de démontrer qu'il est possible d'extraire efficacement de l'information d'un ensemble de données relationnelles grâce à des techniques de visualisation scientifique comme le rendu de volume et l'utilisation de contrôleurs graphiques.

Dans un premier temps, nous avons énoncé la distinction entre la visualisation scientifique et la visualisation d'information. Nous avons par la suite exploré ces deux sous-domaines de recherche de façon plus détaillée en décrivant quelques concepts généraux et en proposant des algorithmes pour chacun d'eux. Suite à l'introduction de la notion de taille d'un ensemble de données (correspondant au nombre de variables ou dimensions le constituant), nous avons regroupé les algorithmes de visualisation d'information en six classes distinctes. Ces derniers nous ont semblé appropriés pour visualiser des ensembles de données constitués d'un grand nombre de variables.

Similairement, nous avons regroupé les algorithmes de visualisation scientifique permettant d'effectuer des rendus d'ensembles de données tridimensionnelles en deux classes distinctes (ie. le rendu de surface et le rendu de volume). Les algorithmes de rendu de surface permettent de représenter des surfaces extraites d'un volume sous

forme de maillages polygonaux. Ce type d'algorithme nous a plus tard été utile pour représenter des régions d'intérêt extraites de volumes permettant d'explorer un graphe de grande taille. Nous avons également regroupé les algorithmes de rendu de volume en deux sous-groupes distincts selon le type de projections qu'ils utilisent (ie. le backward et le forward mapping). Pour chacune de ces classes, nous avons présenté deux algorithmes. Étant donné que le type de volume que nous désirons représenter est constitué d'un ensemble de nuages diffus, nous avons opté pour un algorithme de lancer de rayons. Nous avons ensuite conclu cette étude en effectuant une présentation sommaire de divers concepts et méthodes d'interaction appliquées à la visualisation, notamment le concept de contrôleur.

Nous nous sommes ensuite intéressés à l'exploration de graphes, plus particulièrement aux problèmes engendrés par la visualisation de graphes de grande taille. L'introduction du concept de valuation nous a permis de faire l'étude de quelques méthodes d'exploration de graphe basées sur ce concept, notamment l'association de caractéristiques graphiques à des valuations ainsi que le filtrage. On se rappelle qu'il existe deux types de valuations, soit les valuations sémantiques et structurelles. Nous avons particulièrement porté notre attention sur le concept de valuations structurelles et d'agglomérats structurels dans un graphe.

Suite à l'étude de la littérature reliée à ce sujet, nous avons remarqué que l'utilisation simultanée de plusieurs valuations structurelles pour une même opération de filtrage pouvait nous permettre, entre autres, d'extraire des agglomérats structurels d'un graphe. Ce processus était présenté dans le contexte de l'utilisation d'un couplet de valuations. Notre idée fondamentale était donc de généraliser ce concept à trois valuations. Plutôt que d'explorer le graphe à l'aide d'une image, on utilise un histogramme tridimensionnel auquel on a effectué la convolution d'un noyau Gaussien. Ce

volume est généré par le calcul d'un triplet de valuations sur chacun de ses éléments. La convolution du noyau Gaussien nous permet d'obtenir un ensemble de nuages diffus de niveau d'abstraction plus élevé. Une fois ce processus terminé, l'utilisateur pouvait manipuler le rendu et explorer l'intérieur de celui-ci à l'aide d'un contrôleur introduit dans le troisième chapitre.

Les résultats présentés dans le chapitre 4 ont démontré que notre deuxième objectif a été atteint. Nous avons extrait avec succès des agglomérats stucturels dans les trois exemples présentés. Étant donné les domaines d'application auxquels les graphes étaient associés, on a pu aisément associer une interprétation sémantique à ces agglomérats. Par exemple, les agglomérats structurels extraits de la base de données cinématographique IMDB correspondaient tous à des groupes d'acteurs faisant partie d'un même long métrage ou autre évènement télévisuel. Ces résultats font suite à ceux déjà présents dans la littérature.

Au niveau de l'implémentation, il serait sans doute possible d'obtenir de meilleures performances en utilisant une autre librairie que VTK. Celle-ci a la particularité d'être très vorace en mémoire et peu efficace de façon générale. Ce manque de performance est principalement dû à la polyvalence de son architecture. Nous avons d'ailleurs opté pour cette librairie dans le but de minimiser le temps de développement de notre application. L'optimisation des performances de notre application ne faisait pas partie de nos objectifs initiaux, mais il nous apparaît évident qu'il serait possible de les améliorer.

Il pourrait être intéressant d'étendre la généralisation présentée dans cet ouvrage à un nombre plus élevé de dimensions. En effet, nous croyons que l'utilisation combinée de valuations sémantiques et structurelles permettrait d'effectuer des recherches plus rafinées à l'intérieur du graphe étudié. Cette approche est énoncée à quelques reprises dans la littérature. De plus, il nous serait possible d'utiliser des méthodes présentées

dans le premier chapitre, comme les hypercubes, pour visualiser des histogrammes à plus de trois dimensions. L'utilisateur pourrait alors y naviguer par le biais d'un volume représentant un sous-espace borné de l'histogramme multidimensionnel en réutilisant les mécanismes d'interaction présentés dans ce mémoire.

Bien que le système de visualisation que nous avons introduit permette de visualiser un volume calculé à partir de données relationnelles, les concepts y étant rattachés pourraient être utilisés pour visualiser d'autres types de données. Notamment, il serait intéressant d'utiliser le contrôleur développé dans le troisième chapitre pour explorer des volumes provenant du domaine médical ou d'autres modalités d'imagerie. Par exemple, on peut supposer que le mode d'affichage permettant de restreindre le rendu de volume au contrôleur serait particulièrement efficace pour effectuer des coupes d'orientation arbitraire à l'intérieur du volume.

L'outil présenté dans ce mémoire répond aux objectifs que nous nous étions fixés préalablement. Les mécanismes d'interaction y étant présentés offrent un certain potentiel et il serait intéressant de les utiliser dans d'autres contextes.

ANNEXE A

LIBRAIRIE VTK

VTK[60] est une bibliothèque développée par la compagnie Kitware, offrant une variété d'outils de visualisation, de traitement d'images et d'outils d'interaction. Elle est disponible gratuitement sous forme de projet *source libre*. Cette bibliothèque est conçue de façon à faciliter son développement et est basée sur une architecture fortement modulaire. Les prochaines sections visent à présenter un bref aperçu du fonctionnement de cette librairie ainsi que des outils de visualisation et de traitement d'image qu'elle nous offre.

A.1 Architecture

Afin de permettre une plus grande flexibilité de développement au niveau de la librairie elle-même et des logiciels dérivés, la bibliothèque VTK est constituée d'un noyau compilé et de *wrappers* interprétés. Le noyau est programmé en C++ et utilise la librairie graphique OpenGL. Le développement de nouveaux modules se fait donc lui aussi en C++. Les *wrappers* permettent un développement plus rapide d'applications, tandis que le noyau compilé assure une certaine robustesse et efficacité (Fig. A.1). La bibliothèque VTK supporte des wrappers pour le langage de programmation interprété Java/Awt et les langages scripts Tcl/Tk et Python/Tk. Bien sûr, il est aussi possible de développer des applications entièrement en C++.

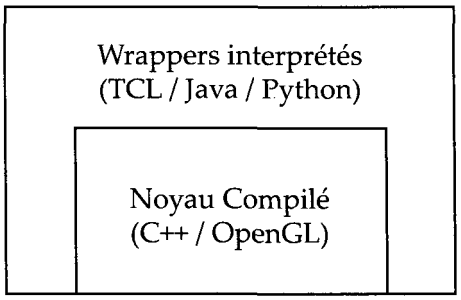


Figure A.1: VTK est divisé en deux grandes composantes : Le noyau compilé en C++ et les *wrappers* interprétés en TCL, Python et Java [60].

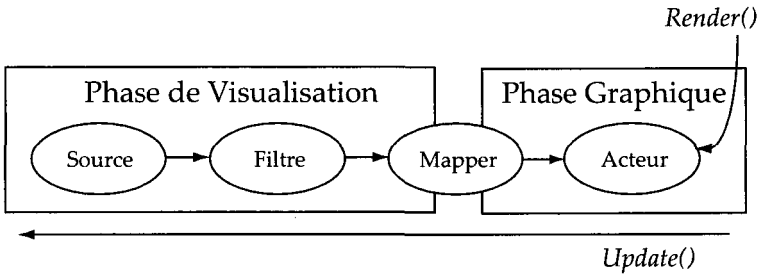


Figure A.2: Le pipeline de VTK est divisé en deux phases : la phase de visualisation et la phase graphique [60].

A.2 Pipeline

Comme pour la majorité des librairies graphiques, VTK utilise un modèle basé sur une architecture en pipeline. Ce type d'architecture consiste à relier plusieurs modules permettant de lire et de modifier des données pour en produire une image. La sortie d'un module peut être utilisée comme entrée pour un autre module. Le pipeline de VTK est divisé en deux grandes phases : *la phase de visualisation* et *la phase graphique* (Fig A.2). Celles-ci sont élaborées plus exhaustivement dans les deux sections suivantes.

A.2.1 Phase de visualisation

La phase de visualisation consiste à transformer les données sources en primitives graphiques. L'exécution de la phase de visualisation est dite "paresseuse" (i.e. le pipeline n'est mis à jour que lorsqu'une requête lui est envoyée par le système ou par l'utilisateur). Par exemple, lorsqu'un objet doit être rendu à l'écran par le système, une demande de mise-à-jour est diffusée à travers le pipeline et les modules rattachés à celui-ci sont exécutés si nécessaire. Le pipeline de visualisation est composé principalement de deux types d'objets : les ensembles de données et les modules de traitement. En voici la description.

Ensembles de données Les ensembles de données gérés par VTK sont différents objets permettant d'encapsuler les données à visualiser de façon générique. Ceux-ci représentent les informations traitées et "transmises" entre les modules du pipeline de VTK. Notons que les ensembles de données ne sont pas réellement transmis entre les différents modules de VTK, chaque module en conserve plutôt une copie. Il est toutefois permis pour un module de disposer de cette copie une fois le traitement terminé. Un ensemble de données est défini par trois composantes : une *géométrie*, une *topologie* et une ou plusieurs *valeur*(s).

La géométrie correspond à l'ensemble des *points* constituant l'objet représenté par l'ensemble de données. Chaque point est affecté d'une position relative à celle de l'objet. Ces points sont associés à des *cellules* définissant leur organisation topologique (ligne, polygones, pixels, voxels, ...). De plus, chaque point peut être associé à une valeur (couleur, valeur scalaire ou vecteur). Par exemple, dans VTK, un volume est constitué d'un ensemble de voxels (cellules), qui eux, sont constitués de cubes de points adjacents disposés uniformément dans une grille tridimensionnelle régulière. Plus généralement, on peut regrouper les ensembles

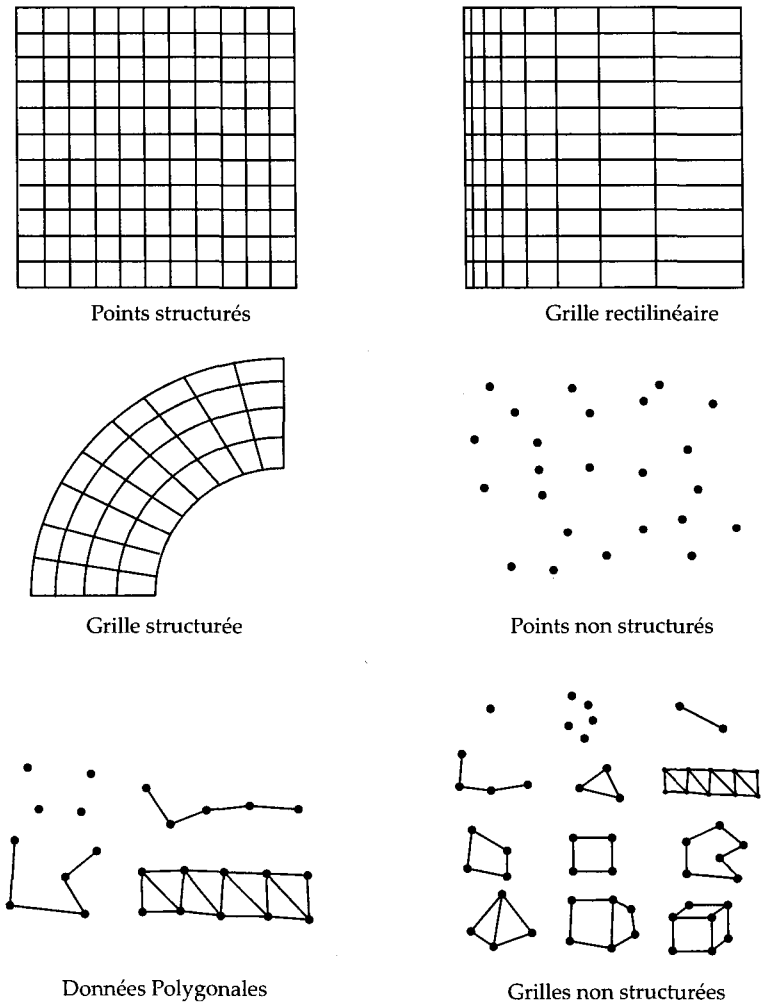


Figure A.3: Types d'ensembles de données utilisés dans VTK [60].

de données en quatre classes distinctes : *données polygonales* (i.e. formées de triangles, lignes, points), *images et volumes* (i.e. grilles de points à topologie et géométrie régulière), *grilles structurées* (i.e. grilles à topologie régulière) et *grilles non structurées* (Fig A.3). On dit qu'un ensemble de données est structuré si sa topologie est régulière.

Modules de traitement Ces objets opèrent généralement sur les données pour les modifier ou en produire de nouvelles. On considère trois types de modules de

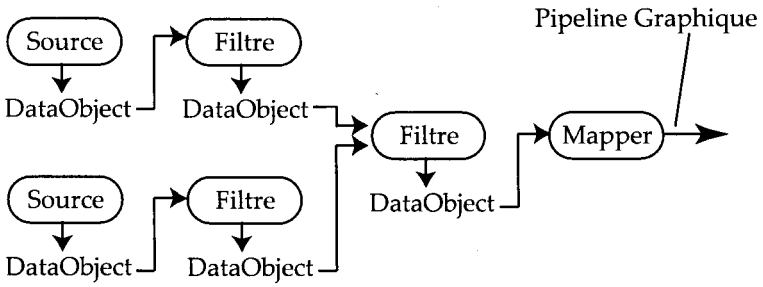


Figure A.4: Exemple de chaîne de filtres utilisée dans VTK [60].

traitement : les *modules sources*, les *filtres* et les *mappers*. Les modules sources sont utilisés afin de débuter la phase de visualisation du pipeline. Ils permettent de générer des données ou encore de les acquérir d'une autre source (ex. : disque dur). Les filtres prennent un ou plusieurs ensembles de données en entrée, les traitent, et en produisent une sortie, qui elle aussi est un ensemble de données. Les filtres forment la majorité des objets présents dans le pipeline et sont connectés entre eux à l'aide de leur(s) entrée(s) et sortie(s) (Fig A.4). À la fin du pipeline de visualisation, les données sont envoyées vers des mappers qui ont pour tâche de les transformer en primitives graphiques (e.g. points, lignes, triangles, ...) pour affichage à l'écran, ou encore de les redirigier vers un fichier. Les mappers servent de ponts entre le pipeline de visualisation et le pipeline graphique (Fig A.2).

A.2.2 Phase graphique

La phase graphique consiste à transformer les primitives graphiques, produites par les *mappers* de la phase de visualisation, en images. Une scène graphique est constituée d'objets (*props*) distribués à travers celle-ci. Ceux-ci ne contiennent pas leur propre représentation géométrique mais sont plutôt associés à des mappers. De plus, ces objets sont aussi associés à des classes décrivant plusieurs de leurs caractéristiques, dont leur position dans l'espace géométrique, leur couleur ainsi que les transformations géomé-

triques qui leur sont appliquées.

La scène graphique est rendue à l'écran à l'intérieur d'une zone d'affichage (*renderer* ou *viewport*). Celle-ci est associée à un objet caméra ainsi qu'à une ou plusieurs sources lumineuses. Ces zones d'affichages sont incluses dans une fenêtre d'affichage pouvant contenir plusieurs zones d'affichage distinctes. De plus, la fenêtre d'affichage est généralement associée à un objet permettant la gestion d'interactions afin que l'utilisateur puisse manipuler la scène 3D ainsi que les objets qui la composent. Étant donné que la bibliothèque VTK utilise OpenGL pour effectuer ses rendus, les zones et les fenêtre d'affichage sont des objets encapsulant les mécanismes de cette librairie et font guise de pont entre le système de fenêtrage (Windows) et le moteur graphique. Afin d'assurer la portabilité et la flexibilité de VTK, la majorité des classes mentionnées précédemments sont abstraites. Cette encapsulation permet d'isoler le plus possible le code dépendant des librairies de bas niveau des autres modules de VTK.

A.3 Outils et techniques de visualisation

Cette section présente un aperçu des modules de traitement, des types de données ainsi que des techniques de visualisation disponibles dans la bibliothèque graphique VTK. Évidemment, VTK comporte un nombre imposant de modules et d'algorithmes de visualisation scientifique, il est donc impossible de les présenter en totalité. Les sections (A.3.1) et (A.3.2) présentent quelques exemples de filtres et quelques algorithmes supportés par VTK permettant de visualiser des données.

A.3.1 Modules de traitement et filtres

Un module de traitement de type filtre accepte un ou plusieurs ensemble(s) de données en entrée, effectue une opération sur ces données et en produit une ou plusieurs sortie(s). Le *color mapping* est un bon exemple de filtre régulièrement utilisé

en visualisation. Ce filtre consiste essentiellement à colorer les points d'un ensemble de données selon leur valeur scalaire à l'aide d'une table de référence (chaque valeur scalaire est associée à une couleur). Par exemple, considérons une grille de points, chacun de ces points étant associés à une valeur scalaire représentant une température. Il est avantageux d'utiliser *color mapping* afin de faire ressortir les points à basse température en bleu et les points à haute température en rouge. Cette approche est beaucoup plus visuelle, comparée à une simple matrice de valeurs numériques.

Un autre processus couramment utilisé en visualisation consiste à extraire des contours (iso-surfaces) d'un ensemble de données afin d'en cibler des régions d'intérêt. Le filtre nécessite la spécification d'une valeur seuil. Les points ayant une valeur scalaire supérieure à la valeur seuil sont alors inclus dans la région d'intérêt tandis que les autres en sont rejetés. La bibliothèque VTK offre plusieurs algorithmes permettant de générer des contours comme les *Marching cubes* (voir section 1.3.1.2) et les *Marching squares.*

VTK offre aussi des outils permettant de rééchantillonner un ensemble de données sur un autre ensemble de données. Cette opération, nommée *probing*, permet entre autres d'effectuer des coupes d'orientation arbitraire dans un volume, ou encore de réechantilloner un ensemble de points non-structuré sur une grille régulière. Ce type de filtre permet entre autres de convertir un ensemble de données en une grille de points structurée pour le visualiser à l'aide d'un algorithme de rendu de volume.

Certains modules de traitement sont réservés uniquement à un type de données en particulier. La bibliothèque VTK supporte une multitude de filtres images-à-images comme des filtres passe-bas et passe-haut, de convolution et de calcul de dérivée première (*gradient*). D'autres filtres permettent de modifier des ensembles de données polygonaux comme le lissage de polygones, le calcul de vecteurs normaux ainsi que la décimation (réduction du nombre de triangles dans un maillage).

A.3.2 Algorithmes de visualisation

La bibliothèque VTK offre une grande variété d'algorithmes de visualisation pour des ensembles de données à une, deux et trois dimensions. Ceux-ci sont généralement spécialisés pour un type de données en particulier. Par exemple, les algorithmes de rendu de volume ne permettent de visualiser que des points disposés dans l'espace sous formes de grilles régulières (volumes). C'est pourquoi il existe plusieurs modules de traitement permettant d'effectuer des conversions d'un type d'ensemble de données à un autre.

Parmi les méthodes de visualisation offertes par VTK, on retrouve le rendu de polygones. Une grande partie des ensembles de données en visualisation scientifique (vecteurs de force, flux, ...) peuvent être représentés à l'aide de primitives graphiques et la majorité des aides visuelles et objets graphiques formant une interface graphique dans une application de visualisation sont dessinées à l'aide de polygones. La librairie permet de convertir la grande majorité des types de données en données polygonales. Dans VTK, les rendus polygonaux sont pris en charge par OpenGL. La librairie permet l'application d'effets de lumière et d'ombrage (Phong et Gouraud), de textures et contient quelques algorithmes utilitaires comme le calcul de vecteurs normaux des polygones formant un maillage.

Les grilles structurées (images et volumes) peuvent quant à elles être visualisées à l'aide d'algorithmes de rendu d'images et de volume. La librairie supporte la lecture de plusieurs types de fichiers images (BMP, JPG, PNG, ...) ainsi que la création d'images à partir de primitives graphiques 2D (lignes, triangles, cercles, ...). Ces images peuvent être rendues dans un espace d'affichage 2D, comme texture appliquée sur un polygone ou encore sous forme de maillages polygonaux. L'image est alors interprétée comme une "surface topographique" pour laquelle l'intensité de chaque pixel de l'image

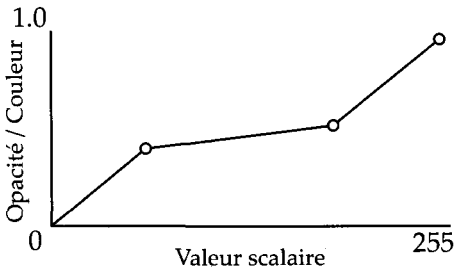


Figure A.5: Les fonctions de transfert sont définies à l'aide d'un ensemble de points dans VTK [60].

correspond à l'élévation d'un point sur la surface. En ce qui concerne les volumes, la bibliothèque VTK incorpore trois mappers permettant d'effectuer des rendus de volume : le *VolumeRayCastMapper*, le *VolumeTextureMapper* et le *VolumeProMapper*. Le *VolumeRayCastMapper* utilise un algorithme de lancer de rayons. Celui-ci doit être associé à une fonction de lancer de rayons, décrivant la contribution de chaque voxel à l'image finale (projection par intensité maximale, extraction de surface ou composition). Le *VolumeTextureMapper* utilise une série de polygones sur lesquels des textures de coupe du volume ont été appliquées. Le *VolumeProMapper* utilise quant à lui l'accélération matérielle produite par la carte graphique VolumePro^{TM} lorsque celle-ci est disponible. De plus, l'apparence du rendu de volume est contrôlée par trois fonctions de transfert. La première fait correspondre la valeur scalaire d'un voxel à une couleur, la deuxième fait correspondre la valeur scalaire d'un voxel à une opacité et la dernière fait correspondre la norme du gradient à la position d'un voxel à un multiplicateur d'opacité. Ces fonctions de transfert sont spécifiées par l'utilisateur en ajoutant et supprimant des points. La fonction est alors interpolée linéairement entre ces points (Fig A.5).

A.4 Interaction

En plus d'outils de visualisation, la librarie VTK offre un système de gestion d'évènements. La fenêtre d'affichage VTK est associée à un objet effectuant cette ges-

tion. Celui-ci saisit les évènements lancés par Windows et les traduit en évènements inteprétables par VTK. La librairie offre un certain nombre de profils d'interaction préconfigurés. Il est aussi possible de créer son propre profil d'interaction en associant des fonctions *callback* à des observateurs d'évènements. Ces fonctions sont passées comme paramètres à l'objet faisant la gestion de l'interaction et sont exécutées par le système lorsque l'évènement qui leur est associé est détecté par l'observateur. De plus, la librairie comporte une collection de contrôleurs graphiques. Ceux-ci répondent à des évènements comme les profils d'interaction, et sont également représentés graphiquement dans la scène.

Bibliographie

[1] Adams, R., and Bischof, L. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence 16*, 6 (1994), 641–647.

[2] Ahlberg, C., and Shneiderman, B. Visual information seeking : tight coupling of dynamic query filters with starfield displays. In *CHI '94 : Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1994), ACM Press, pp. 313–317.

[3] Ahlberg, C., Williamson, C., and Shneiderman, B. Dynamic queries for information exploration : an implementation and evaluation. In *CHI '92 : Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1992), ACM Press, pp. 619–626.

[4] ANGEL, E. *Interactive computer graphics : a top-down approach with OpenGL.* Addison-Wesley, 2006.

[5] Ankerst, M., Keim, D. A., and Kriegel, H.-P. 'circle segments' : A technique for visually exploring large multidimensional data sets. In *Proc. Visualization '96, Hot Topic Session* (San Francisco, CA, 1996).

[6] Arvo, J., Ed. *Graphics gems II.* Academic Press Professional, Inc., San Diego, CA, USA, 1991.

[7] Asimov, D. The grand tour : a tool for viewing multidimensional data. *SIAM J. Sci. Stat. Comput. 6*, 1 (1985), 128–143.

[8] Battista, G. D., Eades, P., Tamassia, R., and Tollis, I. G. Algorithms for drawing graphs : an annotated bibliography. *Comput. Geom. Theory Appl. 4*, 5 (1994), 235–282.

[9] Bederson, B. B., and Hollan, J. D. Pad++ : a zooming graphical interface for exploring alternate interface physics. In *UIST '94 : Proceedings of the 7th annual ACM symposium on User interface software and technology* (New York, NY, USA, 1994), ACM Press, pp. 17–26.

[10] Bergman, L., Rogowitz, B., and Treinish, L. A rule-based tool for assisting colormap selection. *vis 0* (1995), 118.

[11] BIER, E. A., STONE, M. C., PIER, K., BUXTON, W., AND DEROSE, T. D. Toolglass and magic lenses : the see-through interface. In *SIGGRAPH '93 : Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1993), ACM Press, pp. 73–80.

[12] CARD, S. K., MACKINLAY, J. D., AND SHNEIDERMAN, B., Eds. *Readings in information visualization : using vision to think.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[13] CHAMBERS, J. M., CLEVELAND, W. S., KLEINER, B., AND TUKEY, P. A. *Graphical Methods for Data Analysis.* Wadsworth/Brooks Cole, Monterey, CA, 1983.

[14] CHEN, M., MOUNTFORD, S. J., AND SELLEN, A. A study in interactive 3-d rotation using 2-d control devices. In *SIGGRAPH '88 : Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1988), ACM Press, pp. 121–129.

[15] CHERNOD, H. The use of faces to represent points in k-dimensional space graphically. *J. Am. Statistical Assoc. 68* (June 1973), 361–368.

[16] CHIKOFSKY, E. J., AND II, J. H. C. Reverse engineering and design recovery : A taxonomy. *IEEE Softw. 7*, 1 (1990), 13–17.

[17] CHIRICOTA, Y., JOURDAN, F., AND MELANCON, G. Software components capture using graph clustering. In *IWPC '03 : Proceedings of the 11th IEEE International Workshop on Program Comprehension* (Washington, DC, USA, 2003), IEEE Computer Society, p. 217.

[18] CHIRICOTA, Y., JOURDAN, F., AND MELANCON, G. Metric-based network exploration and multiscale scatterplot. In *INFOVIS '04 : Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'04)* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 135–142.

[19] CLEVELAND, W. S. *Visualizing Data.* Hobart Press, 1993.

[20] CONNER, B. D., SNIBBE, S. S., HERNDON, K. P., ROBBINS, D. C., ZELEZNIK, R. C., AND VAN DAM, A. Three-dimensional widgets. In *SI3D '92 : Proceedings of the 1992 symposium on Interactive 3D graphics* (New York, NY, USA, 1992), ACM Press, pp. 183–188.

[21] COOK, B., ASIMOV, D., AND HURLEY, D. Theory and computational methods for dynamic projections in high-dimensional data visualization.

[22] DOS SANTOS, S. R., AND BRODLIE, K. W. Visualizing and investigating multidimensional functions. In *VISSYM '02 : Proceedings of the symposium on Data Visualisation 2002* (Aire-la-Ville, Switzerland, Switzerland, 2002), Eurographics Association, pp. 173–ff.

[23] DREBIN, R. A., CARPENTER, L., AND HANRAHAN, P. Volume rendering. *SIGGRAPH Comput. Graph. 22*, 4 (1988), 65–74.

[24] E., K. Approximating complex surfaces by triangulation of contour lines. *IBM Journal of Research and Development 19* (1975), 2–11.

[25] FEINER, S. K., AND BESHERS, C. Worlds within worlds : metaphors for exploring n-dimensional virtual worlds. In *UIST '90 : Proceedings of the 3rd annual ACM SIGGRAPH symposium on User interface software and technology* (New York, NY, USA, 1990), ACM Press, pp. 76–83.

[26] FEUER, A. R. *MFC Programming with Cdrom.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[27] FOWLER, M., AND SCOTT, K. *UML distilled : applying the standard object modeling language.* Addison-Wesley Longman Ltd., Essex, UK, UK, 1997.

[28] FRIEDMAN, J. H. Exploratory projection pursuit. *Journal of the American Statistical Association 82*, 397 (1987), 249–266.

[29] GELMAN, A., CARLIN, J. B., STERN, H. S., AND RUBIN, D. B. *Bayesian Data Analysis, Second Edition.* Chapman & Hall/CRC, July 2003.

[30] HARARY, F. *Graph theory.* Addison-Wesley, 1969.

[31] HAVELIWALA, T., GIONIS, A., KLEIN, D., AND INDYK, P. Evaluating strategies for similarity search on the web, 2002.

[32] HERMAN, I., MARSHALL, M. S., MELANCON, G., DUKE, D. J., DELEST, M., AND DOMENGER, J.-P. Skeletal images as visual cues in graph visualization. In *Data Visualization '99*, E. Gröller, H. Löffelmann, and W. Ribarsky, Eds. Springer-Verlag Wien, 1999, pp. 13–22.

[33] HERMAN, I., MELANCON, G., AND MARSHALL, M. S. Graph visualization and navigation in information visualization : A survey. *IEEE Transactions on Visualization and Computer Graphics 6*, 1 (2000), 24–43.

[34] HERNDON, K. P., AND MEYER, T. 3d widgets for exploratory scientific visualization. In *UIST '94 : Proceedings of the 7th annual ACM symposium on User interface software and technology* (New York, NY, USA, 1994), ACM Press, pp. 69–70.

[35] HOPCROFT, J. E., AND ULLMAN, J. D. *Introduction to Automata Theory, Languages, and Computation.* Series in Computer Science. Addison-Wesley, Reading, MA, 1979.

[36] INSELBERG, A., AND DIMSDALE, B. Parallel coordinates : a tool for visualizing multi-dimensional geometry. In *VIS '90 : Proceedings of the 1st conference on Visualization '90* (Los Alamitos, CA, USA, 1990), IEEE Computer Society Press, pp. 361–378.

[37] J. EDWARD SWAN, I., MUELLER, K., MÜLLER, T., SHAREEF, N., CRAWFIS, R., AND YAGEL, R. An anti-aliasing technique for splatting. In *Proceedings of the 8th conference on Visualization '97* (1997), IEEE Computer Society Press, pp. 197–ff.

[38] J. M. CHAMBERS, W. S. CLEVELAND, B. KLEINER, AND P. A. TUKEY. *Graphical Methods for Data Analysis.* Chapman and Hall, New York, 1983.

[39] JOHN, M. S., SMALLMAN, H. S., AND OONK, H. M. The use of 2d and 3d displays for shape-understanding versus relative-position tasks. *Human Factors 43*, 1 (2001), 79–98.

[40] KEIM, D. A. Pixel-oriented visualization techniques for exploring very large databases. *Journal of Computational and Graphical Statistics*, March (1996), 58–77.

[41] KEIM, D. A. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics 8*, 1 (2002), 1–8.

[42] KEIM, D. A., ANKERST, M., AND KRIEGEL, H.-P. Recursive pattern : A technique for visualizing very large amounts of data. In *IEEE Visualization* (1995), pp. 279–286.

[43] LEBLANC, B., DION, D., AUBER, D., AND MELANCON, G. Constitution et visualisation de deux réseaux d'associations verbales. In *Actes du colloque ALCAA 2001* (septembre 2001), pp. 37–43.

[44] LEBLANC, J., WARD, M. O., AND WITTELS, N. Exploring n-dimensional databases. In *VIS '90 : Proceedings of the 1st conference on Visualization '90* (Los Alamitos, CA, USA, 1990), IEEE Computer Society Press, pp. 230–237.

[45] LEVOY, M. Display of surfaces from volume data. *IEEE Comput. Graph. Appl. 8*, 3 (1988), 29–37.

[46] LIN, C.-F., YANG, D.-L., AND CHUNG, Y.-C. A marching voxels method for surface rendering of volume data. In *Computer Graphics International 2001* (2001), IEEE Computer Society, pp. 306–316.

[47] LORENSEN, W. E., AND CLINE, H. E. Marching cubes : A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (1987), ACM Press, pp. 163–169.

[48] MARSHALL, M. S. *Methods and tools for the visualization and navigation of graphs.* PhD thesis, Université de Bordeaux, 2001.

[49] MARTIN, A. R., AND WARD, M. O. High dimensional brushing for interactive exploration of multivariate data. In *VIS '95 : Proceedings of the 6th conference on Visualization '95* (Washington, DC, USA, 1995), IEEE Computer Society, p. 271.

[50] MCCORMICK, B. H. Visualization in scientific computing. *SIGBIO Newsl. 10*, 1 (1988), 15–21.

[51] MEISSNER, M., HUANG, J., BARTZ, D., MUELLER, K., AND CRAWFIS, R. A practical evaluation of popular volume rendering algorithms. In *Proceedings of the 2000 IEEE symposium on Volume visualization* (2000), ACM Press, pp. 81–90.

[52] NEY, D. R., FISHMAN, E. K., MAGID, D., AND DREBIN, R. A. Volumetric rendering. *IEEE Comput. Graph. Appl. 10*, 2 (1990), 24–32.

[53] PHONG, B. T. *Illumination for computer-generated images.* PhD thesis, 1973.

[54] PORTER, T., AND DUFF, T. Compositing digital images. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (1984), ACM Press, pp. 253–259.

[55] RHYNE, T.-M., TORY, M., MUNZNER, T., WARD, M., JOHNSON, C., AND LAIDLAW, D. H. Information and scientific visualization : Separate but equal or happy together at last. In *VIS '03 : Proceedings of the 14th IEEE Visualization 2003 (VIS'03)* (Washington, DC, USA, 2003), IEEE Computer Society, p. 115.

[56] ROBERTSON, G. G., CARD, S. K., AND MACKINLAY, J. D. Information visualization using 3d interactive animation. *Commun. ACM 36*, 4 (1993), 57–71.

[57] ROGOWITZ, B. E., AND TREINISH, L. A. Data visualization : the end of the rainbow. *IEEE Spectr. 35*, 12 (1998), 52–59.

[58] ROXBOROUGH, T., AND SEN, A. Graph clustering using multiway ratio cut. In *GD '97 : Proceedings of the 5th International Symposium on Graph Drawing* (London, UK, 1997), Springer-Verlag, pp. 291–296.

[59] SARKAR, M., AND BROWN, M. H. Graphical fisheye views. *Commun. ACM 37*, 12 (1994), 73–83.

[60] SCHROEDER, W. *The VTK User's Guide.* Kitware Inc., 2001.

[61] SIPSER, M. *Introduction to the Theory of Computation.* International Thomson Publishing, 1996.

[62] STOLTE, C., TANG, D., AND HANRAHAN, P. Polaris : A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics 8*, 1 (2002), 52–65.

[63] TORY, M. Mental registration of 3d views and 2d orthographic views in orientation icon displays.

[64] TORY, M. Mental registration of 2d and 3d visualizations (an empirical study). In *VIS '03 : Proceedings of the 14th IEEE Visualization 2003 (VIS'03)* (Washington, DC, USA, 2003), IEEE Computer Society, p. 49.

[65] TORY, M., AND ET AL. Comparing exovis, orientation icon, and in-place 3d visualization techniques, 2003.

[66] TORY, M., MOLLER, T., ATKINS, M. S., AND KIRKPATRICK, A. E. Combining 2d and 3d views for orientation and relative position tasks. In *CHI '04 : Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2004), ACM Press, pp. 73–80.

[67] VAN WIJK, J. J., AND VAN LIERE, R. Hyperslice : visualization of scalar functions of many variables. In *VIS '93 : Proceedings of the 4th conference on Visualization '93* (1993), pp. 119–125.

[68] VIEGA, J., CONWAY, M. J., WILLIAMS, G., AND PAUSCH, R. 3d magic lenses. In *UIST '96 : Proceedings of the 9th annual ACM symposium on User interface software and technology* (New York, NY, USA, 1996), ACM Press, pp. 51–58.

[69] WARD, M. O. Xmdvtool : integrating multiple methods for visualizing multivariate data. In *VIS '94 : Proceedings of the conference on Visualization '94* (Los Alamitos, CA, USA, 1994), IEEE Computer Society Press, pp. 326–333.

[70] WATTENBERG, M., AND FISHER, D. A model of multi-scale perceptual organization in information graphics. *infovis 00* (2003), 4.

[71] WEGMAN, E. The grand tour in k-dimensions. *Center for Computational Statistics, George Mason University*, 68 (1991).

[72] WESTOVER, L. Interactive volume rendering. In *Proceedings of the 1989 Chapel Hill workshop on Volume visualization* (1989), ACM Press, pp. 9–16.

[73] WESTOVER, L. Footprint evaluation for volume rendering. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (1990), ACM Press, pp. 367–376.

[74] WONG, P. C., AND BERGERON, R. D. 30 years of multidimensional multivariate visualization. In *Scientific Visualization, Overviews, Methodologies, and Techniques* (Washington, DC, USA, 1997), IEEE Computer Society, pp. 3–33.

[75] YANG, L. Interactive exploration of very large relational datasets through 3d dynamic projections. In *KDD '00 : Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2000), ACM Press, pp. 236–243.