Rendu narratif en synthèse d'images

Thierry Stein

Université de Grenoble
Laboratoire Jean Kuntzmann (LJK)

Thèse

présentée pour l'obtention du titre de
Docteur de l'Université de Grenoble, spécialité Informatique

par

Thierry Stein

Rendu narratif en synthèse d'images

Soutenue le 17 Mai 2010 devant le jury composé de :

M.	Pascal Guitton	Université de Bordeaux 1	(Rapporteur)
M.	Pierre Alliez	INRIA Sophia-Antipolis	(Rapporteur)
M.	Georges-Pierre Bonneau	Université de Grenoble	(Examinateur)
M.	Pascal Barla	INRIA Bordeaux	(Examinateur)
M.	Nicolas Holzschuch	INRIA Rhône-Alpes	(Directeur)
M.	François Sillion	INRIA Rhône-Alpes	(Co-Directeur)

À mon père

Remerciements

Avant toute considération scientifique, une thèse est une aventure humaine, avec ses hauts et ses bas. Souvent solitaire, cette aventure donne d'autant plus d'importance à ceux qui nous entourent. A eux, je tiens donc à adresser mes plus chaleureux remerciements :

Mes encadrants tout d'abord : Nicolas Holzschuch, pour la liberté qu'il m'a laissée et la pertinence de ses conseils, François Sillion pour sa sagesse et sa bienveillance, et Xavier Décoret, qui a cru en moi en étant à l'origine de ce travail.

Pascal Guitton et Pierre Alliez, qui ont accepté avec enthousiasme d'être les rapporteurs de cette thèse, ainsi que pour la pertinence, la justesse et l'honnêteté de leurs retours. La soutenance fut un moment chaleureux, et ce fut en partie grace à eux.

Chaleureux aussi grace à Pascal Barla, examinateur à l'oeil aceré et à l'enthousiasme communicatif, que j'ai été ravi de revoir à cette occasion ; et à Georges-Pierre Bonneau, indispensable président de ce jury, quatre ans après m'avoir annoncé l'attribution de ma bourse de thèse la veille de mon anniversaire !

Un peu coach mental, un peu mentor, un peu confident, François Rechenman a été tout cela, et un peu plus encore. Ce manuscrit existerait-il sans lui ?

L'équipe ARTIS, dont j'ai partagé le quotidien pendant quatre ans : les thésards passent, parfois repassent, les permanents restent et se souviennent, et les bons souvenirs se partagent. Cela restera une fierté d'avoir effectué ma thèse parmi eux. Mention spéciale à mes premiers guides dans le monde de la recherche : Joelle et FF (et au passage une pensée pour l'équipe EVASION qui m'a accueilli pendant mon DEA).

L'équipe Nano-D, deuxième famille pendant les derniers mois ; Daniel Dumas et la troupe de théatre amateur de l'INRIA, pour l'énergie incroyable avant de monter sur les planches ; les anciens du DEA IVR promotion 2005-2006, car c'est aussi là que tout a commencé.

A tous ceux qui furent présents le 17 Mai, l'après-midi ou le soir, et qui ont contribué à faire de ce jour ce qu'il est et restera dans mes souvenirs !

A mon père, à ma mère, ta présence compte désormais double.

A Evelyne, cette thèse est aussi un peu la tienne.

Table des matières

Et maintenant, tournez la page !
Steve Jackson et Ian Livingston
Le Sorcier de la Montagne de Feu

Le récit graphique

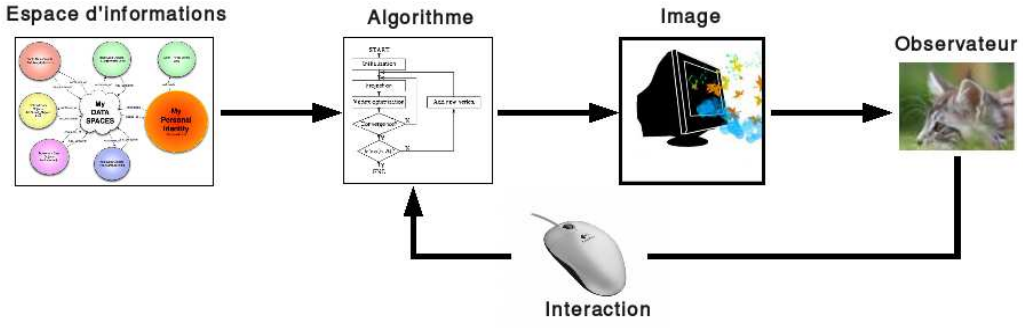
1

Rendu narratif : *Génération d'images pour la communication visuelle d'informations temporelles.*

Image, communication, information temporelle : voici en quelques mots ce qui définit la problématique ayant guidé ces travaux de thèse.

Parler d'information ayant une dimension temporelle, c'est plus précisément parler d'une information qui affiche une évolution au cours du temps, ou encore de mouvement. Dans notre societé numérique actuelle, on peut imaginer une infinité d'illustrations de ce type de données : simulations de fluides ou de solides, capture de mouvement, animations, vidéos, itinéraires. Pourtant, au-delà de cette diversité, la question de leur visualisation reste toujours la même : **comment représenter dans une image, par essence bi-dimensionnelle, une information de mouvement au minimum tri-dimensionnelle ?**

Avant de répondre à cette question, commençons par définir un système de visualisation, ce qu'il considère en entrée et ce qu'il produit en sortie. Indépendemment de la nature des données d'entrées, un tel système peut se décrire par le schéma suivant :



Notre observateur souhaite analyser, comprendre ou plus simplement percevoir un espace d'informations au sens large. Cette espace d'informations est traité par un ou plusieurs algorithmes qui fourniront une image. Pour modifier son point de vue, l'observateur peut interagir avec

l'algorithme, ce qui donnera une nouvelle image (un exemple classique d'interaction consiste à modifier la position et l'orientation de la caméra). Notons que ce schéma est valable dans le cadre d'un utilisateur passif vis-a-vis des données qu'il observe, a contrario d'une application de type modélisation où l'utilisateur peut aussi interagir avec l'espace d'informations. Dans le cadre de la visualisation, celui-ci est considéré comme statique et l'observateur ne peut agir qu'au niveau de l'algorithme.

Le rendu narratif trouve sa spécificité dans l'espace d'informations considéré Thomas Strothotte [Str98] distingue le modèle géométrique qui définit l'apparence physique et la forme des données, et le modèle symbolique, tel que du texte, qui contient les informations n'ayant pas d'influence directe sur cette apparence. A ces deux composantes, le rendu narratif ajoute un aspect chronologique, c'est-à-dire que le modèle géométrique comme le modèle symbolique contiennent une dimension temporelle.

En utilisant la terminologie du théatre, nous pouvons alors décrire notre espace d'informations par trois composantes :

1. **Les acteurs** correspondent aux composantes géométriques animées.
2. **Le décor** décrit la géométrie statique qui vient se greffer autour des acteurs.
3. **Le texte** décrit l'ensemble des données symboliques qui viennent enrichir la géométrie, sans distinction initiale quant à leur caractère dynamique ou non.

Si l'on assimile l'algorithme au metteur en scène et si l'on ommet provisoirement l'aspect interactif de ce système, le schéma exposé plus haut n'est alors en rien spécifique à la jeune discipline de l'informatique.

Fondamentalement, visualiser une information temporelle revient à raconter une histoire en images. Il convient donc, avant de présenter et de définir en détail le rendu narratif sur ordinateur, de parler de son ancêtre de papier.

1.1 Le récit graphique sur papier

Il existe de nombreuses façons de raconter des histoires : avec des mots, avec des sons, avec des gestes, des goûts, des arômes ou des parfums.

Parler de récit en image, c'est selon nous parler tout simplement de

bande dessinée [1]. La définition la plus générique et la plus satisfaisante en a été donnée par Scott MCCould dans *L'art invisible* :

> **Images picturales et autres, volontairement juxtaposées en séquences, destinées à transmettre des informations et/ou à provoquer une réaction esthétique chez le lecteur.**

Suivant cette définition, une bande dessinée commence donc à partir de deux images qui s'enchaînent pour former une séquence. Ainsi, un schéma d'assemblage et une procédure d'évacuation sont des bandes dessinées.

Deux cas extrêmes pourraient contredire cette définition : les aficionados de "La bande pas dessinée" (Plus qu'un concept, un pas concept) objecteront que l'image n'est pas nécessaire :

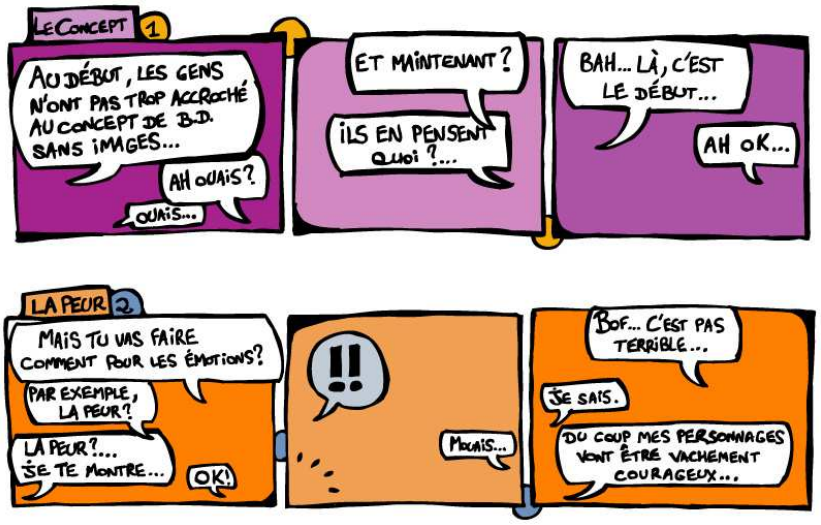


FIGURE 1.1 – *Les deux premiers épisodes de "La bande pas dessinée", extraits de http ://www.labandepasdessinee.com/*

Il n'y a pas d'images, mais le reste du vocabulaire et de la grammaire de la BD est exploité : composition des éléments organisés en séquences, phylactères (nom savant des bulles), onomatopées.

Plus délicat est le cas d'une seule image, comme dans l'exemple de la figure 1.2.

1. Le terme de BD est malheureusement encore marqué d'une connotation péjorative, renvoyant pour certaines personnes à des gags de quelques cases destinés aux enfants. Les francophones ne sont pourtant pas les plus mal lotis : ainsi les américains lisent des comics et les japonais des « images dérisoires ». Will Eisner a proposé les expressions de Récit Graphique ou Art Séquentiel, mais ces termes, bien que probablement plus descriptifs, n'ont guère dépassé le stade du débat d'expert.

FIGURE 1.2 – *Le héros sans-emploi !*

Formellement, c'est-à-dire au sens de McCloud, il ne s'agit plus de BD. Cependant, comme dans l'exemple précédent, bien qu'il n'y ait qu'une seule image, le reste du langage de la bande dessinée est exploité. On notera en particulier l'ensemble des indices visuels au niveau de la voiture et du vélo qui suggèrent le mouvement.

Ces deux exemples contradictoires nous permettent de préciser ce que nous entendons dans cette thèse par le terme "bande dessinée" : un itinéraire visuel formé par l'agencement des différents éléments. Dans "La bande pas dessinée", les bulles de texte sont organisées spatialement de manière à être lues dans un certain ordre. Il en est de même dans l'illustration de Franquin : le lecteur découvre d'abord Longtarin sur son vélo, puis le gaz d'échappement émanant de la voiture du héros sans-emploi, et enfin le dialogue entre Gaston et Jules de chez Smith. Cette notion de séquence est importante : c'est d'elle que naît la temporalité, c'est elle qui nous permet de dire objectivement si un document se rattache à notre problématique.

De par sa nature, la bande dessinée se compare naturellement avec deux médias auxquels elle est étroitement reliée : le texte et la vidéo.

Les amateurs de littérature vous affirmeront que le pouvoir d'évocation du texte est sans comparaison avec l'objectivité froide de l'image, même stylisée. La bande dessinée tire, elle, avantage de l'universalité de son langage. L'exemple des procédures d'urgence en avion est parlant à cet égard. Un vol Air-France est une véritable tour de babel : les passagers n'utilisent ni le même alphabet, ni le même sens de lecture. Pourtant, grâce à l'image, les procédures sont lisibles et compréhensibles par tous.

Une vidéo, objecteront alors les amateurs du septième art, remplit

aussi cet office, et de manière beaucoup plus claire. Mais s'il est vrai qu'une vidéo montrant le mouvement d'une main sera toujours moins ambiguë qu'une suite d'images juxtaposées en séquence, la BD compense par son aspect compact. Elle permet de parcourir et de rechercher une information beaucoup plus facilement et rapidement, sans avoir à visualiser, même en accéléré, les plusieurs minutes qui composent la vidéo.

Universalité du langage, parcours et recherche facilités, la bande dessinée est donc un choix pertinent pour visualiser et communiquer des informations de nature temporelle. Pour illustrer ce constat, la figure 1.3 nous semble être un exemple parfait de récit graphique, au sens d'itinéraire visuel dans une image.

Dessinée par Charles-Joseph Minard, cette carte décrit l'évolution des effectifs de l'armée de l'empereur Napoléon Ier lors de la campagne de Russie de 1812. L'avancée de l'armée jusqu'aux portes de Moscou est décrite par la trajectoire rose-saumon, la retraite correspondant à la trajectoire noire. L'épaisseur de ces flèches est fonction des effectifs. On remarque notamment la brutale réduction d'épaisseur de l'itinéraire noir au moment de la traversée de la Bérézina.

Dans *The Visual Display of Quantitative Information* [Tufo1], Edward Tufte présente cette carte comme étant peut-être "the best statistical graphic ever drawn". Si on peut trouver excessif un tel enthousiasme, il faut néanmoins reconnaître la force de ce graphique ainsi que la manière dont chaque information est représentée :

- Les acteurs sont clairement exposés. On suit sans difficulté le parcours de l'armée napoléonienne, les séparations et les regroupements, ainsi que la réduction tragique de ses effectifs.
- Le décor est réduit au strict minimum ! Aucun relief, aucun polygone symbolisant les villes, on voit juste les fleuves les plus importants dont la Bérézina, en un sens actrice à part entière de cette campagne.
- Le texte apporte plusieurs informations complémentaires : nom des villes et des fleuves, effectifs ponctuels, ainsi que l'évolution des températures lors de la retraite. Contrairement aux autres données précitées, la température a un aspect dynamique. Cependant, la temporalité de cette information vient en grande partie de sa connexion, par l'intermédiaire d'ancres, avec l'itinéraire des troupes.

Cette carte est complétée par le bandeau supérieur exposant titre, nom de l'auteur, date de réalisation et explication du document. On notera que le nom de l'empereur ne figure nulle part sur ce graphique. Une omission qui ajoute beaucoup de sens à une carte d'apparence neutre.

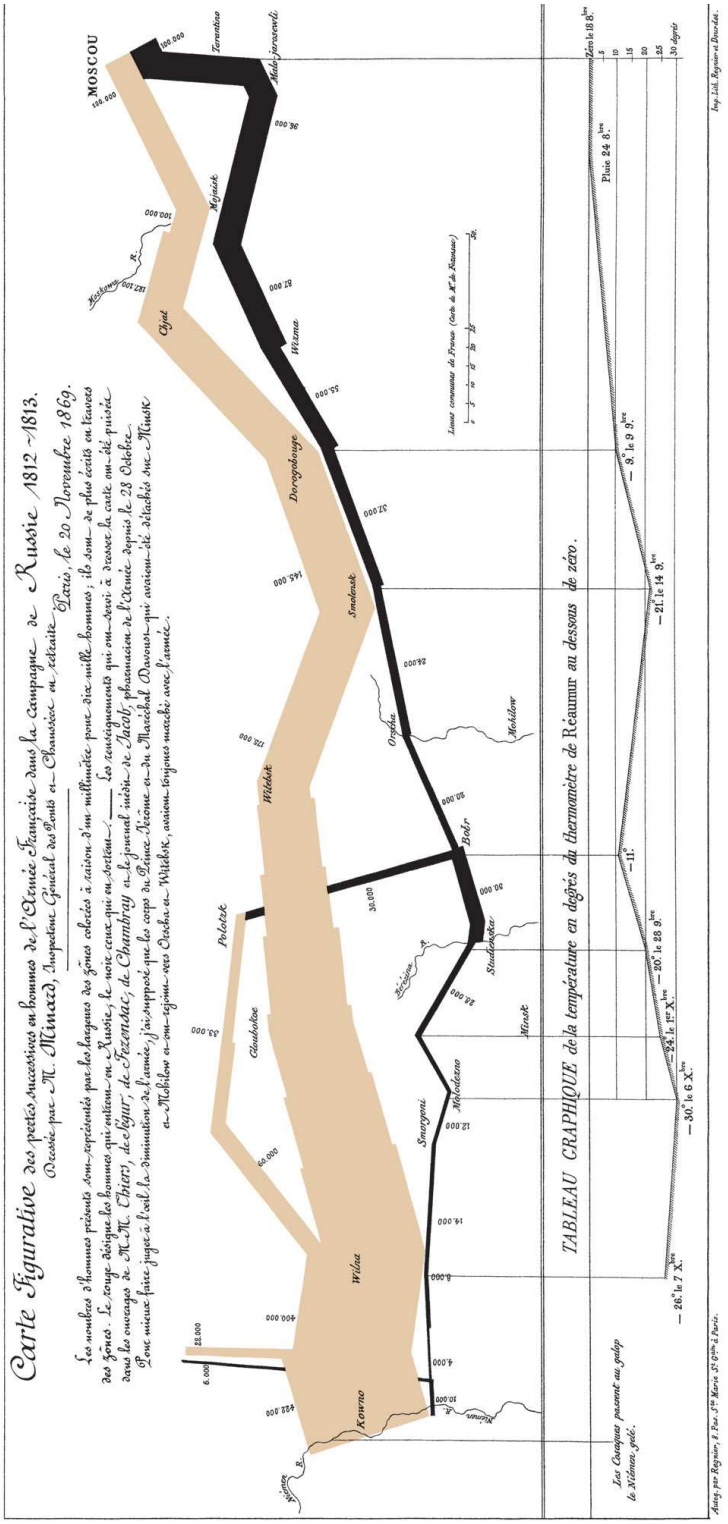


FIGURE 1.3 – *Carte figurative des pertes successives en hommes de l'armée française dans la campagne de Russie 1812-1813, par Charles-Joseph Minard.*

1.2 Le récit graphique sur ordinateur

Quittons maintenant le papier pour nous intéresser à l'objet du vingtième siècle, l'ordinateur, et à la discipline qui le soutient, l'informatique [2].

La notion de récit graphique sur ordinateur nécessite d'aborder plusieurs aspects :

- La représentation de l'information.
- La génération d'images à partir de cette information.
- La dimension nouvelle de l'interactivité apportée par le support informatique.

Notre information initiale se décompose en trois catégories : les acteurs, le décor et le texte. Nous discuterons en détail des aspects spécifiques de représentation et de visualisation que soulève chacune de ces catégories dans les chapitres dédiés. Pour le moment, nous souhaitons aborder deux caractéristiques majeures propres à l'informatique.

La première concerne l'automatisation plus ou moins totale du processus de génération d'images, qui nécessite de traduire en opérations compréhensibles par la machine l'expérience, la perception et le jugement d'illustrateurs. Il est connu que l'ordinateur règne sur le domaine quantitatif mais qu'il est encore difficilement capable d'émuler le jugement qualitatif de l'humain. Les travaux présentés dans ce manuscrit visent à une automatisation totale du processus. Nous avons effectué ce choix car, dans le domaine qualitatif, il est souvent plus simple d'aller d'une solution automatique vers une intervention manuelle. L'étude de l'automatisation est alors enrichissante par les problèmes qu'elle soulève, problèmes qui ne seraient pas envisagés si l'on choisissait directement de confier telle tâche à tel utilisateur.

La deuxième caractéristique réside dans l'interaction, et donc dans l'aspect dynamique conféré par le portage d'un document du papier vers l'écran. Là où, sur le papier, le point de vue sur l'espace d'informations est imposé par l'illustrateur, l'ordinateur permet à l'observateur de changer ce point de vue, et par là-même d'explorer les données à différentes échelles spatiales et temporelles. Cette recherche d'interactivité impose une contrainte sur les algorithmes, qui doivent se montrer suffisamment réactifs et donc rapides pour ne pas nuire à l'exploration.

Nous nous plaçons dans un cadre de visualisation, dans lequel nous avons précisé que l'espace d'informations était statique. Ce constat ouvre la voie au pré-traitement. Il importe peu qu'un calcul prenne plusieurs

2. On parle d'ailleurs aujourd'hui volontiers de Sciences et Technologies de l'Information et de la Communication, soit des termes faisant volontairement écho à notre définition initiale du rendu narratif.

heures si l'information qu'il fournit se révèle suffisamment riche pour permettre une exploration temps-réel et reproductible des données traitées. Cependant, tout pré-calcul peut devenir une limitation à l'exploration en supprimant des degrés de liberté. Il faut donc accepter des compromis. Par ailleurs, le volume de données actuelles est en perpetuelle augmentation. Un précalcul qui nous semble raisonnable lors de nos petites expériences peut rapidement exploser lors de son transfert vers l'extérieur. Ce danger apparaît parfois de manière évidente, parfois plus sournoisement et est, en ce qui nous concerne, délicat à évaluer.

1.3 Le récit de la suite de ce document

Le chapitre 2 sera entièrement consacré à l'étude des différentes techniques inventées et utilisées par les illustrateurs pour créer des images en mouvement.

Dans le chapitre 3, nous présenterons les travaux qui se sont attaqués directement ou partiellement à notre problématique, dans un tour d'horizon qui nous promenera de la visualisation scientifique à la synthèse d'images. La définition de notre espace d'informations nous fournit ensuite le plan de nos contributions.

Les acteurs seront abordés dans le chapitre 4. Nous commencerons par donner une définition générale de données en mouvement puis nous présenterons un pipeline permettant de générer des images résumant un mouvement. Nous présenterons nos résulats dans les cas 2D et 3D + temps, en insistant sur l'aspect générique et flexible de notre approche. Nous terminerons ce chapitre en évoquant des pistes autour du traitement de la dimension temporelle.

Le décor sera développé dans le chapitre 5. L'originalité de ce travail réside dans les interconnexions entre la visualisation de la scène et la visualisation de l'animation qui se joue à l'intérieur. Nous présenterons un algorihtme basé sur la segmentation de la géométrie statique et dynamique et nous montrerons deux applications de visualisation du décor associé aux acteurs.

Le texte fera l'objet du chapitre 6. Nous présenterons un algorithme d'ajout d'annotations à un modèle 3D, permettant une mise à jour majoritairement continue et temps réel dans le cadre d'un scénario interactif. Nous évoquerons ensuite plusieurs pistes de recherche, notamment au niveau de l'ajout d'une chronologie dans les annotations et de ses répercussions sur notre approche.

L'ensemble visualisation/interaction développé dans ces travaux nous permet enfin d'envisager la définition et la conception d'un viewer 4D, c'est-à-dire d'une interface permettant de naviguer dans la dimen-

sion temporelle comme dans les dimensions spatiales. Nous développerons dans le chapitre 7 les perspectives d'un tel outil avant de prendre un peu de recul pour évoquer le futur du rendu narratif sur un plan général et aussi personnel.

Le mouvement en images

2

Une image est par essence fixe. Cependant, son contenu peut créer l'impression du mouvement et évoquer des actions parfois complexes. Dans ce qui suit, nous mettrons de côté les techniques d'illusion d'optique, telles que les quadrillages dont les intersections semblent se promener lorsque l'on fixe l'image, pour nous concentrer sur les techniques objectives et utilisables dans le plus grand nombre d'applications.

2.1 Le mouvement dans une image

Dans [Cuto2], James Cutting se livre à une analyse comparative des cinq grandes techniques picturales permettant de représenter le mouvement dans une image.

Illustrées par la figure 2.1, ces techniques sont :

1. L'ajout de flèches et autres indicateurs de mouvement, c'est à dire l'utilisation d'éléments abstraits qui viennent enrichir l'image originale.
2. L'utilisation de flou, technique issue des expositions longue-durée en photographie.
3. La représentation du sujet dans une position de déséquilibre. Le déséquilibre est le plus souvent suggéré par une rupture de symétrie dans la pose du sujet, les connaissances *a priori* de notre cerveau interprétant alors l'image comme une action en cours et non un moment figé.
4. Une déformation ou cisaillement du sujet en mouvement, très utilisée en illustration. Cette technique est particulièrement adaptée à des objets indéformables, typiquement des véhicules, que l'on ne peut pas représenter dans une pose déséquilibrée.
5. La représentation juxtaposée de plusieurs positions successives, dont on peut trouver l'idée chez Descartes ou chez les peintres impressionnistes tels que Duchamp.

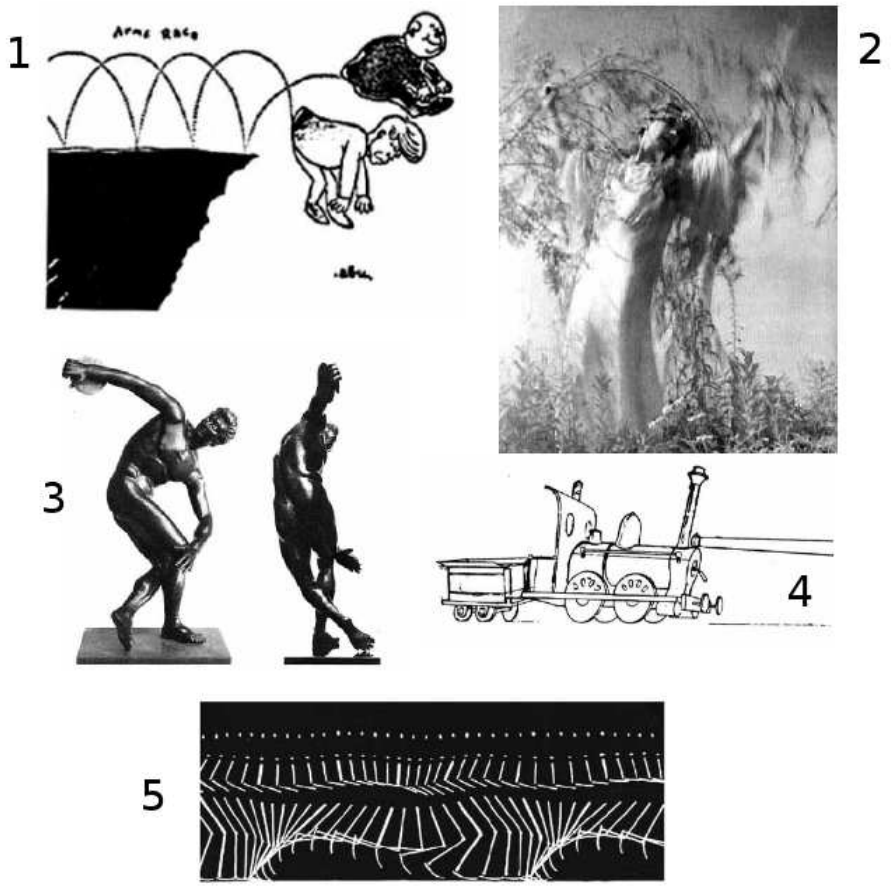


FIGURE 2.1 – *Les 5 types de représentations du mouvement d'après Cutting : en haut à gauche (1), le mouvement et le jeu de saute-mouton des deux personnages sont illustrés par l'ajout de lignes de mouvement ; en haut à droite (2), une exposition prolongée crée un flou dans l'image qui suggère le déplacement ; au milieu à gauche (3), le discobole est modélisé dans une position de déséquilibre et notre cerveau infère l'action ; au milieu à droite (4), la locomotive a subit un cisaillement, et notre double connaisance de la forme d'un train et des techniques d'illustration nous font comprendre le mouvement ; en bas (5), la course du personnage a été échantillonée et les différentes positions juxtaposées.*

Les cinq techniques recensées par James Cutting sont exclusivement centrées sur l'acteur en mouvement. En prenant le contrepied de ces techniques, il est possible de suggérer le mouvement en jouant sur la représentation de l'environnement. L'illusion est un peu celle du voyageur : lorsqu'un train sur la voie voisine se met à bouger, nous avons l'impression confuse que c'est nous qui sommes en mouvement. De la même manière, un illustrateur peut suggérer le mouvement en déformant l'environnement théoriquement fixe, tel que l'illustre la figure 2.2.

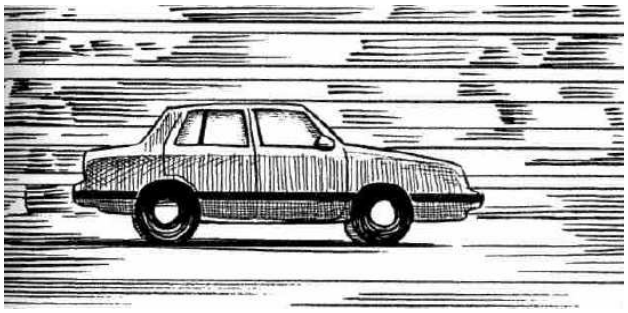


FIGURE 2.2 – *Le mouvement peut être suggéré en déformant ce qui est fixe, c'est-à-dire le décor environnant. Image extraite de [Mcc94].*

Cutting étudie ensuite ces différentes techniques au filtre de 4 critères qui sont la capacité d'évocation, la clarté du sujet, la direction du mouvement et la précision du mouvement. Faisant ressortir les avantages et les inconvénients de chaque méthode, il met en évidence que, d'un point de vue objectif, les lignes de mouvement forment l'indicateur qui remplit le mieux ces 4 critères. D'un point de vue strictement scientifique, nous pouvons donc nous appuyer sur Cutting pour conclure que l'utilisation de flèches, lignes et autres techniques picturales apparentées est la meilleure manière de représenter un mouvement dans une image.

Dans le domaine du cinéma, les auteurs de storyboards ne s'y trompent pas puisque les flèches forment un des éléments essentiels du scénarimage. Que ce soit pour représenter le mouvement d'un personnage ou de la caméra, ce déplacement est explicité à l'aide de flèches qui forment un véritable code graphique dans la discipline. La figure 2.3 en fournit deux exemples.

De même, la représentation d'un itinéraire sur une carte se fait par le dessin d'une trajectoire. Nous pourrions multiplier les exemples de ce type pour confirmer le constat de Cutting : la représentation du mouvement la plus courante et la plus explicite consiste à ajouter une information iconique à une image sous la forme de flèche.

L'étude de Cutting est cependant restreinte au cas d'une image fixe, ce qui occulte une grande partie de ce qui forme la grammaire de la Bande-Dessinée.

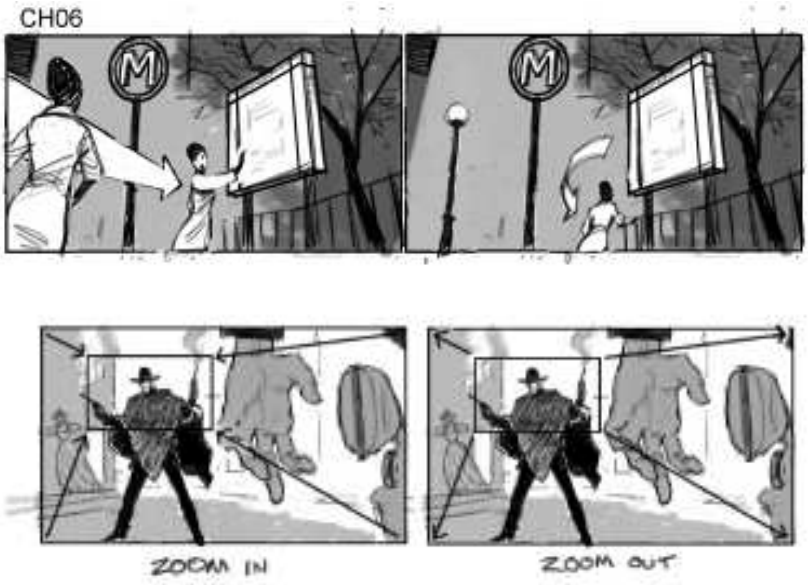


FIGURE 2.3 – *Dans un storyboard, mouvements de personnages et de caméra sont explicités à l'aide de flèches, indicateur le plus explicite du déplacement*

2.2 Le mouvement par interpolation d'images

Scott McCloud [Mcc94] a démontré que sggèrer le mouvement **entre les cases** forme l'outil de base d'un auteur. L'expression "entre les cases" décrit le processus naturel du cerveau consistant à infèrer une action continue à partir de deux images successives. L'étude de McCloud aboutit à une classification en 6 types de transition, exposés dans la figure 2.4, qui sont :

1. La transition « De moment à moment »peut se décrire comme deux instantanés successifs séparés par un très court intervalle de temps. L'utilisation succcessive de ce type de transitions donnera au lecteur une impression de lenteur.
2. La transition « D'action à action »se distingue de la précédante par un intervalle de temps plus long entre les deux images, même s'il n'existe aucune limite temporelle précise entre les deux types. Il s'agit de la forme de transition la plus répandue quel que soit le type de BD considéré.
3. La transition « De sujet à sujet »correspond à un changement de point de vue à l'intérieur d'une même scène.
4. La transition « De scène à scène »présente un écart très large en espace ou en temps (voire les deux) entre deux images successives et marque le plus souvent un changement d'unité narrative plus grand.

5. La transition « De point de vue à point de vue »évacue complètement la notion de temps pour donner différents aperçus d'un même instant et d'un même lieu. Elle est très courante au Japon où l'espace alloué aux auteurs leur permet plus de scènes contemplatives contribuant à créer une atmosphère.
6. La « Solution de continuité »consiste à juxtaposer deux images qui n'ont d'autre rapport entre elles que le fait d'être côte à côte sur une page.

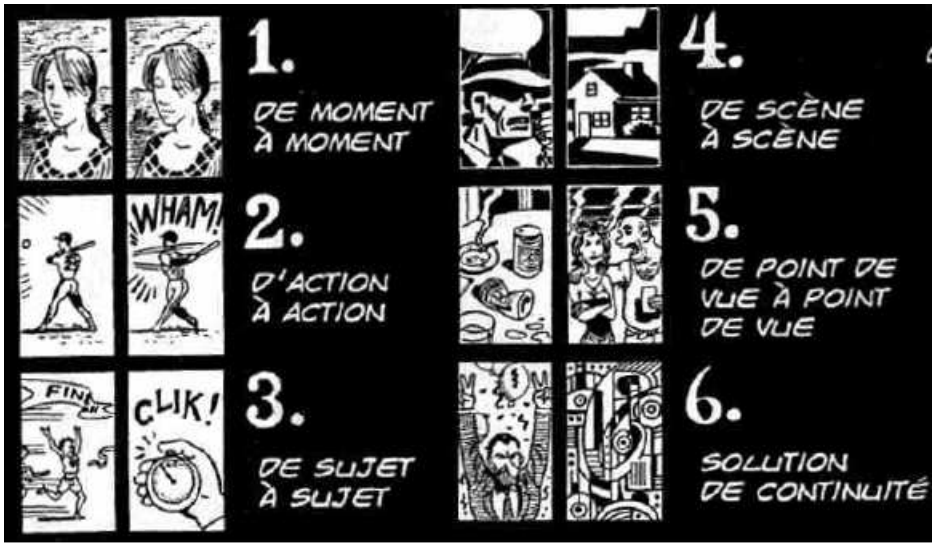


FIGURE 2.4 – *Les 6 types de transition entre images d'après McCloud [Mcc94]*

Un exemple de tous les jours d'utilisation de cette technique concerne les schémas d'assemblage, comme le montre la figure 2.5.

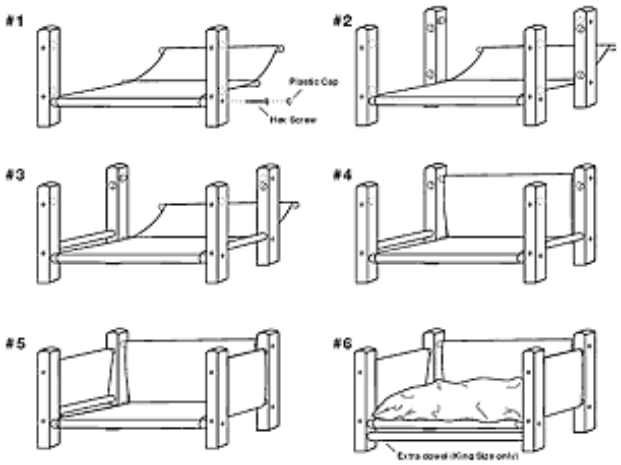


FIGURE 2.5 – *Manuel d'instruction pour le montage d'un futon spécial animal de compagnie, image extraite de http ://www.petfuton.com/*

Le découpage en cases peut servir à créer des effets beaucoup complexes L'exemple de la figure 2.6 est particulièrement fascinant à observer : le positonnement des bulles associé au découpage basé sur le décor de la scène promènent le regard dans une lecture en spirale.

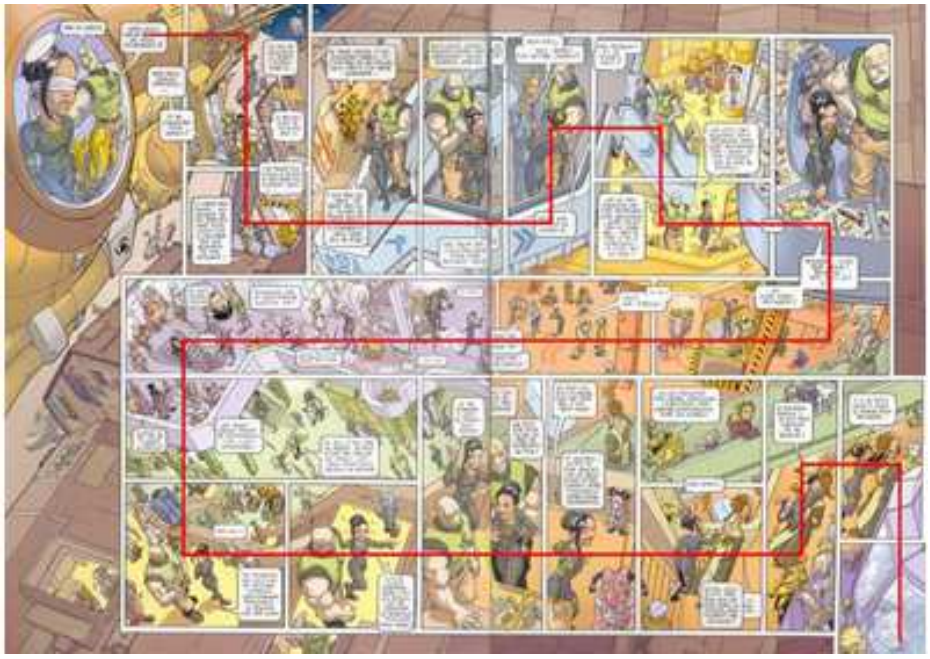


FIGURE 2.6 – *Un extrait de Sillage par Philippe Buchet et Jean-David Morvan : une composition qui promène le regard du lecteur à travers la double-page*

De la même manière, l'utilisation de l'espace entre les cases peut servir à mettre en valeur le décor, créant à nouveau un effet de promenade visuelle.

Si l'on essaye d'aborder de manière scientifique ces diverses transitions, on constate que les variations entre deux images successives sont de deux types : variation en temps, c'est-à-dire l'intervalle de temps qui sépare les deux images ; ou variation en espace, qui correspond au changement de position de la caméra. Comme nous le verrons dans le chapitre suivant, l'usage qu'il a été fait de ces transitions en informatique graphique est relativement restreint, probablement car elles touchent à une perception artistique et subjective des évènements et car elles se prêtent mal à une traduction algorithmique.

FIGURE 2.7 – *un extrait de Promethea par Alan Moore et J.H Williams III. Le décor vu de dessus sert de découpage pour les cases, joignant l'itinéraire physique du déplacement à l'itinéraire visuel de la lecture*

2.3 Mouvement et représentations symboliques

L'invention de la notation musicale moderne est généralement attribuée au moine bénédictin Guido d'Arezzo [1], dans son texte *Micrologus* publié vers 1026. Son système de notes et de portée est progressivement devenu standard.

De la même manière que l'on chercha à transcrire la musique sur papier, de nombreux chorégraphes ont inventé des langages afin d'écrire le mouvement. Les premiers exemples connus datent du Quatroccento. En 1661, Louis XIV crée l'Académie royale de Danse et exige une codification des règles de la danse de cour. En 1700, Raoul-Auger Feuillet publie *Chorégraphie, ou l'art de décrire la danse par caractères, figures et signes démonstratifs* qui fera office de standard pendant près de 150 ans. La figure 2.8 donne un exemple de son système.

En 1887, Friedrich Albert Zorn publie un système de notation à base de pictogrammes. Ses théories feront école et son ouvrage *Grammatik der Tanzkunst* sera traduit en anglais en 1905 et régulièrement réédité.

Le système le plus aboutit actuellement est l'oeuvre de Rudolf Laban qui s'appuie sur le concept de kinésphère. La kinésphère est une

1. Gui d'Arezzo est principalement connu aujourd'hui pour son poème *Ut Queant Laxis / Resonare Fibris* qui, par acrostiche, a fourni les noms des notes de musique actuelles

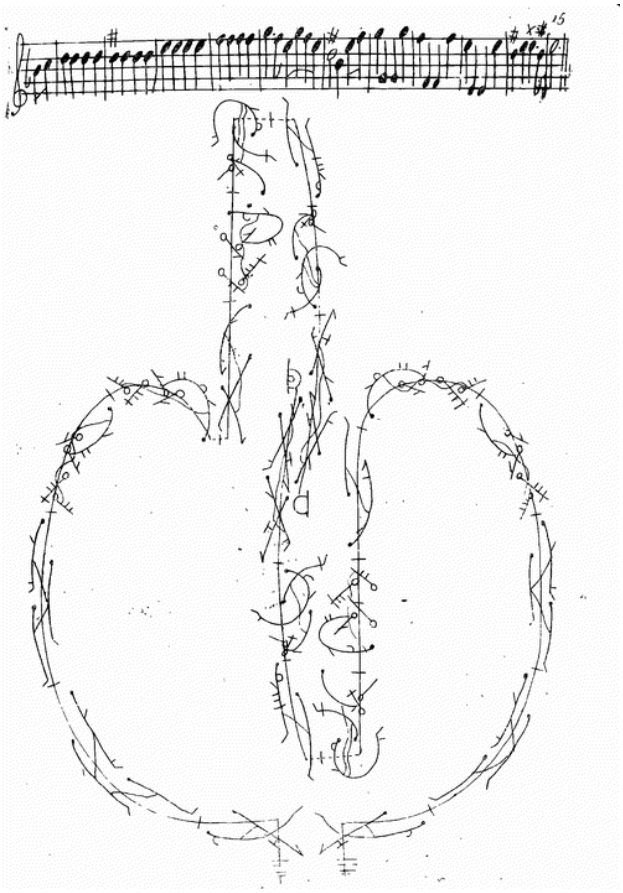


FIGURE 2.8 – *Les contrefaiseurs, Danse de bal pour l'année 1703, par Raoul-Auger Feuillet.*

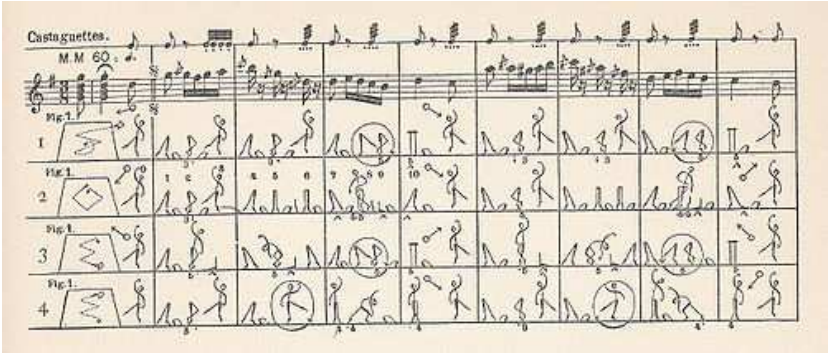


FIGURE 2.9 – *La Cachucha, danse de Fanny Elsser dans le Diable Boiteux, retranscrite avec la notation de Zorn.*

sphère imaginaire centrée sur le centre de gravité du corps et englobant ce dernier. Il distingue ensuite six catégories de mouvements : le corps (qu'est-ce qui bouge et comment), l'espace (où va le mouvement), l'énergie (comment le mouvement est-il exécuté), la forme (quels sont les différents chemins empruntés par le mouvement), le phrasé (à quel rythme s'effectue le mouvement) et l'interrelation (comment l'individu en mouvement est-il en relation avec son entourage). Les signes d'écriture sont placés sur une portée verticale qui se lit de bas en haut. La figure 2.10 donne un exemple de son système.

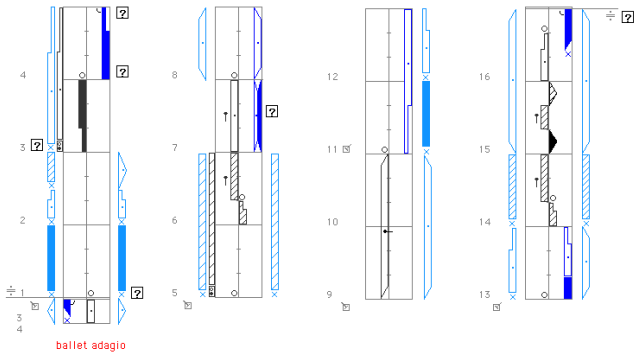


FIGURE 2.10 – *Un exemple de la notation de Rudolf Laban.*

Ce système de notation demande cependant un degré d'expertise élevé et reste apparemment peu répandu dans le monde des danseurs et des chorégraphes. L'explication vient probablement pour partie du fait que le mouvement est une information visuelle et qu'il semble facile d'imaginer un système de transcription, au contraire de la musique. Dès lors, on préfère de beaucoup avoir recours à une notation personnelle d'autant plus claire qu'on en est l'auteur plutôt que de devoir apprendre un système d'apparence complexe.

2.4 Bilan sur la représentation du mouvement

Le choix des techniques utilisées pour représenter graphiquement un mouvement dépend d'abord de l'objectif, du propos de l'image. Les peintres et les auteurs de bande dessinée recherchent une dimension esthétique et une subjectivité qui s'adressent à une lecture de loisir. Les storyboardeurs, les chorégraphes et les auteurs de schémas d'assemblage sont dans la perspective de créer des documents utiles, pour des clients ou des collaborateurs. Le choix de la technique picturale est donc logiquement fonction de l'objectif visé.

Il nous semble que ce choix dépend aussi de l'espace disponible. Les techniques des chorégraphes cherchent généralement à s'aligner avec une portée musicale, ce qui peut représenter beaucoup d'informations à placer dans un espace court. Les schémas d'assemblage doivent tenir dans une feuille au format A4, parfois au détriment du client, circonspect devant un manuel obscur. Cette notion d'espace a aussi pour partie conditionné l'évolution de la bande dessinée. À l'origine, les auteurs remplissaient de gags en trois cases les maigres bandeaux qui leur étaient alloués. Avec la généralisation progressive de l'album, ils ont conquis la page, puis la double page. De ce point de vue, le rendu narratif, ou récit graphique sur ordinateur, n'est alors que le prolongement naturel de cette conquête.

Visualisation du mouvement en image de synthèse

3

On distingue dans la création d'images virtuelles deux grands courants. Le rendu réaliste vise la reproduction exacte de la réalité, c'est-à-dire la génération d'images de synthèse indiscernables d'une photographie. Cet objectif, véritable graal des chercheurs du domaine, semble aujourd'hui atteint aux yeux du grand public [1]. Le rendu non photoréaliste (NPR), qui s'est tout d'abord défini par opposition, s'inspire des techniques d'illustrateurs et d'artistes pour créer des images convoyant un certain degré d'abstraction.

La visualisation scientifique, dont l'objectif final est aussi la production d'images de synthèse, représente curieusement un domaine à part entière distinct de la synthèse d'images. Dans son *Invitation à discuter de la dépiction par ordinateur* [Duro2], Frédo Durand distingue ces deux disciplines en ce que la visualisation s'attache à représenter méthaphoriquement des sujets qui ne sont pas par essence visuels. Suivant cette définition, à quelle discipline se rattache alors la représentation du mouvement dans une image ?

Le mouvement est un élément visuel. Cependant, sa visualisation dans une image, par essence statique, relève de la métaphore. En fait, cette problématique se trouve à cheval entre les deux disciplines :

- Décrire en image un mouvement est une des problématiques majeures de la visualisation scientifique. Elle est la question centrale de tous les travaux en visualisation de flots et de champs de vecteurs.
- Décrire en image un mouvement est une question qui s'est posée de manière ponctuelle en informatique graphique, dans le cas de données de capture de mouvement ou de vidéos par exemple. Cette problématique n'a cependant pas encore fait l'objet d'une approche générale ou systémique.

1. Petite précision : je n'affirme aucunement que la recherche en rendu réaliste est close, mais simplement qu'elle a abouti à des résultats qui confondent aujourd'hui l'oeil du profane

Comme nous allons le développer dans les deux sections suivantes, la visualisation scientifique s'est majoritairement restreinte à la dépiction du mouvement dans une seule image, au moyen d'une représentation vectorielle plus ou moins dense selon les techniques. La synthèse d'images a fait quant à elle plusieurs incursions dans la décomposition en plusieurs images, mais le plus souvent dans le cadre de problématiques et d'applications très spécifiques.

3.1 Travaux précédents en Visualisation Scientifique

La définition exacte du mouvement est une variation de position en fonction du temps. L'étude des travaux de visualisation scientifique présente donc un intéressant paradoxe : un champ de vecteur ou un flot décrit sans aucun doute un mouvement. Pourtant, dans la forme simple des flots stables, le temps n'est pas une variable. Le mouvement est en fait représenté par la vitesse, c'est-à-dire le rapport de la variation de position au temps.

Les flots instables, c'est à dire les flots ou la vitesse en un point peut varier au cours du temps, introduisent une variable temporelle et semblent donc plus proches de notre problématique initiale. Cependant, la visualisation de flots instables est très souvent construite par extension des techniques utilisées pour les flots stables. Cette observation motive notre choix de restreindre cette présentation des travaux antérieurs aux champ de vecteurs dits statiques.

L'affichage compréhensible et analysable de champs de vecteurs est une question qui a vu l'émergence de nombreuses méthodes que l'on peut classifier en cinq catégories :

1. La visualisation directe consiste à utiliser des primitives graphiques simples, telles que le code de couleurs ou des flèches localisées. Elle soulève des problèmes d'échelles car elle peut être à la fois trop détaillée au niveau d'observation global tout en manquant de précision dans les parties les plus fines.
2. La visualisation géométrique est le domaine des *streamlines*, *streaklines* et autres *pathlines*, qui sont toutes des variations autour de l'idée de lancer une particule dans le champ de vecteurs et de suivre son évolution. Comme dans le cas de la visualisation directe, cette méthode peut poser des problèmes de densité de primitives et il n'est pas rare que l'on compare un ensemble de *streamlines* à une soupe de spaghetti.

3. La visualisation à base de textures utilise une image bruitée qui est déformée en fonction du champ de vecteurs sous-jacents. Très détaillée dans le cas 2D, cette technique se révèle délicate à étendre dans le cas 3D.
4. La visualisation structurelle cherche à mettre en évidence la topologie du champ de vecteurs en calculant des points critiques. Cette approche, qui met l'accent sur les singularités du champ, est souvent complétée par un autre rendu de type texture pour afficher son apparence globale.
5. L'idée de la visualisation par partitionnement est de segmenter le champ en parties semblables par le biais d'une mesure de similarité. Ces méthodes s'inspirent, dans l'esprit aussi bien que dans la méthode, des techniques de segmentation d'images et de vidéos.

Avant de passer en revue l'ensemble de ces techniques, nous allons donner quelques précisions mathématiques sur les champs de vecteurs.

3.1.1 Notions de mathématiques des flots

Un champ de vecteurs est une application $\mathbf{V}$ qui, à tout point de l'espace x, associe un vecteur $\mathbf{V}(x)$. On parle alors de flot stable ou de champ statique. Si on ajoute une composante temporelle, on rentre dans le cadre des champs de vecteurs dynamiques qui se définissent par :

$$\begin{array}{ccc} \mathbf{V} : \mathbb{R} \times \mathbb{R}^n & \rightarrow & \mathbb{R}^n \\ (t; x_1, \cdots, x_n) & \longmapsto & \mathbf{V}(t; x_1, \cdots, x_n) \end{array} \tag{3.1}$$

Une streamline, ou trace de particule, peut se voir de manière imagée comme une particule projetée dans le champ de vecteurs et dont on suivrait la trajectoire. Plus formellement, il s'agit d'une courbe partout tangente au champ de vecteurs, *i.e.* une fonction c qui satisfait l'équation différentielle suivante :

$$c \left\{ \begin{array}{ccc} \mathbb{R} & \rightarrow & \mathbb{R}^n \\ s & \longmapsto & c(s) \end{array} \right. / \forall s, \frac{\partial c}{\partial s} = \mathbf{V}(c(s)) \tag{3.2}$$

La définition ci-dessus d'une streamline est valable pour un champ statique. Dans le cas de champs dynamiques, on voit apparaître de nouveaux types de lignes :

- La *streamline* se définit comme une photographie d'un flot à un instant donné.
- Une *pathline* est exactement une particle jetée dans le flot, dont on suit l'évolution au cours du temps.

– Une *timeline* joint les positions à un instant t d'un ensemble de particules jetées dans le flot depuis différentes sources.
– Une *streakline* est tracée par un ensemble de particules qui ont en commun d'être passées par un même point au cours du temps. Elle est en quelque sorte le dual de la *timeline*, qui joint des points d'origines différentes pour un temps constant, alors que la *streakline* joint des points temporellement différents pour une position constante.

L'élégance de ces définitions est qu'elles sont toutes équivalentes à notre streamline de départ dans le cas d'un flot stable.

Les points critiques, ou singularités, sont les éléments les plus caractéristiques d'un champ de vecteurs. Ils correspondent aux points x où $V(x) = 0$. Ces singularités peuvent ensuite être classifiées par une étude locale portant sur les valeurs propres de la Jacobienne. Cette classification est illustrée par la figure 3.1.

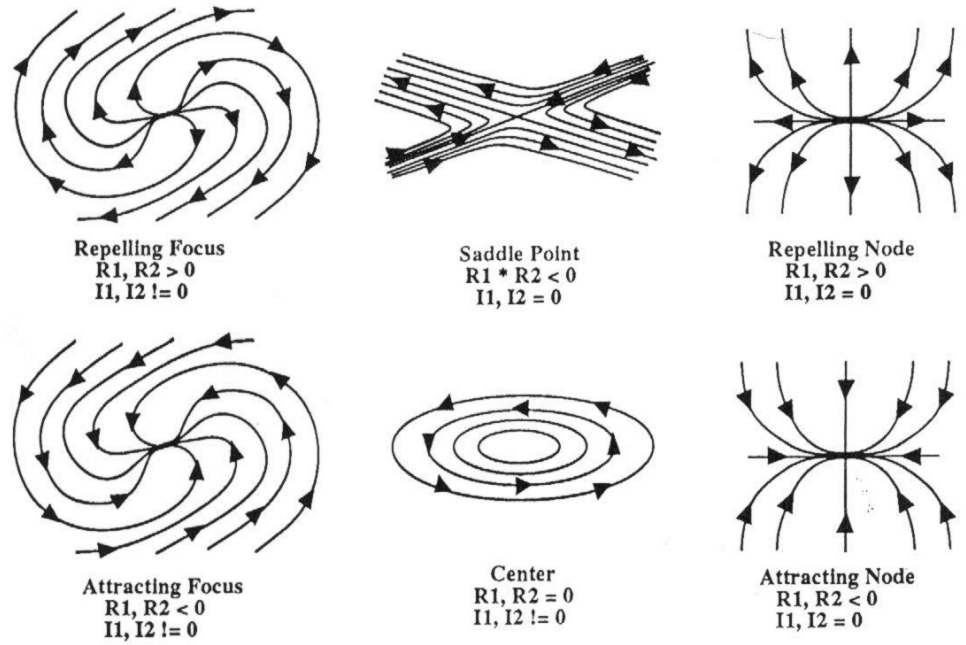


FIGURE 3.1 – *Les types de singularités dans un champ de vecteurs, classés en fonction des valeurs propres de la Jacobienne. R1,I1 et R2,I2 représentent les parties réelles et imaginaires de ces deux valeurs propres.*

3.1.2 Visualisation directe : l'affichage de flèches

La façon la plus directe de visualiser un champ de vecteurs consiste à dessiner des flèches en un certain nombre de points de l'espace. L'orientation d'une flèche est déterminée par la direction du champ en le point considéré. D'autres attributs tels que la vitesse peuvent lui être ajoutés

en jouant sur sa forme, sa taille ou sa couleur. Cette approche, qui a le double mérite d'être peu coûteuse et très simple à implémenter, est encore un outil populaire et disponible par défaut dans un certain nombre d'applications (voir figure 3.2). Elle est de plus très souple en terme de visualisation car elle permet de jouer facilement sur le style des primitives graphiques.

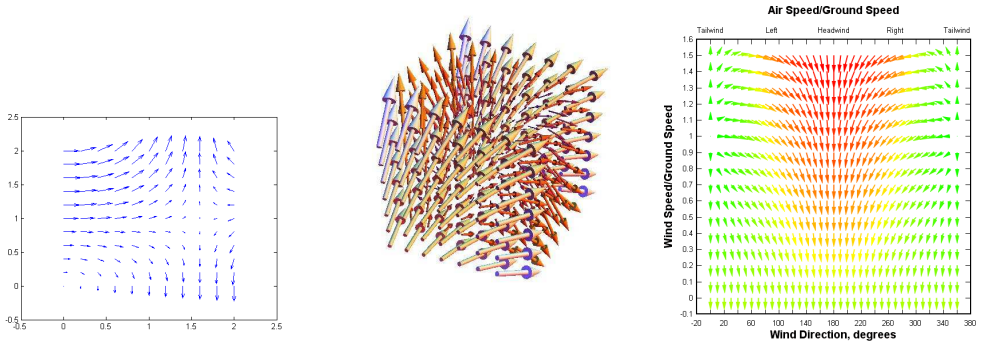


FIGURE 3.2 – *A gauche, une image générée par Matlab, au milieu par Mathematica et à droite avec le logiciel dplot. La visualisation par flèches ponctuelles d'un champ de vecteurs est un outil standard de nombreux logiciels scientifiques.*

L'utilisation de flèches se fait généralement à l'aide d'une grille régulière. Quelques travaux ont cherché à améliorer cette méthode dans le cas de grilles irrégulières ou courbes, citons entre autres le travail de Dovey [Dov95].

En terme de visualisation, cette technique directe se révèle malheureusement rapidement limitée. Une trop faible densité de flèches ne rend pas bien compte des mouvements généraux du flot et inversement, une trop grande densité rend l'image finale difficilement lisible, déjà en 2 dimensions et encore plus en 3D. C'est pourquoi les chercheurs n'ont pas tardé à créer des alternatives à cette technique de visualisation initiale.

3.1.3 Visualisation géométrique à base de streamlines

La primitive de base des techniques de visualisation géométrique est la *streamline*. La question fondamentale qui se pose lors de la visualisation d'un flot à l'aide de streamlines est de savoir où et comment représenter ces lignes. Une approche naïve consiste à partir d'une grille régulière puis de générer une streamline pour chaque point de cette grille. Malheureusement, cette méthode n'apporte aucune garantie quant à la régularité de l'image finale, comme le montre l'image 3.3 de gauche.

Turk et Banks [TB96] sont les premiers auteurs à proposer une méthode pour contrôler la densité des streamlines. Ils proposent une méthode itérative à base de minimisation d'énergie. Leur algorithme au-

torise un certain nombre d'opérations sur les streamlines (déplacement, insertion, suppression, élongation ou rétrécissement, fusion), qui sont répétées jusqu'à ce que les variations de la fonction d'énergie deviennent faibles. Ironiquement, une des images les plus fameuses de leur article est une visualisation des vents au dessus de l'Océanie, où leurs streamlines ont été échantillonées pour servir de support à une visualisation par flèches ponctuelles.

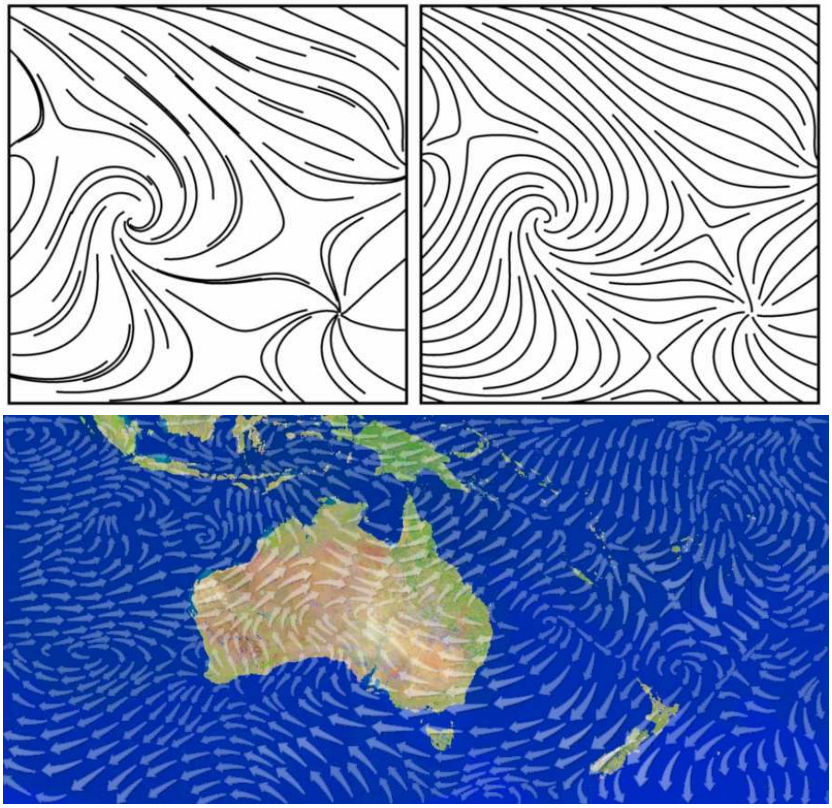


FIGURE 3.3 – *Extrait de [TB96] : en haut à gauche, les streamlines sont calculées en utilisant des points sources disposés selon une grille régulière. En haut à droite, le résulat obtenu avec l'algorithme de Turk et Banks. L'ensemble des streamlines a un aspect beaucoup plus homogène. En bas, les streamlines calculées avec le même algorithme ont été échantillonées et affichées sous la forme d'une suite de flèches courbes qui rendent compte des vents soufflant sur le continent Océanien.*

À leur suite, plusieurs auteurs ont travaillé sur l'optimisation du placement des streamlines. Mentionnons Mebarki *et al.* [MAD05] qui proposent une méthode purement géométrique pour placer les sources des streamlines. Leur algorithme se base sur une heuristique consistant à placer itérativement une streamline en choisissant comme source le point situé le plus loin de toutes les lignes déjà tracées. Le calcul de ce point se fait à l'aide d'une triangulation de Delaunay mise à jour à chaque

nouvelle ligne. L'approche est rapide et produit des résultats esthétiquement satisfaisants. De plus, elle autorise à jouer sur la densité finale des streamlines, permettant un affichage multi-résolution.

Li *et al.*[LHS08] proposent une technique de placement illustratif qui se veut plus qualitative que quantitative. Ils observent que les méthodes précédentes ont cherché à obtenir un placement dense de streamlines ce qui n'est pas nécessairement désirable. Par exemple, une zone large et homogène peut être efficacement réduite à un petit nombre de streamlines, voire une seule, sans perte d'information pour l'observateur. Leur objectif est alors de parvenir au meilleur compromis entre la dépiction du flot et le nombre de streamlines affichées. Pour ce faire, ils définissent une mesure de similarité entre streamlines, puis calculent itérativement des points sources de manière à ce que la streamline résultante soit suffisamment différente des précédentes. Les résultats fournissent une représentation plus abstraite du champ de vecteurs que les méthodes précédentes, dans le sens où les streamlines obtenues ne sont pas uniformément réparties, mais semblent plus concentrées au voisinage des points critiques.

Annen *et al.* [ATR+08] sont aussi dans cette démarche d'abstraction en s'inspirant des techniques de rendu de dessin au trait en rendu non-photoréaliste. Observant que la silhouette d'une forme géométrique donne un bon compromis entre la quantité d'information affichée et la qualité de l'information véhiculée, ils définissent la silhouette d'un champ de vecteurs. Leur méthode, qui s'applique à des champs 3D, se ramène alors à proposer une nouvelle stratégie de placement de streamlines. À la vue de leurs résultats, on peut cependant objecter que, de même que la silhouette n'est pas un indicateur suffisant pour percevoir une forme (les algorithmes de NPR utilisent de nombreux autres types de lignes en complément), la silhouette du champ de vecteur ne fait ressortir qu'une information partielle sur celui-ci.

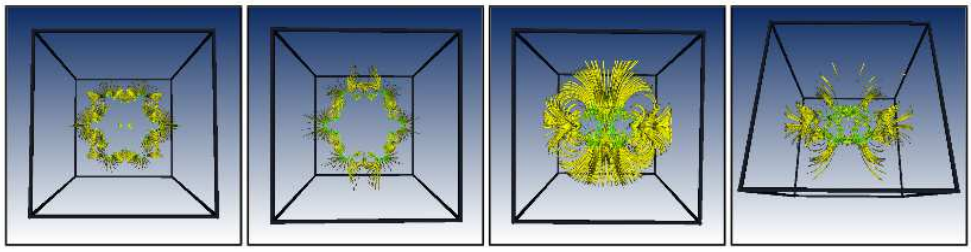


FIGURE 3.4 – *Le teaser de l'article de Annen et al. [ATR+08] montre quatre points de vue différents d'un même champ de vecteurs. Les streamlines correspondant à la silhouette de ce champ sont extraites en fonction de la position de l'observateur.*

Le cas d'un champ de vecteurs 3D, auxquel s'attaquait ce dernier travail, constitue une problématique à part entière car les méthodes dédiées

originalement à la 2D ne s'étendent pas naturellement en 3D. En 3D, un placement homogène de streamlines calculé en utilisant une des méthodes précédemment décrites ne fournira pas nécessairement un placement homogène lorsque celui-ci sera projeté à l'écran pour créer l'image finale.

La visualisation d'un champ de vecteurs en 3D soulève deux problèmes nouveaux :

1. Obtenir un ensemble de streamlines suffisamment dense pour percevoir l'ensemble du mouvement et suffisamment épars pour éviter les effets « plat de spaghetti ».
2. Afficher les streamlines de manière à percevoir leur position spatiale, en particulier en terme de profondeur.

Ye *et al.* [YKP05] s'attaquent au problème du placement en 3D. Ils commençent par calculer les points critiques du champ de vecteurs puis, une fois la nature de ces points identifiée, y positionnent des modèles prédéterminés. Ces modèles servent alors à positionner des sources pour le calcul de streamlines. L'intuition derrière cette aproche est que c'est aux abords des points critiques que l'on a besoin d'afficher le plus d'informations. Pour venir ensuite combler l'espace vide, les auteurs utilisent un échantillonnage de Poisson pour placer les points sources de lignes complémentaires.

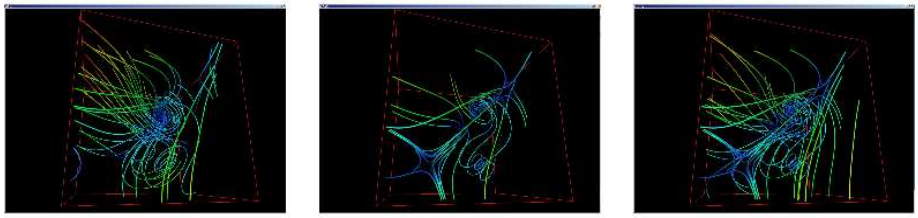


FIGURE 3.5 – *Résulat de l'algorithme de Ye et al.[YKP05]. Des streamlines sont placées autour des singularités, puis l'espace vide est comblé à l'aide d'un échantillonnage de Poisson.*

Zöeckler *et al.* [ZSH96] abordent la partie visuelle du problème : comment afficher un ensemble de streamlines de manière à ce qu'elles soient facilement distinguables et localisables spatialement ? Ils proposent un système d'éclairage des streamlines, montrant comment on peut calculer rapidement les composantes diffuses et spéculaires de lignes.

Mattausch *et al.* [MTHG03] améliorent cette technique en ajoutant des effets de halos, qui améliorent la perception de la profondeur, et une technique de tappering qui permet de facilement localiser les extrêmités des streamlines.

Li et Shen[LwS07], en proposant un placement de streamlines basé image, s'attaquent conjointement à ces deux questions. La sélection des

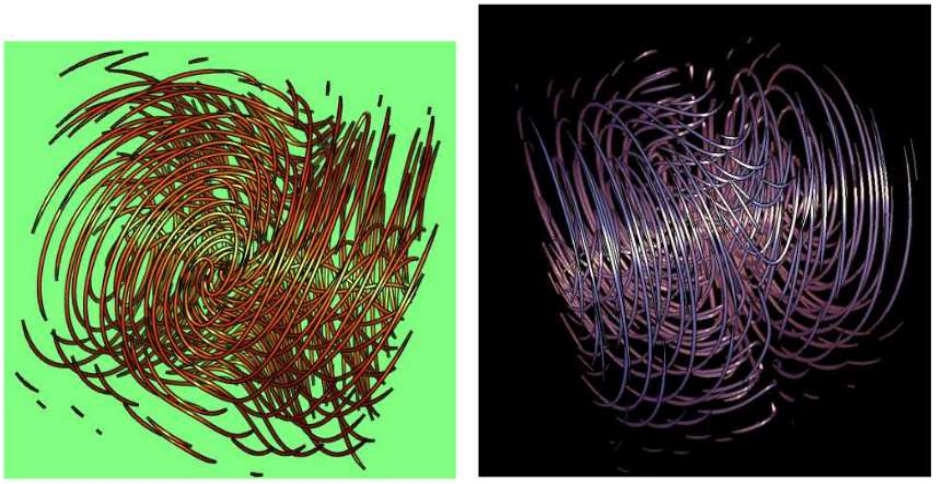


FIGURE 3.6 – *Les Illuminated Streamlines obtenus par Mattausch et al. [MTHG03] permettent de dinstinguer chacune des lignes et de percvoir leur profondeur.*

points sources et le rendu des streamlines sont faits en espace écran. Ceci permet d'obtenir une répartition homogène des streamlines à l'écran et de jouer sur de nombreux effets de rendu.

3.1.4 Visualisation à base de textures

La visualisation par textures d'un champ de vecteurs est un vaste domaine de la littérature de visualition scientifique que nous nous contenterons donc de décrire brièvement.

L'ensemble de ces travaux dérive plus ou moins directement des recherches initiales de Cabral et Leedom [CL93] qui introduisent les *Line Integral Convolution* (LIC) pour la visualisation de champs de vecteurs. La méthode initiale consiste à prendre en entrée un champ de vecteurs discrétisé sur une grille régulière ainsi qu'une texture de bruit de même dimension. La texture est filtrée selon la trajectoire locale des lignes du flot, générant ainsi une visualisation à haute densité du champ de vecteurs.

De nombreux travaux ont cherché à étendre cette méthode pour ajouter des informations de direction et de vitesse, gérer des grilles non-régulières, obtenir des performances temps-réel, ou encore étendre la méthode aux champs 3D.

La visualisation à base de textures rentre en concurrence avec les techniques basées streamlines et on peut se poser la question de leurs performances relatives en terme de visualisation. L'étude utilisateur de Laidlaw *et al.* [LKJ+05], qui évalue initialement six techniques de visualisation différentes, est un bon compte-rendu du match LIC-Streamline.

Les auteurs posent comme critère de performance la rapidité avec

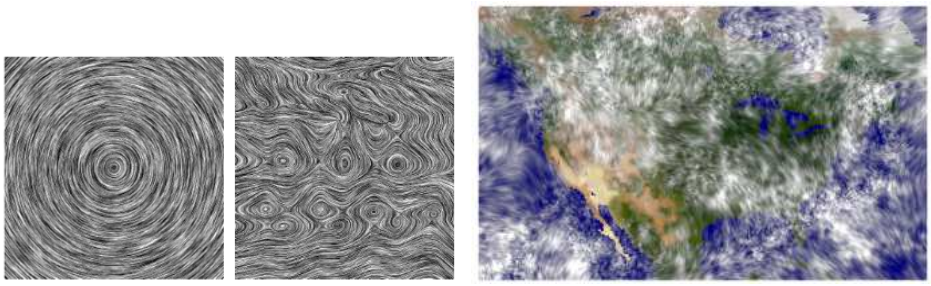


FIGURE 3.7 – *Les LICs sur un bruit blanc pour décrire un champ circulaire et un champ turbulent. A droite, une visualisation des vents au dessus de l'Amérique du Nord*

laquelle un utilisateur parvient à localiser et identifier un point critique, et suivre localement le sens et la direction pris par une particule jetée dans le flot. Les LIC se révèlent très efficaces pour suivre la direction du champ et localiser les points critiques. Cependant, une fois identifiée la direction de la particule, une fois localisée la singularité, les LIC perdent leur avantage au profit des streamlines, qui contiennent une information de sens plus riche et facilitent l'identification du type de singularité. La conclusion est donc que la technique de visualisation la plus efficace est surtout fonction des objectifs de l'utilisateur.

3.1.5 Visualisation structurelle : extraction de la topologie du champ

Les techniques de visualisation structurelle cherchent à expliciter la topologie d'un champ de vecteurs. Ces méthodes s'appuient sur les singularités du champ, lesquelles sont connectées par des lignes particulières nommées « séparatrices ». L'ensemble prend alors la forme d'un graphe tel que représenté par la figure 3.8. Helman et Hesselink [HH91] ont été les premiers à proposer cette représentation pour observer un champ de vecteurs.

Comme dans le cas de la visualisation à base de textures, nous ne rentrerons pas dans les détails de ces recherches. Précisons que la limitation majeure de ces approches est qu'elles sont encore limitées au cas 2D. L'extension à la 3D se heurte aux difficultés à calculer les séparatrices (des surfaces en l'occurence). L'extension à des champs dynamiques souffre d'un problème de définition de la topologie dans la dimension temporelle.

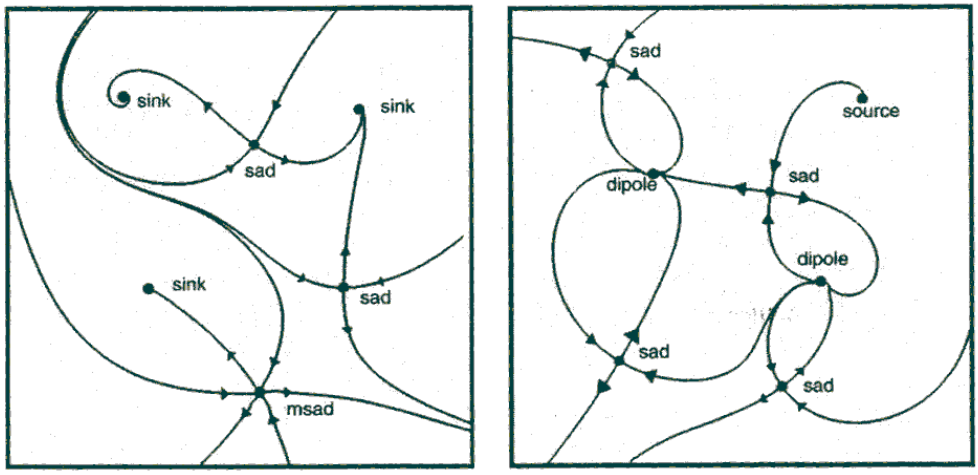


FIGURE 3.8 – *Exemples de graphe points critiques/lignes séparatrices formant la topologie d'un champ de vecteurs.*

3.1.6 Visualisation par partitionnement : segmentation du champ

Mise en évidence par l'état de l'art de Salzbrunn *et al.* [SJWS08], cette nouvelle catégorie regroupe des travaux cherchant à segmenter le champ de vecteurs, cette segmentation servant ensuite de support à la visualisation. Pour qualifier ce pan de recherche comme une sorte de dual de la visualisation structurelle, les auteurs présentent une intéressante analogie avec l'analyse d'images. La visualisation structurelle, qui se concentre sur les singularités et les lignes séparatrices, correspond à l'analyse basée contours des images. Par opposition, la visualisation par partionnement serait l'équivalent de l'analyse basée régions. Bien que cette comparaison relève de la métaphore sans équivalence mathématique, elle permet de donner une intuition de ces travaux.

Les deux premiers articles référencés datent de 1999 et proposent de segmenter de manière hiérarchique un champ de vecteurs discret.

Heckel *et al.* [HWHJ99] présentent une approche descendante où l'ensemble des vecteurs est d'abord considéré comme une entité unique. Les auteurs représentent chaque cluster par une valeur moyenne et définissent une mesure d'erreur entre cette moyenne et le champ initial. Chaque cluster est ensuite découpé hiérarchiquement selon un plan de bisection jusqu'à ce que la mesure d'erreur par rapport au champ initial soit inférieure à une certaine tolérance. Une limitation majeure de leur approche est que chaque cluster est nécessairement convexe, ce qui contraint fortement le partitionnement obtenu.

La même année, Telea et van Wijk [TvW99] proposent une ap-

proche ascendante dans laquelle chaque vecteur est initialement considéré comme une des feuilles d'un arbre en devenir. Ils définissent une mesure de similarité entre clusters et fusionnent itérativement les clusters adjacents ayant la plus forte similarité, ceci jusqu'à ne plus avoir qu'une racine. La visualisation s'effectue ensuite en affichant pour chaque cluster soit une flèche rectiligne, soit une flèche courbe. La figure 3.9 montre leurs résultats dans le cas 3D.

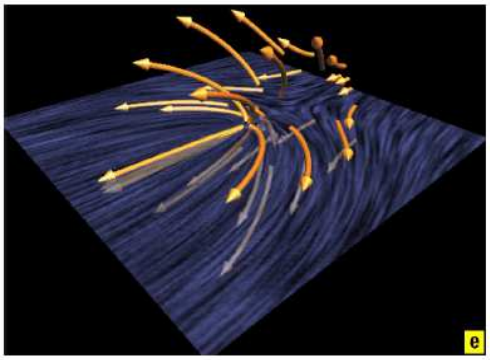


FIGURE 3.9 – *Extrait de [TvW99]*

Telea et van Wijk seront parmi les co-auteurs de Garcke un an plus tard [GPR+] pour proposer un clustering continu basé sur un modèle physique. Pour résumer les clusters obtenus, les auteurs calculent d'abord le squelette géométrique de chaque groupe, puis utilisent le point central de ce squelette comme source pour calculer une streamline. Le résultat final pour un champ 3D est montré figure 3.10.

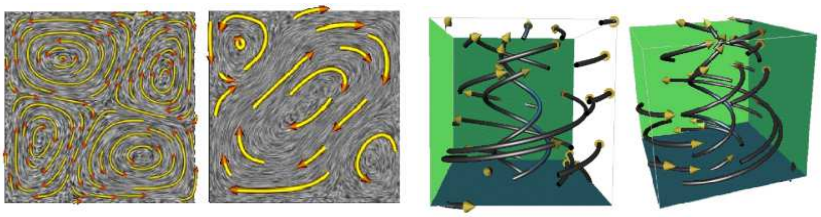


FIGURE 3.10 – *Résultats de la méthode de Garcke et al. [GPR+] pour des champs de vecteurs 2D et 3D.*

3.1.7 Visualisation de champs de vecteurs : un bilan

La dépiction de champs de vecteurs est un domaine majeur de la visualisation scientifique et ce rapide aperçu nous a permis de constater que les techniques proposées étaient nombreuses et, en un sens, complémentaires. Observons que de nombreux algorithmes, tout à fait performants dans le cas 2D, supportent mal une extension immédiate dans

la troisième dimension. D'une manière plus générale, la visualisation de données 3D engendre des problèmes d'ordre visuel et perceptuel pour lesquel l'intégration de solutions dans la boucle de visualisation ne fait pas encore l'objet de consensus. Par exemple dans le cas de streamlines, faut-il appliquer une stratégie de placement purement 3D pour ensuite essayer d'afficher au mieux ces lignes, ou faut-il prendre en compte la projection à l'écran de la streamline résultante lors du placement, quitte à passer à coté d'informations 3D ?

Un tel débat fait inévitablement écho avec ceux qui animent la communauté du rendu non-photoréaliste sur l'opportunité d'appliquer un algorithme en espace objet ou en espace écran. Ce sont ces questions, et les réponses que leur apporte la synthèse d'images, que nous allons maintenant passer en revue.

3.2 Travaux précédents en Informatique Graphique

Contrairement à la visualisation scientifique, les travaux de synthèse d'images se distinguent d'abord par la variété des données qu'ils considèrent en entrée, et dont la nature définit le plus souvent l'objectif final et l'application. Nous distinguerons les données issues de vidéos, les animations 3D et enfin des exemples spécifiques et pas uniquement géométriques.

3.2.1 Visualisation de vidéos

Parler de vidéos ne consiste pas à parler de données 2D+t. La grande spécificité de cette problématique est au contraire que l'information d'entrée est déjà tronquée d'une dimension (puisqu'une vidéo est déja la projection de données 3D+t), ce qui n'est pas sans soulever de nombreuses ambiguïtés dues à cette perte d'information.

Dony *et al.* [DMR05] présentent l'idée du storyboarding inverse et proposent un pipeline pour générer une image qui résume une séquence vidéo. Goldman *et al.* [GCSS06] reprennent cette idée en se basant sur une étude des différentes techniques de storyboarding (voir figure 3.11). Ils proposent de plus d'utiliser les flèches de mouvement calculées comme interface pour sélectionner un instant particulier de la séquence. Ces algorithmes sont limités par la perte d'information sur la profondeur liée à l'utilisation d'une vidéo comme donnée d'entrée.

Cette idée de manipulation a été reprise et amplifiée par Dragicevic

FIGURE 3.11 – *Storyboard résumant un segment de vidéo, généré par la méthode de Goldman et al.[GCSS06]*

et al. [DRB+08]. Les résultats obtenus offrent une interface intuitive pour explorer la dimension temporelle.

La Palette Plus (figure 3.12), qui permet d'enrichir le contenu d'une retransmission sportive au moyen d'annotations incrustées dans l'image, pose des problèmes similaires. L'information d'entrée est cependant plus riche car le système s'appuie sur une reconstruction 3D effectuée grâce à une dizaine de caméras placées autour du terrain. De manière similaire, le système LucentVision [POJC01] permet de visualiser la trajectoire des joueurs et des balles lors d'une rencontre de tennis, fournissant un outil d'analyse puissant pour arbitres, spectateurs, journalistes, joueurs et entraineurs.

FIGURE 3.12 – *Deux exemples d'applications commerciales : La Palette Plus pour les rencontres de football et le système LucentVision pour le tennis*

3.2.2 Visualisation statique d'animations

À notre connaissance, le premier article en synthèse d'images qui s'est intéressé à la représentation du mouvement dans une image est celui de Masuch *et al.*[MSS99]. Les auteurs présentent une méthode qui, partant de plusieurs positions clefs, calcule et affiche des lignes d'actions venant enrichir l'image d'un sujet en mouvement.

Kawagishi *et al.* [KHK03] modélisent différentes techniques de dessinateur pour représenter le mouvement dans une image fixe. Leur mé-

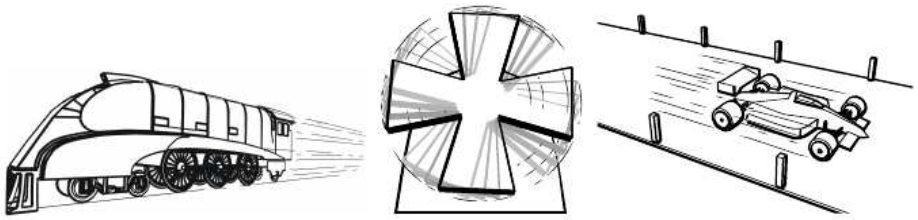


FIGURE 3.13 – *Lignes de vitesse obtenues avec l'algorithme de Masuch et al.[MSS99]*

thode est semi-automatique et permet de générer des lignes de vitesse, une réplication de contours et des déformations dans la direction du mouvement.

FIGURE 3.14 – *Représentation du mouvement par lignes de vitesse, réplication de contours et déformation, issues des travaux Kawagishi et al.[KHK03]*

Les travaux de Nienhaus et Döllner [ND05] sont allés plus loin en travaillant à partir du graphe de scène et en créant un graphe d'évènements, offrant une grande souplesse pour la génération de l'illustration finale. Une telle méthode suppose que l'on a accès à une information de structure sur la scène, ce qui n'est pas nécessairement le cas dans toutes les applications.

Plusieurs travaux se sont concentrés sur le cas spécifique de données issues de capture de mouvement. Ainsi, Assa *et al.* [ACCO05] mettent en place une stratégie pour sélectionner dans l'animation les positions clefs les plus pertinentes. Leur algorithme se fonde sur une réduction dimensionnelle de l'espace des positions/orientations/vitesses de toutes les articulations. Ayant projeté ces données dans un espace réduit (typiquement entre 5 et 9 dimensions), les auteurs en extraient un ensemble d'extrema locaux décrivant les positions extrêmes du mouvement. La juxtaposition des poses sélectionnées rend alors compte fidèlement de l'animation originelle. Lu et Shen [LS08] ont repris cette stratégie dans le cadre de données volumiques. Associée à un slider temporel, leur technique permet une visualisation interactive de la masse de données initiales.

Bouvier-Zappa *et al.*[BZOP07] proposent une approche complémentaire pour résoudre le même problème. Les auteurs présentent une utilisation combinée de flèches de mouvement, ondes de bruit et suivi stroboscopique pour résumer une animation par une image fixe. Le système,

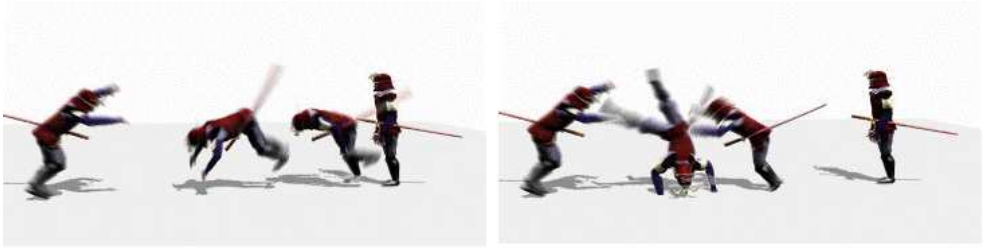


FIGURE 3.15 – *A gauche, un échantillonage régulier du temps peut faire rater des moments importants de l'animation ; à droite, l'algorithme de Assa et al.[ACCO05] extrait les poses les plus significatives.*

partiellement manuel, permet à un utilisateur de sélectionner une pose particulière puis d'afficher un ensemble d'indices visuels dont le rendu s'adapte en fonction du point de vue. La méthode s'appuie de manière extensive sur la hiérarchie du squelette, ce qui rend sa généralisation non-triviale.

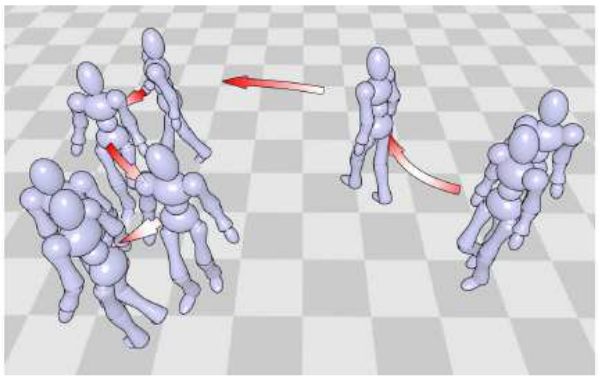


FIGURE 3.16 – *Bouvier-Zappa et al.[BZOP07] résument une animation issue de capture de mouvement par la combinaison de flèches et d'extraction de poses clefs.*

3.2.3 Cartographie, tutoriels : une inclassable liste de données

Depuis quelques années paraissent de nombreuses méthodes dédiées à des applications spécifiques, ayant des données d'entrées originales et proposant des problématiques rarement abordées. Cependant, on trouve derrière tous ces travaux une trame commune : la visualisation en image d'une information de nature temporelle. Petit tour d'horizon presque chronologique :

Maneesh Agrawala a consacré l'ensemble de sa thèse à la visualisation d'itinéraires, ses travaux étant synthétisés dans un article publié avec Chris Stolte à Siggraph 2001 [AS01]. Observant que la plupart des logiciels standards dans la représentation d'itinéraires souffrent d'un

manque d'abstraction dans le rendu final, il a proposé une méthode visant à reproduire les tracés effectués par un humain, qui mettent naturellement en avant les informations pertinentes telles que les bifurcations ou les distances relatives. Sa méthode se décompose en plusieurs sous-problèmes : simplification de la géométrie de l'itinéraire, respect de la topologie, annotation et enrichisement contextuel.

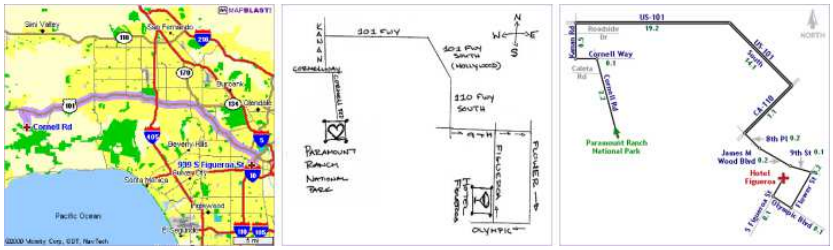


FIGURE 3.17 – *A gauche, une carte obtenue par un logiciel standard, au milieu une carte dessinée manuellement, à droite le résulat de la méthode d'Agrawala et Stolte [AS01]. Leur algorithme émule l'abstraction et la simplification produites naturellement par un dessinateur pour obtenir un rendu d'itinéraire dans lequel les informations pertinentes sont mises en évidence.*

Deux ans plus tard, Agrawala et ses co-auteurs [APH+03] s'attaquent à la création automatique de schéma d'assemblage. La contribution majeure de leurs travaux consiste à reconstituer le processus d'assemblage à partir du modèle final, problématique géométrique qu'ils résolvent à l'aide de notion de visibilité. Le document final (figure 3.18) est un très bel exemple de mouvement entre les cases : un schéma d'assemblage est une bande dessinée !

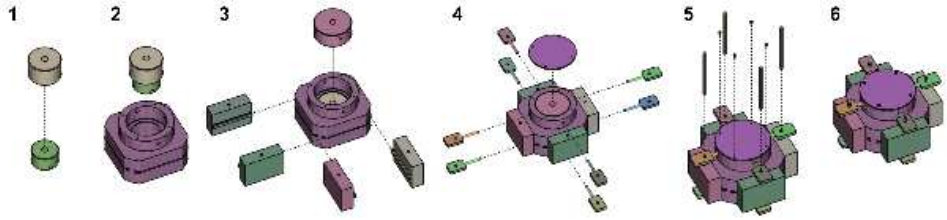


FIGURE 3.18 – *Un schéma d'assemblage généré automatiquement avec la méthode d'Agrawala et al. [APH+03]. Pour chaque partie du modèle, les auteurs considèrent sa boite englobante ; les lignes de constructions sont tracées en connectant les centres des faces adjacentes .*

Shamir et al. [SRL06] se réfèrent directement à la bande dessinée pour résumer une séquence de jeu vidéo. Ils mettent en place un pipeline complet mimant le processus créatif et utilisant les différentes transitions définies par MacCloud. Leur méthode s'appuie sur la définition d'un niveau d'interaction entre les différents éléments. Un type de transition est ensuite affecté en fonction du degré et de la nature de cette interaction.

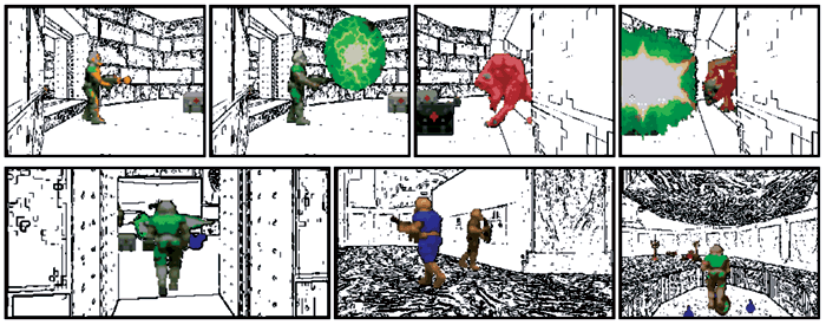


FIGURE 3.19 – *Extrait de [SRL06] : une séquence de Doom résumée sous la forme d'une bande dessinée. Une transition d'action à action illustre le tir, une transition de sujet à sujet focalise l'attention sur le monstre qui reçoit le résultat du tir après une nouvelle transition d'action à action. Plusieurs transitions de sujet à sujet présentent une suite moins mouvementée. Le décor a été rendu de manière très abstraite pour mettre en relief les personnages.*

Grabler *et al.* [GASP08] s'attaquent à la génération automatique de cartes touristiques. Le cœur de leur méthode consiste à identifier à l'aide de techniques de vision les éléments géométriques (immeubles principalement) les plus pertinents pour un touriste. La forme des immeubles est suffisamment simplifiée pour s'adapter au format d'une carte tout en restant identifiable sur place. Il n'y a pas directement de trajectoire dans les cartes obtenues. Cependant, un itinéraire est suggéré par l'agencement des bâtiments mis en évidence.

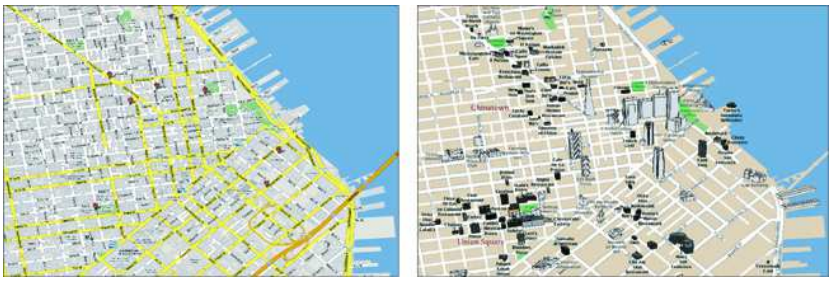


FIGURE 3.20 – *Extrait de [SRL06] : à gauche, une carte basique de San Francisco produite par Microsoft live se révèle difficile à lire ; à droite, le résultat de la méthode de Grabler et al. De nombreux repères visuels pertinents sont automatiquement ajoutés au plan de base*

Actuellement, un type de données populaire émane des logiciels de dessin numérique.

Dans ses travaux de thèse, Sara Su [Su07] et [SPA+09] propose de représenter l'historique d'opérations d'un logiciel de dessin vectoriel sous la forme d'un storyboard. Cet historique sert ensuite d'interface pour la modification et se révèle particulièrement performant pour effectuer des "undos sélectifs".

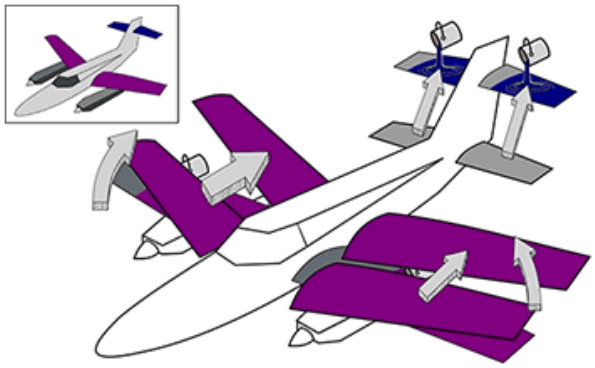


FIGURE 3.21 – *Extrait de [Su07] : les flèches et les icones de pots de peinture décrivent les différentes opérations à effectuer pour obtenir l'illustration finale visible en haut à gauche.*

Grabler *et al.* [GAL+09] proposent de générer automatiquement un tutoriel de manipulation d'images à partir de l'enregistrement d'une séance de travail.

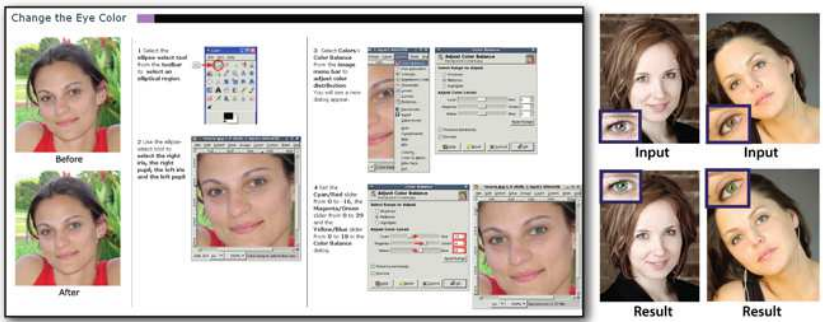


FIGURE 3.22 – *Tutoriel obtenu avec la méthode de Grabler et al. [GAL+09]. Les auteurs proposent de plus un système pour reproduire automatiquement les opérations sur d'autres images.*

Bien loin d'être des *one-shot*, ces travaux s'incrivent dans une tendance qui va probablement aller en s'accentuant. L'efficacité de la communication par images n'est plus à démontrer et il suffit de quelques minutes de recherches pour trouver de nouveaux logiciels, de nouveaux types de données qui pourraient bénéficier de ces approches.

3.2.4 Synthèse d'images : un bilan

Une observation sommaire laisserait penser que les recherches en synthèse d'images se caractérisent par une grande créativité mais manque d'un fondement commun, les approches étant chaque fois très spécifiques. Beaucoup de ces travaux cherchent soit à transposer sur le support

informatique des techniques traditionnelles de dessin technique ou artistique, soit à automatiser des tâches fastidieuses jusqu'à présent dévolues à l'humain.

Il y a cependant un mécanisme commun derrière ces travaux : dans un premier temps, on simplifie l'espace d'informations pour en extraire les éléments les plus pertinents à l'aide d'une métrique spécifique ; dans un deuxième temps, on organise visuellement ces éléments pour générer le document final.

3.3 Discussion comparative

La mise en parallèle Visusalisation Scientifique/ Informatique Graphique fait alors ressortir sur cette question une différence structurelle d'approche : les méthodes de visualisation scientifique sont bien balisées en terme d'objectifs et de définitions mathématiques. Mais la plupart d'entres elles se révèlent difficiles à lire du fait du manque d'abstraction et d'expressivité de l'image finale. Ce sont ces deux buts, abstraction et expressivité, qui sont au coeur de la plupart des algorithmes d'informatique graphique présentés. Cependant, la recherche dans ce secteur s'est développée de manière éparse, le plus souvent pour des applications précises. En conséquence, on ne trouve pas dans la littérature d'analyse globale des méthodes existantes, au contraire de la profusion d'etat de l'art existant en visusalisation.

Parmi les approches présentées en Visusalition Scientifique, celles qui se basent sur une partition des données nous semblent les plus à même de former un pont avec les approches d'informatique graphique, car ce sont elles qui, selon nous, s'insèrent le mieux dans le mécanisme simplification/rendu. C'est donc un algorithme dans l'esprit de cette dernière catégorie que nous allons présenter dans le chapitre suivant, en y ajoutant la part de symbolisme et d'abstraction propres au NPR qui permet d'accentuer le "sens" de l'image affichée.

4 Les acteurs : abstraction et simplification de données animées

La notion de données en mouvement peut couvrir un grand nombre de cas et de représentations différentes, plus ou moins déterminés par l'application. Notre choix dans cette thèse est d'aborder ce type d'informations de manière globale, c'est-à-dire sans connaissance du contexte spécifique. Pour respecter ce soucis de généralité, nous avons opté pour un ensemble discret et fini de particules dont on connait à chaque instant les caractéristiques telles que position et vitesse.

Cette définition nous permet de considérer de la même manière une simulation de foule, une animation par squelette (chaque particule correspondant à une articulation), ou encore une simulation numérique de type Lagrangienne ou Eulérienne (en échantillonnant les données dans ce dernier cas).

Formellement, nous considérons un ensemble de n particules $p_i^t \in \mathbb{R}^{d+1}$, p_i^t représentant l'état de la particule i à l'instant t. L'espace $\mathbb{R}^d$ correspond généralement aux paramètres de position, orientation, vitesse et vitesse angulaire, la dimension supplémentaire décrivant le temps. L'espace final, l'image, étant basiquement équivalent à $\mathbb{R}^2$, notre problème se ramène alors à minimiser la perte d'information engendrée par la projection de $\mathbb{R}^{d+1}$ vers $\mathbb{R}^2$.

Si nous commençons par projeter toutes nos données dans $\mathbb{R}^d$ en supprimant la dimension temporelle, nous nous ramenons alors au cas des champs de vecteurs stables, où le mouvement est représenté par la vitesse. C'est ce cas que nous allons commencer par étudier (section 4.1), en présentant un nouvel algorithme de visualisation par streamlines.

Supprimer la dimension temporelle en projetant nos données dans l'espace est une simplification qui présente de nombreuses limitations. Dans la section 4.2, nous chercherons à adapter notre méthode de visualisation pour prendre en compte cette dimension supplémentaire ô combien particulière.

Finalement, nous poserons dans la section 4.3 un autre regard sur notre problématique, à la lueur des enseignements apportés par nos travaux.

4.1 Placement illustratif de streamlines

Nous considérons en entrée un ensemble de particules $p_i = (x_i, v_i)$, x_i étant le vecteur position et v_i le vecteur vitesse. Nous disposons de plus d'une méthode f qui, pour un point x, nous fournit la vitesse v correspondante. Cette méthode peut être un schéma d'interpolation ou, dans les exemples que nous présenterons, une fonction $v = f(x)$.

Il est courant de visualiser un tel ensemble de données à l'aide de lignes de courant. La figure 4.1, extraite des *Feynman Lectures on Physics* [FLS64], présente les caractéristiques importantes d'un flot autour d'un cylindre.

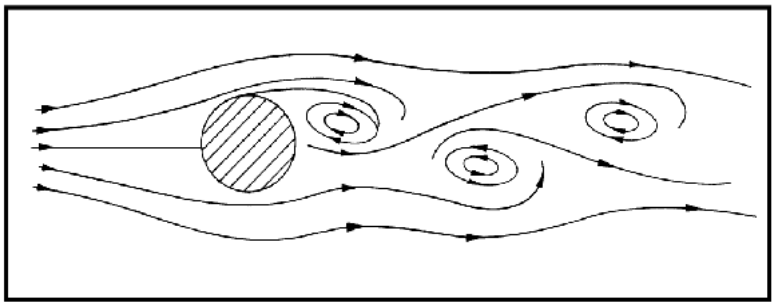


Figure 4.1 – *Illustration d'un flot autour d'un cylindre, extraite de [FLS64].*

Cet exemple classique, dessiné manuellement, est intéressant à étudier par plusieurs aspects :

1. Il n'y a qu'un petit ensemble de lignes représentées, douze pour être exact. Ce faible nombre est cependant suffisant pour décrire le mouvement global du flot. Les singularités sont marquées et l'observateur, même non-expert, infère facilement le mouvement dans les zones laissées vides. Cette observation rentre en contradiction avec les méthodes classiques de visualisation scientifique qui tentent de représenter un ensemble dense de streamlines et privilégient le quantitatif au qualitatif.
2. L'information est hiérarchisée. Plusieurs lignes s'accumulent au niveau des tourbillons, soulignant nettement les singularités du flot. De longues lignes largement espacées résument le mouvement général dans les zones où un supplément de détail serait superflu.
3. Les courbes sont agrémentées de flèches qui ajoutent un sens aux directions décrites par les lignes de flots. À nouveau, cette observa-

tion contredit les résulats exposés traditionnellement en visualisation scientifique. Bien évidemment, ajouter une flèche à une ligne ne relève pas de la prouesse scientifique. Cependant, un ensemble trop dense de lignes limite les informations supplémentaires que ces lignes pourraient porter, en jouant sur leur épaisseur ou en ajoutant des icones par exemple. En affichant trop de lignes, on ne peut en fait dessiner que des lignes.

Ces observations nous motivent à mettre en place une méthode visant non pas à saturer l'espace en fonction d'un seuil de densité mais à représenter au mieux l'ensemble de l'information avec un faible nombre de lignes. Pour ce faire, il nous faut commencer par étudier le champ de vecteurs pour en extraire une information qualitative.

Une première solution consiste à s'appuyer sur la topologie du champ de vecteurs, determinée par les singularités et les séparatrices les reliant. Nous avons rejeté cette solution pour plusieurs raisons :

- L'absence de singularité ne signifie pas qu'il n'y ait rien à visualiser. Elle nous laisserait cependant nu dans une approche connectée à la topologie.
- Le calcul des séparatrices est bien maîtrisé en 2D mais soulève encore de nombreux problèmes théoriques et pratiques lorsque l'on monte en dimension.

Inspirée par l'algorithme de Bezerra *et al.*[BEDT08] pour la stylisation de scènes 3D, la méthode que nous proposons se décompose en deux étapes :

1. Un partitionnement de l'espace basé sur une segmentation du champ de vecteurs, présenté dans la section 4.1.1.
2. Un placement de streamlines illustrant le mouvement du flot avec un petit nombre de lignes, exposé en section 4.1.2.

Nous présenterons nos résultats dans la section 4.1.3 avant de discuter des améliorations possibles en section 4.1.4.

4.1.1 Partitionnement du champ de vecteurs

La première étape de notre méthode consiste à partitionner le champ de vecteurs en groupes homogènes, c'est à dire de réunir les positions ayant une vélocité proche. Les techniques de clustering traditionnellement utilisées pour traiter un champ de vecteurs exhibent une des deux limitations suivantes :

1. Le nombre de groupes est fixé par l'algorithme
2. Les groupes ont une géométrie convexe

Pour contourner ces deux limitations, nous avons choisi de partionner nos données en utilisant l'algorithme du *Mean Shift* [CM02]. Cet algorithme ne fait aucun a priori sur le nombre, ni sur la forme des groupes qui seront créés. Les clusters peuvent avoir une forme tout à fait arbitraire, ce qui les distingue des approches de type octrees ou KD-trees.

Le principe du *Mean Shift*, illustré par la figure 4.2, est le suivant : on considère un ensemble de n points $x_i \in \mathbb{R}^d$ et un point source y_0 quelconque. On construit une série y_j en calculant les moyennes successives des points de données, ponderées par un noyau K :

$$y_{j+1} = \frac{\sum_{i=1}^{n} K(y_j - x_i) x_i}{\sum_{i=1}^{n} K(y_j - x_i)} \tag{4.1}$$

Pour grouper les données x_i, on commence par positionner un point source y_0^i en chaque x_i et on calcule les séries $y_j^i \forall i$. Deux points x_{i_1} et x_{i_2} appartiennent alors au même groupe si et seulement si :

$$\lim_{j \to +\infty} y_j^{i_1} = \lim_{j \to +\infty} y_j^{i_2} \tag{4.2}$$

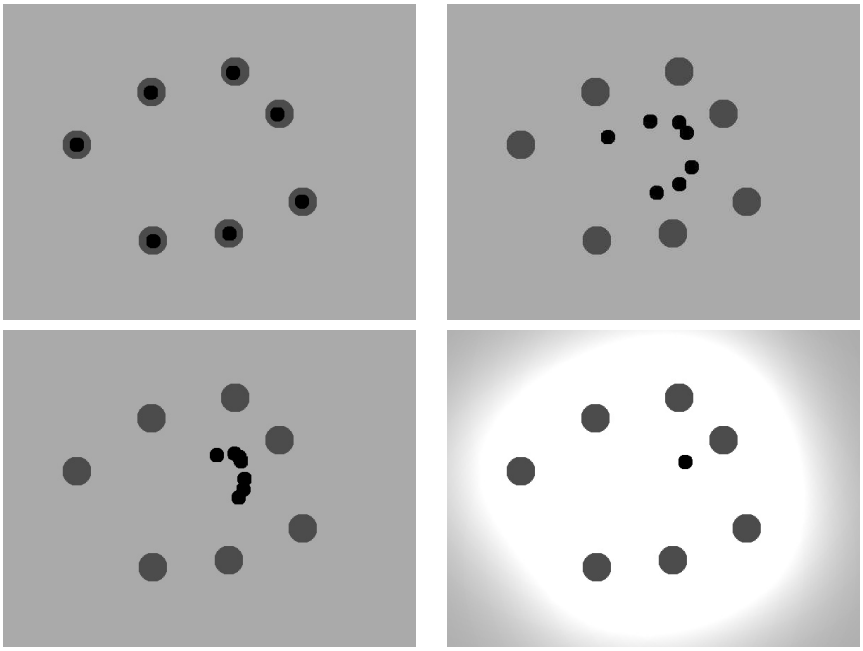


FIGURE 4.2 – *Principe de l'algorithme du Mean Shift : les sources (noir) sont positionnées sur chaque point de donnée (gris). Un groupe est créé lorsque les sources convergent vers un même point.*

Le choix classique pour K est d'employer une gaussienne :

$$K(t) := \frac{1}{(h\sqrt{2\pi})^d} e^{-\frac{1}{2}\|\frac{t}{h}\|^2} \tag{4.3}$$

La variance h de la gaussienne définit l'échelle à laquelle on effectue le partitionnement : une faible valeur créera de nombreux petits groupes tandis qu'une forte valeur réunira l'ensemble des points de données en un seul groupe compact. Aux erreurs d'approximation près, le clustering est entièrement déterminé par cette seule valeur de variance.

Celle-ci nous permet donc de faire évoluer le partionnement en fonction de l'échelle qui intéresse l'observateur. Si celui-ci choisit un rayon large, il regarde en quelque sorte ses données d'un point de vue lointain et perçoit alors de vastes groupes résumant le mouvement général du champ. *A contrario*, un rayon petit, symbolisant un observateur proche, mettra en évidence des mouvements fins qui ne seraient pas pertinents visuellement pour une vue plus lointaine. Précisons que l'interactivité de ce scénario est évidemment soumise au temps de calcul du *Mean Shift* et que celui-ci n'est généralement pas temps-réel. Nous reviendrons sur cette question dans la discussion 4.1.4.

Le seul reproche que nous pourrions formuler vis-a-vis du *Mean Shift* est qu'il ne nous donne pas explicitement les contours de nos groupes. Pour les calculer, nous construisons un diagramme de Voronoi de nos points de données. Le contour des groupes est alors donné par les arêtes qui sont adjacentes à deux cellules dont les centres appartiennent à des groupes différents.

Un problème survient alors. Lors de cette extraction de contour, nous effectuons implicitement une projection des contours des clusters, calculés dans l'espace 4D des paramètres, vers l'espace 2D des positions. Or, un groupe continu dans l'espace des paramètres peut se retrouver fractionné dans l'espace des positions. Ce phénomène se visualise facilement en 2D par la figure 4.3.

Pour résoudre ce problème, nous commençons par établir une liste des cellules de Voronoi appartenant à un groupe donné. Nous sélectionnons une cellule aléatoire et traversons les cellules voisines qui appartiennent au même groupe en les marquant. Si à la fin de ce processus, certaines cellules ne sont pas marquées, nous nous trouvons dans le cas présenté plus haut. Nous créons alors un nouveau groupe et répétons le processus de marquage de cellule, ré-itérant ce processus jusqu'à ce que toutes les cellules du diagramme de Voronoi aient été affectées à un groupe. Nous obtenons au final une partition de nos positions, présentée dans la figure 4.4.

La dernière étape de notre partitionnement consiste à calculer, pour chaque groupe, un vecteur position/vitesse (x, v) représentatif. Pour ce faire, nous sélectionnons aléatoirement un point x_i et répétons la procédure du *Mean Shift* en calculant la série y_i^j jusqu'à convergence. Cependant, nous calculons les moyennes pondérées en prenant en compte uni-

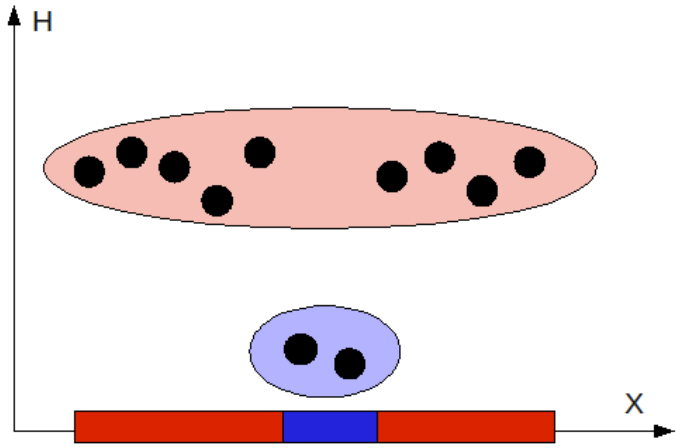


FIGURE 4.3 – *Le clustering est calculé dans l'espace des paramètres (X, H) sur des points échantillonnant régulièrement X. Lorsque l'on projette les deux groupes rouge et bleu sur la droite X, on obtient trois groupes, le rouge étant scindé en deux.*

quement les x_i appartenant effectivement au groupe i. Ceci nous assure que le vecteur limite y_i^∞ appartient bien au groupe et qu'il est représentatif de celui-ci.

Notons que nous ne pouvons pas ré-utiliser les valeurs limites calculées lors de la segmentation par *Mean Shift* pour deux raisons :

1. Les groupes initiaux peuvent être partagés. Il est alors tout à fait possible que la projection de la moyenne se retrouve dans un autre groupe, supprimant toute représentativité spatiale.
2. Tous les points ont une influence, même faible, sur le point limite. Bien que la différence ne soit pas énorme, nos expériences révèlent tout de même un écart perceptible entre la moyenne locale, calculée uniquement sur la base des points du groupe, et la moyenne globale, influencée par l'ensemble des données. Pour choisir une valeur représentative, il parait plus logique d'utiliser un calcul local.

4.1.2 Placement illustratif de streamlines

Pour chaque groupe i de notre partition, nous disposons d'une valeur moyenne y_∞^i qui se compose d'une position moyenne p_i et d'une vitesse moyenne v_i. Nous utilisons ce couple pour calculer un support qui va nous servir de base pour générer des lignes de flots.

Nous définissons le support d'un groupe i comme étant la droite othogonale à la vitesse moyenne et passant par le point moyen du groupe. Il se calcule de la manière suivante : soit d_i le vecteur orthogonal à v_i et

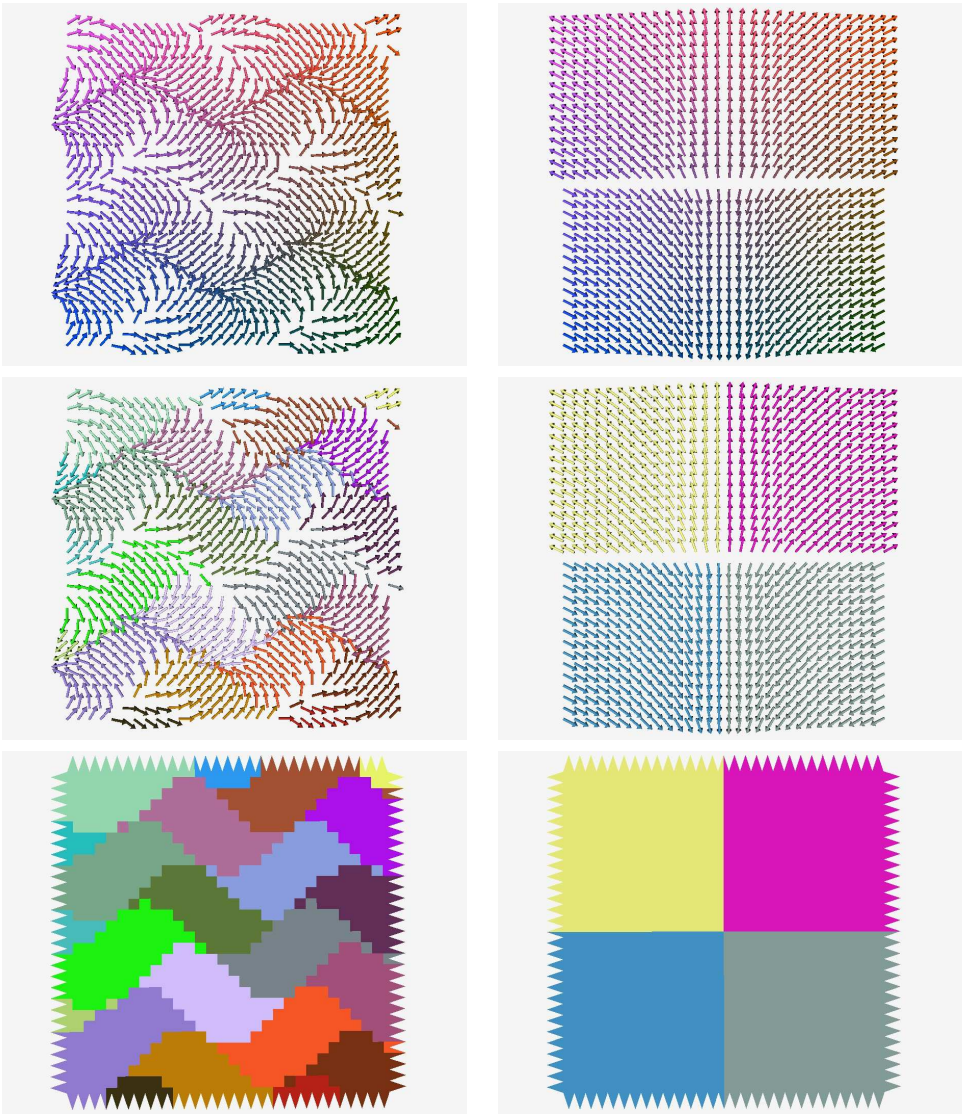


FIGURE 4.4 – *Deux exemples de partionnement de champ de vecteurs avec notre méthode : en haut les champs initiaux ; au milieu le même champ, mais avec une couleur spécifique pour chaque groupe obtenu avec le Mean Shift ; en bas le partionnement obtenu sur les cellules de Voronoi. Notre champ ayant été échantilloné sur une grille régulière, les cellules sont carrées.*

$s_i^j = [a_i^j b_i^j]$ les arêtes qui forment le contour du groupe i. L'intersection k_i entre s_i^j et le rayon $[p_i d_i)$ est donnée par :

$$k_i = p_i + \alpha d_i$$

avec

$$\alpha = -\frac{det(p_i - a_i^j, b_i^j - a_i^j)}{det(d, b_i^j - a_i^j)}$$

On valide cette intersection en vérifiant que le point k_i se trouve bien sur le segment s_i^j. Notre support se compose alors des deux points k_{i_1} et k_{i_2} donnés par :

$$\alpha_1 = \min \alpha_j / \alpha_j < 0$$

et

$$\alpha_2 = \min \alpha_j / \alpha_j > 0$$

Nous échantillonnons ensuite ce support pour obtenir un ensemble de points sources à partir desquels nous faisons partir des streamlines. Plusieurs stratégies d'échantillonnages sont envisageables :

- Échantillonner régulièrement ce support en fonction d'une distance seuil qui détermine la densité de lignes par groupe.
- Utiliser le point moyen pour couper le support en deux et subdiviser régulièrement ces deux parties.
- Echantillonner selon une certaine distance à partir du point moyen.

Dans la pratique, nous n'affichons guère plus de 4 streamlines par groupe, ce qui fait que ces diverses stratégies donnent des résultats sensiblement équivalents.

Une fois établie cette liste de points, nous les utilisons comme source pour calculer des streamlines. Une streamline se conçoit généralement comme une particule sans masse lancée dans dans le champ de vecteurs et dont on suit le parcours. Mathématiquement, cela revient à résoudre une équation différentielle définie ainsi :

$$\frac{dp}{dt} = f(p(t)), p(0) = p_0$$

où $p(t)$ est la position de la particule à l'instant t, f la fonction qui à toute position p fait correspondre la vitesse locale et p_0 la position initiale.

Connaissant $p(t)$, on détermine la nouvelle position après un intervalle de temps Δt par :

$$p(t + \Delta t) = p(t) + \int_t^{t+\Delta t} f(p(u))du$$

En pratique, on approxime cette équation par une suite de points p_k en spécifiant un pas d'intégration h et en utilisant un schéma d'intégration d'Euler ou de Runge-Kutta. Le schéma de Runge-Kutta à l'ordre 2 gagne en précision par rapport à celui d'Euler en introduisant un point intermédiaire p'_k entre p_k et p_{k+1}, soit :

$$p'_k = p_k + \frac{1}{2}hf(p(t))p_{k+1} = p_k + hf(p'_k)$$

L'intégration se fait en marche avant puis en marche arrière en prenant une valeur h positive puis négative. Elle s'arrête lorsque la streamline fait une boucle ou sort de la boite englobante du champ de vecteurs. Il est aussi courant d'arrêter l'intégration lorsque la streamline se retrouve trop proche d'une ligne déja tracée, mais nous n'avons pas implémenté cette condition d'arrêt.

Les méthodes classiques de visualisation par partitionnement restreignent les streamlines à la région qu'elles décrivent. Nous avons mis en place cette stratégie en filtrant les streamlines calculées pour ne conserver que les parties qui se situent dans le groupe du point source. Cependant, les frontières de nos groupes ne sont pas des séparatrices, et il peut arriver qu'une streamline sorte provisoirement de son groupe avant d'y ré-entrer. Si un tel cas se produit, il nous a semblé préférable de conserver la partie intermédiaire afin de favoriser les lignes les plus longues.

Finalement, nous affichons nos streamlines en leur ajoutant des flèches directionnelles. Nous plaçons une première flèche au début de la ligne, ce qui permet de marquer son origine et de faciliter son suivi. Puis, si la longueur de la ligne est supérieure à un certain seuil, nous ajoutons régulièrement quelques flèches supplémentaires. Le résultat final donne une apparence proche de l'exemple de Feynman figure 4.1.

4.1.3 Résultats

La figure 4.5 présente nos résultats pour quatre champs de vecteurs différents. Dans le premier exemple, les streamlines ont été conservées telles quelles car le filtrage par groupe supprimait les formes en champignon sur le coté gauche. Dans les trois autres exemples, seules sont affichées les parties des streamlines qui appartiennent au groupe qu'elles décrivent.

Ces résultats nous laissent sceptiques sur l'intérêt réel de filtrer les lignes de courants. Le principal avantage de ce filtrage est d'empêcher l'accumulation de lignes dans un faible espace, accumulation qui crée un effet de confusion. Mais une telle confusion peut être évitée en stop-

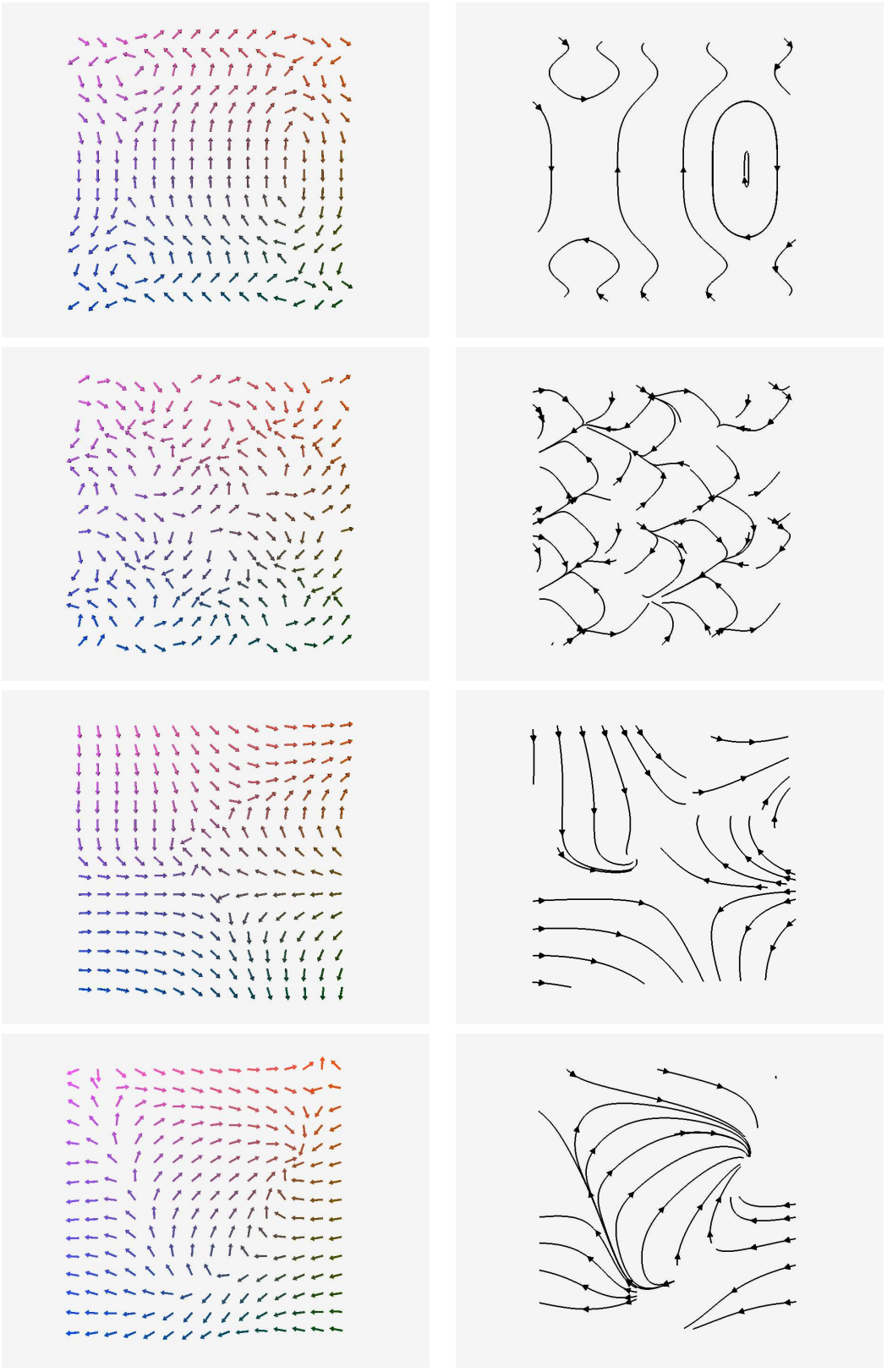


FIGURE 4.5 – *Quatre champs de vecteurs et la visualisation par streamlines calculée par notre méthode.*

pant le calcul d'une ligne lorsque celle-ci s'approche trop de lignes déjà présentes. Cette dernière solution semble donc préférable.

La qualité de la visualisation est variable :

1. Le premier exemple est encourageant. Les caractéristiques du champ ressortent nettement malgré un nombre de lignes très faible. Les deux champignons suggèrent facilement la singularité à l'intérieur, malgré l'absence de dépiction explicite. Toute la partie du milieu se suit facilement à l'aide de seulement deux lignes.
2. Le deuxième exemple est chaotique et met au défi un observateur de suivre une particule au hasard dans le champ. Ce résulat est décevant car le partionnement donne des groupes homogènes et bien détachés.
3. Le troisième exemple est déroutant car les lignes de flots semblent s'amuser à éviter la singularité située au milieu du champ. Nous avons presque l'impression de tracer des lignes la où il ne se passe rien, et de laisser vides les zones plus actives.
4. Le quatrième exemple est réconfortant. Avec un nombre de lignes certes plus important que dans le premier exemple mais malgré tout réduit, l'ensemble semble suffisant pour percevoir le mouvement général du flot et imaginer la trajectoire d'une particule lancée depuis un point quelconque.

L'ensemble de notre méthode a été programmé en C++ de manière non-optimisée. Le temps de calcul du *Mean Shift* écrase toutes les autres parties, mettant plusieurs secondes pour partitionner 250 échantillons... Ceci provient en partie de notre implémentation actuelle en n^2. Des solutions existent pour réduire ce coût à $n \log n$. De plus, notre code est très largement améliorable indépendamment de la complexité théorique de l'algorithme. L'ensemble du calcul des streamlines est temps-réel, c'est à dire qu'une fois le partionnement effectué, nous pouvons faire varier interactivement la densité de streamlines.

4.1.4 Discussion

Chaque étape de notre pipeline soulève des questions et est sujette à discussion. Le premier point à développer concerne l'étape de segmentation.

Le *Mean Shift* s'est révélé au cours de nos expériences d'une maniablité délicate. Les temps de calcul de notre implémentation se révèlent rédhibitoires pour toute application d'envergure même moyenne. Des versions plus rapides existent cependant. En montrant les similitudes entre le *Mean Shift* et la ligne de partage des eaux, Paris et Durand [PD07]

ont proposé un algorithme rapide qui calcule, pour un surcoût faible, une segmentation hiérarchique complète, permettant une exploration interactive des clusters obtenus avec différentes tailles de noyau.

Dans notre méthode, la segmentation calculée ne fait pas ressortir une hiérarchie claire lorsque nous faisons varier le rayon du noyau. A faible rayon, nous obtenons généralement un agrégat de petits clusters peu représentatif. En augmentant la taille du noyau, ces petits clusters fusionnent jusqu'à atteindre un état de stabilité. Cette stabilité se traduit par une acalmie dans le processus de fusion jusqu'à un certain stade où tout se regroupe, ne laissant plus qu'un seul groupe contenant toutes les données. Nous espérions une hiérarchie claire, nous nous retrouvons à devoir déterminer quelle est la bonne échelle pour la segmentation.

À la réflexion, cette absence de hiérarchie n'est pas surprenante. Pour un champ de vecteurs, il existe mathématiquement un clustering défini par la topologie. Le fait de ne pas s'appuyer sur des calculs de la topologie n'implique pas que celle-ci n'est pas présente. Notre objectif initial en utilisant le *Mean Shift* était de pouvoir considérer des structures vectorielles quelconques, sans nécesssaire cohérence du flot. Or, nos expériences se sont faites sur des structures cohérentes, pour lesquelles de nouveaux outils mathématiques, beaucoup plus performants et aux fondations solides, émergent depuis quelques années. Ainsi en est-il de l'exposant de Lyapunov, introduit par Haller [Hal01], qui consiste à mesurer la dispersion entre les trajectoires de particules. Le coût initial du calcul est prohibitif car nécessitant un très grand nombre de particules pour révéler la struture des données. Les travaux de Garth *et al.* [GGTH07] et [GLT+09] proposent des solutions pour alléger ce calcul et l'utiliser pour la visualisation de flots.

Le placement de streamlines résultant de notre partitionnement et de notre calcul de support laisse insatisfait. Nous avons souligné qu'un manque de notre implémentation réside dans l'absence de condition d'arrêt du calcul des lignes lorsqu'elles arrivent à proximité de lignes existantes. Cet ajout apporterait certainement une clarté supérieure à notre rendu final, bien meilleur qu'un filtrage par groupe et plus générique. Le risque demeurera cependant de rater des informations et de dessiner des lignes là où rien ne se produit. Nos expériences révèlent une qualité de dépiction globale du champ plutôt aléatoire, très variable selon le champ considéré en entrée, et donc non satisfaisante d'un point de vue scientifique. En conclusion, une segmentation à base de *Mean Shift* en tant que guide semble ne répondre que partiellement à notre question initiale : comment placer un nombre restreint de lignes décrivant l'ensemble du mouvement du flot ?

Deux pistes nous semblent envisageables :

1. Nous pourrions utiliser une information topologique partielle en détectant les singularités du champ. Cette information à elle seule ne permet sans doute pas de répondre à notre problématique. Par contre, son introduction dans notre pipeline pourrait permettre une combinaison intelligente des clusters et des singularités, améliorant de manière sensible la qualité des résultats.
2. Nous pourrions envisager une approche radicalement différente qui se déroulerait en deux étapes : dans un premier temps, nous saturerions l'espace de streamlines, selon une certaine densité. Dans un deuxième temps, nous utiliserions une méthode de simplification dans l'esprit des techniques utilisées en NPR pour le dessin au trait, par exemple [BTS05] ou [SC08]. Il n'est pas impossible que cette problématique de simplification, qui nécessitera un critère pour déterminer si deux lignes fusionnent, nous ramène vers l'exposant de Lyapunov.

4.2 Résumé d'une animation de particules

Nous augmentons maintenant la dimension de notre problème en considérant des particules en 3D et en conservant la dimension temporelle. Nous utilisons le même pipeline que pour les champs de vecteurs, à quelques adaptations près. Celui-ci est exposé dans la section 4.2.1. Nous présentons et discutons des premiers résultats dans la section 4.2.2.

4.2.1 Algorithme pour des données 3D + temps

L'adaptation du pipeline que nous proposons est résumée par la figure 4.6 :

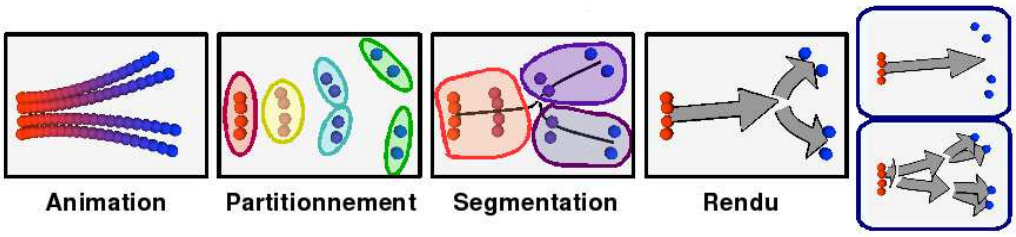


Figure 4.6 – *Aperçu de notre algorithme : nous prenons en entrée un ensemble de particules animées. Les trajectoires sont découpées par l'algorithme de Mean Shift, segmentées, avant d'être affichées sous forme de flèches représentatives du mouvement. En faisant varier l'échelle du clustering, on obtient soit une indication globale du mouvement (en haut à droite), soit un rendu plus détaillé (en bas à droite).*

Nous commençons par traiter indépendamment chaque pas de temps. Pour chaque instant, nous regroupons les positions et les vitesses de nos particules à l'aide du *Mean Shift*. Nous obtenons, pour chaque groupe ainsi créé, la valeur y_i^∞ qui le résume au mieux.

Nous obtenons ensuite la trajectoire d'un groupe en chaînant ces moyennes au cours du temps. Lorsque nous détectons un évènement du type séparation d'un groupe en deux ou fusion de deux groupes, nous stoppons le chaînage et recréons de nouvelles trajectoires pour cette nouvelle topologie. Cette détection produit donc une segmentation temporelle de nos données.

Cet ensemble de trajectoires nous sert de support pour calculer et afficher des flèches de mouvement. La construction d'une flèche s'effectue en considérant trois points successifs p_{t-1}, p_t et p_{t+1} de la trajectoire et $\vec{V}$ la direction de la caméra. Nous utilisons une direction d'*offset* $\vec{b}$, donnée par le produit vectoriel entre la tangente de la trajectoire et le point de vue, soit :

$$\vec{b} = (p_t - p_{t-1}) \otimes \vec{V} + (p_{t+1} - p_t) \otimes \vec{V}$$

Cette direction nous sert à calculer deux courbes *offset* à la trajectoire, dont les points sont donnés par :

$$k_t = p_t \pm w * \vec{b}$$

Le paramètre w définit l'épaisseur locale de la flèche affichée comme une bande de triangles, la prise en compte du point de vue permettant de la maintenir comme faisant face à l'observateur dans un soucis de lisibilité.

4.2.2 Résultats et discussion

L'implémentation actuelle de notre méthode donne des résultats satisfaisants pour plusieus cas basiques, illustrés par la figure 4.7.

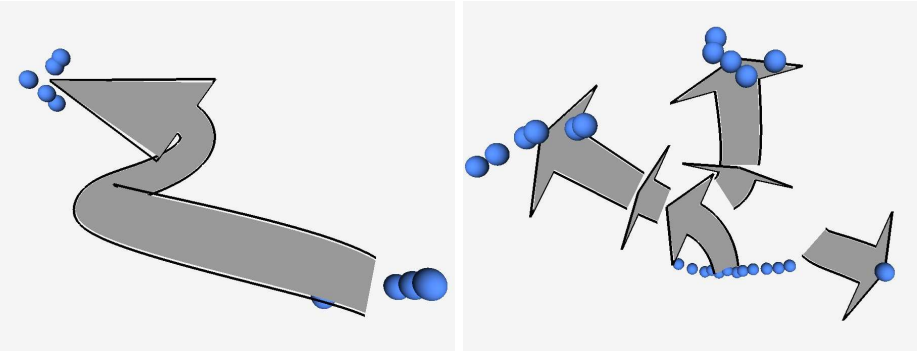


FIGURE 4.7 – *Deux exemples de résultats obtenus avec notre méthode.*

Le mouvement hélicoïdal ressort de manière claire, ainsi que la séparation des particules en plusieurs groupes. Dans ce deuxième cas, on remarque cependant un trou dans la trajectoire affichée de la particule de droite. Ce trou est le résultat d'un des deux artefacts expliqués par la figure 4.8.

Analysons cet exemple : au commencement de l'animation, nos 3 particules ne forment qu'un seul groupe homogène, et leur trajectoire est grosso-modo approximée par la trajectoire de la particule centrale (la 2). Puis la particule 1 diverge lentement vers le nord tandis que les particules 2 et 3 continuent ensemble leur chemin vers le sud. Les 3 particules ne forment encore qu'un seul groupe mais, à cause du moyennage basé sur un noyau gaussien, la particule 1 perd de l'influence sur le calcul de la moyenne. En conséquence, on voit apparaitre une bosse dans la trajectoire moyenne, bosse qui illustre cette perte d'influence. Puis la distance entre la particule 1 et les particules 2 et 3 devient trop grande, un deuxième groupe se forme. L'effet « bosse »s'estompe continuement jusqu'à disparaître lors de la création de ce deuxième groupe, et la trajectoire moyenne des particules 2 et 3 redevient correcte. Le nouveau groupe, qui contient uniquement la particule 1, n'a aucun antécédent, ce qui provoque un trou dans l'affichage des trajectoires.

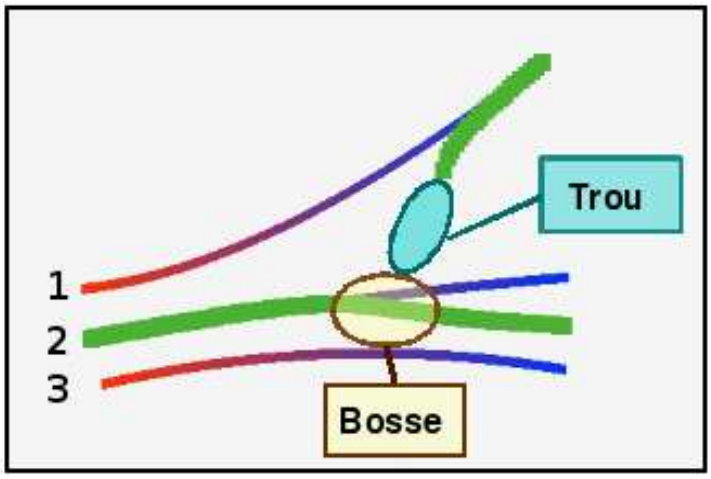


FIGURE 4.8 – *Deux artefacts causés par un chaînage basique : une bosse apparaît lorsque la trajectoire d'une particule commence à diverger de celle des autres ; lorsqu'un deuxième groupe se crée, celui-ci n'a aucun antécédant ce qui produit un trou dans les trajectoires*

Cet algorithme a été initalement conçu avec l'intention de séparer traitement spatial et temporel. L'artefact de trou observé en est la conséquence directe. Il semble donc plus approprié de segmenter l'ensemble espace-temps des données à l'instar de ce qui se fait en traitement de vidéos. À la condition de se doter d'un schéma d'interpolation adéquat, le

calcul des trajectoires pourrait ensuite se faire avec une ou plusieurs pathlines, extension temporelle d'une streamline. Nous limiterions au groupe la portée de la pathline, sauf si cette limitation se révélait mauvaise en terme de visualisation.

Il y a une ironie dans cette discussion due au fait que nous avons d'abord abordé ce problème 3D+temps avant de nous restreindre au cas des champs de vecteurs. Les solutions que nous avons développées dans ce dernier cadre sont alors celles que nous souhaiterions transposer au problème plus général. Cependant, indépendamment de la complexité supérieure induite par l'ensemble espace/temps, est-il souhaitable de réutiliser des outils qui demandent encore validation dans un cas simple ?

Nous pensons qu'une solution rigoureuse et générale développée dans le cas de champs de vecteurs 2D peut servir de base pour aborder le problème supérieur. Encore faut-il que les fondations soient solides !

4.3 Discussion : segmentation temporelle

Dans ce chapitre, nous avons limité notre utilisation des techniques de dépiction du mouvement au cas d'une seule image. Selon nous, le recours à une séquence d'images peut se révéler utile, voire indispensable, dans deux cas :

1. L'affichage de plusieurs positions clefs est plus lisible que l'ajout de primitives graphiques dans une image. Ceci est le cas pour un objet qui se déforme, par exemple qui gonflerait et rétrécirait. Trois images montrant les différentes tailles seront plus claires que des petites lignes dessinées.
2. L'accumulation d'informations dans une image rend celle-ci illisible. Par exemple, le trajet d'une personne revenant sur ses pas se superpose avec son itinéraire initial.

Sans vouloir mettre en équation le travail de découpage d'un auteur de bande dessinée, une perspective de recherche intéressante consisterait à définir des critères qui permettraient de dire qu'une séparation en plusieurs images seraient mieux adaptée à la visualisation de nos données.

La justification de cette séparation peut être dépendante du point de vue, auquel cas le travail en espace écran sera plus adapté. Mais à moins de savoir traiter l'ensemble de points de vue possibles, cette approche rend difficile le pré-traitement, limitant fatalement la quantité de calculs engageables dans cette segmentation. La question du choix entre travail en espace écran et en espace objet étant presque aussi vieille que le rendu et toujours d'actualité, il n'est guère étonnant qu'elle se pose dans le projet d'une bande dessinée dynamique.

Le décor : représentation et visualisation de la scène

5

Nous avons jusqu'à présent traité un système de particules isolées dans un espace vide. Cependant, un certain nombre d'applications ont lieu dans un contexte géométrique qu'il est primordial de mettre en relief, tel qu'illustré par la figure 5.1. Cet exemple illustre l'importance du décor et de sa représentation pour donner du sens à l'information de mouvement qu'il contient.

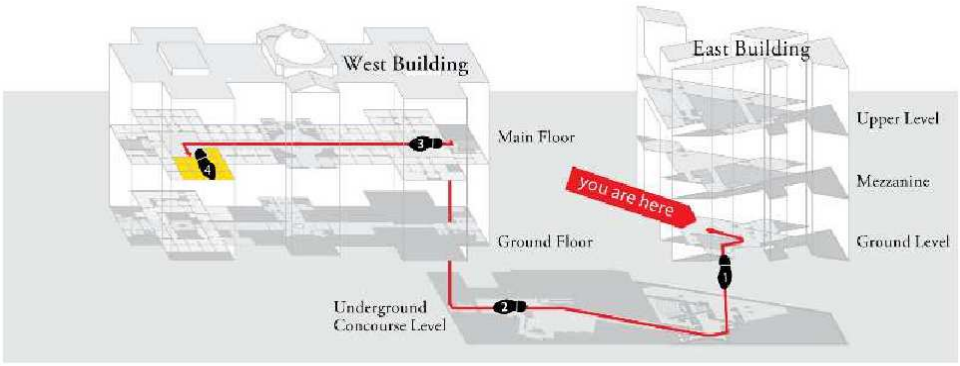


Figure 5.1 – *Museum Map Kiosk extraite de [Tuf97]. L'itinéraire en rouge qui représente l'acteur n'aurait aucun sens si l'illustration ne s'accompagnait pas d'une représentation abstraite des batiments.*

La problématique générale que nous souhaitons développer dans ce chapitre est la suivante : étant donné un décor, représenté par un modèle géométrique, et un acteur se déplaçant dans ce décor, représenté par son itinéraire, quelles techniques de visualisation proposer pour afficher ce couple acteur/décor et la relation qu'ils entretiennent ?

A notre connaissance, Niederauer et al. [NHAH03] ont été les premiers à s'attaquer à une problématique voisine. Leur méthode consiste à décomposer une scène en étages, appelés scénarios dans leur article, pour en obtenir une vue explosée.

Dans ce chapitre, nous considérons que la géométrie dont nous disposons en entrée nous est fournie de manière brute, sous la forme d'un maillage par exemple, sans information de sémantique. Nous considérons le cas d'un acteur, représenté par son itinéraire, se promenant dans ce décor. La méthode de visualisation que nous présentons se décompose en trois étapes :

1. Décomposition de la scène en une structure cellule/portail
2. Segmentation de l'itinéraire en fonction de la structure précédemment calculée
3. Visualisation du couple Acteur/Décor

Notre implémentation actuelle étant exclusivement 2D, 2D sera aussi la présentation de notre algorithme. Nous discuterons des problématiques qui se posent dans le cadre d'une extension à la 3D.

5.1 Segmentation du décor

La décomposition en cellules et portails est généralement utilisée dans le cadre de problèmes de visibilié, l'idée étant qu'une cellule ne peut en voir une autre que par l'intermédiaire d'un portail. Il s'agit donc d'une méthode de pré-processing très courante, dont la mise en oeuvre manuelle peut être suffisamment fastidieuse pour motiver la création d'algorithmes dédiés.

Dans notre contexte, ce sont moins les problèmes de visibilité que la segmentation partiellement sémantique qui motive notre utilisation de cette structure. Pour effectuer cette décomposition, nous utilisons l'algorithme présenté par Haumont *et al.* [HDS03].

Le principe de leur méthode est à la fois simple et élégant. Les auteurs commencent par calculer une carte de distance de la scène. La scène est alors représentée par un relief, la hauteur d'un point étant définie par la distance aux bords (figure 5.2).

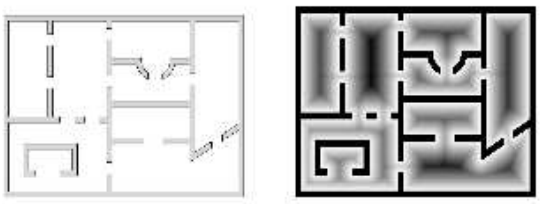


Figure 5.2 – *Carte de distance d'une scène d'un modèle architectural. Les valeurs de distance ont été inversées pour représenter le relief sous la forme de bassins. Image extraite de [HDS03]*

À ce relief, qui n'est finalement rien d'autre qu'une image en niveaux de gris, les auteurs appliquent une ligne de partage des eaux (illustrée figure 5.3). La ligne de partage des eaux est une opération issue de la morphologie mathématique qui consiste à concevoir une image en niveau de gris comme un relief qui serait progressivement rempli d'eau. On a au début autant de bassins que de minima locaux d'altitude. Lorsque le contenu de deux bassins rentrent en contact, les auteurs créent un portail séparant les deux cellules. L'algorithme se termine lorsque l'ensemble du relief se retrouve plongé sous l'eau. La segmentation finale de la scène n'est peut-être pas optimale en terme de calcul de visibilité mais ses avantages sont en contre-partie nombreux : la décomposition obtenue est fortement correlée à la structure de la scène et s'adapte en 3D à un grand nombre de représentations surfaciques ou volumiques.

FIGURE 5.3 – *Différentes étapes de la ligne de partage des eaux. Un portail est créé chaque fois que deux bassins rentrent en contact. Image extraite de [HDS03]*

Cet algorithme nous fournit automatiquement une segmentation sémantiquement pertinente de la scène. La carte de distance utilisée nous permet de plus pour chaque cellule de calculer un centre, soit le point de distance minimale, soit le milieu du squelette dans le cas fréquent où plusieurs points ont cette propriété.

5.2 Segmentation de l'acteur

Une fois obtenue la segmentation de la scène, nous pouvons segmenter de manière simple l'itinéraire de l'acteur. Si on considère raisonnable que l'acteur se déplace de cellule en cellule en utilisant les portails, il nous suffit d'échantilloner l'itinéraire puis, pour chaque segment, de voir s'il intersecte un portail. Nous pouvons alors découper précisément la trajectoire pour assigner à chaque morceau la cellule dans laquelle il se trouve.

5.3 Visualisation du couple Acteur/Décor

Les segmentations de la scène et du trajet de l'acteur nous permettent de proposer deux méthodes de visualisation complémentaires du couple acteur/décor :

1. Une représentation du trajet de l'acteur par déformation du décor.
2. Une abstraction dynamique du décor guidée par un intervalle du trajet de l'acteur.

5.3.1 Visualisation de l'acteur guidée par le décor

Représenter un mouvement dans une image peut se faire en jouant sur l'affichage du décor. Dans le domaine du storyboard, une telle transformation est courante, ainsi que l'illustre la figure 5.4

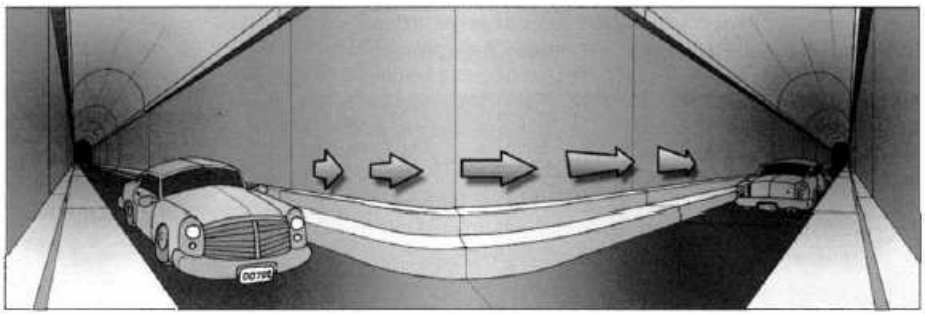


Figure 5.4 – *Dans cet extrait de storyboard, le tunnel est déformé pour montrer la courbe effectuée par la trajectoire de la voiture.*

Le principe de notre visualisation consiste à dérouler l'itinéraire que l'on souhaite observer et à déformer la scène en conséquence.

Pour ce faire, nous traitons la trajectoire de l'acteur comme un squelette et nous utilisons un algorithme de skinning. Nous commençons par échantillonner l'itinéraire. Pour chaque cellule du décor, chaque point de la scène est ensuite mis en correspondance avec le point de l'itinéraire le

plus proche. Dérouler l'itinéraire revient à appliquer une translation et une rotation spécifiques à chaque point de la trajectoire pour que celle-ci forme une ligne droite. La même transformation est appliquée à chaque point de la scène associée, déformant celle-ci de manière appropriée. La figure 5.5 montre la partie de l'itinéraire selectionné et le déroulé associé.

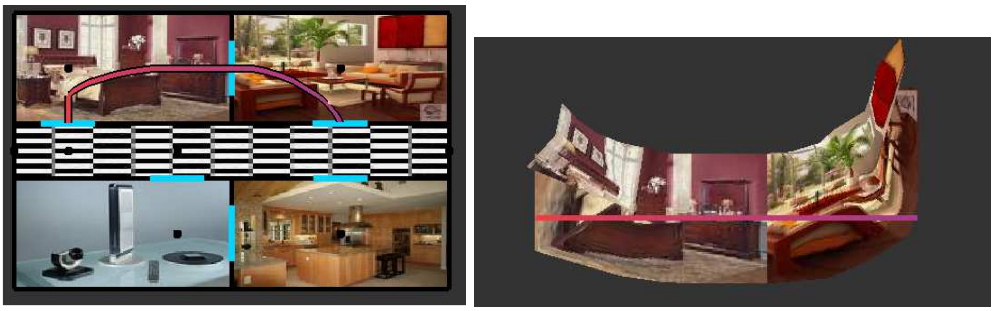


FIGURE 5.5 – *A gauche, un extrait d'itinéraire dans une scène découpée en Cellules et Portails ; A droite, le déroulé de l'itinéraire et la transformation de la scène résultant.*

Bien qu'ouvrant des perspectives intéressantes, il faut bien admettre que le résulat en 2D n'est guère impressionnant. Cependant, notre méthode de mise en correspondance et skinning est parfaitement généralisable à la 3D où la déformation du décor prend beaucoup plus de sens. Mais la 3D ouvre des questions supplémentaires sur lesquelles nous reviendrons en fin de chapitre.

5.3.2 Visualisation du décor guidée par l'acteur

Une autre manière de visualiser le décor consiste à mettre en évidence, pour un intervalle de la trajectoire, uniquement les cellules concernées par cet intervalle.

Indépendamment du style de rendu (plus ou moins grand niveau d'abstraction ou de transparence selon le statut de la cellule), une visualisation consiste à zoomer uniquement sur les cellules actives. Le mécanisme de zoom mis en oeuvre s'apparente à une explosion du modèle géométrique : nous commençons par calculer un centre d'explosion, qui peut être soit le centre de la zone, soit le point du milieu du morceau d'itinéraire concerné. Nous appliquons ensuite un changement d'échelle à partir de ce centre d'explosion pour zoomer sur l'ensemble des pièces du décor. Nous utilisons ensuite le centre de chacune des cellules calculée précédemment à l'aide de la carte de distance. Pour chaque cellule passive, nous appliquons alors la transformation d'échelle inverse à partir de ce centre. Le résulat produit, illustré par la figure 5.6, s'apparente à une vue explosée du décor où les pièces actives sont mises en évidence.

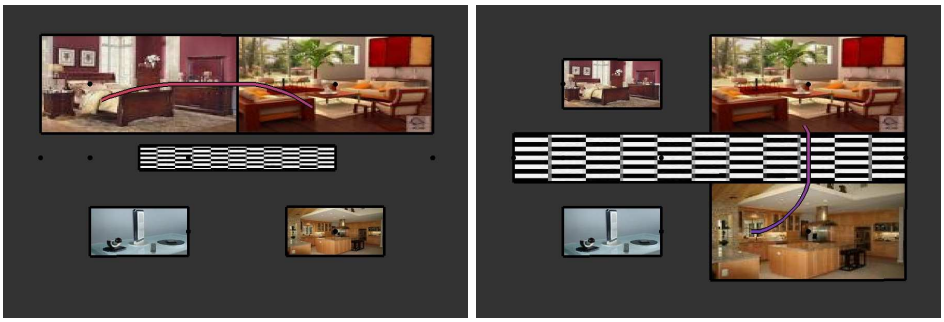


FIGURE 5.6 – *Visualisation des cellules du décor à différentes échelles selon la portion d'itinéraire sélectionnée.*

Notons qu'un tel zoom géométrique peut être complété et amplifié par un zoom sémantique sur lequel nous reviendrons dans le chapitre 6.

Si des questions autour du passage à la 3D se posent aussi pour cette application, une perspective nous semble explorable. Nous avons vu que nous nous contentons de calculer un centre pour chaque cellule à l'aide de la carte de distance. Mais cette carte nous permet aussi de calculer un squelette pour chaque pièce. De plus, nous disposons d'une information de relation entre les cellules grâce aux portails.

La combinaison de ces données permet d'imaginer d'autres modèles d'explosion en plaçant par exemple des ressorts au niveau de chaque porte. Nos premiers tests dans ce sens se sont cependant révélés peu concluants du fait d'un système masse/ressort tendant à rendre quelque peu aléatoire la position finale de chaque pièce après explosion. De même, le squelette de chaque pièce pourrait permettre d'appliquer à celle-ci une déformation dans le même esprit que la visualisation présentée dans la section précédente. Mais il ne s'agit pour l'instant que d'une idée.

5.4 Discussion : décor et visibilité

Un metteur en scène vous expliquera que le décor est sans importance et que ce sont les acteurs qui font la pièce. Il vous dira cela surtout si vous êtes acteur ! Il ne pourra cependant nier que le décor, ou son absence, contribue pleinement à rendre l'atmosphère du récit. Autrement dit, le choix du décor dépend de ce que l'on souhaite raconter.

De la même manière, la visualisation de la géométrie statique dépend de l'application finale. Cependant, une écrasante majorité de ces applications se déroule dans des environnements 3D, créant des problèmes de visibilité à cause des murs qui peuvent dissimuler l'intérieur d'une pièce. Pour résoudre ce problème, il est courant en illustration scienti-

fique, notamment dans les domaines techniques et médicaux, de découper un modèle pour mettre en évidence les différentes couches qui le composent(par exemple figure 5.7).

FIGURE 5.7 – *Deux exemples de modèles découpés pour mettre en évidence leurs différentes couches.*

Explorer un modèle à l'aide de ces techniques de visualisation est une problématique très actuelle en synthèse d'images. Li *et al.* [LRA+07] ont présenté un système complet permettant de générer des découpes adaptées à la géométrie globale du modèle. Burns et Finkelstein [BF08] ont proposé un algorithme permettant d'effectuer des découpes en temps réel sur la carte graphique, permettant une exploration continue et interactive. Knödel *et al.* [KHG09] ont mis en place une interface permettant à un utilisateur de définir les découpes qui l'intéressent à l'aide de sketchs.

La convergence de ces travaux permet alors de réaliser la découpe d'une scène 3D adaptée aux trajectoires des éléments en mouvement dans cette scène.

Le texte : annotations du modèle géométrique

6

Les historiens spécialisés s'accordent aujourd'hui pour dater la naissance de la bande dessinée avec les oeuvres de Rodolphe Töpffer. Politicien et écrivain suisse de la première moitié du dix-huitième siècle, Rodolphe Töpffer commence *Histoire de monsieur Jabot* en 1833. L'oeuvre se distingue par son utilisation conjointe mais surtout complémentaire du texte et de l'image, ce qui en fait plus qu'un simple "roman illustré". C'est d'ailleurs la vision qu'en a l'auteur lui-même, déclarant : *Les dessins, sans le texte, n'auraient qu'une signification obscure ; le texte, sans les dessins, ne signifierait rien*.

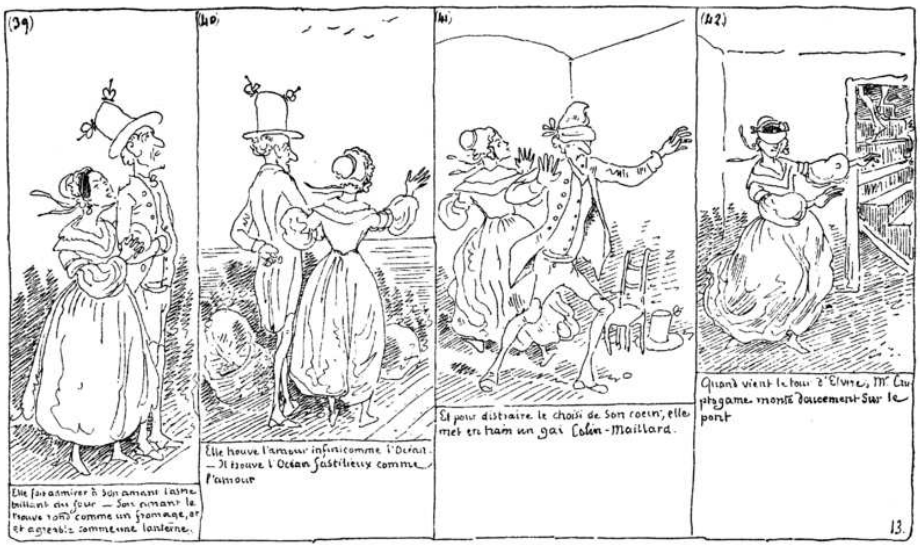


Figure 6.1 – *Histoire de Monsieur Cryptogame de Rodolphe Töpffer*

Cette phrase de l'auteur suisse est valable pour de nombreux autres domaines, tels que la cartographie, l'illustration médicale ou le dessin technique. L'exemple de gauche de la figure 6.2 montre comment le texte

est généralement utilisé dans l'illustration médicale pour décrire les différentes parties d'un organe. Le texte est aligné avec la silhouette de l'objet et connecté par une ligne à la région du modèle qu'il décrit.

Cette technique d'ajout d'information n'est pas limitée au texte, comme le montre l'exemple droit de cette même figure 6.2. Dans ce qui suit, nous utiliserons les termes de label ou annotation pour désigner du texte ou des éléments graphiques venant compléter l'illustration principale. Le problème qui nous intéresse est de venir placer un label autour d'un modèle géométrique, en le connectant à l'aide d'une ligne à la partie qu'il décrit, symbolisée par un point d'intérêt.

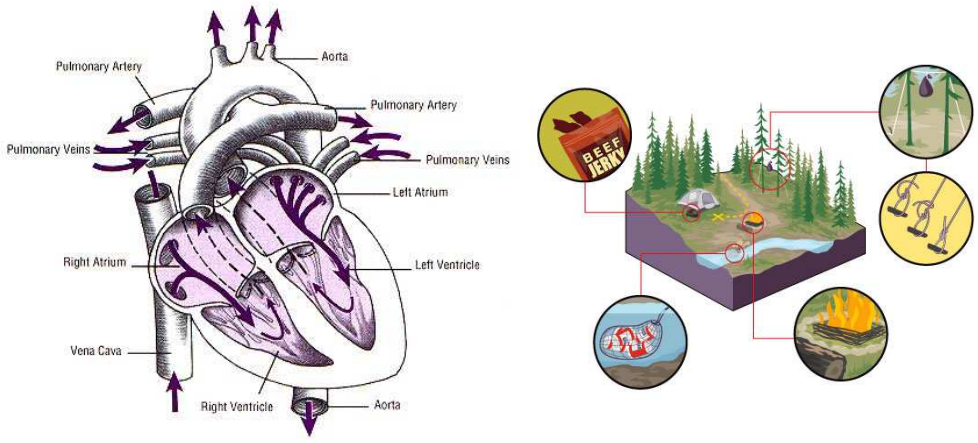


FIGURE 6.2 – *Deux exemples d'étiquetage de scène, qui peut prendre la forme de texte ou d'images.*

Dans le contexte d'une utilisation interactive, ces annotations doivent revêtir un caractère dynamique, c'est-à-dire réactualiser leurs positions en fonction des modifications de l'utilisateur telles qu'un changement de point de vue par exemple. Cette contrainte a une influence sur la **complexité** des algorithmes utilisables en terme de temps de calcul et sur la stratégie de positionnement des annotations, qui doivent maintenir une **continuité** dans leur mouvement.

Dans ce chapitre, nous commençons par passer en revue les diverses techniques de placement d'annotations dans les cas 2D puis 3D (section 6.1). Nous présentons ensuite un nouvel algorithme de placement temps-réel d'annotations dans une scène 3D préservant au maximum la continuité temporelle de leur mouvement (section 6.3). Nous parlerons ensuite de l'ajout d'une chronologie sur nos annotations et discuterons de l'influence de cette dimension supplémentaire sur les différentes parties de notre algorithme (section 6.4)

6.1 Annotation : de la 2D statique à la 3D interactive

L'annotation d'un modèle géométrique, ou étiquetage de scène, est un problème vaste qui va du cas 2D statique, problème classique de cartographie, à celui d'un modèle 3D dynamique. Pour introduire les différents algorithmes d'étiquetage qui existent dans la littérature, il convient de commencer par présenter le problème classique du *Map Labeling* en 2D.

6.1.1 Annotation interne

Dans l'histoire de la cartographie, deux problèmes ont mobilisé l'énergie des géographes et géomètres : la connaissance la plus précise possible du relief, et la projection de la sphère sur une feuille. La question de l'ajout des noms de villes, de fleuves, de rivières semble s'être limitée à quelques considérations esthétiques. Ironiquement, dans notre société actuelle de l'information numérique, le relief est connu et accessible au plus grand nombre et la projection n'est que l'affaire de quelques paramètres d'une matrice. Ajouter du texte à une carte relève par contre du challenge pour l'ordinateur.

Le problème du *Map Labeling* se formule basiquement ainsi : étant donné un ensemble de villes symbolisées par des points, on cherche à attacher à chaque ville son nom, sous la forme d'une petite étiquette, en cherchant à minimiser le nombre d'intersections entre deux noms de villes. D'autres formulations de ce problème existent, dans le cas de lignes représentant des rivières par exemple, ou de régions (pays). Le nombre de positions possibles pour le nom de la ville peut être discret (par exemple deux placements possibles, au-dessus et en-dessous du point) ou continu (le nom est attaché au point, mais peut se promener sur les 360 degrés autour de la ville). Trouver la solution dans un cas simple (4 positions possibles) a été démontré comme étant NP-Dur [MS91]. Les solutions proposées sont extrêmement variées, allant du glouton customisé aux algorithmes génétiques ou à base de recuit simulé. L'état de l'art de Christensen *et al.*[CMS95] est un bon point de départ pour découvrir ces approches. Récemment, Daiches [Dai06] s'est attaqué au problème du zoom sur la carte, qui nécessite une mise à jour consistante en temps-réel du labeling. Sa méthode se base sur le calcul de plusieurs solutions à des échelles différentes, certains labels étant supprimés lorsque l'échelle devient de plus en plus grossière.

6.1.2 Annotation externe 2D

Le placement "extérieur" des labels a été introduit par Fekete et Plaisant [FP99] comme une alternative au placement interne des labels dans le cas d'ensemble de points dense. Les auteurs définissent une région d'intérêt à l'aide d'un cercle, métaphore d'une loupe, englobant un sous-ensemble des points présents à l'écran. Les informations associées à ces points sont ensuite positionnées sur le bord du cercle, une ancre connectant l'information avec son point associé pour éviter toute ambiguïté. Le placement se fait sur la base d'une projection radiale, ce qui prévient automatiquement les intersections d'ancres. Leur algorithme est cependant entièrement limité au cas 2D.

C'est aussi le cas 2D qui mobilise les recherches de Bekos *el al.* [BKSW05]. Les auteurs formalisent le problème en contraignant les labels à être placés sur le bord d'une boîte englobante. Ils cherchent alors à minimiser, selon les cas, la longueur cumulée des ancres ou le nombre de créneaux dans le cas d'ancres en forme de marches d'escalier. Une limitation majeure de leur approche est que les labels doivent tous avoir la même taille.

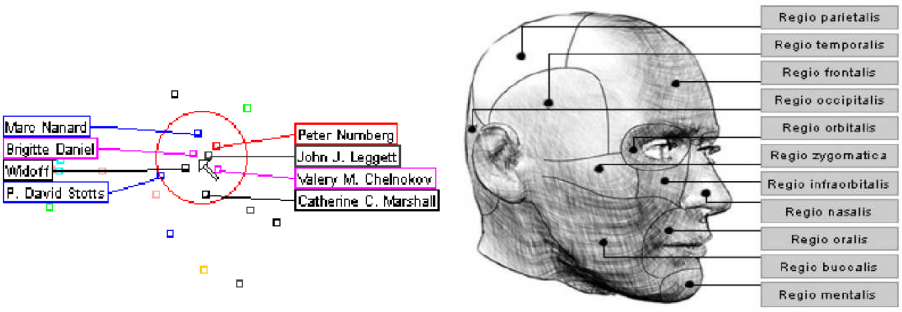


FIGURE 6.3 – *A gauche, un exemple de résultat de [FP99], à droite un labeling sur un seul bord extrait de [BKSW05]*

Vollick *et al.* [VVAH07] modélisent le problème sous la forme d'une minimisation d'une fonction d'énergie. Leur motivation première est de capturer le "style" d'un labeling établi par un utilisateur et de le réutiliser pour calculer l'étiquetage d'autres modèles. Ils utilisent un algorithme de recuit simulé pour résoudre la minimisation, calcul prenant plusieurs minutes et rendant conséquamment leur approche inadaptée pour le temps-réel.

6.1.3 Annotation 3D

Une première mention de l'étiquetage externe dans le cas 3D se trouve dans Preim *et al.* [PRS97]. Les auteurs limitent l'interaction au cas du

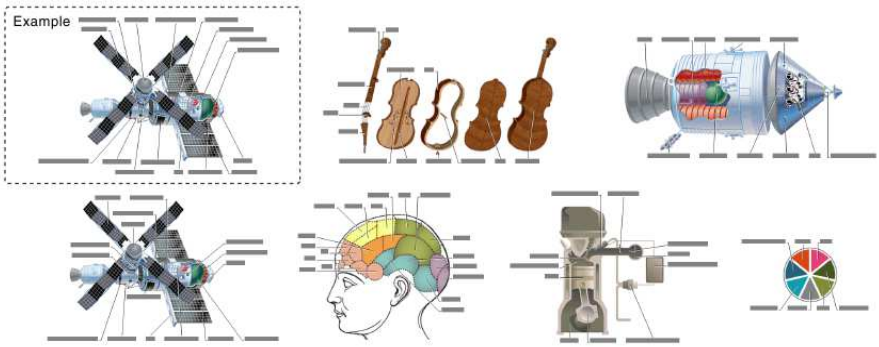


FIGURE 6.4 – *La capture de style de [VVAH07]. L'utilisateur commence par composer un labeling pour un modèle particulier, l'image encadrée en haut à gauche. Une minimisation inverse transforme cette image en paramètres de style qui servent à calculer d'autres labeling ayant la même "apparence" que le premier.*

zoom et cherchent à maintenir un labeling cohérent lors des changements d'échelle. Dans le cas d'interactions plus complexes (rotation...), Strothotte [Str98] a mis en évidence les problèmes de cohérence temporelle d'un labeling dynamique en montrant deux exemples basiques pour lesquels il est impossible d'éviter du popping dans le cas de rotation autour du modèle étiquetté. Les figures 6.5 et 6.11 illustrent ces cas "impossibles".

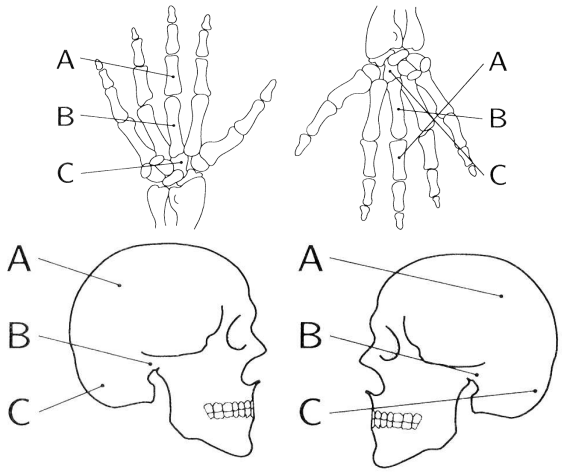


FIGURE 6.5 – *Deux exemples de problème de cohérence temporelle tirés de [Str98].*

Dans [BFo0], Bell et Feiner introduisent une stratégie de gestion de l'espace libre pour mettre à jour la position d'un ensemble de fenêtres présentes à l'écran. Cette méthode leur sert de support pour la mise en place d'un algorithme permettant le placement en temps-réel de labels dans une scène 3D, dans un contexte de réalité virtuelle [BFH01]. Pour ce faire, les auteurs approximent la scène par un ensemble de boîtes englobantes, et découpent ensuite l'écran en rectangles vides. Les labels sont

ensuite positionnés itérativement dans le rectangle vide le plus proche, et la structure est remise à jour. Une telle simplification de la scène, qui peut se révèler très grossière dans certains cas, introduit un placement des labels sous-optimal en terme de qualité visuelle. De plus, la structure intrinsèquement discontinue et discrète de l'appoximation de la scène entraîne des problèmes de discontinuité lors de la navigation. Pour reduire ce dernier artefact, les auteurs utilisent une technique d'hystérésis qui permet de grandement lisser l'animation.

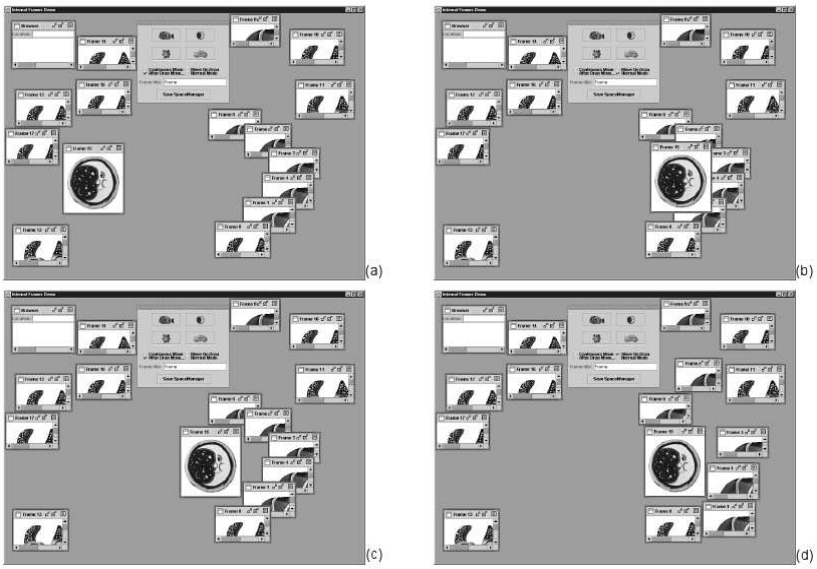


FIGURE 6.6 – *Extrait de [BFoo]*

Hartmann *et al.* [HAS04] modélisent le problème à l'aide de champs de potentiels. La position des labels est initialisée à la position de leur point d'accroche. Une énergie est diffusée en partant du centre de la scène et repousse les labels jusqu'à ce qu'ils atteignent une position valide, c'est-à-dire respectant les contraintes de non-recouvrement. Si la formulation ne manque pas d'élégance, elle génère cependant certains artefacts du fait qu'elle ne prend absolument pas en compte les intersections d'ancres. Les auteurs procèdent alors à un post-traitement en permutant les positions des labels dont les ancres s'intersectent, étape qui peut potentiellement faire perdre toute la consistance esthétique du labeling initial.

Ali *et al.* [AHS05] présentent un pipeline très général permettant de calculer un labeling adoptant de nombreux styles différents. Les étiquettes qu'ils peuvent placer se limitent par hypothèse à une seule ligne de texte, limite qu'exploite donc leur algorithme. Par ailleurs, les labels sont constamment placés à l'extérieur de la scène.

Götzelmann *et al.* [GHS07] présentent une solution dédiée aux mo-

dèles 3D animés. La principale restriction de leur approche est qu'elle nécessite d'analyser l'intégralité de l'animation, rendant ainsi la méthode inutilisable pour une navigation interactive. Les mêmes auteurs ont mis en place un algorithme temps-réel [GHS06] pour éviter cet écueuil.

6.1.4 Bilan sur les techniques d'annotation

À l'exception du travail de Bell et Feiner, toutes les méthodes d'annotation interactive restreignent le placement à une enveloppe englobant l'ensemble du modèle. Dans le cas d'un archipel, une étiquette ne peut donc pas être positionnée entre deux îles, solution qui serait pourtant la plus lisible.

Par ailleurs, les méthodes d'optimisation globale, plus performantes en terme de qualité, sont inaplicables à un usage interactif. De plus, de telles méthodes doivent intégrer la nécessaire continuité de mouvement du label, ce qui rajoute une dimension au problème et augmente encore sa complexité.

6.2 Modélisation du problème

Dans ce qui suit, nous noterons $\mathbf{L}_i$ un label, qui sera assimilé à une image de dimensions $W_i \times H_i$. À chaque label est associé un point d'intérêt $\mathbf{P}_i$. Nous noterons $p_i = (X_i, Y_i)$ les coordonnées 2D à l'écran du point d'intérêt i. Nous appelerons $l_i = (x_i, y_i)$ la position 2D du label i. Le segment $[p_i, l_i]$ définit alors l'ancre connectant le label à son point d'intérêt. Nous noterons enfin $\mathbf{S}$ la scène, c'est-à-dire le modèle 3D auquel sont attachés les points d'intérêt.

Les contraintes esthétiques et techniques définissant un bon étiquetage sont alors les suivantes :

- Deux labels ne doivent pas se superposer, ils doivent rester complètement visibles pour que l'information qu'ils contiennent reste accessible.
- Deux ancres ne doivent pas se croiser afin de faciliter l'association visuelle entre un label et son point d'intérêt.
- Un label ne doit pas être placé sur la scène, ou en tout cas pas sur les parties visuellement importantes de la scène, toujours dans le but de rendre accessible l'information présente.
- Il est souhaitable de réduire au minimum la longueur de l'ancre afin que l'association label-point d'intérêt soit claire. De plus, la tradition du dessin technique amène à positionner les labels le long de la silhouette de l'objet ou le long d'une boîte englobante.

– Enfin, dans le cadre d'une exploration interactive, le mouvement des labels doit apparaître aussi naturel que possible. Leurs trajectoires doivent être continues, et leurs mouvements doivent suivre le mouvement apparent des points d'intérêts associés.

En traduisant mathématiquement ces différentes requêtes, nous nous ramenons donc à minimiser une fonction :

$$E(\{x_i, y_i, \}) = \sum_i |p_i l_i| \tag{6.1}$$

en respectant les contraintes :

$$\forall i \neq j \;\; [p_i, l_i] \cap [p_j, l_j] = \emptyset \tag{6.2}$$

$$\forall i \neq j \;\; \mathbf{L}_i \cap \mathbf{L}_j = \emptyset \tag{6.3}$$

$$\forall i \;\; \mathbf{L}_i \cap \mathbf{S} = \emptyset \tag{6.4}$$

Nous nous retrouvons donc devant un problème de minimisation sous contrainte que nous devons potentiellement résoudre en temps-réel pour permettre une exploration interactive. Cette contrainte de vitesse rend inutilisables les algorithmes d'optimisation classiques tel que le recuit simulé. C'est pourquoi notre choix s'est porté vers un algorithme glouton, implémenté sur GPU afin de tirer parti de la puissance des processeurs graphiques actuels.

6.3 Optimisation gloutonne sur carte graphique

Le principe de notre algorithme est basiquement très simple : nous disposons d'une pile de labels $\{\mathbf{L}_i, i = 1 \dots n\}$. Nous évaluons la fonction d'énergie du premier label $\mathbf{L}_1$ en tout point de l'espace libre, puis nous sélectionnons la position qui minimise cette fonction. Nous répétons ce processus pour les labels suivants jusqu'à ce que la pile soit vide. Plusieurs points de ce processus demandent une attention particulière :

– Évaluer les contraintes entre le label courant et la scène, c'est-à-dire savoir estimer si leur intersection est nulle ou pas.
– Évaluer les contraintes entre le label courant et ceux déja placés, c'est-à-dire savoir estimer s'ils s'intersectent ou pas.
– Calculer un ordre intelligent sur nos labels, l'ordre d'insertion conditionnant fatalement la qualité finale du résultat.

6.3.1 Contraintes entre les labels et la scène

Soit $\mathbf{L}_i$ le label courant, $\mathbf{S}$ la scène et $\mathbf{I}(x)$ l'image 2D de la scène telle que $\mathbf{I}(x) = 1$ si x est un pixel appartenant à $\mathbf{S}$ et $\mathbf{I}(x) = 0$ sinon. On a

alors :

$$\mathbf{L}_i \cap \mathbf{S} = \{\} \Leftrightarrow \int_{\mathbf{L}_i} \mathbf{I}(x)dx = 0$$

L'évaluation de notre contrainte est donc équivalente au calcul d'une intégrale sur un masque binaire de la scène. Sachant que nos labels sont exclusivement rectangulaires, ce calcul peut être effectué de manière efficace en ayant recours à une *Summed-Area Table* (SAT).

Les SAT ont été introduites à l'origine par Crow [Cro84] pour servir d'alternative au mip-mapping dans le cas de filtrage de texture. Elles se définissent ainsi : Soit $T(x, y)$ un tableau d'entiers. La représentation de T sous forme d'une SAT est un tableau S de même dimension tel que chaque case $S(x, y)$ contient la somme des valeurs contenues dans le rectangle défini par la diagonale $[T(0, 0), T(x, y)]$. Supposons maintenant que nous souhaitions connaître la somme des pixels de T contenus dans un rectangle défini par deux points (X_1, Y_1) et (X_2, Y_2). Cette somme Σ est donnée par :

$$\Sigma = S(X_2, Y_2) - S(X_2, Y_1 - 1) - S(X_1 - 1, Y_2) + S(X_1 - 1, Y_1 - 1)$$

le dernier terme est ajouté pour corriger le fait que le rectangle $(X_1 - 1, Y_1 - 1)$ est soustrait deux fois. La figure 6.7 illustre de calcul.

Une SAT permet donc de calculer une intégrale sur un rectangle de taille quelconque aligné avec les axes de l'image à l'aide d'une addition et de deux soustractions. Hensley *et al.* [HSC+05] ont présenté un algorithme temps-réel permettant de calculer une SAT sur carte graphique.

De par sa nature, une SAT répond parfaitement à ce que nous cherchons : une structure de données portable sur GPU permettant l'évaluation rapide de la contrainte scène/label. Pour la mettre en oeuvre, nous commençons par effectuer un premier rendu de la scène dans une texture, sous la forme d'un masque binaire. L'algorithme de [HSC+05] nous fournit ensuite une texture qui nous permet de tester en temps constant l'intersection entre la scène et un label de taille quelconque.

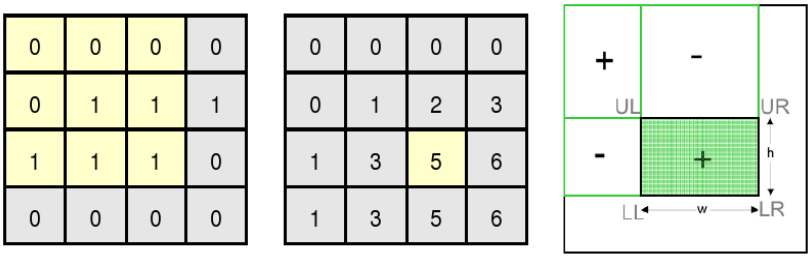


FIGURE 6.7 – *Evaluation de la contrainte Label/Scène : la scène est représentée par un masque binaire à partir duquel nous calculons une Summed-Area Table qui nous permet une évaluation efficace de la contrainte*

6.3.2 Contraintes entre deux labels

Pour prévenir l'intersection entre le label courant l_i et les labels déjà placés l_j, nous définissons l'espace des positions valides pour l_i. Cet espace se définit en prenant en compte l'ensemble des labels l_j déjà placés, le point d'accorche p_i et la taille (W_i, H_i) du label l_i. La figure 6.8 illustre la construction de cet espace.

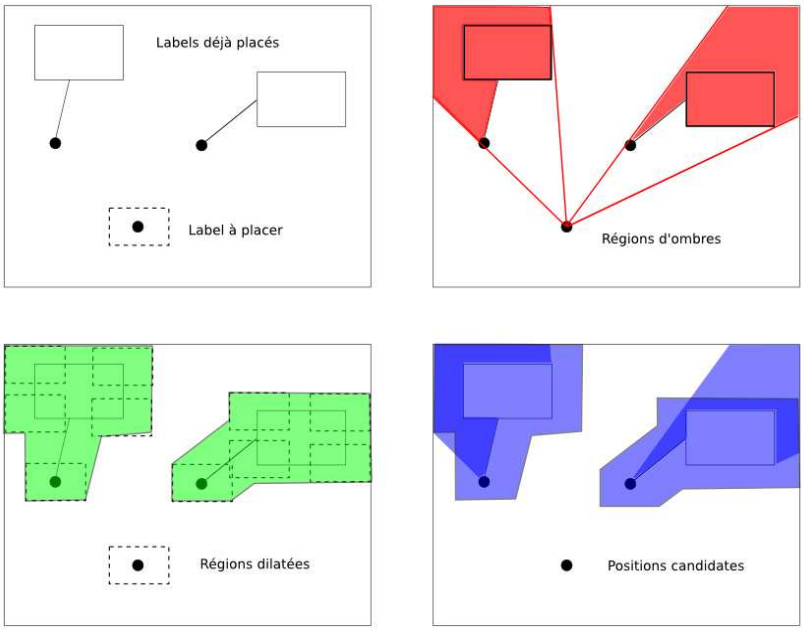


FIGURE 6.8 – *Pour respecter la condition de non-intersection entre labels, nous commençons par calculer la région d'ombre en haut à droite, qui correspond à l'ensemble des points où nous pouvons placer le label l_i sans que son ancre ne s'intersecte avec les autres labels. Nous dilatons ensuite cette région par un rectangle de taille $W_i \times H_i$, ce qui prévient les intersections entre le label l_i et les labels l_j. La région vide représente alors l'ensemble des positions admissibles par rapport à nos contraintes labels/labels.*

En pratique, la région « interdite »peut être calculée de manière analytique sous la forme d'un polygone pour chaque label l_j. En rendant ce polygone dans une texture, nous pouvons alors tester notre contrainte de manière atomique sur la carte graphique. De plus, nous pouvons contrôler la taille de la dilatation, par exemple en ajoutant une petite marge aux dimensions (W_i, H_i) du label l_i, ce qui nous permet de définir une distance minimum entre labels, ce qui permet de renforcer la lisibilité de l'étiquetage final.

6.3.3 Ordre d'insertion des labels

Insérer les labels sans ordre particulier peut donner des résultats peu satisfaisants. Dans le pire des cas, un label peut devenir impossible à insérer sans violer une des contraintes. La figure 6.9 de gauche montre un exemple de "mauvais" résulat.

Notre statégie de tri s'articule autour de l'idée de placer en premier les labels a priori difficiles et de garder les plus simples pour la fin. Intuitivement, les labels les plus durs à positionner sont ceux pour lesquels on aurait peu de pixels candidats, ou bien ceux qui auraient une ancre longue augmentant donc la valeur de la fonction d'énergie.

Deux situations peuvent entraîner un placement insatisfaisant. La première a lieu quand un point d'intérêt se trouve encerclé par d'autres points d'accroche. Si nous positionnons en premier les labels des points extérieurs, le label interne se retrouvera complètement bloqué et n'aura plus aucune position valide. La deuxième provient des points d'intérêt pour lesquels l'ancre sera longue, ce qui correspond aux points qui sont le plus à l'intérieur de la scène.

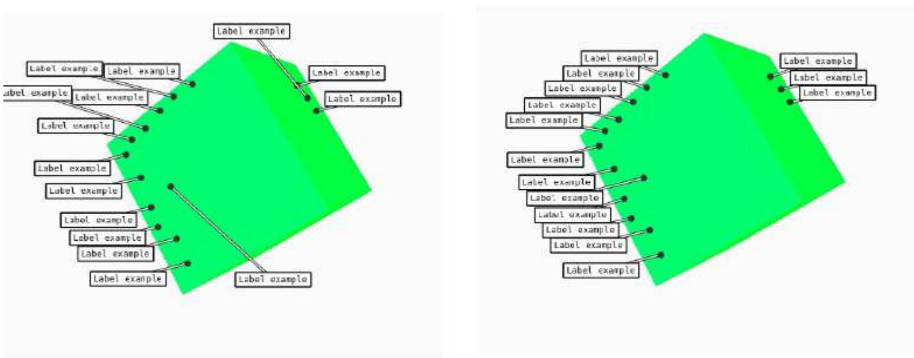


FIGURE 6.9 – *À gauche, notre algorithme glouton effectue le placement des labels dans un ordre aléatoire. On constate certains artefacts : dans le coin Nord-Ouest, les étiquettes semblent se battre et se repousser. Dans le coin Nord-Est et au Sud, deux labels se retrouvent à des positions très éloignées de leurs points d'intérêt. A droite, l'ordre d'insertion préalablement calculé permet un placement beaucoup plus propre et cohérent avec les positions relatives des points d'intérêt.*

Le premier problème se résout à l'aide d'un diagramme de Voronoï des points d'intérêt. Nous considérons toutes les cellules d'aire infinie de ce diagramme que nous parcourons dans le sens trigonométrique par exemple. Par définition du diagramme de Voronoï, chaque cellule infinie a exactement deux voisines ayant la même propriété. Une telle opération est équivalente à parcourir l'enveloppe convexe des points d'intérêt. Cette liste circulairement ordonnée est alors stockée dans une pile. Nous effeuillons alors notre diagramme de Voronoï en supprimant les

cellules d'aire infinie et ré-effectuons le même traitement sur les cellules restantes, jusqu'à ce qu'il ne reste plus de point d'intérêt non placé dans la pile. Le dépilage nous garantit alors que les labels seront insérés dans un ordre cohérent par rapport à leurs voisins.

Le deuxième problème se résout en ayant recours à une carte de distance. Pour chaque point d'intérêt, nous calculons sa distance par rapport au plus proche pixel n'appartenant pas à la scène. Nous trions ensuite les points d'intérêt par ordre décroissant de distance, ce qui garantit que les labels les plus à l'intérieur seront placés en premier.

Les deux solutions présentées aboutissent à deux ordres différents et présentent chacune des avantages et des inconvenients. Pour fusionner ces deux approches, nous conservons notre statégie d'effeuillage, mais en l'appliquant non plus à un diagramme de Voronoï mais à un diagramme d'Apollonius, c'est à dire un diagramme de Voronoï pondéré.

On dispose de sites c_i auxquels on associe des poids w_i. Le diagramme d'Apollonius est une subdivision du plan en cellules associées aux sites, la cellule d'un site (c_i, w_i) étant composée de l'ensemble des points du plan qui sont plus proches de c_i que de tout autre c_j. La distance d'un point x dans le plan à c_i est définie par :

$$d(x, c_i) = |x - c_i| - w_i$$

Dans le cas $w_i = w_j \ \forall i, j$, le diagramme d'Apollonius est équivalent à celui de Voronoi.

A chaque point d'intérêt, nous associons comme poids sa distance par rapport au plus proche pixel vide. La stratégie d'effeuillage sur cette stucture donne alors un ordre très naturel et qui minimise les artefacts que nous mentionnions. La figure 6.10 permet de comparer les diagrammes de Voronoi et d'Apollonius dans un cas simple.

6.3.4 Gestion de la cohérence temporelle

Strothotte[Str98] a montré que la cohérence temporelle parfaite était impossible à obtenir. Cette impossiblité est en partie issue de nos contraintes puisque nous exigeons que

– Le label ne doit jamais se superposer avec la scène.
– Le mouvement du label doit être continu.

La figure 6.11 illustre alors l'incompatibilité de ces deux requêtes.

Notre solution à ce problème consiste à lisser le mouvement du label. Lors de l'affichage du label, nous prenons en compte à la fois sa nouvelle position et sa position dans l'image précédente. La position effective à l'écran du label devient alors une moyenne pondérée entre l'ancienne et la nouvelle position. La pondération nous permet de jouer sur la vitesse du lissage, un label pouvant alors nécessiter plusieurs images pour

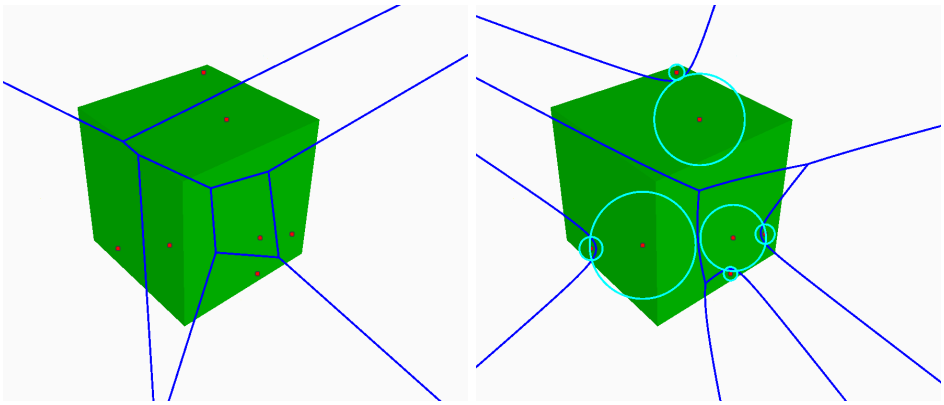


FIGURE 6.10 – *A gauche, le diagramme de Voronoï contient quatre cellules infinies et trois cellules internes, ces dernières correspondant aux points qui seront placés en premier par notre algorithme glouton. A droite, le diagramme d'Apollonius ne contient que des cellules infinies. L'ordre obtenu devient alors beaucoup plus naturel pour placer les labels.*

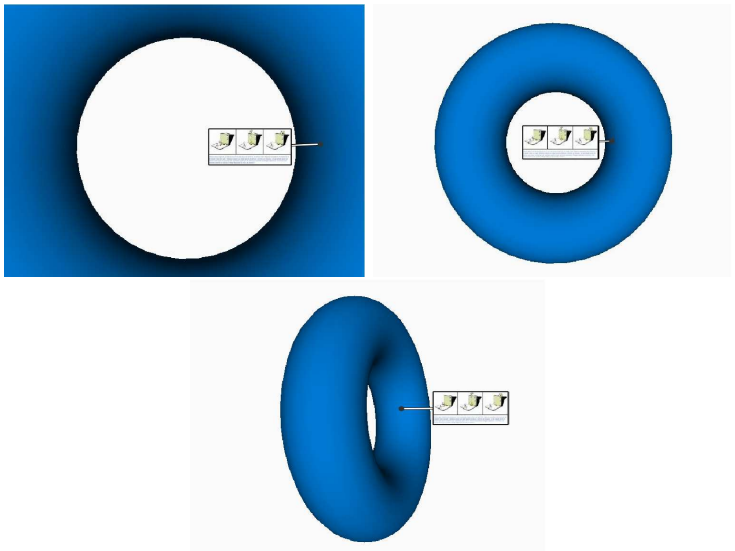


FIGURE 6.11 – *Le placement des labels ne peut pas être continu. Ces trois images montrent trois points de vue consécutifs (dézoom puis rotation) d'un tore auquel est attaché un label. Dans la première image, le label doit être placé à l'intérieur du tore. Dans la dernière, il doit être à l'extérieur. La contrainte imposant que le label ne doit pas se superposer à la scène (i.e. le tore), il devra nécessairement se téléporter au cours de l'animation, brisant la cohérence temporelle.*

atteindre sa position idéale. Cette solution ne résout pas fondamentalement notre problème puisque le label est toujours téléporté, mais le lissage permet de minimiser les discontinuités visuelles. Ainsi, la violation de la contrainte de non-recouvrement n'est que temporaire et lorsque l'utilisateur stabilise son point de vue, les labels convergent doucement vers leur position finale dans un effet de relaxation.

L'implémentation de cette unique stratégie a cependant pour effet de faire flotter les labels, petites vignettes tiraillées entre leur ancienne maison et leur nouveau domicile. La source de ce problème est notre algorithme glouton, aveugle à la continuité du mouvement. La solution idéale calculée pourrait par exemple envoyer un label à droite du modèle alors que celui-ci se trouvait à gauche. Nous procédons à l'interpolation précedemment décrite mais soudain, notre algorithme décide que le label doit retourner à gauche, à cause d'une variation minime dans la fonction d'énergie. Le résultat devient alors désastreux visuellement. Pour stabiliser nos labels, nous définissons leur position "idéale" en terme d'interaction utilisateur. Soient $p_i(t)$ la position du point d'intérêt associé au label l_i à l'instant courant et $p_i(t-1)$ la position du même point d'intérêt dans l'image précédente, et $l_i(t-1)$ la position du label dans l'image précédante. Nous définissons la position idéale à l'instant courant $l_i(t)$ comme étant :

$$l_i(t) = l_i(t-1) + (p_i(t) - p_i(t-1))$$

c'est à dire que le label l_i devrait idéalement avoir le même mouvement à l'écran que son point d'intérêt associé. Nous modifions alors notre fonction d'énergie en lui ajoutant un terme prenant en compte cette position idéale $l_i(t)$

$$E(\{x_i, y_i, \}) = \sum_i \alpha |p_i l_i| + \beta |l_i l_i(t)| \tag{6.5}$$

Les paramètres α, β peuvent être fixés de manière globale, mais ils peuvent aussi être dynamiques. Tant que l'utilisateur est en train de changer de point de vue autour de la scène, on donne plus d'importance au facteur β afin de minimiser les discontinuités et assurer un mouvement lisse des labels au détriment de la qualité effective du placement. Lorsque l'utilisateur stabilise son point de vue, on annule le paramètre β de façon à ne plus rechercher que le meilleur positionnement absolu des labels. Notre stratégie de lissage s'appliquant toujours, on asssure une transition douce entre la position finale de l'animation et la position idéale selon le point de vue donné. Ceci assure ainsi que le placement définitif des labels pour un point de vue donné ne dépendra pas de l'historique de navigation.

Une autre source de discontinuité provient du changement de statut d'un point d'intérêt, visible à un instant t et invisible (masqué par la scène) à un instant $t+1$. L'action simple de supprimer le label de l'affichage entraine naturellement une discontinuité ; une oscillation de la navigation provoque un clignotement du label, effet indésirable. Nous traitons ce dernier cas de manière basique en procédant à *un alpha_blending* lors de l'affichage d'un label qui change d'état de visibilité. Ainsi, un label dont le point d'intérêt devient invisible n'est plus pris en compte par l'algorithme glouton mais continue à rester un certain temps visible à l'écran. On pourrait imaginer une stratégie plus intelligente en considérant les points d'intérêt comme des disques faisant face à l'utilisateur. Une requête d'occlusion nous permettrait de déterminer la proportion visible du disque et ainsi de moduler la transparence du label en fonction de cette proportion. Faire varier de manière continue la visibilité des points d'intérêts aurait l'avantage supplémentaire d'éviter les problèmes de précision numérique inhérents au test d'un seul pixel.

6.3.5 Détails de programmation

Notre algorithme ne se déroule pas dans sa totalité sur la carte graphique. Le calcul et le parcours du graphe d'Appolonius ainsi que la gestion de la pile ordonnée de labels ont lieu sur CPU. Cette utilisation conjointe du processeur central et du processeur graphique nécessite un transfert entre les deux. Or, si les communications CPU→GPU sont naturelles et peu coûteuses, elles peuvent handicaper les performances dans le chemin inverse. Les différentes étapes de notre algorithme sont :

1. **CPU→GPU** : Rendre la scène sous la forme d'un masque binaire. En pratique, nous pourrions taguer certaines parties de la scène avec un critère d'importance et assigner des valeurs plus élevées à ces zones. Le reste de la scène serait alors traité comme étant à recouvrir au minimum mais sans interdit. Cette extension ne modifierait pas notre pipeline.
2. **GPU** : Calculer la SAT associée au masque de la scène. Le calcul de la fonction distance se fait dans le même temps et avec le même fragment shader. Pour cela, nous utilisons un algorithme de *jump flooding* (proposé en version GPU par Rong et Tuan[RT06]) qui permet d'effectuer le même parcours logarithmique pour le calcul des deux structures. Nous obtenons au final une texture stockant la valeur de SAT dans le canal rouge et la valeur de distance à la silhouette dans le canal vert.
3. **GPU→CPU** Pour chaque point d'intérêt, nous allons lire dans la texture précédemment calculée la valeur de la fonction distance.

4. **CPU** Nous calculons le graphe d'Appolonius en utilisant l'implémentation de la librairie CGAL. Nous effectuons ensuite l'effeuillage de ce graphe pour introduire notre ordre d'insertion des labels.
5. Pour chaque label de cette pile :
   (a) **GPU** Pour chaque pixel de l'image, nous évaluons notre fonction d'énergie associée au label courant, en mettant cette valeur à un infini prédéterminé si le pixel entraine une violation des contraintes de placement.
   (b) **GPU→CPU** Nous récupérons le pixel de coût minimal par un étape de réduction matricielle (voir figure 6.12).
   (c) **GPU** Nous mettons à jour la SAT pour tenir compte des labels déja positionnés. Cette mise à jour se fait de manière atomique.

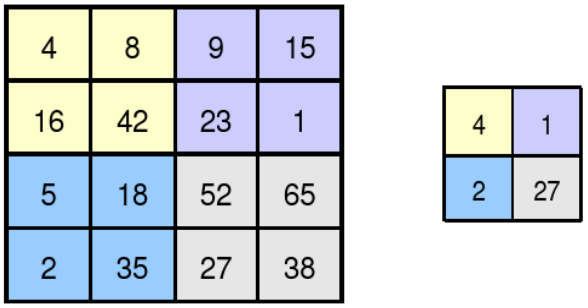


FIGURE 6.12 – *Le principe de la réduction matricielle pour calculer la valeur minimale d'un tableau. On découpe le tableau en blocs de 2x2 pixels, et le minimum de chaque bloc est stocké dans un nouveau tableau quatre fois plus petit. On obtient le minimum en* $\log n$ *itérations, n étant la dimension du tableau (nxn cases).*

Suivant ce mécanisme, les tranferts GPU→CPU se limitent donc à la lecture de quelques pixels dans une texture sans jamais nécessiter le couteux rapatriement de la texture entière sur le CPU.

Un des points clefs de notre algorithme est que la résolution de la texture dans laquelle sont effectués les calculs est complètement décorellée de la résolution de la fenêtre de l'utilisateur. Elle aura une influence uniquement sur le nombre de positions (x, y) possibles pour un label. En conséquence, le coût de notre méthode dépend uniquement de cette résolution. Soit n la taille de notre texture et L le nombre de labels à placer. Notre algorithme va nécessiter $1 + \log n + L * (\log n + 2)$ passes de rendu. Ceci peut sembler énorme au premier abord.

En pratique, nous avons implémenté notre méthode en C++ non-optimisé et effectué nos tests sur une GeForce 7800 en utilisant une texture de 512×512 pixels. Notre programme traite alors jusqu'à 20 labels

à une fréquence de 30 images par seconde. Ceci nous permet de dire que notre méthode fonctionne en temps-réel pour la majorité des cas pratiques, un nombre de labels supérieur à 20 à l'écran devenant peu lisible. La figure 6.13 montre des exemples de résultats.

FIGURE 6.13 – *Quelques résultats obtenus avec notre méthode. L'exemple basique du camion montre que notre algorithme se comporte bien dans le placement de labels autour d'un object compact, les étiquettes étant disposées harmonieusement autour de la silhouette. Les trois autres exemples montrent que notre algorithme parvient généralement à exploiter le moindre espace qui permet de placer une étiquette sans provoquer une ancre trop longue.*

6.3.6 Discussion

Positionner des étiquettes en respectant des contraintes esthétiques et techniques est un problème légèrement plus délicat que de rendre la monnaie. En conséquence, notre algorithme glouton échoue dans un certain nombre de cas. La figure 6.14 illustre deux cas pathologiques.

Le cas de droite de la figure 6.14 nous laisse quelque peu démunis. Il s'agit typiquement d'une situation où une méthode à base de diffusion d'énergie, qui ferait interagir dynamiquement les labels entre eux en se repoussant jusqu'à atteindre une position d'équilibre, serait plus adaptée. On pourrait considérer que le problème vient de l'ordre d'insertion :

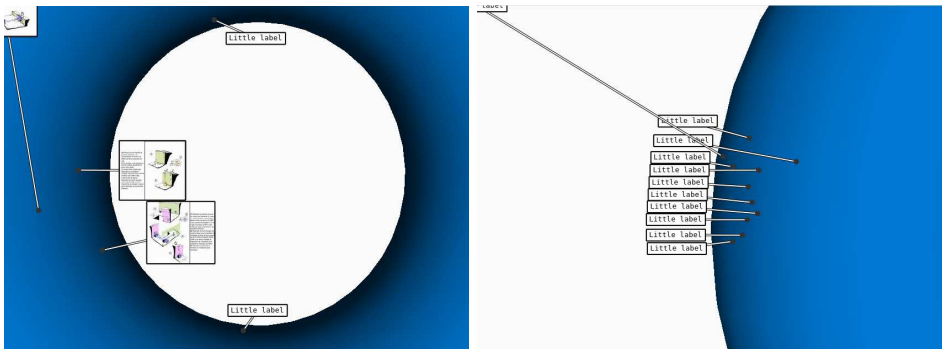


FIGURE 6.14 – *Deux exemples d'échecs pour notre algorithme. A gauche, l'ordre calculé est à l'opposé de l'ordre logique car notre algorithme considère en premier lieu un placement externe et non interne ; à droite, trop de labels saturant une zone peuvent rendre une étiquette impossible à placer sans violer les contraintes.*

pourquoi le label fautif n'a-t-il pas été placé avant ses deux acolytes situés juste au dessus ? La raison vient du fait que nous ne contrôlons pas la première cellule choisie par notre algorithme pour traverser le graphe d'Apollonius. Comme les deux cellules extrêmes sont adjacentes à l'infini, le mauvais label se retrouve propulsé au fond de la pile. Ainsi, le label juste au-dessus et celui juste au-dessous se retrouvent déjà placés quand arrive son tour et il n'y a plus de place disponible.

Le cas gauche est beaucoup plus intéressant. Il illustre le fait que dans notre stratégie d'effeuillage, nous formulons implicitement l'hypothèse que les labels se positionnent de préférence à l'extérieur de la scène, ce pourquoi il faut positionner en premier les plus à l'intérieur. Mais dans le cas de ce tore occupant tous les bords de l'image, les seules positions valides se trouvent dans le trou interne. Notre hypothèse au niveau de l'effeuillage n'est plus du tout correcte, il faudrait faire l'inverse. On pourrait imaginer de nombreuses stratégies pour contourner cette difficulté, mais la solution la plus simple est probablement la meilleure, et la formuler est l'occasion de discuter une hypothèse extrême de notre approche.

Regardons à nouveau notre figure de gauche et essayons d'aller au bout du problème :

« Pourquoi notre placement est-il mauvais ?

-Parce que les labels ont été triés dans le mauvais ordre.

-Pourquoi les labels ont-ils été placés dans le mauvais ordre ?

-Parce que nous faisons l'hypothèse qu'ils iront à l'extérieur de la scène, or dans ce cas précis ils doivent aller à l'intérieur.

- Pourquoi diantre doivent-ils aller à l'intérieur de la scène ?

- Mais voyons, parce qu'il n'y a pas de place à l'extérieur !

- Certes, mais pourquoi n'y a-t-il pas de place à l'extérieur ? »

Nous voici arrivés au coeur du problème. Voila que nous nous plaçons dans le cadre d'une application interactive cherchant à positionner à l'écran un ensemble d'étiquettes, et nous n'envisageons même pas que notre interface graphique puisse réserver une bordure entièrement dédiée au placement de ces étiquettes.

Au delà de cette question de bordure, il est ironique de constater que vouloir placer nos annotations de manière complètement libre dans l'espace 2D (sous respect des contraintes de lisibilité et d'esthétique) complique le problème. En comparaison, il est relativement facile de placer des annotations de manière contrainte sur une boite englobante ou une enveloppe convexe. Une solution explorable pour notre problème plus général consiterait alors à découper notre problématique de base en plusieurs sous-problèmes faciles. Nous commençons par partitionner notre espace 2D et nos différents morceaux de scènes en polygones convexes. Au sein de chacun de ces polygones, nous appliquons ensuite un algorithme plus simple de positionnement le long de la silhouette par exemple. Une telle stratégie se heurte alors à des problèmes de raccords spatiaux et temporels : spatiaux car le placement des labels se trouve alors contraint par les limites du polygone défini, faisant alors rater un placement optimal débordant sur une région voisine ; temporels car les variations de ce découpage au cours du temps introduiront fatalement des causes supplémentaires de discontinuités dans le mouvement des annotations.

Une autre stratégie, orientée vers l'exploration, consiste à adopter un algorithme mixte selon que l'observateur est fixe ou en mouvement. Lorsque l'utilisateur stabilise son point de vue, on peut se permettre d'avoir recours à un algorithme de placement non interactif, c'est-à-dire dont les temps de calcul seraient de l'ordre de deux ou trois secondes. Lorsque l'utilisateur est en mouvement, nous appliquons juste aux labels une translation équivalente au mouvement à l'écran de leurs points d'accroche. Le labeling peut bien évidemment perdre toute cohérence et lisibilité au bout d'un certain temps, auquel cas nous compterions sur l'utilisateur pour stopper sa marche effreinée et ainsi recharger un positionnement plus cohérent. Un tel scénario semble cependant peu souhaitable.

Plus lourd en terme de calcul serait d'effectuer un précalcul qui déterminerait plusieurs labelings selon différents points de vue. Un algorithme intelligent pourrait alors effectuer des transitions douces entre les différents positionnements de labels en fonction du point de vue de l'utilisateur. Cependant, on perdrait totalement la capacité d'éditer interactivement les labels avec des opérations d'ajout, de suppression ou de modification.

6.4 Discussion : Labels narratifs

Jusqu'à présent, nous avons considéré des labels statiques connectés à un nuage de points sans relation. Nous allons maintenant discuter de deux extensions ainsi que de leurs répercussions en terme algorithmique.

En fait de labels statiques, il serait plus juste de parler de hiérarchie. Prenons l'exemple de la figure 6.15. Lorsqu'un utilisateur zoome dans *Google Map*, le changement d'échelle spatiale entraine un changement d'échelle sémantique au niveau de l'information affichée. Aux numéros des autoroutes se substituent les numéros de départementales et les noms d'avenues.

Figure 6.15 – *Un zoom sur la ville de Grenoble. Dans l'image de gauche sont affichés les noms des communes avoisinantes et des axes autoroutiers. Dans l'image de droite, on lit les numéros des départementales ainsi que les noms des principales avenues. Au zoom géométrique est venu s'ajouter un zoom "sémantique" de l'information affichée.*

Cet exemple est un petit aperçu de ce qu'on nomme le "zoom sémantique". L'exemple ici est relativement simpliste car le niveau hiérarchique d'information est directement relié à l'échelle spatiale.

Le zoom sémantique, domaine important de la visualisation d'informations, ne fait pas l'objet d'une définition définitive puisque le wiki InfoVis recence sept définitions différentes. Notre préférée est celle de Modjeska :

"Another graphical technique to balance detail and context is known as semantic zooming or multi-scale interfaces. A physical zoom, on the one hand, changes the size and visible detail of ob jects. A semantic zoom, in the other hand, changes the type and meaning of information displayed by the object."

Pour illustrer la notion de zoom sémantique, on peut aussi évoquer un utilisateur gourmand zoomant d'abord sur sa destination de vacances et voyant s'afficher les différents noms de restaurants, puis avoir accès au menu du restaurant sélectionné.

Le niveau de hiérarchie sémantique n'est pas nécessairement corrélé à une hiérarchie physique et se pose alors la question de la navigation

dans un tel espace. Ces questions sont plus particulièrement abordées par les communautés d'Interface Homme-Machine de Visualisation d'Informations. Dans notre contexte, cela implique cependant d'autant plus la nécessité d'un algorithme dynamique pour le placement des labels, car l'introduction d'un degré de liberté supplémentaire entraine une démultiplication de ce qu'il serait nécessaire de calculer en pré-traitement.

La deuxième extension dont nous souhaiterions discuter concerne le nuage de points auxquels sont rattachées les annotations. Un cas courant d'application concerne les itinéraires, par exemple un trajet de visite dans un musée avec des labels évoquant les principales oeuvres observées. Dans un tel scénario, nous ne sommes plus en présence d'un nuage de points mais d'une ligne brisée : nos points d'accroche possèdent un ordre naturel. Et cela change tout !

Cet ordre naturel rend tout d'abord obsolète notre stratégie de tri à base de graphe d'Appolonius, car si un ordre d'insertion doit être appliqué, il semble beaucoup plus logique de privilégier le chemin tracé par les points d'accroche. Ensuite, cet ordre induit une nouvelle contrainte sur le placement des labels, car on souhaitera dans ce contexte que la structure visuelle des labels corresponde à la structure des points d'accroche. En quelque sorte, on souhaitera que la ligne brisée formée par les centres des labels soit géométriquement proche de la ligne brisée formée par les points d'accroche. Enfin, le corollaire de cette contrainte est qu'un algorithme de placement pourra travailler beaucoup plus localement puisque ses labels adjacents, donc ceux avec lesquels il y a risque d'intersection, sont connus et limités.

Un premier article, publié par Karnick *et al.* [KCJ+09], aborde cette question dans un cadre strictement 2D et en contraignant les labels à se placer le long d'une boite englobante. La figure 6.16 présente leur résultat.

Leur algorithme cherche au maximum à préserver l'équivalence entre ordre des labels et ordre des points d'accroche. Cependant, lorsque le placement respectant cette contrainte provoque un trop grand écart entre le label et son point d'accroche, les auteurs effectuent un split, créant une nouvelle chaîne indépendante.

Dans le cadre d'une application interactive, il est probable que ce splitting serait une cause d'instabilité. De plus, comme nous l'avons exposé précédemment, nous pensons que dans le cadre d'une application 3D, les labels doivent être placés le long d'une silhouette et non le long d'une boite englobante afin que leurs mouvements soient en adéquation avec le mouvement des points d'accroche.

Finalement, la combinaison d'un zoom sémantique dans un labeling interactif d'itinéraire ouvre la porte à de passionnantes applications de

FIGURE 6.16 – *Visualisation d'itinéraire avec la méthode de Karnick et al. [KCJ+09]. Les labels placés le long de l'image correspondent aux points d'intérêts de l'itinéraire et sont placés de manière à respecter l'ordre du parcours.*

visites virtuelles à plusieurs échelles, qui seront autant d'occasions de se projeter dans des voyages bien réels.

Conlusion : Vers le récit graphique sur ordinateur

7

On n'a pas encore trouvé de meilleur moyen que de raconter des histoires pour aller dans d'autres mondes, d'autres temps, d'autres corps et d'autres esprits que les nôtres. Les histoires multiplient et enrichissent la nôtre.

Pierre Jourde

Cette thèse a esquissé les contours du rendu narratif en synthèse d'images. En modélisant le schéma de visualisation général, nous avons proposé une description de l'espace d'information qui lui est associé. Nous avons défini les acteurs, le décor et le texte et avons abordé les problématiques de visualisation de ces trois composantes. L'étape suivante consiste à fusionner ces travaux pour obtenir un ensemble que nous pourrions qualifier de viewer 4D.

L'originalité de cet outil réside dans l'interaction avec la dimension temporelle. On ne visualise pas uniquement un instant, mais on observe un moment dont on peut faire varier la durée. Cette invention n'est pas la nôtre, l'état de l'art a montré plusieurs exemples utilisant ce principe. Les perspectives qu'elle ouvre sont vastes, voire vertigineuses. Pour les appréhender, il faut peut-être commencer par observer la période que nous sommes en train de vivre.

A l'occasion des 40 ans de l'INRIA, le philosophe Michel Serre tint une conférence sur les Sciences et Technologies de l'Information et de la Communication [1]. Selon l'académicien, nous vivons actuellement, avec l'invention de l'ordinateur, une troisième révolution de l'information, comparable aux inventions de l'écriture et de l'imprimerie qui entrainèrent de profondes modifications dans l'histoire de l'humanité.

1. Visionnable en intégralité à l'adresse http ://interstices.info/jcms/c_33030/les-nouvelles-technologies-revolution-culturelle-et-cognitive

Ces trois révolutions, que le philosophe exprimait au niveau du couplage entre le support et le message, ne nous semblent cependant pas complètement comparables en termes techniques. L'imprimerie a modifié la diffusion du savoir mais pas réellement le support, qui est resté le papier. L'ordinateur, lui, transforme complètement ce support et par conséquent le rapport que nous entretenons avec lui.

Pour appréhender cette tranformation, revenons à notre définition de la bande dessinée. Nous avons généralisé la définition de Scott McCloud en considérant comme récit graphique toute oeuvre dont l'organisation des éléments picturaux forme un itinéraire visuel, ce qui permet d'inclure les cas limites des cartes, des itinéraires, de l'image unique telle que le dessin de presse, ou encore l'expérimentale bande pas dessinée. Nous pouvons progresser d'un cran supplémentaire si, comme Scott McCloud, nous incluons dans les éléments picturaux l'alphabet, qui n'est rien d'autre qu'une abstraction extrême de la représentation, conservant le sens mais supprimant toute ressemblance. Nous pouvons dès lors considérer qu'un texte est une bande-dessinée. Et par conséquent, si la transformation du support doit avoir un effet sur tout document présentant un récit graphique, alors nous devons déja constater ces effets sur ce « noble sous-ensemble »qu'est le texte.

L'écriture commence déjà à se modifier avec les nouvelles technologies. On peut rapidement en pointer les évolutions négatives telles que le langage SMS ou certains commentaires qu'on lit sur les forums de discussions. On oublie alors de se réjouir du fait que, grâce à l'ordinateur, de plus en plus de personnes renouent avec l'écrit. Cette évolution n'est en rien un substitut en attendant la généralisation de la vidéo, et ceci pour deux raisons : d'une part, l'information écrite se transmet plus rapidement que l'information orale ; d'autre part, l'écrit permet de s'exprimer silencieusement. Prenons un exemple : vous souhaitez expliquer à un ami combien votre collègue de bureau vous exaspère. Par écrit, vous pourrez sans censure deverser votre bile, à la simple condition que ce dernier ne jette pas un coup d'oeil par dessus votre épaule. A l'oral, l'expression de vos sentiments sera plus délicate. Ce silence nous laisse dire que le SMS n'est en rien une étape vers le MMS. L'écrit retrouve droit de cité de manière non provisoire.

L'écriture s'enrichit. Les smileys ne sont pas une simple mode, ils viennent compléter la ponctuation et transmettre des émotions. Ces petits éléments picturaux font aujourd'hui partie intégrante de l'écriture numérique.

Cependant, l'écriture seule nous limite dans notre reflexion car ce mode d'expression reste strictement bi-dimensionel, toujours sur un plan strictement technique. Ainsi l'hypertexte et la navigation par lien ne sont

pas réellement nouveaux, la technologie ne faisant que faciliter le mécanisme des « livres dont vous êtes le héros ». Mais si nous enrichissons notre vocabulaire graphique au-delà de l'alphabet, nous basculons alors sur la représentation en 2D d'un univers de dimension supérieur. Sauf que, là où le papier nous contraignait à un point de vue arbitraire et à une projection unique, l'ordinateur nous offre la capacité de faire varier ce point de vue et de nous promener dans cette troisième dimension.

Là encore, l'évolution est en cours. De nombreux sites internet proposent aujourd'hui des applets java permettant de se promener dans un modèle 3D (voir par exemple le site de Carcassone). Dans le même temps, l'utilisation de Google Map et Google Street ne rend plus absurde l'idée de Borges d'une carte à l'échelle 1.

Et si nous rajoutons une dimension ?

On imagine facilement une promenade en trois dimensions, c'est notre quotidien. On pense sans mal au concept de changement de point de vue, c'est la métaphore de la caméra. Mais qu'en est-il du temps, cette dimension que nous ne faisons que subir ?

Nous pourrons choisir d'observer une période plus ou moins longue, de comparer plusieurs moments, de faire varier notre échelle temporelle. Un schéma d'assemblage deviendra un document dynamique que nous lirons sur notre téléphone. La carte de Charles-Joseph Minard deviendra dynamique. Nous observerons et étudierons l'histoire avec des outils nouveaux, et donc un oeil nouveau. Nous obtiendrons un nouvel outil de pédagogie. Les artistes s'empareront de ces techniques et produiront des oeuvres nouvelles, à mi-chemin entre la bande dessinée, le jeu vidéo et le cinéma. C'est aussi dans ce sens que va le courant du *Serious Gaming*, qui cherche à transposer dans des applications dites sérieuses les techniques de narration et d'interaction apparues dans le jeu vidéo. Les outils modernes d'établissement d'un portrait-robot en sont issus.

Le récit nait de la temporalité et de l'enchainement des évènements, et l'art du conteur réside dans la façon dont celui-ci met à disposition du lecteur/spectateur les éléments de l'histoire. En nous permettant d'interagir avec le temps et de nous promener librement dans cette dimension, l'informatique nous ouvre donc de nouvelles voies pour raconter des histoires et naviguer dans un récit. Et s'il est aujourd'hui impossible de dire quelles formes exactes adopteront ces perspectives, nous pouvons affirmer sans crainte que le siècle qui commence sera d'une richesse narrative qui n'aura rien à envier à ses prédécesseurs.

Bibliographie

[ACCO05] Jackie Assa, Yaron Caspi, and Daniel Cohen-Or. Action synopsis : pose selection and illustration. *ACM Trans. Graph.*, 24(3) :667–676, 2005.

[AHS05] K. Ali, K. Hartmann, and T. Strothotte. Label Layout for Interactive 3D Illustrations. *Journal of the WSCG*, 13(1) :1–8, 2005.

[APH+03] Maneesh Agrawala, Doantam Phan, Julie Heiser, John Haymaker, Jeff Klingner, Pat Hanrahan, and Barbara Tversky. Designing effective step-by-step assembly instructions. *ACM Trans. Graph.*, 22(3) :828–837, 2003.

[AS01] Maneesh Agrawala and Chris Stolte. Rendering effective route maps : improving usability through generalization. In *SIGGRAPH*, pages 241–249, 2001.

[ATR+08] T. Annen, H. Theisel, C. Rössl, G. Ziegler, and H.-P. Seidel. Vector field contours. In *Graphics Interface 2008*, pages 97–105, 2008.

[BEDT08] Hedlena Bezerra, Elmar Eisemann, Xavier Décoret, and Joëlle Thollot. 3d dynamic grouping for guided stylization. In *NPAR '08 : 6th International Symposium on Non-photorealistic Animation and Rendering*, pages 89–95. ACM, 2008.

[BF00] Blaine A. Bell and Steven K. Feiner. Dynamic space management for user interfaces. In *UIST '00 : Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 239–248, New York, NY, USA, 2000. ACM.

[BF08] Michael Burns and Adam Finkelstein. Adaptive cutaways for comprehensible rendering of polygonal scenes. In *SIGGRAPH Asia '08 : ACM SIGGRAPH Asia 2008 papers*, pages 1–7, New York, NY, USA, 2008. ACM.

[BFH01] Blaine Bell, Steven Feiner, and Tobias Höllerer. View management for virtual and augmented reality. In *UIST '01 : Proceedings of the 14th annual ACM symposium on User interface*

software and technology, pages 101–110, New York, NY, USA, 2001. ACM.

[BKSW05] Michael A. Bekos, Michael Kaufmann, Antonios Symvonis, and Alexander Wolff. Boundary labeling : Models and efficient algorithms for rectangular maps. In János Pach, editor, *Proc. 12th Int. Symposium on Graph Drawing (GD'04)*, volume 3383, pages 49–59. Springer, 2005.

[BTS05] Pascal Barla, Joëlle Thollot, and François Sillion. Geometric clustering for line drawing simplification, 2005.

[BZOP07] Simon Bouvier-Zappa, Victor Ostromoukhov, and Pierre Poulin. Motion cues for illustration of skeletal motion capture data. In *NPAR '07 : Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, pages 133–140. ACM, 2007.

[CL93] Brian Cabral and Leith Casey Leedom. Imaging vector fields using line integral convolution. In *SIGGRAPH '93 : Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 263–270, New York, NY, USA, 1993. ACM.

[CM02] Dorin Comaniciu and Peter Meer. Mean shift : A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5) :603–619, 2002.

[CMS95] Jon Christensen, Joe Marks, and Stuart Shieber. An empirical study of algorithms for point-feature label placement. *ACM Trans. Graph.*, 14(3) :203–232, 1995.

[Cro84] Franklin C. Crow. Summed-area tables for texture mapping. In *SIGGRAPH '84 : Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 207–212, New York, NY, USA, 1984. ACM.

[Cut02] James E Cutting. Representing motion in a static image : constraints and parallels in art, science, and popular culture. *Perception advance online publication*, 31(10) :1165–1193, 2002.

[Dai06] Eli Daiches. Dynamic map labeling. *IEEE Transactions on Visualization and Computer Graphics*, 12(5) :773–780, 2006. Member-Ken Been and Senior Member-Chee Yap.

[DMR05] R.D. Dony, J.W. Mateer, and J.A. Robinson. Techniques for automated reverse storyboarding. *Vision, Image and Signal Processing, IEEE Proceedings* -, 152(4) :425–436, Aug. 2005.

[Dov95] Don Dovey. Vector plots for irregular grids. In *VIS '95 : Proceedings of the 6th conference on Visualization '95*, page 248, Washington, DC, USA, 1995. IEEE Computer Society.

[DRB+08] Pierre Dragicevic, Gonzalo Ramos, Jacobo Bibliowitcz, Derek Nowrouzezahrai, Ravin Balakrishnan, and Karan Singh. Video browsing by direct manipulation. In *CHI '08 : Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 237–246, New York, NY, USA, 2008. ACM.

[Dur02] Frédo Durand. An invitation to discuss computer depiction. In *NPAR '02 : 2nd international symposium on Non-photorealistic animation and rendering*, pages 111–124, New York, NY, USA, 2002. ACM.

[FLS64] Richard P. Feynman, Robert B. Leighton, and Matthew Sands. *The Feynman Lectures on Physics*. Addison-Wesley, Reading, Massachusetts, 1964.

[FP99] Jean-Daniel Fekete and Catherine Plaisant. Excentric labeling : dynamic neighborhood labeling for data visualization. In *CHI '99 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 512–519, New York, NY, USA, 1999. ACM Press.

[GAL+09] Floraine Grabler, Maneesh Agrawala, Wilmot Li, Mira Dontcheva, and Takeo Igarashi. Generating photo manipulation tutorials by demonstration. *ACM Trans. Graph.*, 28(3), 2009.

[GASP08] Floraine Grabler, Maneesh Agrawala, Robert W. Sumner, and Mark Pauly. Automatic generation of tourist maps. *ACM Trans. Graph.*, 27(3), 2008.

[GCSS06] Dan B Goldman, Brian Curless, David Salesin, and Steven M. Seitz. Schematic storyboarding for video visualization and editing. *ACM Trans. Graph.*, 25(3) :862–871, 2006.

[GGTH07] Christoph Garth, Florian Gerhardt, Xavier Tricoche, and Hagen Hans. Efficient computation and visualization of coherent structures in fluid flow applications. *IEEE Transactions on Visualization and Computer Graphics*, 13 :1464–1471, 2007.

[GHS06] Timo Götzelmann, Knut Hartmann, and Thomas Strothotte. Agent-based annotation of interactive 3d visualizations. In *6 th International Symposium on Smart Graphics. Springer-Verlag (in print*, pages 24–35. Springer Verlag, 2006.

[GHS07] Timo Götzelmann, Knut Hartmann, and Thomas Strothotte. Annotation of animated 3d objects. In Thomas Schulze, Bernhard Preim, and Heidrun Schumann, editors, *SimVis'07*, pages 209–222. SCS Publishing House e.V., 2007.

[GLT+09] Christoph Garth, Guo-Shi Li, Xavier Tricoche, Charles D. Hansen, and Hans Hagen. Visualization of coherent structures in transient 2d flows. *Topology-Based Methods in Visualization II*, 13 :1–13, 2009.

[GPR+] H. Garcke, T. Preusser, M. Rumpf, A. Telea, U. Weikard, and J. van Wijk.

[Hal01] G. Haller. Distinguished material surfaces and coherent structures in three-dimensional fluid flows. *Physica D : Nonlinear Phenomena*, 149(4) :248 – 277, 2001.

[HAS04] Knut Hartmann, Kamran Ali, and Thomas Strothotte. Floating labels : Applying dynamic potential fields for label layout. In *Smart Graphics*, pages 101–113, 2004.

[HDS03] Denis Haumont, Olivier Debeir, and François Sillion. Volumetric cell-and-portal generation, 2003.

[HH91] James L. Helman and Lambertus Hesselink. Visualizing vector field topology in fluid flows. *IEEE Comput. Graph. Appl.*, 11(3) :36–46, 1991.

[HSC+05] Justin Hensley, Thorsten Scheuermann, Greg Coombe, Montek Singh, and Anselmo Lastra. Fast summed-area table generation and its applications. *Computer Graphics Forum*, 24(3) :547–555, 2005.

[HWHJ99] Bjoern Heckel, Gunther Weber, Bernd Hamann, and Kenneth I. Joy. Construction of vector field hierarchies. In *VIS '99 : Proceedings of the conference on Visualization '99*, pages 19–25, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.

[KCJ+09] Pushpak Karnick, David Cline, Stefan Jeschke, Anshuman Razdan, and Peter Wonka. Route visualization using detail

lenses. *IEEE Transactions on Visualization and Computer Graphics*, 99(RapidPosts) :235–247, 2009.

[KHG09] Sebastian Knödel, Martin Hachet, and Pascal Guitton. Interactive generation and modification of cutaway illustrations for polygonal models. In *SG '09 : Proceedings of the 10th International Symposium on Smart Graphics*, pages 140–151, Berlin, Heidelberg, 2009. Springer-Verlag.

[KHK03] Yuya Kawagishi, Kazuhide Hatsuyama, and Kunio Kondo. Cartoon blur : Non-photorealistic motion blur. *Computer Graphics International Conference*, 0 :276, 2003.

[LHS08] Liya Li, Hsien-Hsi Hsieh, and Han-Wei Shen. Illustrative streamline placement and visualization. In *Visualization Symposium, 2008. PacificVIS '08. IEEE Pacific*, pages 79–86, March 2008.

[LKJ+05] David H. Laidlaw, Robert M. Kirby, Cullen D. Jackson, J. Scott Davidson, Timothy S. Miller, Marco da Silva, William H. Warren, and Michael J. Tarr. Comparing 2d vector field visualization methods : A user study. *IEEE Transactions on Visualization and Computer Graphics*, 11(1) :59–70, 2005.

[LRA+07] Wilmot Li, Lincoln Ritter, Maneesh Agrawala, Brian Curless, and David Salesin. Interactive cutaway illustrations of complex 3d models. In *SIGGRAPH '07 : ACM SIGGRAPH 2007 papers*, page 31, New York, NY, USA, 2007. ACM.

[LS08] Aidong Lu and Han-Wei Shen. Interactive storyboard for overall time-varying data visualization. *Visualization Symposium, 2008. PacificVIS '08. IEEE Pacific*, pages 143–150, March 2008.

[LwS07] Liya Li and Han wei Shen. Image-based streamline generation and rendering. *IEEE Trans. Visualization and Computer Graphics*, 13 :630–640, 2007.

[MAD05] A. Mebarki, P. Alliez, and O. Devillers. Farthest point seeding for efficient placement of streamlines. *Visualization, 2005. VIS 05. IEEE*, pages 479–486, Oct. 2005.

[Mcc94] Scott Mccloud. *Understanding Comics*. Perennial Currents, April 1994.

[MS91] Joe Marks and Stuart Shieber. The computational complexity of cartographic label placement. Technical Report TR-05-91, 1991.

[MSS99] Maic Masuch, Stefan Schlechtweg, and Ronny Schulz. Speedlines : depicting motion in motionless pictures. In *SIGGRAPH 99 Conference abstracts and applications*, page 277. ACM, 1999.

[MTHG03] Oliver Mattausch, Thomas Theussl, Helwig Hauser, and Eduard Gröller. Strategies for interactive exploration of 3d flow using evenly-spaced illuminated streamlines. In *SCCG '03 : Proceedings of the 19th spring conference on Computer graphics*, pages 213–222, New York, NY, USA, 2003. ACM.

[ND05] Marc Nienhaus and Jürgen Döllner. Depicting dynamics using principles of visual art and narrations. *IEEE Computer Graphics and Applications*, 25(3) :40–51, 2005.

[NHAH03] Christopher Niederauer, Mike Houston, Maneesh Agrawala, and Greg Humphreys. Non-invasive interactive visualization of dynamic architectural environments. *ACM Trans. Graph.*, 22(3) :700, 2003.

[PD07] S. Paris and F. Durand. A topological approach to hierarchical segmentation using mean shift. *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007.

[POJC01] Gopal Pingali, Agata Opalach, Yves Jean, and Ingrid Carlbom. Visualization of sports using motion trajectories : providing insights into performance, style, and strategy. In *VIS '01 : Proceedings of the conference on Visualization '01*, pages 75–82. IEEE Computer Society, 2001.

[PRS97] Bernhard Preim, Andreas Raab, and Thomas Strothotte. Coherent zooming of illustrations with 3d-graphics and text. In *Proceedings of the conference on Graphics interface '97*, pages 105–113, Toronto, Ont., Canada, Canada, 1997. Canadian Information Processing Society.

[RT06] Guodong Rong and Tiow-Seng Tan. Jump flooding in gpu with applications to voronoi diagram and distance transform. In *I3D '06 : Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 109–116, New York, NY, USA, 2006. ACM.

[SCo8] Amit Shesh and Baoquan Chen. Efficient and dynamic simplification of line drawings. *Computer Graphics Forum*, 27(2) :537–545, April 2008.

[SJWSo8] Tobias Salzbrunn, Heike Jänicke, Thomas Wischgoll, and Gerik Scheuermann. The state of the art in flow visualization : Partition-based techniques. In *SimVis*, pages 75–92, 2008.

[SPA+09] Sara L. Su, Sylvain Paris, Frederick Aliaga, Craig Scull, Steve Johnson, and Frédo Durand. Interactive visual histories for vector graphics. Technical Report MIT-CSAIL-TR-2009-031, Massachusetts Institute of Technology, Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, June 2009.

[SRLo6] Ariel Shamir, Michael Rubinstein, and Tomer Levinboim. Generating comics from 3d interactive computer graphics. *IEEE Computer Graphics and Applications*, 26(3) :53–61, 2006.

[Str98] Thomas Strothotte. *Computational Visualization : Graphics, Abstraction, and Interactivity*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[Suo7] Sara L. Su. Visualizing, editing, and inferring structure in 2d graphics. In *Adjunct Proceedings of the 20th ACM Symposium on User Interface Software and Technology*, pages 29–32, 2007.

[TB96] Greg Turk and David Banks. Image-guided streamline placement. In *SIGGRAPH '96 : Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 453–460, New York, NY, USA, 1996. ACM.

[Tuf97] Edward R. Tufte. *Visual explanations : images and quantities, evidence and narrative*. Graphics Press, 1997.

[Tufo1] Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, Conn., 2nd ed. edition, 2001.

[TvW99] Alexandru Telea and Jarke J. van Wijk. Simplified representation of vector fields. In *VISUALIZATION '99 : Proceedings of the 10th IEEE Visualization 1999 Conference (VIS '99)*, Washington, DC, USA, 1999. IEEE Computer Society.

[VVAHo7] Ian Vollick, Daniel Vogel, Maneesh Agrawala, and Aaron Hertzmann. Specifying label layout style by example. In *UIST '07 : Proceedings of the 20th annual ACM symposium on*

User interface software and technology, pages 221–230, New York, NY, USA, 2007. ACM.

[YKP05] Xiangong Ye, D. Kao, and A. Pang. Strategy for seeding 3d streamlines. *Visualization, 2005. VIS 05. IEEE*, pages 471–478, Oct. 2005.

[ZSH96] Malte Zöckler, Detlev Stalling, and Hans-Christian Hege. Interactive visualization of 3d-vector fields using illuminated stream lines. In *VIS '96 : Proceedings of the 7th conference on Visualization '96*, pages 107–ff., Los Alamitos, CA, USA, 1996. IEEE Computer Society Press.

Titre Rendu narratif en synthèse d'images

Résumé Créer une séquence d´images qui résume une animation est une tache effectuée dans de nombreux domaines, allant de la bande-dessinée à la conception de schémas d´assemblage. Une telle représentation repose sur la projection de données 3D plus temps vers un support 2D, et induit inévitablement une perte d´informations. Compenser cette perte relève encore aujourd´hui du traitement au cas par cas, et est très généralement fait de manière manuelle. L´objectif de cette thèse est de développer des algorithmes automatisant cette tache et restant les plus généraux possible. Dans ce contexte, nous nous sommes intéressés à la visualisation en images d´animations de particules, en essayant d´identifier puis d'illustrer leurs mouvements généraux. Une animation se déroulant généralement dans un décor géométrique spécifique, nous avons proposé une méthode pour afficher ce décor en tenant compte du résumé de l'animation. Enfin, afin d´enrichir nos images d´informations textuelles ou contextuelles, nous avons étudié le problème de l´annotation dynamique d´une scène 3D et les difficultés de cohérence temporelle soulevées.

Title Narrative Rendering in Computer Graphics

Abstract Creating a sequence of images that illustrates an animation is a task that finds applications in many fields, from comics to assembly schemes. Such a representation is based on the projection of 3D plus time data to 2D and necessarily involves loss of information. Compensation for this loss is generally handmade and is currently performed on an individual case basis. The objective of this thesis is to develop algorithms to automate this task as generally as possible. In this context, we are interested in pictorial visualization of particle animations by identifying and illustrating their aggregate movement. For animations taking place in a specific geometrical context, we propose a method to display this context while taking into account the animation summary. Finally, to enhance our pictures with textual or contextual information, we studied the problem of dynamic labeling of 3D scenes and the associated problems of temporal discontinuities and incoherence.