Université Lumière Lyon 2
École Doctorale Informatique et Information pour la Société

Numéro : |_|_|_|_|_|_|_|_|

THÈSE

pour obtenir le grade de
Docteur en Informatique

présentée par
Antoine Vacavant

Géométrie discrète sur grilles irrégulières isothétiques

Soutenue le 4 décembre 2008

JURY

Rapporteurs	Eric Andres	Professeur des Universités	XLIM, Poitiers
	Edouard Thiel	Professeur des Universités	LIF, Marseille
	Walter G. Kropatsch	Professeur des Universités	TU Wien, Autriche
Examinateurs	Annick Montanvert	Professeure des Universités	Gipsa-Lab, Grenoble
	Jean Michel Jolion	Professeur des Universités	LIRIS, Lyon
Directeurs	Laure Tougne	Professeure des Universités	LIRIS, Lyon
	David Coeurjolly	Chargé de Recherches CNRS	LIRIS, Lyon

Université Lumière Lyon 2
École Doctorale Informatique et Information pour la Société

Numéro : |_|_|_|_|_|_|_|_|

THÈSE

pour obtenir le grade de
Docteur en Informatique

présentée par
Antoine Vacavant

Géométrie discrète sur grilles irrégulières isothétiques

Soutenue le 4 décembre 2008

JURY

Rapporteurs	Eric Andres	Professeur des Universités	XLIM, Poitiers
	Edouard Thiel	Professeur des Universités	LIF, Marseille
	Walter G. Kropatsch	Professeur des Universités	TU Wien, Autriche
Examinateurs	Annick Montanvert	Professeure des Universités	Gipsa-Lab, Grenoble
	Jean Michel Jolion	Professeur des Universités	LIRIS, Lyon
Directeurs	Laure Tougne	Professeure des Universités	LIRIS, Lyon
	David Coeurjolly	Chargé de Recherches CNRS	LIRIS, Lyon

Remerciements

Ca y est : le manuscrit est terminé ! Mais il me reste encore à remercier tous ceux qui, de près ou de loin, m'ont permis d'arriver jusqu'en thèse et d'aller jusqu'au bout . . .

Je commencerai par remercier mes deux directeurs de thèse, Laure et David, qui ont toujours su me guider, me conseiller et me soutenir pendant ces trois ans. Par contre, je crois que je ne les remercierai jamais assez d'avoir autant relu et corrigé (avec beaucoup de stylos rouges ϟ) mon manuscrit. J'espère pouvoir encore travailler avec eux dans le futur, car ce fut un véritable plaisir.

Ensuite, j'aimerais remercier tous les doctorants du LIRIS et d'ailleurs avec lesquels nous avons supporté ensemble ces années de dur labeur, dans la joie et la bonne humeur (?). Entre autres : Amélie, Anne-Laure, Brice, Céline, Guillaume, Jean-Marie, Lionel, Loris, Mathieu, Pierre, Régis, Rémi, Thibault, Tristan. Je souhaite bon courage aux petits nouveaux, que je connais moins malheureusement. Je pense aussi à Annick, David, Nicolas, et Robin avec lesquels j'ai discuté, collaboré, réfléchi (!?) pour résoudre pleins de casse-têtes de chercheurs. Une petite pensée va également à Thierry et à tous les autres membres du LASMEA de Clermont-Ferrand qui m'ont fait découvrir la recherche en image, et la rédaction d'articles pour avant-hier ☺.

Enfin, je remercie tous les membres de ma famille pour leur soutien et leurs encouragements. Ces dernières lignes vont à ma femme Sarah et au petit Jérémi qu'elle m'a donné, qui me remplissent la vie d'amour et de bonheur ☼.

Résumé

Les systèmes d'acquisition de données image en deux ou trois dimensions (2-D ou 3-D) fournissent généralement des données organisées sur une grille régulière, appelées données discrètes. Que ce soit pour la visualisation ou l'extraction de mesures, la géométrie discrète définit les outils mathématiques et géométriques pour de nombreuses applications.

Dans cette thèse, nous nous intéressons à l'adaptation des algorithmes de la géométrie discrète aux grilles irrégulières isothétiques. Ce modèle de grille permet de représenter de manière générique les structurations d'images en pixels ou voxels de taille et de position variable (ou cellules) : les grilles anisotropes, très répandues en imagerie médicale, les décompositions hiérarchiques telles que quadtree/octree, les techniques de compression comme le run length encoding, *etc.* Nous proposons d'étendre deux méthodologies largement étudiées pour analyser les formes discrètes à cette représentation : la reconstruction d'objets binaires complexes et la transformée en distance.

Pour réaliser la reconstruction topologique et géométrique d'objets irréguliers, nous construisons le graphe de Reeb discret irrégulier, qui résume la forme de l'objet traité. Ensuite, nous calculons une structure polygonale exacte. Ces deux représentations peuvent être rapidement mises à jour, lors du raffinement d'une cellule de l'objet, ou quand plusieurs cellules sont groupées ensemble. Nous proposons deux applications de ce système : la distinction de caractères ambigus dans un outil de reconnaissance de plaques minéralogiques, et l'approximation de courbes implicites planaires.

La transformée en distance d'une image régulière peut être calculée par de nombreux algorithmes, dont le point commun est généralement le calcul d'un diagramme de Voronoï discret. Dans cette thèse, nous étudions le calcul de cette décomposition sur grilles irrégulières isothétiques, et nous proposons un algorithme optimal linéaire (en le nombre de cellules de la grille) en 2-D. Nous avons également développé deux techniques extensibles à n dimensions, dont la complexité (temps et espace) dépend principalement de l'irrégularité de la grille traitée.

Mots clés : Géométrie discrète, grilles irrégulières isothétiques, reconstruction polygonale, arithmétique d'intervalles, transformée en distance, diagramme de Voronoï.

Abstract

The systems that perform the acquisition of two or three (2-D or 3-D) dimensional images usually provide data structured on a regular grid, also called discrete data. Those discrete objects are now efficiently handled: for measure extraction or visualization, discrete geometry defines geometrical and mathematical tools for many applications.

In this PhD thesis, our goal is to adapt algorithms developed in discrete geometry to irregular isothetic grids. This generic grid model allows to represent the image structures based on pixels or voxels with variable position and size (or cells): anisotropic grids, commonly used in medical imagery, hierarchical decompositions like quadtree/octree, the compression methods like the run length encoding, *etc.* We propose to extend two methodologies, widely studied for discrete shape analysis, to this representation: the reconstruction of complex binary objects and distance transformation.

To process the topological and geometrical reconstruction of irregular objects, we build the irregular discrete Reeb graph, that sums up the shape of the input object. Then, we compute and exact polygonal structure. Those two representations can be quickly updated, when a cell of the object is refined, or when several cells are grouped together. We propose two applications of our system: the distinction of ambiguous characters in a software for licence plate recognition, and planar implicit curve approximation.

The distance transformation of a regular image may be computed with many algorithms, their common point is generally that they build the discrete Voronoi diagram. In this PhD thesis, we study the computation of this decomposition on irregular isothetic grids and we propose an optimal and linear algorithm (in respect to the number of cells in the grid) in 2-D. We have also developed two algorithms extensible to n dimensions, which complexity (time et space) principally depends on the irregularity of the input grid.

Keywords: Discrete geometry, irregular isothetic grids, polygonal reconstruction, interval arithmetic, distance transformation, Voronoi diagram.

Table des matières

Table des figures

Liste des tableaux

Liste des Algorithmes

Introduction générale

Les systèmes d'acquisition de données image en deux ou trois dimensions (2-D ou 3-D) fournissent généralement des données organisées sur une grille régulière, appelées données discrètes. Depuis les années 60 et la naissance des écrans matriciels, notre capacité à représenter et manipuler des objets dans le monde discret (droites, cercles, *etc.*) est un challenge permanent, puisqu'elle permet d'analyser et de traiter les images. Par exemple, J. Bresenham a été l'un des premiers à étudier le tracé de droites [Bresenham, 1965] et de cercles [Bresenham, 1977] sur une grille régulière dans les années 60-70. Dans la même période ont également émergé des recherches où les données ne sont plus régulières. En imagerie médicale, le scanner tomodensitométrique est inventé par G. N. Hounsfield *et al.* [Peeters *et al.*, 1979] et permet de reconstruire en 3-D l'anatomie d'un patient. Cette reconstruction induit la création d'une grille anisotrope, où les voxels ont un côté plus grand que les autres suivant un axe. Des techniques de subdivision du plan et de l'espace (par exemple le quadtree [Finkel et Bentley, 1974]), ont vu le jour afin de faciliter entre autres les requêtes de localisation dans un ensemble de points. En plus de leurs applications dans les systèmes d'information géographiques, ces structurations irrégulières ont été appliquées à d'autres domaines de l'image et de la modélisation, grâce notamment aux travaux de H. Samet [Samet, 1990a, Samet, 1990b]. Par exemple, les grilles de subdivision [Jevans et Wyvill, 1989] ont été l'une des premières structures utilisées pour accélérer l'algorithme du lancer de rayons. Le point commun de toutes ces méthodologies de représentation du plan ou de l'espace est la grille irrégulière obtenue, constituée de pixels (voxels) dont les côtés (faces) sont alignés aux axes : une grille *irrégulière* et *isothétique*. Les éléments qui la constituent sont nommés *cellules*.

Que ce soit pour la visualisation, l'extraction de mesures ou l'analyse de formes, la *géométrie discrète* définit les objets élémentaires, et des outils mathématiques et géométriques adaptés pour de nombreuses applications. Cette théorie est née dans les années 60 avec les travaux de A. Rosenfeld [Rosenfeld et Pfaltz, 1966, Rosenfeld et Pfaltz, 1968], puis, dans les années 80, elle a véritablement été connue avec entre autres les travaux de J.-P. Réveillès [Réveillès, 1991]. La géométrie discrète [Coeurjolly *et al.*, 2007] s'est d'abord concentrée sur le modèle le plus répandu : la grille régulière. Dans ce cas, les outils développés sont supportés par l'arithmétique sur $\mathbb{Z}^2$, puisque les coordonnées des pixels sont des entiers. Puis, ces outils ont été étendus à d'autres grilles, citons principalement les grilles triangulaires, hexagonales, anisotropes (voir par exemple [Luczak et Rosenfeld, 1976, Freeman, 1979]).

Les travaux présentés dans cette thèse visent à accomplir une généralisation de certaines notions et de certains algorithmes utilisés en géométrie discrète aux grilles irrégulières isothétiques. Ainsi, nous nous intéressons aux notions élémentaires de la géométrie discrète (objets, distances discrètes), et nous développons des algorithmes en accord avec ce modèle. De plus, nous cherchons à établir un lien avec l'arithmétique d'intervalles.

Le manuscrit est structuré en trois parties, qui s'enchaînent de manière linéaire (voir la figure 1). La première partie a pour objectif de présenter les principales grilles irrégulières isothétiques et de montrer certaines de leurs utilisations à travers des applications. Nous y proposons un modèle générique global permettant de les représenter. Dans la partie II, nous

proposons d'étendre à ce modèle deux méthodologies largement étudiées pour analyser les formes discrètes : la reconstruction d'objets binaires complexes et la transformée en distance. Enfin, nous exposons dans la dernière partie des applications dans lesquelles nous employons le modèle et les outils que nous avons développés.

Organisation détaillée du document

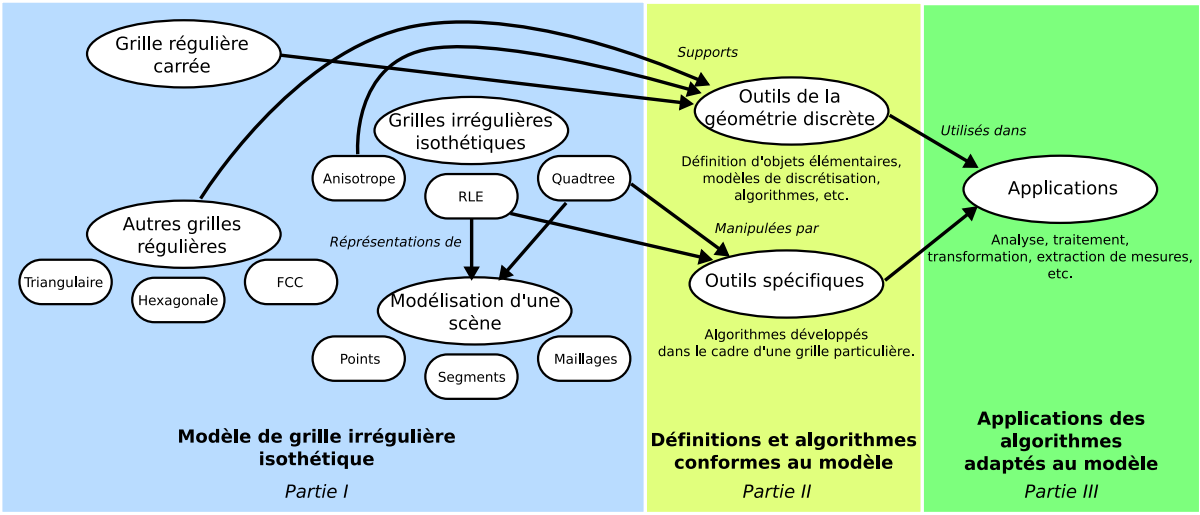


Fig 1: Schéma global d'organisation du document en trois parties. Les flèches indiquent les dépendances entre les tâches nécessaires aux applications classiques liées à l'image 2-D ou 3-D.

La première partie vise à décrire les structurations en grilles irrégulières isothétiques les plus répandues dans les domaines de l'image 2-D ou 3-D ; nous détaillons en particulier la technique du *run length encoding*, ainsi que les structures hiérarchiques *kd-tree* et *quad-tree, octree*. Puis, nous donnons quelques exemples de l'application de ces représentations : la simulation de phénomènes naturels par des approches multi-grilles, la modélisation géométrique par arithmétique d'intervalles et la modélisation d'environnements 3-D pour la réalité augmentée et les jeux vidéos. Le point commun de ces domaines est que les grilles irrégulières permettent d'accélérer les traitements et les algorithmes (par exemple, nous détaillons l'accélération de la simulation de fluides). Enfin, nous proposons un modèle générique qui permet de représenter toutes les grilles qui ont été présentées précédemment. Grâce à l'adaptation des outils mathématiques et géométriques de la géométrie discrète, nous exposons également comment manipuler la grille et les objets qu'elle contient.

Dans la seconde partie, nous développons dans un premier temps un système de reconstruction topologique et géométrique d'objets complexes sur grilles irrégulières isothétiques. Il permet de construire le graphe de Reeb discret irrégulier associé à l'objet, et

détermine une reconstruction polygonale exacte. Nous détaillons enfin comment mettre à jour ces deux représentations, lorsqu'une cellule de l'objet traité est subdivisée, ou lorsque plusieurs cellules sont groupées ensemble. Dans un deuxième temps, nous cherchons à calculer la transformation en distance d'une grille irrégulière isothétique. La transformée en distance d'une image régulière peut être calculée par de nombreux algorithmes, dont le point commun est généralement le calcul d'un diagramme de Voronoï discret. Dans cette thèse, nous étudions le calcul de cette décomposition sur grilles irrégulières isothétiques, et nous proposons un algorithme optimal linéaire (en le nombre de cellules de la grille) en 2-D. Nous développons également deux techniques extensibles à n dimensions, dont la complexité (temps et espace) dépend principalement de l'irrégularité de la grille traitée.

Dans la partie III, nous présentons trois applications où nous employons le modèle et les algorithmes que nous avons développés. Tout d'abord, nous étudions l'accélération de simulations de type Monte Carlo grâce à un codage de l'espace 3-D adaptés aux phénomènes simulés. Puis, dans le cadre d'une collaboration avec l'entreprise *Foxstream* (spécialisée dans la vidéo-surveillance), nous développons un algorithme d'analyse structurelle de caractères. Il permet de distinguer les lettres et chiffres ambigus (*e.g.* '8' et 'B') dans un logiciel de reconnaissance de plaques minéralogiques. Enfin, nous nous intéressons à l'approximation de courbes implicites. Grâce à l'arithmétique d'intervalles, il est en effet possible de construire un pavage du plan en rectangles isothétiques qui encadrent la courbe. À l'aide de cette décomposition, nous sommes ensuite capables de proposer une représentation topologique et une structure polygonale simple associée à la courbe.

Fig 2: Une œuvre de Richard Schur, *Anywhere Else*, 2006, acrylique sur toile de 340 cm x 340 cm, exposée dans la *Galerie der Künstler*, Munich, Allemagne. Source : <http ://www.richard-schur.de/ >

Première partie

Représentations et utilisations de grilles irrégulières isothétiques

SOMMAIRE DE LA PARTIE

CHAPITRE 1

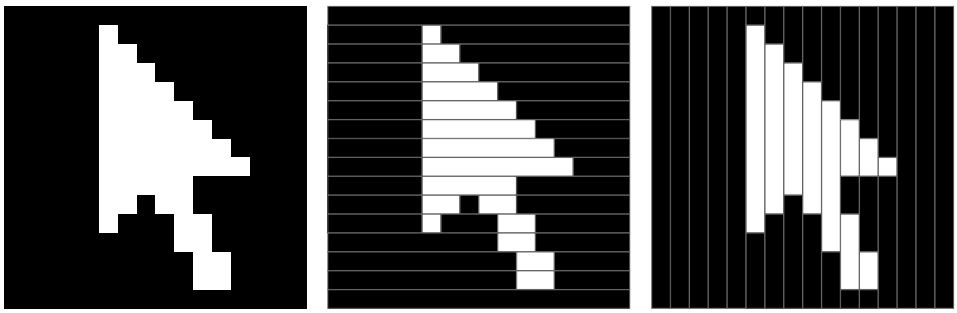
Structurations irrégulières isothétiques classiques et exemples de leurs applications

Dans ce chapitre, nous nous intéressons aux structurations sous forme de grilles irrégulières isothétiques, classiques dans les domaines de l'image. Plus précisément, nous décrivons en premier lieu une technique de codage simple pour compresser une image, puis les décompositions kd-tree et quadtree très répandues dans l'informatique graphique et la modélisation. Nous donnons également deux algorithmes qui permettent de réaliser ces structurations. Pour chaque structure, nous indiquons quelques applications possibles, dans les domaines liées à l'image et à la modélisation géométrique.

1.1 Structurations irrégulières isothétiques classiques

1.1.1 Compression d'images par regroupement de pixels

Une technique simple à implémenter pour compresser une image 2-D consiste à regrouper ensemble les pixels homogènes en une seule cellule, en choisissant un sens de parcours (par exemple suivant un axe de l'image). Dans toute cette partie, nous désignons par « pixels homogènes » des pixels « proches » suivant un critère défini à l'avance. Par exemple, on peut décider que deux pixels en niveaux de gris sont homogènes si leurs valeurs sont proches à ϵ près, où $0 \leq \epsilon \leq 255$. Cet codage, nommé *run length encoding* (ou RLE) est inspiré des travaux de S. W. Golomb [Golomb, 1966], et reste une référence dans la compression sans perte des données. Le format BMP (ou *bitmap*) d'images 2-D créé par Microsoft [Charlap, 1995] peut inclure un codage RLE afin d'en réduire la taille. On trouvera dans la figure 1.1 un exemple de codage RLE d'une image 2-D suivant les axes X et Y. Cette compression induit la construction de pixels de tailles variables suivant l'axe choisi. Par la suite (sans entrer dans un état de l'art exhaustif), la plupart



(a) (b) (c)

Fig. 1.1: Codage RLE de l'image binaire *cursor* suivant les axes X et Y respectivement en (c) et (d).

des algorithmes de compression sont basés sur une transformation qui permet de réduire l'information dans un autre espace. Ainsi, la norme JPEG (ou *joint photographic experts group*) [Wallace, 1991] propose d'utiliser une transformée en cosinus discrète (ou DCT pour *discrete cosine transform*), et de sélectionner ensuite ses coefficients pour compresser l'image de manière efficace. On peut observer que ces approches ne prennent plus en compte la géométrie des pixels de l'image. En fait, il ne s'agit plus de regrouper des pixels homogènes, comme dans le RLE. Généralement, ces algorithmes traitent l'image par blocs [1], mais ils n'induisent pas la création d'une *structure spatiale irrégulière* de l'image. Quelques travaux récents utilisent encore des regroupements irréguliers des pixels comme [Jackson *et al.*, 2007] avec une décomposition par *quadtree* (voir la section suivante pour plus de détails sur le quadtree).

Lorsque l'on traite des images 3-D, comme dans les applications en imagerie médicale, les compressions par grilles irrégulières peuvent être très utiles pour accélérer les traitements de ces images. En effet, celles-ci sont dans ce cadre très volumineuses, et une représentation par des regroupements de voxels permet par exemple de segmenter ces images efficacement [Droske *et al.*, 2001]. Nous détaillons dans la section 5 nos travaux sur l'accélération de simulation numérique grâce à un codage RLE de la géométrie d'un patient modélisée initialement sur une grille régulière de voxels. Nous allons nous intéresser dans la section suivante à des structurations qui impliquent la construction d'une arborescence. Comme le RLE, elles sont très utilisées en modélisation pour accélérer les traitements et compresser les images.

1.1.2 Structurations hiérarchiques de l'espace

Dans les années 70-80, de nombreuses recherches ont porté sur la manière de structurer un ensemble $\mathcal{S}$ de points, de segments, de polygones, *etc.* [de Berg *et al.*, 2000]. Étant donné un nouveau point p dans cet espace, un des objectifs de cette *structure de données spatiale* est d'accélérer la recherche de l'élément le plus proche de p dans $\mathcal{S}$ (on notera

[1] Dans JPEG, l'image est d'abord scindée en blocs de taille 8×8 ou 16×16 pixels.

cette requête $\mathtt{NN}(p)$). Une telle structure de données est généralement organisée sous forme d'une hiérarchie de volumes (ou d'une manière plus générale de *cellules*) qui permet de faciliter les tests d'inclusion. Pour effectuer $\mathtt{NN}(p)$ dans $\mathcal{S}$, il n'est plus utile de tester les n objets de $\mathcal{S}$. On parcourt la hiérarchie en cherchant dans quelle partie du plan se situe p, pour ensuite trouver l'élément de $\mathcal{S}$ le plus proche de p. Ces structures de données ont des applications diverses, comme le test de visibilité [de Berg *et al.*, 2000], la synthèse d'images [Foley *et al.*, 1997] ou le traitement d'images [Fu *et al.*, 2004]. Nous présentons dans cette section les structures les plus couramment utilisées dans ces domaines. Elles sont illustrées en 2-D, mais elles sont généralement simples à étendre aux dimensions supérieures. En particulier, nous verrons dans la section 1.2.2 une application de ces structures en 3-D.

Le k-dimensional tree (kd-tree)

Dans la version originale de J. L. Bentley [Bentley, 1975, de Berg *et al.*, 2000], le kd-tree d'un ensemble de points $\mathcal{S}$ est construit tout d'abord en subdivisant le plan par une droite parallèle à l'un des axes, et passant par un point de $\mathcal{S}$. De manière récursive, on continue à subdiviser une cellule de l'espace en considérant un nouveau point de $\mathcal{S}$ présent dans celle-ci. On alterne l'orientation de la droite (suivant l'axe des X ou des Y) à chaque itération. On décrit ainsi une hiérarchie de cellules rectangulaires que l'on représente également par un arbre binaire T. En posant $n = |\mathcal{S}|$, la recherche d'un élément dans une telle structure de données est en $\mathcal{O}(\log_2 n)$ (hauteur de l'arbre), et la construction est en $\mathcal{O}(n \log_2 n)$. Le kd-tree a ensuite été défini d'une manière différente pour d'autres ensembles d'objets. En particulier, si l'on considère un ensemble d'objets triangulés dans l'espace 3-D, on construit le kd-tree en choisissant toujours le plan médian [2] pour diviser une cellule en deux [Foley *et al.*, 1997, Gandoin et Devillers, 2002, Havran, 2000, Malgouyres, 2002, Szécsi et Benedek, 2002, Pharr et Humphreys, 2004]. Le choix du critère d'arrêt du processus de récursivité peut être la profondeur de l'arbre, ou la densité de triangles (ou de points) présents dans ses feuilles. Par exemple, si le nombre de triangles est inférieur à un seuil, on ne subdivise pas la cellule en deux. On peut également choisir de continuer jusqu'à ce qu'il n'y ait plus qu'un seul élément dans chaque feuille (voir l'algorithme 1.1 pour plus de détails). Pour distinguer les deux modes de partitionnement de l'espace que nous venons de décrire, on peut les nommer comme H. Samet [Samet, 1990a] PR-kd-tree pour le premier, *i.e. point region*-kd-tree, et BPR-kd-tree pour le second, *i.e. bucket point region*-kd-tree. Dans l'algorithme 1.1, nous avons décrit la découpe d'un nœud n, qui est associé à une cellule dont la taille en X est notée $[min_x(n), max_x(n)]$. Nous considérons alors le milieu de n suivant l'axe des X pour construire les nœuds fils n_1 et n_2 (on suit un raisonnement analogue pour les axes Y et Z).

La découpe du plan par un kd-tree permet de construire une grille irrégulière isothétique (voir figure 1.2 pour un exemple). Nous l'utiliserons dans la partie II, cha-

[2] Pour une subdivision suivant l'axe des X, et une cellule dont les bornes en X sont $[x_1, x_2]$, ce plan aura pour équation $x = (x_1 + x_2)/2$.

Algorithme 1.1 : Procédure récursive `subdiviser_kd_tree()` de construction du kd-tree [Havran, 2000].

entrée : un nœud n du kd-tree, de niveau l et l'axe de découpe actuel a
sortie : le nœud n est mis à jour et on appelle `subdiviser_kd_tree()` sur ses fils

```
début
    si n contient trop d'objets et l < l_max alors
        nœud n_1 ← n ;
        nœud n_2 ← n ;
        fils_1(n) ← n_1 ;
        fils_2(n) ← n_2 ;
        si a = X alors
            min_x(n_1) ← (min_x(n) + max_x(n))/2 ;
            max_x(n_2) ← (min_x(n) + max_x(n))/2 ;
            prochain axe de subdivision a' ← Y;
        sinon si a = Y alors // idem que pour X
        sinon si a = Z alors // idem que pour X
        pour chaque objet O ∈ n faire
            si O appartient à n_1 alors ajouter O aux objets référencés par n_1 ;
            si O appartient à n_2 alors ajouter O aux objets référencés par n_2 ;
        subdiviser_kd_tree(n_1, l + 1, a') ;
        subdiviser_kd_tree(n_2, l + 1, a') ;
fin
```

pitre 4 pour la génération aléatoire de grilles irrégulières binaires. De nombreuses recherches récentes en synthèse d'images ont permis d'adapter le kd-tree sur GPU (*graphical processor unit*) [Foley et Sugerman, 2005, Horn *et al.*, 2007]. Cette structure reste une référence dans ce domaine. Il existe aussi des variantes du kd-tree où l'on ne choisit pas forcément le plan médian pour couper une cellule [MacDonald et Booth, 1990] (voir la figure 1.3). Dans ce cas, il est nécessaire de calculer un critère permettant de caractériser la géométrie des objets représentés. D'autre part, le kd-tree est utilisé pour le traitement, la compression ou l'indexation d'images 2-D (voir par exemple [Al-Abudi *et al.*, 2005, Kubica *et al.*, 2005]) ou encore la compression de maillages 3-D [Gandoin et Devillers, 2002]. On peut enfin noter que le kd-tree est généralisable simplement à k dimensions (d'où son nom) [Bentley, 1975]. Dans ce cas, on cherche à subdiviser l'espace $\Omega \subset \mathbb{R}^k$ en un ensemble de sous-espaces convexes grâce à des hyper-plans (de dimension $k-1$). Le processus récursif de construction des hyper-plans reste similaire à celui que nous venons de détailler, c'est-à-dire que l'on créé une arborescence binaire de cellules de dimension k.

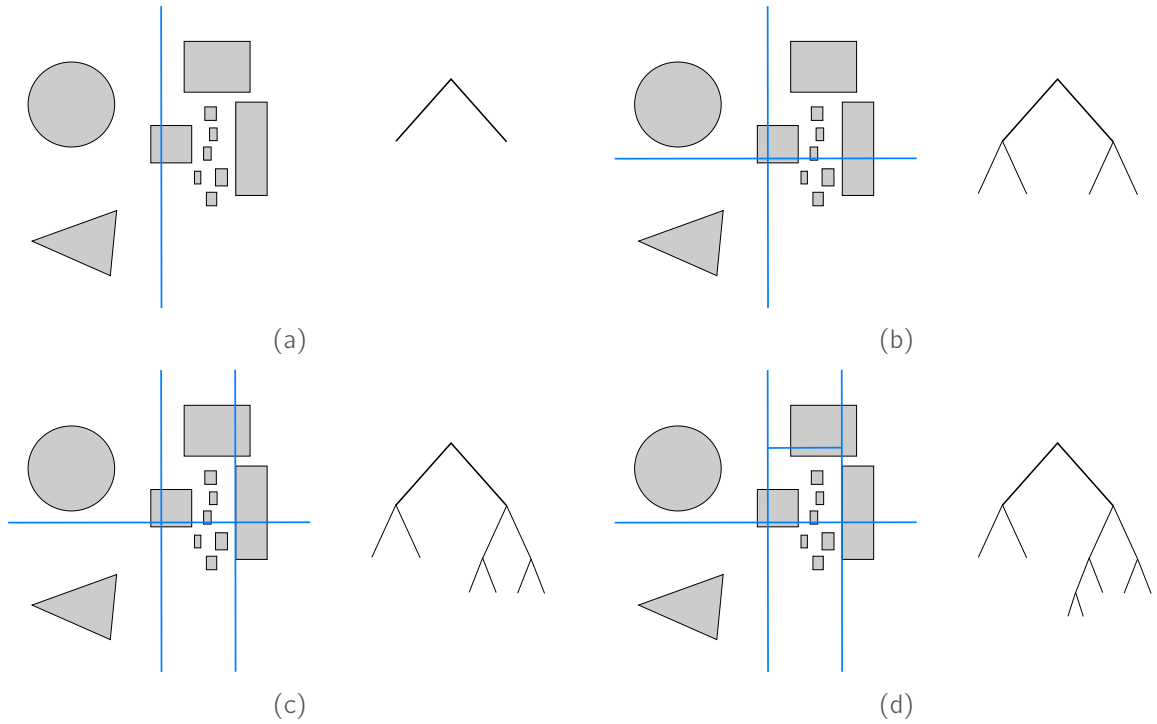


Fig. 1.2: Un exemple de découpe du plan par un *k*d-tree. A chaque étape, on représente également l'arbre binaire construit. Le critère d'arrêt peut être dans notre cas le nombre d'objets présents exclusivement dans la cellule.

Arbre quaternaire (quadtree), arbre octal (octree)

Le quadtree est une structure de données que l'on doit au départ à R. A. Finkel [Finkel et Bentley, 1974], mais qui a été approfondie par H. Samet [Samet, 1990a, Samet, 1990b]. Dans ses ouvrages, il a montré que le quadtree pouvait être utilisé dans des domaines variés, comme les systèmes d'information géographique (SIG) ou la synthèse d'images. La construction du quadtree est une subdivision du plan en une hiérarchie de

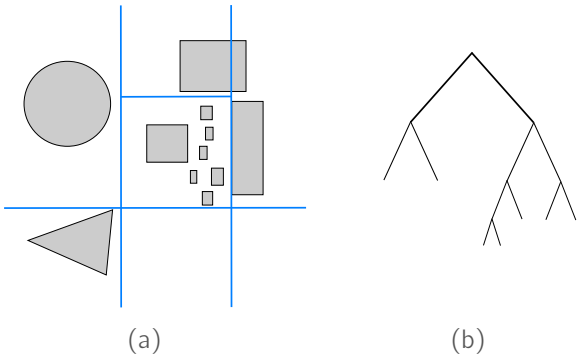


Fig. 1.3: Un *k*d-tree adaptatif, dont l'arbre est le même que dans la figure 1.2.

cellules, où l'on découpe à chaque étape en quatre sous-cellules. Comme dans le *k*d-tree, on choisit généralement les droites médianes parallèles aux axes.

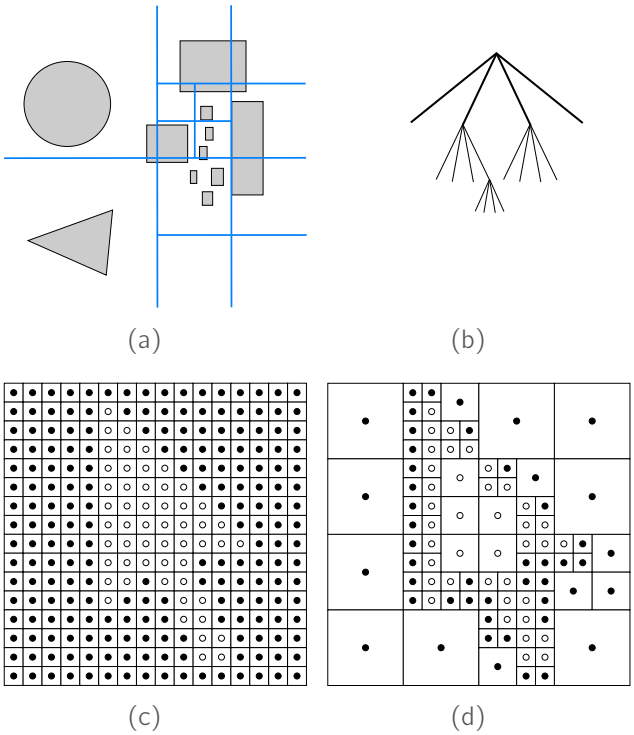


Fig. 1.4: En (a) et (b) sont présentés un exemple de découpe du plan par un quadtree. L'arbre correspondant est également représenté. En (c) et (d), nous avons illustré la même structure employée sur l'image *cursor*.

On peut énoncer quelques remarques sur le quadtree, analogues à celles sur le *k*d-tree. Tout d'abord, cette structure spatiale est très utilisée dans le traitement, la compression ou la segmentation d'images 2-D [Preusser et Rumpf, 1999, Droske *et al.*, 2001, D'Haene *et al.*, 2004, Jackson *et al.*, 2007]. Dans la figure 1.4, nous donnons un exemple de la décomposition en quadtree d'une image 2-D. Dans ce cas, lors de la découpe d'une cellule, on considère l'ensemble des pixels qu'elle contient. Si ces pixels sont homogènes, alors on ne subdivise pas cette cellule. Généralement, les cellules feuilles de la structure sont soit homogène et de taille quelconque, soit de taille 1 pixel. On peut distinguer là encore les différentes variantes de quadtrees grâce aux livres de H. Samet. Tout comme le *k*d-tree, il existe les PR-quadtrees et les BPR-quadtrees (voir section précédente pour plus de détails), et la décomposition d'une image 2-D est nommée R-quadtree pour *region*-quadtree (voir algorithme 1.2). Dans ce cas, le critère d'arrêt est l'homogénéité des pixels contenus dans la sous-image associée à un nœud de l'arbre. Dans cet algorithme, on note généralement la taille de `I` par des puissances de deux, par exemple $2^k \times 2^k$, pour faciliter le calcul de la taille des fils d'un nœud. Ainsi, il est facile de déterminer, à partir de la taille de `I`, la position et la taille d'une cellule, en prenant en compte sa place dans l'arbre.

Algorithme 1.2 : Procédure récursive `subdiviser_quadtree()` de construction du quadtree d'une image 2-D `I`.

entrée : un nœud n du quadtree, $\mathtt{I}_n$ est la sous-image de `I` associée à n
sortie : le nœud n est mis à jour et on appelle `subdiviser_quadtree()` sur ses fils

début
  si $\mathtt{I}_n$ *n'est pas homogène et* $|\mathtt{I}_n| > 1$ **alors**
    nœud $n_1 \leftarrow n$, $\mathtt{fils}_1(n) \leftarrow n_1$;
    nœud $n_2 \leftarrow n$, $\mathtt{fils}_2(n) \leftarrow n_2$;
    nœud $n_3 \leftarrow n$, $\mathtt{fils}_3(n) \leftarrow n_3$;
    nœud $n_4 \leftarrow n$, $\mathtt{fils}_4(n) \leftarrow n_4$;
    $mid_x \leftarrow \big(\min_x(n) + \max_x(n)\big)/2$;
    $mid_y \leftarrow \big(\min_y(n) + \max_y(n)\big)/2$;
    // n_1 représente le quart haut-gauche de n
    $\min_x(n_1) \leftarrow mid_x$;
    $\min_y(n_1) \leftarrow mid_y$;
    // n_2 représente le quart haut-droit de n
    $\max_x(n_2) \leftarrow mid_x$;
    $\min_y(n_2) \leftarrow mid_y$;
    // n_3 représente le quart bas-gauche de n
    $\min_x(n_3) \leftarrow mid_x$;
    $\max_y(n_3) \leftarrow mid_y$;
    // n_4 représente le quart bas-droit de n
    $\max_x(n_4) \leftarrow mid_x$;
    $\max_y(n_4) \leftarrow mid_y$;
    `subdiviser_quadtree`(n_1) ;
    `subdiviser_quadtree`(n_2) ;
    `subdiviser_quadtree`(n_3) ;
    `subdiviser_quadtree`(n_4) ;
  sinon si $\mathtt{I}_n$ *est homogène* **alors**
    calculer une couleur c représentante de n ;
    // on peut calculer la moyenne des pixels contenus dans $\mathtt{I}_n$
    `couleur`$(n) \leftarrow c$;
fin

Nous donnons dans la dernière section de ce chapitre une formalisation du calcul de ces éléments pour une cellule quelconque.

La version 3-D, ou octree, est également très répandue en synthèse d'images [Havran, 2000, Frisken et Perry, 2002, Knoll, 2006]. L'octree est construit en subdivisant chaque cellule en huit sous-cellules grâce à trois plans parallèles aux axes X, Y et Z. La complexité de la recherche d'un élément dans cette structure est en $\mathcal{O}(\log_8 n)$, et peut être optimisée à $\mathcal{O}(\log_2 l)$, où l est le nombre de feuilles grâce à un

octree linéaire [Gargantini, 1982]. Enfin, il est possible de moduler le placement des plans de découpe, comme dans [Whang *et al.*, 1995].

Nous avons dépeint les structurations en grilles irrégulières isothétiques les plus répandues dans les domaines de l'image. Ces structures ont des applications diverses : compression d'images, représentation d'un ensemble d'objets (points, segments, triangles entre autres) qui permettent d'accélérer la localisation, *etc.* Nous avons également évoqué la possibilité de les optimiser en fonction de l'application envisagée (par exemple construire un octree linéaire). Nous détaillons maintenant quelques applications des structurations dans des domaines liés à l'image et à l'informatique graphique.

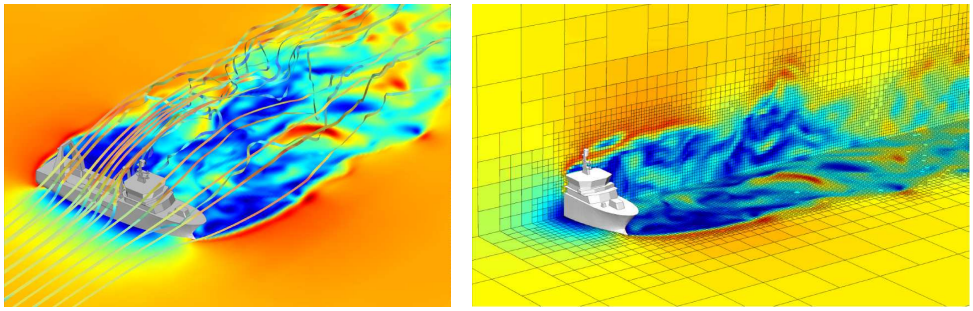
1.2 Exemples d'applications des grilles irrégulières isothétiques en image et modélisation

1.2.1 Grilles irrégulières dans les méthodes numériques et la modélisation géométrique

Dans cette section, nous allons nous intéresser à l'utilisation des structures de données que nous venons de décrire dans (1) les méthodes numériques, qui ont pour but de résoudre des systèmes d'équations complexes, et (2) la modélisation géométrique, où les courbes et surfaces implicites peuvent être rendues grâce à des techniques de subdivision du plan et de l'espace.

Méthodes numériques et simulation de phénomènes naturels

Les approches par grilles irrégulières sont aujourd'hui les plus rapides pour calculer les solutions des équations aux dérivées partielles, des équations intégrales, *etc.* [Plewa *et al.*, 2005]. Dans cette classe d'approches, on parle également de schémas multi-résolution, ou encore multi-grilles. En général, on utilise des schémas de subdivision comme ce qui est réalisé dans un octree [Popinet, 2003] ou dans un *k*d-tree [Ham et Young, 2003]. Ces techniques ont pour but de simuler des phénomènes naturels, régis par ces types d'équations, tels que les gaz [Pruess et Garcia, 2000], les liquides [Ham et Young, 2003, Popinet, 2003, Losasso *et al.*, 2006], la fumée [Losasso *et al.*, 2004], *etc.* Dans un espace de résolution discret donné, on calcule en chaque point, temps par temps la « valeur » du phénomène (chaleur, pression, quantité de fluide ou de fumée, *etc.*). Cette approche par grille régulière consommant beaucoup de ressources, on fait appel à d'autres représentations, sous forme de grilles irrégulières isothétiques. L'objectif principal est de réaliser des calculs précis uniquement dans les parties de l'espace où ils sont nécessaires. Dans la figure 1.5, nous montrons un exemple d'utilisation d'un octree pour résoudre les équations d'Euler. Dans ce cas, elles servent à simuler le vent qui contourne un bateau. La principale problématique que soulèvent les approches par grilles irrégulières est le moyen



(a) (b)

Fig. 1.5: La simulation du vent par les équations d'Euler dans [Popinet, 2003]. L'utilisation d'un octree (b) permet d'accélérer la simulation du phénomène schématisé en (a). La couleur représente la vitesse (de valeur croissante du bleu au rouge).

de calculer le transfert du phénomène entre cellules, qui ne sont pas de la même taille. Dans [Losasso *et al.*, 2004, Losasso *et al.*, 2006], les auteurs proposent de résoudre avec un octree les équations de Navier Stokes pour les fluides incompressibles

$$\mathbf{u}_t + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \mathbf{f} \tag{1.1}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{1.2}$$

où $\mathbf{u} = (u, v, w)$ est le champ de vitesse, $\mathbf{f}$ est l'ensemble des forces extérieures et p représente la pression. Rappelons que l'opérateur ∇ est le gradient d'un vecteur (*i.e.* $\nabla \mathbf{u} = \overrightarrow{grad}(\mathbf{u})$). Le but de ces travaux est de calculer le gradient, la pression, et finalement l'écoulement du fluide en lui-même (variable $\mathbf{u}$) sur l'octree pour accélérer les calculs. Ainsi, les auteurs montrent comment le fluide peut passer d'une cellule de niveau k dans l'octree à des cellules voisines de niveau $k+1$. Dans l'exemple pris en figure 1.6, l'équation

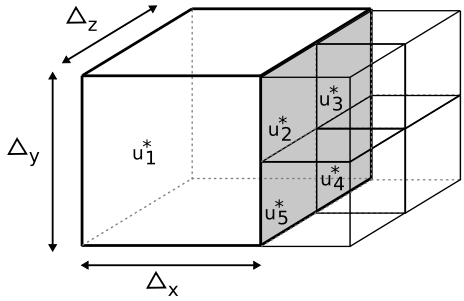


Fig. 1.6: Comment transférer le fluide entre des cellules de tailles différentes. Ici, il s'agit de considérer une cellule de niveau k qui a une face commune avec quatre cellules de niveau $k+1$ dans l'octree. Le fluide défini à l'interface est noté $\{\mathbf{u}_i^*\}_{i=1,\dots,4}$ pour chacune des quatre cellules. Les auteurs de [Losasso *et al.*, 2004, Losasso *et al.*, 2006] montrent comment calculer le fluide passant à travers $\{\mathbf{u}_i^*\}_{i=1,\dots,4}$ et $\mathbf{u}_1^*$.

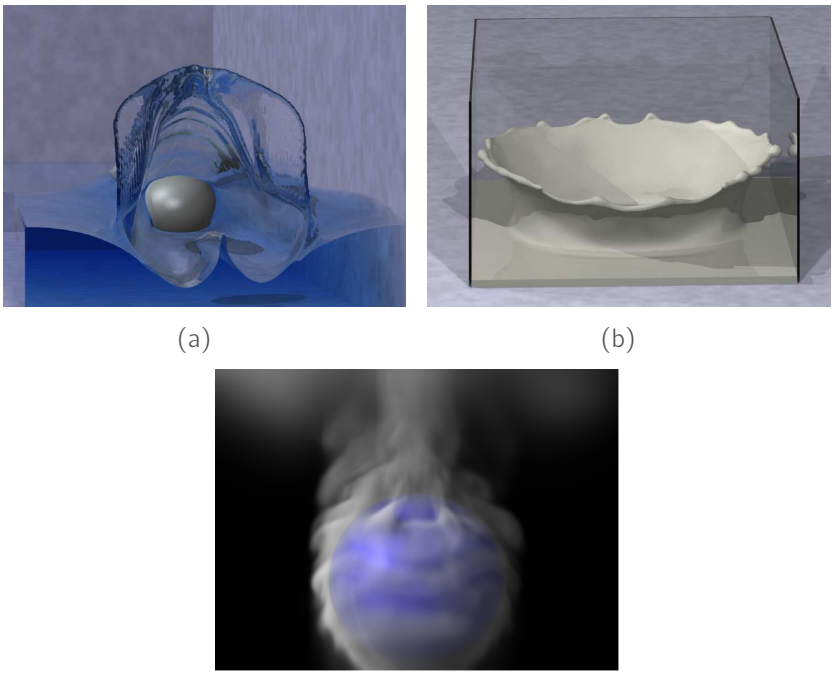
précédente peut se résumer à

$$V_{cell}\nabla \cdot \mathbf{u}^* = \sum_{faces} (\mathbf{u}_f^* \cdot \mathbf{n}) A_f, \tag{1.3}$$

où V_{cell} est le volume de la plus grosse cellule, $\mathbf{n}$ le vecteur unité issu de cette cellule, et f représente l'indice de la face traitée. Enfin A_f est la surface de la face d'indice f. Dans le cas de la figure 1.6, la discrétisation de la divergence suivant l'axe des X est

$$\frac{\partial \mathbf{u}^*}{\partial x} = \frac{1}{\Delta_x}\left(\frac{1}{4}(\mathbf{u}_2^* + \mathbf{u}_3^* + \mathbf{u}_4^* + \mathbf{u}_5^*) - \mathbf{u}_1^*\right), \tag{1.4}$$

et se calcule de manière équivalente dans les autres axes. Grâce à cette ré-écriture, les auteurs enchaînent en adaptant le calcul de la pression sur l'octree, et donnent enfin l'application de leur système pour la simulation de la fumée et de l'eau (voir figure 1.7).



(a) (b)

(c)

Fig. 1.7: Les résultats de simulation de phénomènes naturels illustrés dans [Losasso *et al.*, 2004, Losasso *et al.*, 2006]. On peut voir en (a) un ellipsoïde traversant de l'eau, en (b) une couronne de lait, et en (c) une fumée contournant une sphère.

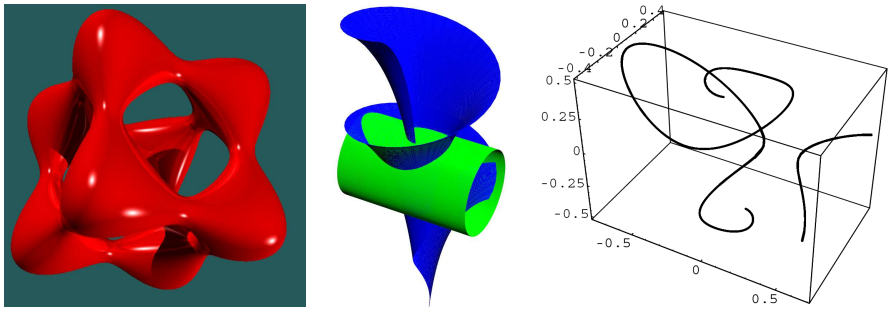
Dans le chapitre 5 de la partie III, nous décrivons une contribution réalisée lors de notre stage de Master 2 recherche au Centre Léon Bérard. Nous étudions l'accélération des simulations de type Monte Carlo grâce à des objets géométriques plus adaptés pour modéliser le patient lors d'une séance de radiothérapie. Nous y proposons une étude où une image acquise par un examen scanner tomodensitométrique d'un patient est d'abord

représentée par une grille régulière classique pour être ensuite insérée dans un simulateur de radiothérapie. On groupe les voxels par intensité grâce à un RLE pour compresser les données de l'image, et ainsi accélérer le processus stochastique. Nous présentons finalement une expérimentation où les temps de la simulation adaptée à cette nouvelle géométrie sont sensiblement réduits et nous garantissons une précision satisfaisante dans les calculs issus de cette séance d'irradiation virtuelle.

Modélisation géométrique par arithmétique d'intervalles

Nous nous intéressons ici à la modélisation de *courbes et surfaces implicites*, *i.e.* définies par les solutions d'une équation $f(x, y) = 0,\ f : \mathbb{R}^2 \to \mathbb{R}$ et $f(x, y, z) = 0,\ f : \mathbb{R}^3 \to \mathbb{R}$ respectivement. L'enjeu des recherches actuelles est de représenter ces objets grâce à une polygonalisation ou un maillage correct aux sens géométrique et topologique [Boissonnat et Teillaud, 2006]. En effet, l'approximation des courbes et surfaces peut être utilisée par la suite pour les visualiser rapidement ou réaliser une approximation des opérations booléennes (intersection, union, *etc.*) entre elles. Une manière de construire une approximation efficace de ces objets est d'utiliser *l'arithmétique d'intervalles* introduite par R. E. Moore [Moore et Yang, 1959, Moore, 1966] dans les années 60-70. Cette arithmétique redéfinit les opérateurs classiques et des fonctions plus complexes en considérant des intervalles contenant la valeur exacte (inconnue). Ces intervalles sont définis par deux nombres représentables en virgule flottante. Ce principe permet d'éviter les erreurs induites par l'arithmétique flottante, en encadrant les quantités réelles manipulées. En combinant des opérateurs et des fonctions redéfinies ainsi, une fonction d'inclusion peut être associée à f. Avec cette fonction, on peut tester si un hyper-intervalle 2-D intersecte ou non la courbe implicite (et un hyper-intervalle 3-D pour une surface implicite). Ensuite, on peut récursivement subdiviser un hyper-intervalle initial en un ensemble d'hyper-intervalles qui intersectent la courbe ou la surface. Enfin, on calcule une approximation en segments de droites ou de plans en joignant les points d'intersection avec les hyper-intervalles. Dans la figure 1.8 est illustrée l'utilisation de l'analyse par intervalles pour deux applications : le tracé d'une surface implicite, et le calcul de l'intersection de deux surfaces. Les applications de cette technique ont été initialement décrites par J. M. Snyder [Snyder, 1992b], dont les travaux restent une référence en informatique graphique.

Nous donnons dans le chapitre 7, partie III, de plus amples explications sur le tracé de courbes implicites grâce à l'analyse d'intervalles. Nous proposons également une méthode originale de tracé basée sur les outils de géométrie discrète irréguliers que nous allons exposer dans la suite de ce mémoire de thèse. En effet, l'ensemble d'hyper-intervalles construit peut être assimilé à une grille irrégulière isothétique. Comme ces intervalles sont construits par une subdivision en deux ou en quatre sous-intervalles, ils sont finalement semblables à une décomposition par kd-tree ou par quadtree respectivement. Dans la figure 1.9, la courbe implicite donnée par la fonction $f : (x, y) \to y^2 - x^3 + x$ est approximée grâce à l'algorithme de [Lopes *et al.*, 2001], basé sur une décomposition en quadtree du plan.



(a) (b)

Fig. 1.8: Deux exemples d'application de l'analyse en intervalles pour l'approximation de surfaces implicites. En (a) est illustré le tracé d'une surface implicite rendue par un lancer de rayons [Stolte, 2005], et en (b) le calcul de l'intersection entre deux surfaces [Bühler, 2001].

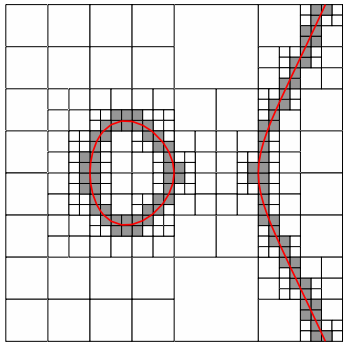
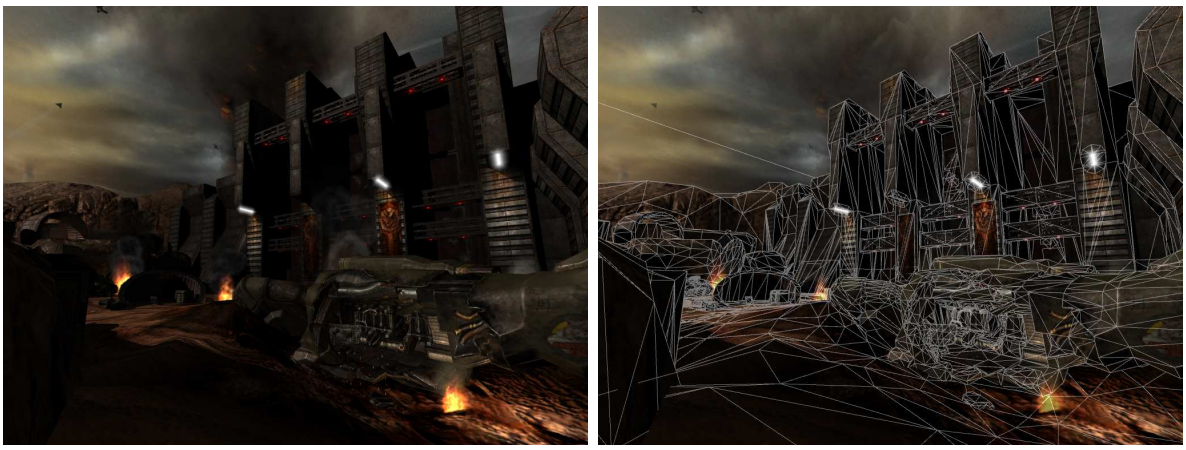


Fig. 1.9: Un exemple de tracé de courbe implicite avec la méthode de [Lopes *et al.*, 2001]. Nous avons choisi la fonction $f : (x, y) \rightarrow y^2 - x^3 + x$ dans le domaine $[-2.0, 2.0] \times [-2.0, 2.0]$. En gris sont représentés les hyper-intervalles qui intersectent la courbe, et les segments de droite qui l'approximent également.

1.2.2 Structures spatiales accélératrices dans un environnement 3-D pour la réalité augmentée et les jeux vidéos

Comme nous l'avons vu précédemment, les requêtes géométriques nécessaires à la manipulation d'un environnement concernent généralement les *tests de visibilité* (peut-on voir un objet d'un point de vue donné ?), *de proximité* (par exemple deux objets sont-ils proches dans l'espace ?), et *d'inclusion* (quels sont les objets inclus dans un autre ?). Grâce aux structures spatiales hiérarchiques, ces requêtes sont significativement accélérées. Nous décrivons ici de manière détaillée des problèmes classiques de l'informatique graphique, de la réalité augmentée et du jeu vidéo ainsi que les tests qu'elles impliquent. On considère ici que les objets sont représentés par des maillages triangulaires, modèle très utilisé dans les jeux vidéos (voir la figure 1.10). Cette étude a été réalisée dans le cadre de notre monitorat, et plus exactement pour réaliser un enseignement sur les structures spatiales

utiles dans les algorithmes d'informatique graphique et les jeux vidéos. Il a été dispensé aux étudiants de Master 2 professionnel « Programmation et Développement » des formations Gamagora [3].



(a) (b)

Fig. 1.10: Une scène de Quake 4 et sa structure triangulée sous-jacente. Source : <http ://www.kamarov.com/levels_quake4.html>

Deux algorithmes classiques de l'informatique graphique nécessaires aux jeux vidéos

Dans les jeux vidéos, le *test de collision* fait partie des requêtes qu'il faut implémenter avec soin, car il conditionne le fonctionnement global du jeu, et permet une interface précise avec le joueur. L'algorithme de base consiste à tester l'intersection entre les triangles des deux objets testés P et Q, jusqu'à ce que l'intersection soit trouvée. On peut approximer ces objets grâce à des volumes englobants : l'enveloppe convexe, la sphère, la boîte englobante alignée aux axes (ou AABB pour *axis-aligned bounding box*), *etc.* Une fois ce test vérifié, on peut ensuite calculer exactement l'intersection entre P et Q. La procédure de test et de calcul de collision devient complexe quand on ne connaît pas *a priori* quels objets de l'espace peuvent s'intersecter (tests d'inclusion et de proximité).

Que ce soit dans une cinématique, un film d'animation ou en pré-calcul d'un environnement de jeu vidéo, *l'algorithme de lancer de rayons* est une technique performante de rendu d'images de synthèse réalistes. Le principe général peut être résumé de la manière suivante (voir figure 1.11). On place tout d'abord les objets et les sources de lumières dans la scène à représenter, et on initialise un plan image I et un centre de projection c (point de vue de l'utilisateur). On lance ensuite un rayon r par pixel p de I en traçant la demi-droite $[cp)$. Dès que r rencontre un objet P de la scène, trois nouveaux rayons sont créés : un rayon de calcul de la couleur de P modifiée par les sources lumineuses, un rayon réfléchi et un rayon réfracté (si l'objet est transparent par exemple). A chaque fois qu'un

[3] http ://gamagora.univ-lyon2.fr/

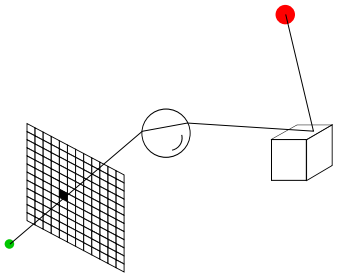


Fig. 1.11: Illustration du comportement de l'algorithme de lancer de rayons. Le rayon a touché un ou plusieurs objets, et une source lumineuse : le pixel p a la couleur de l'objet le plus proche, transformée.

rayon est lancé dans la scène, il faut déterminer quels sont les objets que ce rayon peut atteindre ensuite (test de visibilité).

Comparatif des structures accélérant ces algorithmes

Nous comparons brièvement les hiérarchies précédentes générant une grille irrégulière isothétique avec d'autres structures très utilisées pour répondre aux problématiques citées ci-dessus (voir aussi figure 1.12) :

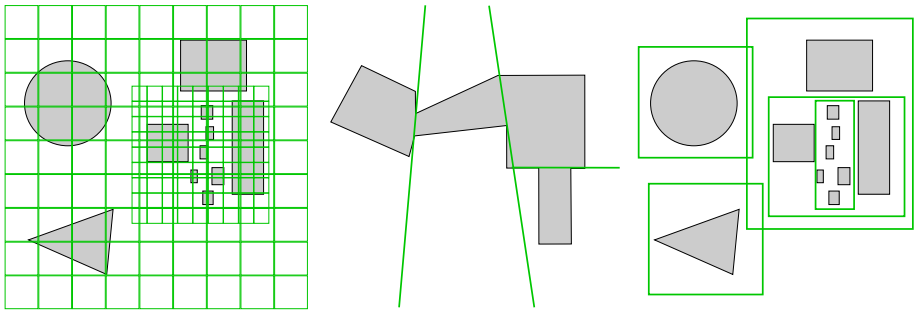


Fig. 1.12: Exemples d'autres structurations n'engendrant pas une grille irrégulière isothétique. (a) et (c) représentent une HGU et une HVE (avec des AABB) du plan. (b) est un BSP tree construit sur une scène de bâtiment. On remarque que les plans de coupe sont alignés avec les issues des pièces, afin d'accélérer les requêtes de parcours de la scène.

- ▷ Les *hiérarchies de grilles uniformes* (HGU) [Cazals *et al.*, 1995, Klimaszewski et Sederberg, 1997, Jevans et Wyvill, 1989, Dmitriev, 2000] sont des approches simples pour discrétiser l'espace 3-D. Dans la grille uniforme, chaque cellule peut contenir ou non les objets de l'espace. On peut par la suite employer des grilles de résolution plus fines pour structurer de manière plus précise une partie de l'espace qui contient de nombreux objets.

- ▷ Le *BSP tree* (pour *binary space partitioning tree*) [Fuchs *et al.*, 1980], où l'on découpe récursivement l'espace 3-D en deux sous-espaces grâce à un plan placé de manière à répartir les objets de manière équilibrée. Ce plan est déterminé grâce à la face d'un objet par exemple. Cette description de l'espace est très répandue dans les jeux vidéos (Doom, Quake 3, *etc.*).
- ▷ La *hiérarchie de volumes englobants* (ou HVE) [Rubin et Whitted, 1980] permet de décrire une hiérarchie de volumes prédéfinis (sphères, AABB, *etc.*) soit de manière manuelle, soit en utilisant des méthodes de classification pour regrouper les objets de la scène proches les uns des autres.

Les caractéristiques de chacune de ces structures dans le cadre d'un environnement 3-D (dynamique ou non) sont résumées dans la table 1.1. Nous avons également ajouté les optimisations possibles pour les différents critères choisis. D'une manière générale,

Structure	Occupation mémoire	Tests de collision	Traversée de la structure	Mise à jour
HGU	o	+	+	o
HVE	+	o	o	+
BSP tree	+		+	
*k*d-tree	+	+	+	
Octree	o	+	o	o

Tab. 1.1: Comparatif simplifié des structures de données spatiales dans un environnement 3-D. Les "+" indiquent les points forts de chaque structure, dans leur description initiale. Chaque signe "o" indique une amélioration possible grâce à des recherches récentes (voir le texte pour plus de détails).

on peut remarquer que les structures arborescentes (BSP tree, *k*d-tree et octree) ne sont pas très adaptées pour une modélisation dynamique de la scène. En effet, mise à part la représentation par un *loose octree* [Thatcher, 2000], la modification de l'arbre est une tâche complexe. L'octree est une structure qui a été longuement étudiée, et qui peut être aujourd'hui très efficace pour le lancer de rayons [Frisken et Perry, 2002]. De plus, l'occupation mémoire peut être optimisée de nombreuses manières [Knoll, 2006] comme dans le cas de l'octree linéaire [Gargantini, 1982]. Le *k*d-tree reste une référence par son codage léger et les opérations élémentaires qui sont rapides [Havran, 2000]. Le BSP tree permet une navigation rapide dans la scène et une réponse efficace au test de visibilité [de Berg *et al.*, 2000, Fu *et al.*, 2004]. Par contre, les cellules sont finalement représentées par des polyèdres convexes, et non des volumes simples, ce qui ralentit les tests d'intersections. La mise à jour des HUG est simple (déplacement d'un objet d'une cellule à une autre). Malheureusement, cette structure contient de nombreuses cellules vides, et requiert généralement plus de mémoire que nécessaire. Pour résoudre ce problème de place mémoire, on peut recourir à un codage par une table de hachage, dont les clés représentent la position des objets de la scène [Teschner *et al.*, 2003]. Enfin, la HVE contient des chevauchements entre volumes englobants, ce qui ralentit les tests de collision et le parcours. Cette représentation a connu des avancées récentes im-

portantes pour les opérations de collision et de parcours pour les HVE (respectivement dans [Larsson et Akenine-Möller, 2006] et [Wald *et al.*, 2007]). De plus, cette structure est très simple et peut être mise à jour rapidement [Kovalcík et Tobola, 2005].

De cette étude, nous avons montré que des recherches méritaient d'être menées pour comparer et améliorer les structures accélératrices pour les jeux vidéos. Ces structures ont été longuement étudiées pour l'algorithme de lancer de rayons par exemple. Il serait nécessaire qu'un comparatif à l'échelle d'un ou plusieurs jeux vidéos soit réalisé, pour connaître de manière plus pragmatique les structures les plus adaptées aux calculs nécessaires dans ces jeux. En effet, les jeux vidéos nécessitent des données volumineuses, et attendent différentes manipulations qui peuvent être significativement accélérées grâce à des outils optimisés. Les structures générant des grilles irrégulières isothétiques (kd-tree et octree) semblent pouvoir répondre à ces problématiques de manière très efficace.

1.3 Conclusion

Nous avons décrit dans cette partie les structurations classiques en grilles irrégulières isothétiques en 2-D et en 3-D que l'on trouve dans la littérature, ainsi que quelques applications utilisant ces grilles. Nous avons pris pour premier exemple les méthodes numériques, avec une étude plus détaillée de la simulation de fluides par [Losasso *et al.*, 2004]. Puis, nous avons dépeint rapidement la problématique de la modélisation de courbes et surfaces implicites, grâce à l'arithmétique d'intervalles. Ces aspects sont expliqués plus en détails dans le chapitre 7 de la partie III. Enfin, nous avons étudié finement l'emploi des structures hiérarchiques dans la réalité augmentée et les jeux vidéos. Elle a permis également de citer les différentes optimisations possibles de ces structures.

Dans le chapitre suivant, nous proposons un modèle générique qui permet de représenter toutes les grilles que nous avons étudiées jusqu'à présent. Grâce à ce modèle, nous sommes ensuite capables de caractériser et de manipuler les objets contenus dans une telle grille. Ces éléments seront ensuite employés pour développer des algorithmes adaptés, très répandus dans l'analyse d'images et de formes, dans la partie II.

CHAPITRE 2

Modélisation générique pour l'analyse d'images 2-D sur grilles irrégulières isothétiques

Dans ce chapitre, nous nous intéressons à présenter deux théories qui se complètent, car elles répondent à des problèmes différents avec des outils mathématiques et algorithmiques adaptés. Nous présentons d'abord la théorie du signal non-uniforme, qui permet de généraliser des outils de transformation et des représentations d'images 2-D sur des modèles irréguliers. Comme cette thématique est éloignée de la géométrie discrète, nous n'avons pas développé d'outils issus de cette théorie sur grille irrégulière isothétique. Néanmoins, nous tenons à présenter cette modélisation, car elle répond à des problématiques complémentaires à celles que nous rencontrons dans cette thèse. Dans la seconde section, nous exposons la théorie que nous avons développée pour traiter, transformer et analyser des grilles irrégulières isothétiques grâce à la géométrie discrète. Nous présentons les éléments de base et les structures de données nécessaires à ces tâches.

2.1 Théorie du signal non-uniforme

Dans cette section, nous prenons un point de vue plus général grâce à l'étude d'une théorie du signal généralisée. En premier lieu, considérons qu'une image 2-D régulière $\mathtt{I}$ peut être représentée comme l'échantillonnage d'une fonction continue (le *signal*) u : $\mathbb{R}^2 \to \mathbb{R}$ avec une *brosse de Dirac*

$$\mathtt{I}(x, y) = \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} u(x, y).\delta(x - m\Delta x, y - n\Delta y) \tag{2.5}$$

où la brosse de Dirac est définie par

$$\delta_{\Delta x \Delta y} = \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} \delta(x - m\Delta x, y - n\Delta y). \tag{2.6}$$

On notera que Δx et Δy sont les espacements des impulsions de Dirac δ en X et Y respectivement. A partir de cette formulation, on peut savoir quel échantillonnage choisir pour convertir le signal continu u en une forme discrète $\mathtt{I}$. En fait, cette forme doit permettre de retrouver le signal u initial, grâce à une reconstruction. Il faut en premier lieu rappeler la transformée de Fourier discrète (ou DFT pour *discrete Fourier transform*). Ce processus, noté F, permet de représenter une image 2-D $\mathtt{I}$ de taille $M \times N$ dans le domaine fréquentiel :

$$F(u, v) = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} \mathtt{I}(x, y) e^{-2\pi j(\frac{ux}{M} + \frac{vy}{N})}, \tag{2.7}$$

et on peut retrouver l'image par la transformée de Fourier discrète inverse (ou IDFT pour *inverse discrete Fourier transfom*) de la manière suivante :

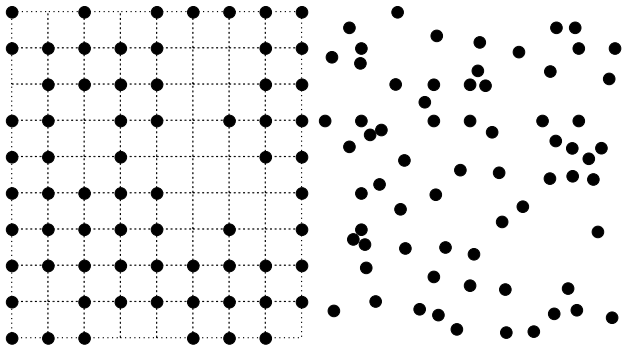
$$\mathtt{I}(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{2\pi j(\frac{ux}{M} + \frac{vy}{N})}. \tag{2.8}$$

On suppose que le signal est à *bande limitée*, c'est-à-dire que sa représentation dans le domaine fréquentiel est contenue dans un rectangle de côté $2u_{max}2v_{max}$ (*i.e.* $|F(u, v)| = 0$ pour $|u| > u_{max} \wedge |v| > v_{max}$). Si Δx et Δy respectent les conditions de Shannon-Nyquist [Nyquist, 1928]

$$\frac{1}{\Delta x} > 2u_{max} \wedge \frac{1}{\Delta y} > 2v_{max}, \tag{2.9}$$

alors toute l'information est conservée, et l'image écrite par l'équation 2.8 est bien définie.

Nous allons nous intéresser maintenant à la représentation des images issues d'un signal *non-uniforme*, c'est-à-dire échantillonnées de manière non régulière (comme dans la figure 2.13). Dans ce cas, il est nécessaire d'adopter une nouvelle formalisation du signal. Bien que la notion de pavage du plan par des cellules irrégulières disparaisse dans cette modélisation et laisse place à une représentation impulsionnelle, elle permet néanmoins de généraliser les outils que nous venons de décrire. Cela pourrait être ensuite appliqué sans problème aux images discrétisées sur des grilles irrégulières isothétiques. Le problème de la définition d'un signal non-uniforme a été longuement étudié pour la reconstruction d'images partielles ou altérées. H. J. Landau [Landau, 1967] (voir également [Grochenig et Razafinjatovo, 1996]) fut certainement le premier à exposer ces problématiques, sous l'étude de l'échantillonnage et de l'interpolation de fonctions entières. Dans la plupart des approches récentes [Strohmer, 1993, Vázquez, 1999, Early et Long, 2001, Marvasti, 2001, Ramponi et Carrato, 2001, Fadili *et al.*, 2007], à



(a) (b)

Fig. 2.13: Les problèmes du manque d'échantillons et des échantillons épars. Dans un cas (a), les données sont incomplètes, mais alignées sur une grille régulière (en pointillés), dans l'autre (b), les données sont placées de manière chaotique, sans structure régulière sous-jacente.

partir du signal incomplet, on cherche à combler itérativement les échantillons manquants en considérant éventuellement les propriétés du signal (valeurs maximales, minimales, convexité, durée, *etc.*). On peut également inclure des propriétés liées aux textures présentes dans l'image, comme dans les techniques récentes d'*image inpainting* [Bertalmío *et al.*, 2000, Fadili *et al.*, 2007], c'est-à-dire « combler les parties manquantes d'une image ». Un exemple de reconstruction d'une image bruitée est donné dans la figure 2.14 avec un algorithme de T. Strohmer [Strohmer, 1993]. Dans la figure 2.15, nous présentons un exemple de résultat d'une technique d'in-painting issue de [Fadili *et al.*, 2007].

(a) (b) (c)

Fig. 2.14: Une illustration de la reconstruction d'une image altérée par T. Strohmer [Strohmer, 1993]. L'image (b) est issue de *lena* (a) avec 60177 points choisis aléatoirement. L'auteur propose de reconstruire cette image irrégulière avec une erreur de 6% (c).

(a) (b) (c)

Fig. 2.15: Technique de in-painting d'une image issue de [Fadili *et al.*, 2007]. L'image (b) est construite à partir de *lena* (a) avec un masque, où 80% des pixels sont manquants (en noir). L'image (c) est l'image finalement reconstruite.

Une première représentation d'un signal non-uniforme consiste à considérer un ensemble d'échantillons distribués non uniformément dans $[0, M-1] \times [0, N-1] \cap \mathbb{Z}^2$. Dans [Marvasti, 2001], on peut lire que si l'on réécrit l'équation 2.8 avec un polynôme trigonométrique de la forme

$$p(x', y') = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{2\pi j(ux' + vy')}, \tag{2.10}$$

ce qui signifie qu'une image `I` s'écrit

$$\mathtt{I}(x, y) = p(x/M, y/N). \tag{2.11}$$

Cette transformation permet d'utiliser des outils mathématiques *continus*, et de considérer l'ensemble des points échantillons de départ dans le domaine continu. Enfin, cela signifie encore que l'on peut résoudre la reconstruction d'un signal issu d'échantillons manquants (mais distribués sur une grille régulière) ou d'échantillons éparses (sans structures particulières). On peut se référer à la figure 2.13 pour un exemple de ces échantillonnages irréguliers. Dans cet ouvrage, on peut lire également d'autres méthodes de reconstructions basées sur des modèles différents, comme les *frames*. Ce modèle permet également d'étendre le théorème de Shannon-Nyquist sur des images irrégulières (voir [Vázquez, 1999] pour un état de l'art). On peut enfin noter que la DFT non-uniforme y est clairement expliquée. Dans ce cas, on se base sur la représentation des échantillons par une séquence $x[u, v]$ de taille M, N. Sans rentrer dans les détails, la formulation générale de la NDFT (*non-uniform DFT*) d'une telle séquence 2-D est alors basée sur sa transformée en z :

$$\widehat{F}(z_{1k}, z_{2k}) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} x[u, v] z_{1k}^{-u} z_{2k}^{-v}; \;\; k = 0, 1, \ldots, MN-1. \tag{2.12}$$

De cette égalité très générale, les auteurs de [Marvasti, 2001] indiquent comment calculer la NDFT sur des séquences avec une forme plus contrainte. Par exemple, dans le cas d'échantillons placés sur des droites parallèles. D'autres modèles ont été étendus aux images irrégulières, et permettent de les définir pour les reconstruire. Citons par exemple les ondelettes et les bandelettes [Bernard et le Pennec, 2002, le Pennec et Mallat, 2005].

Dans cette section, nous avons mis en évidence des travaux issus de la théorie du signal non-uniforme qui permettent de généraliser des outils définis sur des images régulières. Ces modélisations irrégulières pourraient aider à généraliser les traitements d'images classiques (segmentation, détection de contours, *etc.*) comme ce qui a été réalisé pour la compression d'images [le Pennec et Mallat, 2005] ou de vidéos [Marvasti, 2001]. Dans cette thèse, nous nous concentrerons sur la généralisation des outils issus de la géométrie discrète sur des grilles irrégulières pour l'analyse dans le domaine spatial avec des images 2-D.

2.2 Géométrie discrète sur grilles irrégulières isothétiques

2.2.1 Éléments de base

En géométrie discrète, les maillages réguliers sont les fréquemment étudiés [Coeurjolly *et al.*, 2007], car (1) ils représentent la majorité des images, et (2) l'arithmétique définie sur $\mathbb{Z}^2$ (et plus généralement sur $\mathbb{Z}^n$) permet de manipuler les objets sur la grille. En effet, les coordonnées des pixels sont des entiers, et il est plus facile et plus stable de conserver les calculs nécessaires dans $\mathbb{Z}^2$. Une première modélisation des grilles régulières consiste à considérer le *pavage* des pixels dans le plan. Ainsi, plusieurs formes de pixels peuvent être considérées, et les plus étudiées sont sans aucun doute les grilles carrées, triangulaires et hexagonales (voir figure 2.16). Une autre manière de considérer

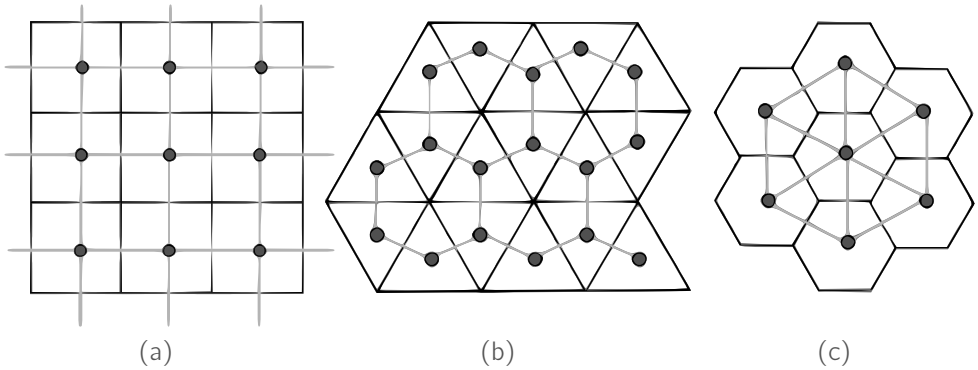


Fig. 2.16: Exemples de pavages réguliers en 2-D. De gauche à droite : la grille carrée, triangulaire et hexagonale. Le maillage associé est illustré en gris clair.

les grilles régulières est de représenter le *maillage* de la grille, qui est un graphe dont les sommets sont les centres des pixels, et les arêtes sont les relations d'adjacence entre eux

(on parle également de graphe d'adjacence). Ainsi, on peut noter que la grille pavée par des triangles est supportée par un maillage hexagonal et réciproquement. Pour la grille carrée, le maillage correspondant est lui aussi carré.

2.2.2 Un modèle général de $\mathbb{I}$-grille

Dans cette partie de la thèse, nous cherchons à modéliser les grilles irrégulières isothétiques, et à définir les différents objets qu'elle peut contenir. Ainsi, nous donnons la définition suivante d'une telle grille (ou $\mathbb{I}$-grille) :

Définition 2.1 ($\mathbb{I}$-grille 2-D)**.** *Soit $I \subset \mathbb{R}^2$ un support fermé rectangulaire. Une $\mathbb{I}$-grille 2-D $\mathbb{I}$ est un pavage de I en cellules rectangulaires (donc sans chevauchement), dont les côtés sont parallèles aux axes du repère (OXY). La position (x_R, y_R) et la taille (l_R^x, l_R^y) d'une cellule R de $\mathbb{I}$ est définie sans contrainte. On a donc :*

$$(x_R, y_R) \in \mathbb{R}^2,\ (l_R^x, l_R^y) \in \mathbb{R}^2. \tag{2.13}$$

Les bords gauche, droit, haut et bas d'une cellule R sont notés respectivement R^L, R^R, R^T et R^B. Par exemple, le bord gauche R^L de R a pour abscisse $x_R - (l_R^x/2)$, et le bord haut R^B a pour ordonnée $y_R - (l_R^y/2)$. À partir de cette définition très générale de $\mathbb{I}$-grille, nous définissons deux classes de relations d'ordre total :

Définition 2.2 (Relations d'ordre total basées sur les centres)**.** *Soient R_1 et R_2 deux cellules d'une $\mathbb{I}$-grille $\mathbb{I}$. On définit les deux relations d'ordre total $\preceq_x$ et $\preceq_y$ sur les centres des cellules de la manière suivante :*

$$\forall R_1, R_2 \in \mathbb{I},\ R_1 \preceq_x R_2 \Longleftrightarrow x_{R_1} < x_{R_2} \vee (x_{R_1} = x_{R_2} \wedge y_{R_1} \leq y_{R_2}), \tag{2.14}$$

$$\forall R_1, R_2 \in \mathbb{I},\ R_1 \preceq_y R_2 \Longleftrightarrow y_{R_1} < y_{R_2} \vee (y_{R_1} = y_{R_2} \wedge x_{R_1} \leq x_{R_2}). \tag{2.15}$$

Définition 2.3 (Relations d'ordre total basées sur les bords)**.** *Soient R_1 et R_2 deux cellules d'une $\mathbb{I}$-grille $\mathbb{I}$. On définit les deux relations d'ordre total $\preceq_{lx}$ et $\preceq_{ly}$ sur les bords des cellules de la manière suivante :*

$$\forall R_1, R_2 \in \mathbb{I},\ R_1 \preceq_{lx} R_2 \Longleftrightarrow R_1^L < R_2^L \vee \left(R_1^L = R_2^L \wedge R_1^B \leq R_2^B\right), \tag{2.16}$$

$$\forall R_1, R_2 \in \mathbb{I},\ R_1 \preceq_{ly} R_2 \Longleftrightarrow R_1^B < R_2^B \vee \left(R_1^B = R_2^B \wedge R_1^L \leq R_2^L\right). \tag{2.17}$$

Le choix du bord gauche et du bord haut dans ces deux dernières relations est arbitraire. Des définitions équivalentes pourraient être énoncées avec les bords droit et bas des cellules pour les ordonner. D'une manière générale, il est clair que ces relations sont des relations d'ordre total, car, si l'on considère par exemple $\preceq_x$, on a

$\triangleright$ $\forall R \in \mathbb{I},\ R \preceq_x R$ (*relation réflexive*),

▷ $\forall (R_1, R_2) \in \mathbb{I}^2,\ R_1 \preceq_x R_2 \wedge R_2 \preceq_x R_1 \Longrightarrow R_1 = R_2$ (*relation antisymétrique*),
▷ $\forall (R_1, R_2, R_3) \in \mathbb{I}^3,\ R_1 \preceq_x R_2 \wedge R_2 \preceq_x R_3 \Longrightarrow R_1 \preceq_x R_3$ (*relation transitive*),
▷ $\forall (R_1, R_2) \in \mathbb{I}^2,\ R_1 \preceq_x R_2 \vee R_2 \preceq_x R_1$ (*relation d'ordre total*).

On peut également noter que les quatre relations $\preceq_x$, $\preceq_y$, $\preceq_{lx}$ et $\preceq_{ly}$ sont des ordres lexicographiques sur $(\mathbb{R}, \leq) \times (\mathbb{R}, \leq)$. De plus, comme le pavage des cellules est borné par un support 2-D fermé, nous savons qu'il existe une borne supérieure et une borne inférieure pour chaque coordonnée. Cela implique qu'il existe, pour chaque relation, une cellule de borne inférieure unique et une cellule de borne supérieure unique [1]. On peut constater que ces notions s'adaptent aux grilles régulières, puisque dans ce cas, deux pixels peuvent être comparés par les mêmes relations, qui impliquent un ordre lexicographique sur $(\mathbb{Z}, \leq) \times (\mathbb{Z}, \leq)$. Nous avons illustré dans la figure 2.17 la distinction entre les relations $\preceq_y$ et $\preceq_{ly}$ pour une $\mathbb{I}$-grille. Dans cette thèse, plutôt que de manipuler le graphe d'adjacence

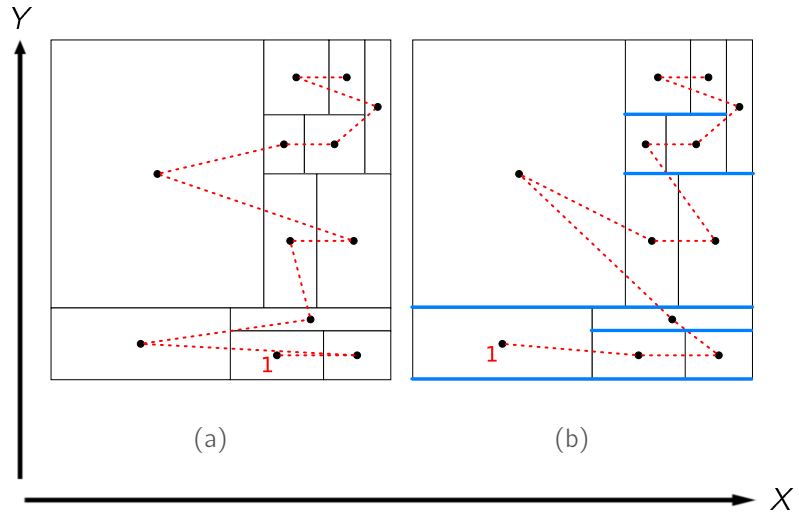


Fig. 2.17: Distinction entre les relations d'ordre sur $\mathbb{I}$-grilles. En (a) est présenté l'ordre lexicographique (en pointillés) induit par la relation $\preceq_y$, le « 1 » indiquant la première cellule dans l'ordre. De même, nous avons présenté en (b) l'ordre par $\preceq_{ly}$. Dans ce cas, les lignes en gras indiquent les bords bas de cellules (R^B pour une cellule R) qui régissent cet ordre.

d'une $\mathbb{I}$-grille $\mathbb{I}$, nous nous basons sur un ordre lexicographique pour manipuler les cellules de $\mathbb{I}$. Cet ordre est implémenté sous forme de liste chaînée, ce qui permet un parcours simple de la structure. Supposons maintenant que l'on choisisse l'ordre $\preceq_y$ sur $\mathbb{I}$ (des raisonnements analogues pourraient être énoncés pour les autres ordres). Les opérations élémentaires suivantes sont nécessaires aux algorithmes que nous développons dans la partie II :

▷ La procédure `suivant`(R), pour tout R dans $\mathbb{I}$, permet de connaître la cellule suivante R' dans l'ordre lexicographique $\preceq_y$. Cette opération est réalisée en $\mathcal{O}(1)$.

[1] On peut par conséquent affirmer que chacune de ces relations permet de construire un treillis représentant l'ordre des cellules sur $\mathbb{I}$.

- ▷ $\texttt{min}(\mathbb{I})$ et $\texttt{max}(\mathbb{I})$ renvoient respectivement la plus petite cellule et la plus grande cellule de $\mathbb{I}$ suivant $\preceq_y$. Ces deux opérations sont effectuées en temps contant $\mathcal{O}(1)$.
- ▷ À partir des deux éléments précédents, nous pouvons parcourir toutes les cellules de $\mathbb{I}$, suivant l'ordre $\preceq_y$. En particulier, nous pouvons accéder à la liste des cellules dont le centre (un bord pour une relation basée bord) est aligné avec $\alpha \in \mathbb{R}$. Nous notons $\mathcal{E}_\alpha$ cette liste, qui est déterminée au pire cas en $\mathcal{O}(n)$, si N est le nombre total de cellules dans $\mathbb{I}$. De plus, si l'on note $R_1 = \texttt{min}(\mathbb{I})$ et $R_n = \texttt{max}(\mathbb{I})$, nous accédons directement à $\mathcal{E}_\alpha$, lorsque $\alpha = y_{R_1}$, et lorsque $\alpha = y_{R_n}$.
- ▷ Soit maintenant une liste $\mathcal{E}_\alpha$. Grâce à la procédure `suivant()`, il est facile d'accéder à la liste des cellules $\mathcal{E}_\beta$, telle que $\beta > \alpha$ et la première cellule de $\mathcal{E}_\beta$ est la suivante de la dernière cellule de $\mathcal{E}_\alpha$. Ainsi, si nous avons les ensembles $\mathcal{E}_{\alpha_1}, \ldots, \mathcal{E}_{\alpha_T}$, nous pouvons déterminer en temps constant les T valeurs successives de α sur $\mathbb{I}$ (*i.e.* les ordonnées possibles du centre des cellules pour $\preceq_y$). Plus précisément, par un système de listes chaînées, nous stockons les différents ensembles $\mathcal{E}_{\alpha_1}, \ldots, \mathcal{E}_{\alpha_T}$, avec $\alpha_1, \ldots, \alpha_T \in \mathbb{R}, \alpha_1 < \alpha_2 < \cdots < \alpha_{T-1} < \alpha_T$. Et, passer de $\mathcal{E}_{\alpha_t}$ à $\mathcal{E}_{\alpha_{t+1}}$, $t \in \{1, T-1\}$, est réalisé en $\mathcal{O}(1)$.

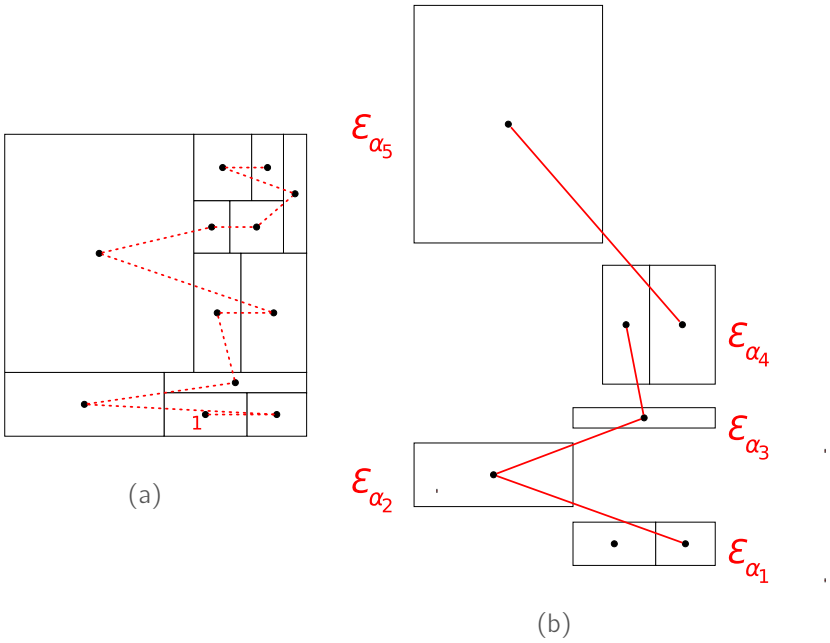


Fig. 2.18: Structure de listes chaînées permettant de représenter la relation $\preceq_y$ entre les cellules. Nous illustrons en (b) les premières listes $\mathcal{E}_{\alpha_1}, \ldots, \mathcal{E}_{\alpha_5}$ associées à l'ordre des cellules en (a).

Bien qu'un parcours linéaire en appelant `suivant()` puisse être suffisant pour manipuler une $\mathbb{I}$-grille, dans la suite de la thèse, nous avons développé des algorithmes basés sur notre structure de listes, comme dans l'algorithme 2.3. Il permet lui aussi de parcourir en temps linéaire les cellules de $\mathbb{I}$ suivant un ordre lexicographique donné (ici nous avons encore choisi $\preceq_y$ sans perte de généralité), et de réaliser des opérations sur chaque liste $\mathcal{E}_{\alpha_t}$. Dans la

partie II de cette thèse, nous détaillons les opérations réalisées sur $\mathcal{E}_{\alpha_t}$ en fonction des besoins de l'algorithme développé. L'algorithme 2.3 est similaire à une approche *scanline*

Algorithme 2.3 : Algorithme générique `parcourir()` de parcours d'une $\mathbb{I}$-grille $\mathbb{I}$.

entrées : une $\mathbb{I}$-grille $\mathbb{I}$, et l'ordre lexicographique $\preceq_y$ sur $\mathbb{I}$

début
  $R \leftarrow \texttt{min}(\mathbb{I})$; // $\mathcal{O}(1)$
  $t \leftarrow 1$;
  tant que $t \leq T$ **faire**
    parcourir les éléments de $\mathcal{E}_{\alpha_t}$ avec `suivant()` ; // $\mathcal{O}(|\mathcal{E}_{\alpha_t}|)$
    $t \leftarrow t + 1$;
fin

dans le cas discret régulier. On peut remarquer que la première commande est en $\mathcal{O}(1)$ car nous connaissons la plus petite ordonnée y_R de $\mathbb{I}$, issue de la plus petite cellule $\texttt{min}(\mathbb{I})$. Puis, nous parcourons les différentes listes $\mathcal{E}_{\alpha_t}$, ce qui implique que la complexité globale de cet algorithme est en $\mathcal{O}(\sum_{t=1}^{T} |\mathcal{E}_{\alpha_t}|) = \mathcal{O}(n)$, où n est le nombre total de cellules dans $\mathbb{I}$.

Généralisation des $\mathbb{I}$-grilles classiques en image et modélisation

Revenons maintenant sur les $\mathbb{I}$-grilles 2-D les plus communes dans divers domaines de l'imagerie, et vérifions que notre définition 2.1 permet de les obtenir. Dans la figure 2.19 sont également illustrés ces différents types de grilles. Il est clair qu'une description analogue pourrait être dépeinte pour des grilles 3-D. On peut remarquer que la structure de listes que nous avons décrite précédemment est adaptée à ces grilles. En effet, elles ont généralement un ordre implicite (comme pour la grille régulière ou la grille anisotrope) qui permet d'utiliser directement notre système. Si l'on est en présence d'une grille issue d'une décomposition hiérarchique (*e.g.* quadtree, *k*d-tree), il est facile de définir un ordre de parcours des feuilles de l'arborescence qui respecte les relations que nous énoncées. Dans le pire cas, où la $\mathbb{I}$-grille ne possède aucune structure sous-jacente (ou grille sans contrainte), il est nécessaire d'effectuer un tri des cellules pour ensuite construire notre structure.

- ▷ La grille régulière classique $\mathbb{D}$ se définit par $(x_R, y_R) \in \mathbb{Z}^2$ et $l_R^x = l_R^y = 1$, quel que soit $R \in \mathbb{D}$.
- ▷ Dans la grille anisotrope (ou rectangulaire) $\mathbb{E}$, très répandue en imagerie médicale, les cellules ont une dimension plus grande suivant un axe. On peut alors généraliser par $l_R^x = \mu$, $l_R^y = \lambda$ et $(x_R, y_R) = (\mu i, \lambda j)$ avec $(i, j) \in \mathbb{Z}^2$, quel que soit $R \in \mathbb{E}$.
- ▷ Dans le cas de l'encodage RLE d'une image régulière en X (un raisonnement analogue peut être fait pour l'encodage en Y), une cellule $R \in \mathbb{L}$ est définie par $l_R^x \in \mathbb{Q}$ et $l_R^y = 1$ et $(x_R, y_R) \in \mathbb{Q} \times \mathbb{Z}$. En effet, l'abscisse et la largeur des cellules suivant l'axe X peuvent être des demi-entiers.

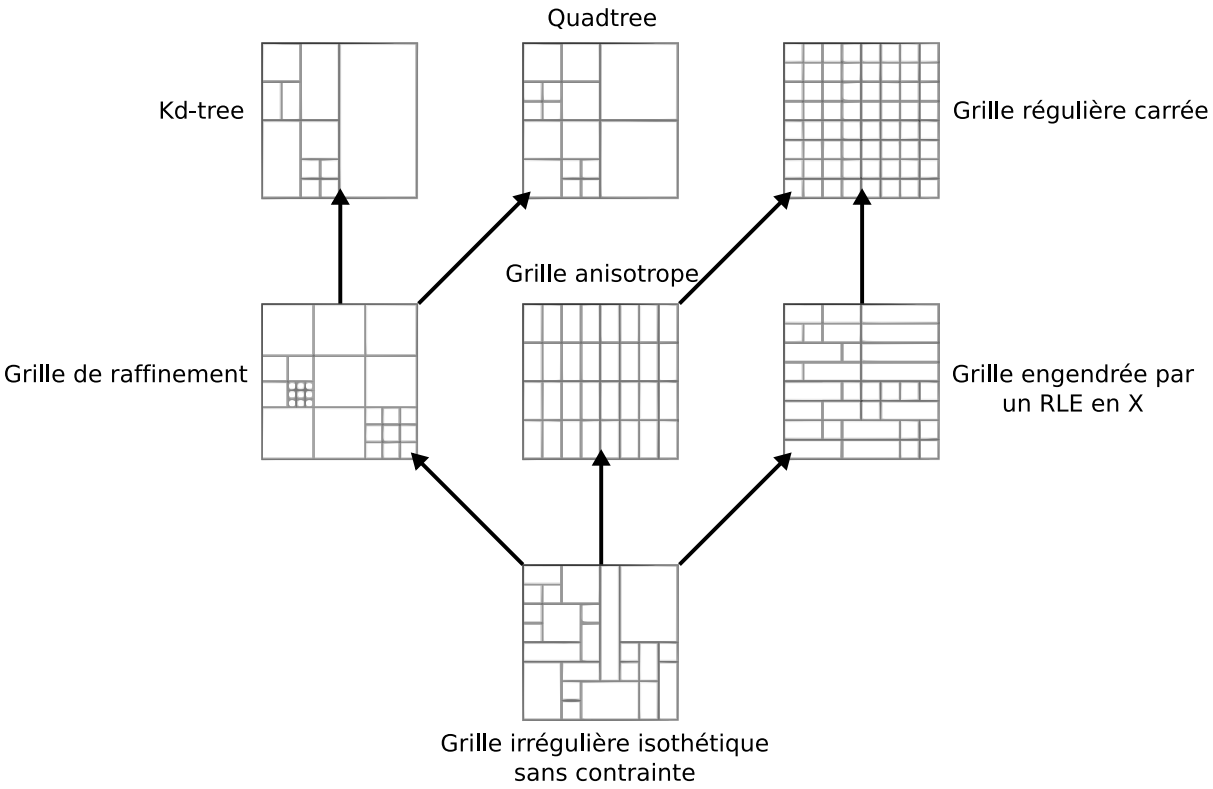


Fig. 2.19: Différents types de $\mathbb{I}$-grilles très communs en imagerie. Les contraintes sur la grille augmentent en montant dans le schéma.

▷ Dans une décomposition par quadtree $q\mathbb{T}$ d'une image 2-D de taille $2^m \times 2^n$, les cellules suivent les règles suivantes. Pour une cellule de niveau k, $k \leq \min(m, n)$ dans le quadtree $q\mathbb{T}$, sa taille est $(2^{m-k}, 2^{n-k})$ et sa position est de la forme $(i \times 2^{m-k-1}, j \times 2^{n-k-1})$, où $(i, j) \in \mathbb{Z}^2$. De la même manière, on peut représenter les cellules d'un kd-tree suivant son niveau dans l'arborescence. Pour une cellule de niveau k impair et une découpe suivant $X, Y, X, \ldots$ pour $k = 1, 2, 3, \ldots$, on observe que sa taille (l_R^x, l_R^y) est de la forme $(2^{m-k+1}, 2^{n-k})$ et sa position (x_R, y_R) est $(i \times 2^{m-k}, j \times 2^{n-k-1})$, où $(i, j) \in \mathbb{Z}^2$. Dans le cas de ces techniques de subdivision d'une image 2-D, les coordonnées et la taille des cellules peuvent être des demi-entiers.

2.2.3 Objets élémentaires et notion de distance sur $\mathbb{I}$-grilles

Maintenant que nous avons défini une $\mathbb{I}$-grille en deux dimensions, pour définir les objets appartenant une telle grille, il faut considérer la relation d'adjacence suivante, entre deux cellules de la grille :

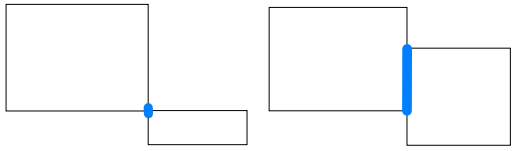
Définition 2.4 ($ve-$adjacence et $e-$adjacence)**.** *Soit R_1 et R_2 deux cellules. R_1 et R_2*

sont $ve-$adjacentes (vertex and edge adjacent) *si :*

$$ou \begin{cases} |x_{R_1} - x_{R_2}| = \frac{l^x_{R_1}+l^x_{R_2}}{2} \ et \ |y_{R_1} - y_{R_2}| \leq \frac{l^y_{R_1}+l^y_{R_2}}{2} \\ |y_{R_1} - y_{R_2}| = \frac{l^y_{R_1}+l^y_{R_2}}{2} \ et \ |x_{R_1} - x_{R_2}| \leq \frac{l^x_{R_1}+l^x_{R_2}}{2} \end{cases}$$

R_1 et R_2 sont $e-$adjacentes (edge adjacent) *si nous considérons un "ou" exclusif et des inégalités strictes dans la définition de $ve-$adjacence ci-dessus. Par la suite, k indique une $k-$adjacence avec $k = e$ ou $k = ve$.*

Dans la figure 2.20, on peut voir une illustration de ces deux types d'adjacence. On peut remarquer que ces relations peuvent être comparées aux adjacences à huit et quatre voisins (pour $k = ve$ et $k = e$ respectivement) dans le cas régulier classique. A partir



(a) (b)

Fig. 2.20: Exemples de ve-adjacence (a) et e-adjacence (b) entre deux cellules.

de cette relation d'adjacence, on peut définir différents objets discrets irréguliers, nécessaires au développement des outils que nous verrons dans la partie suivante. Dans la géométrie discrète classique, une fois que l'on a choisi une règle d'adjacence entre les cellules ($\alpha \in \{4, 8\}$ pour une grille 2-D), on peut définir un $\alpha-$chemin, un α-objet, et une $\alpha-$courbe [Coeurjolly *et al.*, 2007]. De manière analogue, nous énumérons les différentes structures discrètes irrégulières régies par la $k-$adjacence. Dans la figure 2.21 sont illustrés des exemples de ces structures irrégulières de base.

Définition 2.5 ($k-$chemin)**.** *Soit $\mathcal{E}$ un ensemble de cellules, $\mathcal{E}$ est un $k-$chemin si et seulement si pour tout élément de $\mathcal{E} = \{R_i\}_{i\in\{1,\dots,n\}}$, R_i est $k-$adjacent à R_{i-1} pour tout $i \geq 2$.*

Définition 2.6 ($k-$arc)**.** *Soit $\mathcal{E}$ un ensemble de cellules, $\mathcal{E}$ est un $k-$arc si et seulement si pour tout élément de $\mathcal{E} = \{R_i\}_{i\in\{1,\dots,n\}}$, R_i a exactement deux cellules $k-$adjacentes, sauf R_1 et R_n qui sont appelées extrémités du $k-$arc.*

Définition 2.7 ($k-$courbe)**.** *Soit $\mathcal{E}$ un ensemble de cellules, $\mathcal{E}$ est une $k-$courbe si et seulement si pour tout élément de $\mathcal{E} = \{R_i\}_{i\in\{1,\dots,n\}}$, R_i a exactement deux cellules $k-$adjacentes. On peut également noter $R_1 = R_n$ par commodité.*

Définition 2.8 ($k-$objet)**.** *Soit $\mathcal{E}$ un ensemble de cellules, $\mathcal{E}$ est un $k-$objet si et seulement si pour tout couple de cellules (R_1, R_2) appartenant à $\mathcal{E} \times \mathcal{E}$, il existe un $k-$chemin entre R_1 et R_2 dans $\mathcal{E}$.*

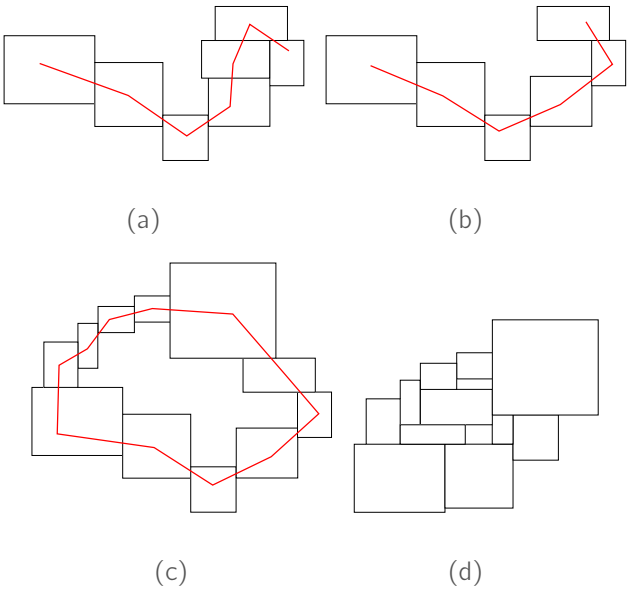


Fig. 2.21: Structures irrégulières de base. De la gauche vers la droite et de haut en bas : un $k-$chemin , un $k-$arc, une $k-$courbe et un $k-$objet. Le cas échéant, l'ordre des cellules R_i deux à deux $k-$adjacentes est illustré par le tracé rouge à l'intérieur de l'objet.

Si l'on suppose que les cellules de $\mathbb{I}$ sont de taille 1x1 et que leur position coïncide avec $\mathbb{Z}^2$, nous pouvons remarquer que ces définitions sont identiques à celles énoncés dans le cas discret régulier.

Dans la géométrie discrète classique, plusieurs modèles de discrétisation existent et permettent de déterminer quel type d'objet discret représente un objet euclidien discrétisé suivant ce modèle. Le *modèle de supercouverture* [Chassery et Montanvert, 1991, Cohen-Or et Kaufman, 1995, Andres, 2000] est celui que nous considérons dans cette thèse. Il s'énonce initialement de la manière suivante :

Définition 2.9 (Supercouverture sur grille discrète régulière)**.** *Soit F un objet euclidien dans $\mathbb{R}^2$. La supercouverture $\mathbb{S}_r(F)$ est définie sur une grille discrète régulière $\mathbb{D}$ par :*

$$\begin{aligned}\mathbb{S}_r(F) &= (F \oplus B^\infty(1/2)) \cap \mathbb{Z}^2 \\ &= \left\{p \in \mathbb{Z}^2 \mid F \cap B^\infty(p, 1/2) \neq \emptyset\right\} \\ &= \left\{p \in \mathbb{Z}^2 \mid d^\infty(p, F) \leq 1/2\right\}\end{aligned}$$

où $B^\infty(r)$ est la boule centrée sur l'origine, de rayon r pour la norme L_∞. De même, $B^\infty(p, r)$ est la boule centrée en p, de rayon r pour la norme L_∞.

La définition suivante concerne l'*extension du modèle de supercouverture* sur $\mathbb{I}$-grille [Coeurjolly, 2005], pour discrétiser des objets euclidiens sur une grille $\mathbb{I}^2$. Chacun des éléments énoncés est illustré par un exemple dans la figure 2.22.

[2]On notera que c'est un processus de discrétisation *basé pavage* [Coeurjolly *et al.*, 2007].

Définition 2.10 (Supercouverture sur $\mathbb{I}$-grille)**.** *Soit F un objet euclidien dans $\mathbb{R}^2$. La supercouverture $\mathbb{S}(F)$ est définie sur une $\mathbb{I}$-grille $\mathbb{I}$ par :*

$$\begin{aligned}\mathbb{S}(F) &= \left\{R \in \mathbb{I} \mid \mathbb{B}^\infty(R) \cap F \neq \emptyset\right\} \\ &= \left\{R \in \mathbb{I} \mid \exists (x,y) \in F, |x_R - x| \leq I_R^x/2 \right. \\ &\quad \left. et\ |y_R - y| \leq I_R^y/2\right\}\end{aligned}$$

où $\mathbb{B}^\infty(R)$ est le rectangle centré sur (x_R, y_R) de taille (I_R^x, I_R^y) (si $I_R^x = I_R^y$, $\mathbb{B}^\infty(R)$ est la boule centrée en (I_R^x, I_R^y) de taille I_R^x pour la norme L_∞).

Ce modèle de supercouverture possède plusieurs propriétés intéressantes.

Proposition 2.1 (Preuve dans [Coeurjolly, 2005])**.** *Soit F, G deux objets euclidiens dans $\mathbb{R}^2$, et une $\mathbb{I}$-grille $\mathbb{I}$, on a :*

$$\begin{aligned}\mathbb{S}(F \cup G) &= \mathbb{S}(F) \cup \mathbb{S}(G) \\ \mathbb{S}(F \cap G) &\subseteq \mathbb{S}(F) \cap \mathbb{S}(G) \\ si\ F \subseteq G &\quad alors\ \mathbb{S}(F) \subseteq \mathbb{S}(G)\end{aligned}$$

On peut également s'intéresser à la discrétisation d'une droite sur une $\mathbb{I}$-grille. Grâce au modèle de supercouverture, on peut donner une définition simple de cette opération.

Proposition 2.2 (Preuve dans [Coeurjolly, 2005])**.** *Soit I une droite euclidienne et une $\mathbb{I}$-grille, $\mathbb{S}(I)$ est un $ve-objet$.*

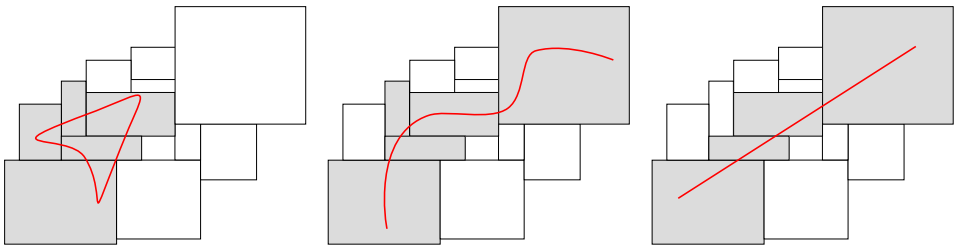
Définition 2.11 (Droite discrète isothétique irrégulière)**.** *Soit S un ensemble de cellules dans $\mathbb{I}$, S est appelé un morceau de droite discrète irrégulière (ou DDI) si et seulement si il existe une droite euclidienne I telle que :*

$$S \subseteq \mathbb{S}(I)$$

En d'autres termes, S est un morceau de DDI si et seulement si il existe I telle que pour tout $R \in S$, $\mathbb{B}^\infty(R) \cap I \neq \emptyset$.

Dans la partie suivante, nous décrivons un algorithme de segmentation d'un $k-$arc sous forme de segments de droites, basé sur les éléments précédents. Maintenant que nous savons comment se discrétisent d'une manière générale les objets euclidiens dans une $\mathbb{I}$-grille, on peut s'intéresser à l'analyse des objets discrets irréguliers que l'on obtient. Ici, nous nous concentrerons sur la notion de distance discrète, qui joue un rôle primordial dans la description des formes. Elle sert par exemple à mesurer l'épaisseur ou la longueur des formes discrètes que l'on manipule. On se concentrera essentiellement sur la distance euclidienne entre deux points $p = (p_1, \ldots, p_n)$ et $q = (q_1, \ldots, q_n)$ de $\mathbb{R}^n$, définie par

$$d_e(p,q) = \sqrt{(q_1 - p_1)^2 + \cdots + (q_n - p_n)^2}, \tag{2.18}$$



(a) (b) (c)

Fig. 2.22: Discrétisation de divers objets euclidiens sur une $\mathbb{I}$-grille par le modèle de supercouverture. En (a), un objet quelconque qui aboutit à un k-objet, en (b) une courbe donnant un $k-$chemin et en (c) une droite implique la formation d'un $k-$arc.

dont nous garderons la version en deux dimensions, à savoir

$$d_e(p, q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}, \tag{2.19}$$

pour deux points $p, q \in \mathbb{R}^2$. D'une manière plus générale, la notion de distance est définie par quatre axiomes :

Définition 2.12 (Distance [Coeurjolly *et al.*, 2007])**.** *Soit E un ensemble non vide et F un sous-groupe de $\mathbb{R}$. Une distance sur E à valeurs dans F, notée (d, E, F), est une application $d : E \times E \to F$ vérifiant :*

$$\begin{aligned}
(positive) \quad & \forall\ p, q \in E, \quad d(p, q) \geq 0\ ; \\
(définie) \quad & \forall\ p, q \in E, \quad d(p, q) = 0 \Leftrightarrow p = q\ ; \\
(symétrique) \quad & \forall\ p, q \in E, \quad d(p, q) = d(q, p)\ ; \\
(triangulaire) \quad & \forall\ p, q, r \in E, \quad d(p, q) \leq d(p, r) + d(r, q).
\end{aligned}$$

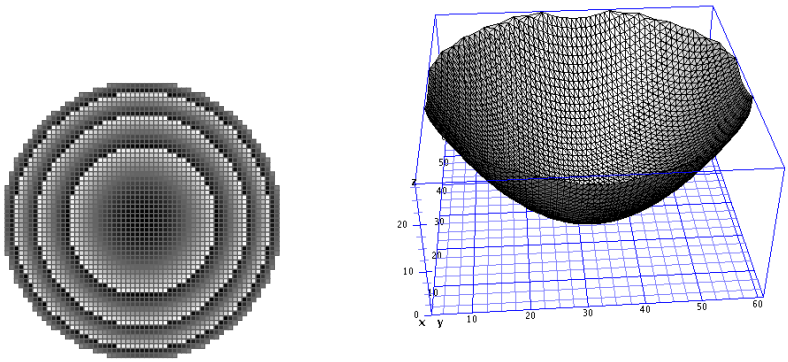
Dans une $\mathbb{I}$-grille, on se basera sur la définition suivante de boule discrète irrégulière, qui peut servir par exemple à caractériser l'axe médian d'une forme :

Définition 2.13 (Boule sur une $\mathbb{I}$-grille)**.** *Soit $(d_e, \mathbb{R}^2, \mathbb{R})$ la distance euclidienne définie sur le plan, $p \in \mathbb{R}^2$, et $r \in \mathbb{R}$. La boule B_{d_e} de centre p et de rayon r est :*

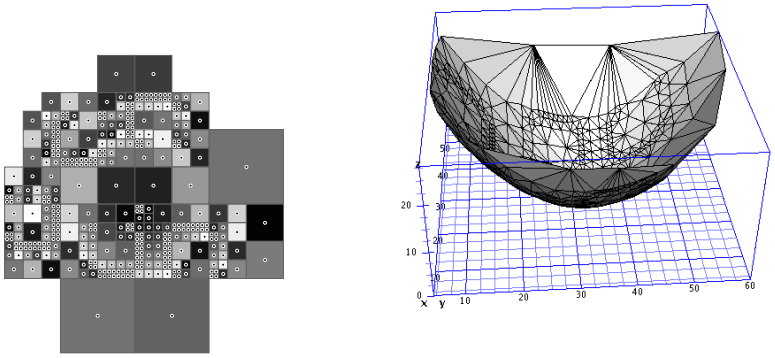
$$B_{d_e}(p, r) = \{q \in \mathbb{R}^2 : d_e(p, q) < r\}.$$

On peut noter que cette définition est très semblable au cas régulier, où les points sont à valeurs dans $\mathbb{Z}^2$. Dans la figure 2.23, deux boules discrètes sont présentées. On peut noter que cette définition est appliquée aux centres de cellules (nous verrons dans la partie suivante que l'on peut considérer d'autres distances dans le cas irrégulier).

Dans cette section, nous avons décrit la discrétisation et la notion de distance sur $\mathbb{I}$-grilles, nécessaires aux outils que nous développons dans les parties suivantes. Par exemple,



(a)



(b)

Fig. 2.23: Distinction entre la boule discrète classique et une boule irrégulière quelconque. Pour chaque boule est présentée la carte de distance où chaque cellule de distance d a pour valeur $v = d \mod 255$. Nous avons inclus également le tracé en 3-D de cette carte, où chaque point a pour altitude v. En (b), dans le cas de la boule irrégulière, on peut noter que les centres de cellule permettent de voir que la définition 2.13 est appliquée à ces points.

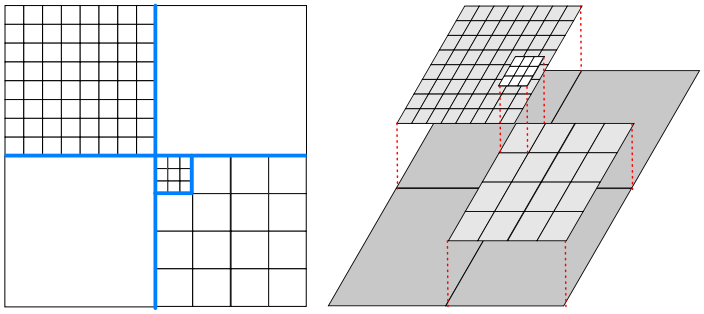
la représentation d'un objet irrégulier par un ensemble de $k-$arcs permettant de le simplifier et de décrire sa forme sera traitée dans le chapitre 3.

On peut maintenant continuer en décrivant les structures de données qui permettent de gérer de manière générique n'importe quelle $\mathbb{I}$-grille qui respecte notre définition, et donc celles que nous avons citées dans le début de ce chapitre.

2.2.4 Structures de données pour représenter une $\mathbb{I}$-grille

Grilles de raffinement

Une première structure qui serait intéressante à développer est le modèle de grilles de raffinement. Cette structure générique permet de représenter toutes les structures hiérarchiques que nous avons vues dans le chapitre précédent. En effet, on cherche à généraliser les règles de subdivision d'une cellule à un niveau k en un nombre variable de sous-cellules de niveau $k+1$. Dans les techniques que l'on peut lire dans le chapitre précédent, ce nombre est généralement fixé (par exemple, quatre pour le quadtree et deux pour le kd-tree). Dans la figure 2.24 est donné un exemple de structuration par une grille de raffinement. L'intérêt d'une telle structure est que l'on peut appliquer directement



(a) (b)

Fig. 2.24: Une grille de raffinement comportant trois niveaux. On remarquera que chaque subdivision de cellule peut comporter un nombre variable de sous-cellules, contrairement au quadtree par exemple, où l'on décompose toujours en quatre sous-cellules. En (a), les traits gras illustrent les interfaces entre les sous-grilles.

sur chaque nœud de l'arborescence les algorithmes développés sur une grille régulière. En effet, chaque nœud est une sous-grille régulière incluse dans la grille globale. La principale problématique est ensuite de mettre en accord les résultats obtenus sur chaque sous-grille. Cela implique donc de considérer les *interfaces* entre ces ensembles pour que le résultat global soit correct.

Cette grille a déjà été utilisée de manière très sporadique dans des recherches de synthèse d'images [Dmitriev, 2000, Jevans et Wyvill, 1989], en géométrie algorithmique [Park *et al.*, 2005] ou en méthodes numériques [Plewa *et al.*, 2005]. Grâce à la construction d'une $\mathbb{I}$-grille, les outils qui sont détaillés dans cette thèse pourraient être utilisés et généralisés à toutes les structures hiérarchiques.

Graphe d'adjacence

Une autre approche pour représenter une $\mathbb{I}$-grille peut être de construire le graphe d'adjacence associé à la grille. De la définition 2.4 de la k-adjacence entre cellules, on

peut construire un graphe $G = (V, E)$ associé à une $\mathbb{I}$-grille $\mathbb{I}$ où chaque sommet $v \in V$ représente une cellule R de $\mathbb{I}$, et une arête $e = (v_1, v_2) \in E, v_1, v_2 \in V$ indique qu'il existe une adjacence entre les cellules R_1 et R_2 dans $\mathbb{I}$. On peut voir dans la figure 2.25 un exemple du graphe d'adjacence associé à une grille quelconque. Pour construire ce

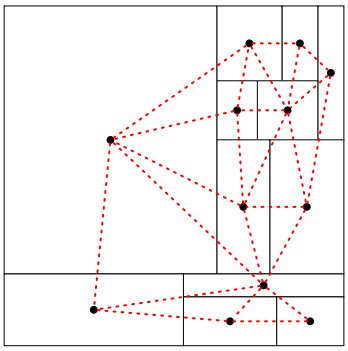


Fig. 2.25: Le graphe d'adjacence d'une grille irrégulière quelconque. Les arêtes du graphe sont illustrées en pointillés, et joignent les centres des cellules adjacentes dans la grille.

graphe d'adjacence, nous proposons l'algorithme 2.4 qui utilise les notions d'ordre que nous avons définies précédemment. Cet algorithme se base sur le parcours d'une $\mathbb{I}$-grille (algorithme 2.3). À chaque étape, nous considérons un ensemble de cellules à traiter, $\mathcal{E}$, et un ensemble $\mathcal{E}_{\alpha_t}$ de cellules qui ont le même bord bas $R^B = \alpha_t$ (on parle ici d'approche par *ligne de front*). Pour chaque cellule R_i encore non traitée, on prend en compte la cellule suivante dans l'ordre $\preceq_{ly}$ et l'ensemble $\mathcal{E}_{\alpha_t}$, qui peut contenir des cellules k-adjacentes à R_i (ligne 1). S'il y a effectivement k-adjacence entre R_i et une cellule de $\mathcal{F}$, nous mettons à jour le graphe en ajoutant un nouveau nœud et une arête (ligne 2). À la fin de cette boucle, nous vérifions si R_i doit être encore considérée dans les itérations suivantes. Cela implique que l'on teste si la ligne de front (d'ordonnée α_t) dépasse la cellule ou non. Nous évitons ainsi de traiter inutilement les cellules dont on est sûr qu'elles ne sont pas k-adjacentes avec les cellules de la ligne de front. À la fin de l'algorithme, on peut noter que les cellules de plus haut bord R^T sont supprimées de $\mathcal{E}$, car elles ont atteint la valeur maximale de $\alpha_t = \alpha_T$. Dans la figure 2.26, nous présentons quelques itérations de notre algorithme sur un exemple. Dans cette figure, lors de l'initialisation (a), l'ensemble $\mathcal{E}$ est formé par les cellules de plus petit bord bas $\alpha_1 = \texttt{min}(\mathbb{I})_B$, *i.e.* $\mathcal{E} = \{R_1, R_2, R_3\}$. Pendant la première itération (b), nous considérons l'ensemble $\mathcal{E}_{\alpha_2}$ des cellules alignées sur la « ligne suivante » d'ordonnée α_2 (dans cet exemple, $\mathcal{E}_{\alpha_2} = \{R_4\}$). Pour chaque cellule R_i de $\mathcal{E}$, nous construisons les adjacences dans le graphe en parcourant les cellules de $\mathcal{F}$. Par exemple, soit la cellule $R_1 \in \mathcal{E}$, alors on a $\mathcal{F} = \mathcal{E}_{\alpha_2} \cup \{\texttt{suivant}(R_1)\} = \{R_4, R_2\}$. On construit ici les adjacences (v_1, v_2) et (v_1, v_4) dans l'ensemble E des arêtes du graphe. On peut noter que lorsque nous parcourons les cellules R_2 et R_3, une fois que leurs adjacences sont mises à jour, nous choisissons ensuite de les supprimer de $\mathcal{E}$, car $R_2^T = R_3^T = \alpha_2$. Cela signifie qu'elle ne toucheront plus la ligne de front ultérieurement. À l'itération suivante (c), on a $\mathcal{E} = \mathcal{E} \cup \mathcal{E}_{\alpha_3} = \{R_1, R_4\}$. Nous obtenons la liste des cellules suivantes $\mathcal{E}_{\alpha_3} =$

Algorithme 2.4 : Algorithme de construction du graphe d'adjacence d'une $\mathbb{I}$-grille.

entrées : une $\mathbb{I}$-grille $\mathbb{I}$, dont les cellules sont notées $\{R_i\}_{i=1,n}$, et l'ordre $\prec_{ly}$ sur $\mathbb{I}$
sorties : le graphe $G = (V, E)$ représentant les adjacences de cellules dans $\mathbb{I}$

début
  $E \leftarrow \emptyset$;
  $V \leftarrow \emptyset$;
  $t \leftarrow 2$;
  $\mathcal{E} \leftarrow \mathcal{E}_{\alpha_1}$;
  tant que $t \leq T$ **faire**
    pour chaque $R_i \in \mathcal{E}$ **faire**
      $V \leftarrow V \cup \{v_i\}$;
1     $\mathcal{F} \leftarrow \mathcal{E}_{\alpha_t} \cup \{\texttt{suivant}(R_i)\}$;
2   **pour chaque** *cellule* $R_k \in \mathcal{F}$ **faire**
        si R_k *est k-adjacent à* R_i **alors**
          $V \leftarrow V \cup \{v_k\}$;
          $E \leftarrow E \cup \{(v_i, v_k)\}$;
3   **si** $R_i^T = \alpha_t$ **alors**
        $\mathcal{E} \leftarrow \mathcal{E} \backslash \{R_i\}$;
    $\mathcal{E} \leftarrow \mathcal{E} \cup \mathcal{E}_{\alpha_t}$;
    $t \leftarrow t + 1$;
  retourner $G = (V, E)$
fin

$\{R_5, R_6, R_7\}$. Nous continuons ainsi ces itérations jusqu'à obtenir le graphe d'adjacence que nous avons illustré dans la figure 2.25.

On peut faire le lien entre cette structure de données et les techniques impliquant des graphes d'adjacence pour la segmentation, comme dans les travaux de J. Cousty [Cousty *et al.*, 2008]. Dans cette approche, le graphe G est construit à partir d'une image en niveaux de gris, et on ajoute aux arêtes de G la différence de valeur entre les pixels adjacents. Pour tester notre algorithme, nous avons construit le graphe d'adjacence de grilles organisées sur une image en niveaux de gris simple. On parle alors de $\mathbb{I}$-grille étiquetée avec des niveaux de gris. Dans la figure 2.27, un autre point de vue est également donné à cette structure, car on peut la visualiser comme une triangulation 3-D, où l'altitude des sommets correspond à la valeur de la cellule associée dans l'image initiale.

Dans le cadre d'un TER (travaux d'étude et de recherche) co-encadré avec David Coeurjolly en 2005-2006, nous nous sommes intéressés à l'emploi de cette structure de données pour généraliser des traitements comme le filtrage. Considérons par exemple la

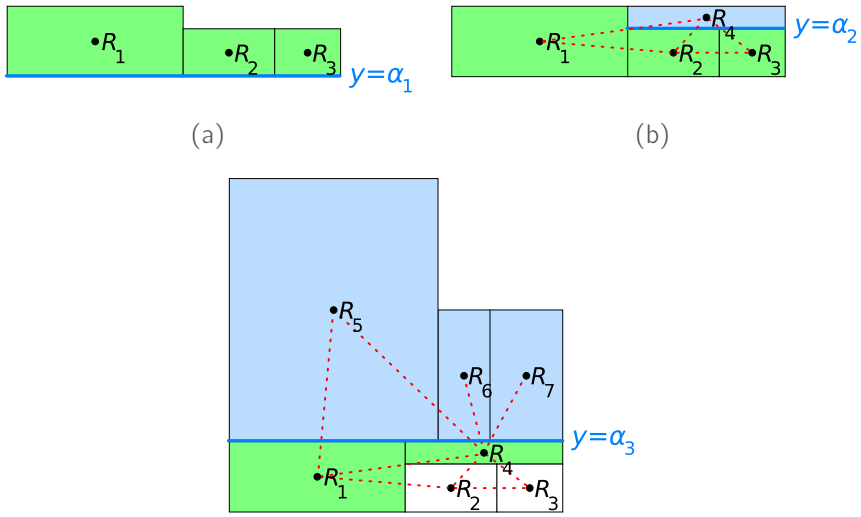
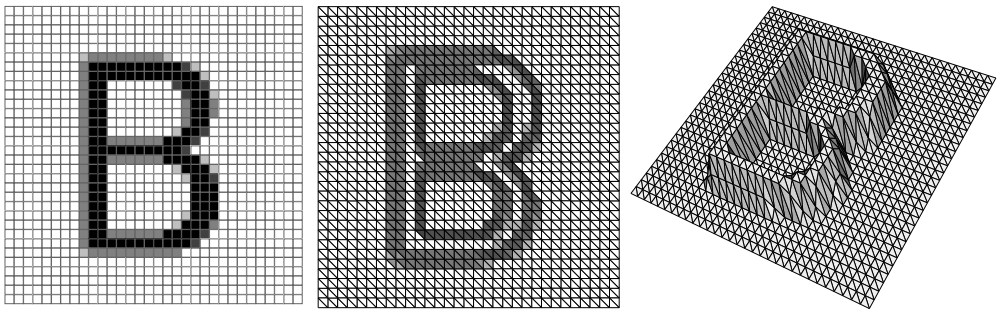


Fig. 2.26: Construction du graphe d'adjacence d'une $\mathbb{I}$-grille sur quelques itérations. L'ensemble $\mathcal{E}$ est illustré par les cellules vertes, l'ensemble $\mathcal{E}_\alpha$ par les cellules bleues, et la ligne de front d'ordonnée α par le trait gras.

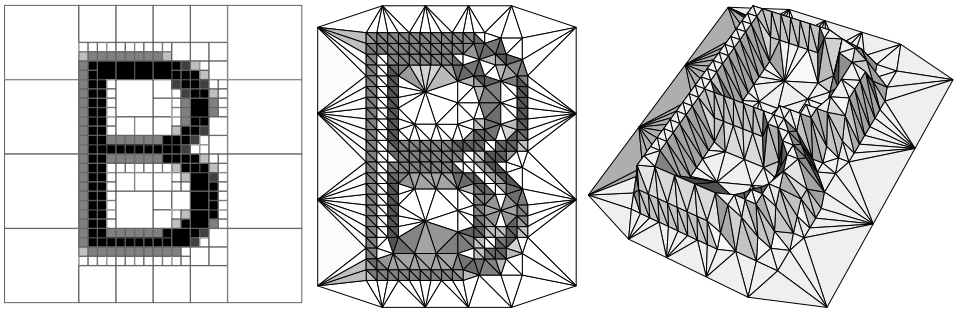
fonction F suivante, appliquée à chaque cellule R de la grille

$$F(R) = \frac{1}{|gauche(R)| + |droite(R)|} \sum_{R_1 \in gauche(R)} F(R_1) + \sum_{R_2 \in droite(R)} F(R_2) \qquad (2.20)$$

où $droite(R)$ ($gauche(R)$) représente les cellules adjacentes à R par son bord droit (gauche respectivement). Pour calculer F en chaque cellule R de la grille, on peut parcourir le graphe d'adjacence, et mettre à jour la valeur de R en fonction de son voisinage. Les autres structures basées sur des graphes que l'on peut citer (voir aussi [Coeurjolly *et al.*, 2007]) ont généralement pour objectif de segmenter une image grâce à des règles de fusion de régions : les graphes de fusion [Cousty *et al.*, 2008], les cartes combinatoires [Lienhardt, 1991, Damiand *et al.*, 2004], les pyramides régulières et irrégulières [Manzanera et Jolion, 1995, Montanvert *et al.*, 1991, Marfil *et al.*, 2006], *etc.* Ces modèles ne construisent pas tous une $\mathbb{I}$-grille associée à l'image. Néanmoins, il pourrait être intéressant de modifier les règles de construction pour aboutir à des cellules isothétiques, plutôt que des régions quelconques. On pourrait ainsi utiliser les outils décrits dans cette thèse sur ces structures, sans adaptation préalable.



(a)



(b)

Fig. 2.27: Construction du graphe d'adjacence associé à des grilles simples. En (a) est représenté ce graphe pour une grille régulière associée à une image en niveaux de gris. La visualisation par un modèle 3-D de face et en perspective permet de mieux apprécier la forme du graphe. Dans ce cas, l'altitude d'un sommet correspond à la valeur dans la grille initiale. Il en est de même dans (b), où le graphe a été construit à partir d'un quadtree de la même image.

2.3 Conclusion

Dans cette partie, nous avons dressé un état des lieux des différentes grilles irrégulières isothétiques qui existent dans la littérature, et exposé quelques-unes de leurs applications. Puis, nous avons présenté la théorie du signal non-uniforme, qui semble une voie intéressante pour généraliser des techniques de traitement d'images définies pour un signal uniforme. Nous avons ensuite proposé un modèle générique de $\mathbb{I}$-grille 2-D, et les définitions des notions élémentaires (ordre, discrétisation, distance) indispensables à l'élaboration d'algorithmes sur ces grilles. Dans la partie II, nous utilisons ces éléments pour développer deux méthodologies classiques de la géométrie discrète. La reconstruction d'objets complexes sur une $\mathbb{I}$-grille $\mathbb{I}$ fait appel principalement aux notions de k-arc et de droite discrète pour proposer une structure polygonale exacte représentant l'objet traité $\mathcal{E} \in \mathbb{I}$. Grâce à l'ordre que nous avons défini, nous pouvons faire le lien avec le graphe de Reeb associé à $\mathcal{E}$. La transformée en distance se base elle-aussi sur un ordre de

$\mathbb{I}$ pour parcourir la grille et étiqueter les cellules.

On pourrait s'intéresser dans le futur à définir d'autres notions pour l'analyse et la description de formes sur $\mathbb{I}$-grilles, comme par exemple, la convexité, la circularité, *etc.* Le graphe d'adjacence que nous avons mis en place est un outil qui pourrait servir à segmenter une $\mathbb{I}$-grille, grâce à l'adaptation des règles de fusion d'arêtes.

Deuxième partie

Outils de la géométrie discrète sur $\mathbb{I}$-grilles pour la description et la reconnaissance de formes 2-D

SOMMAIRE DE LA PARTIE

CHAPITRE 3

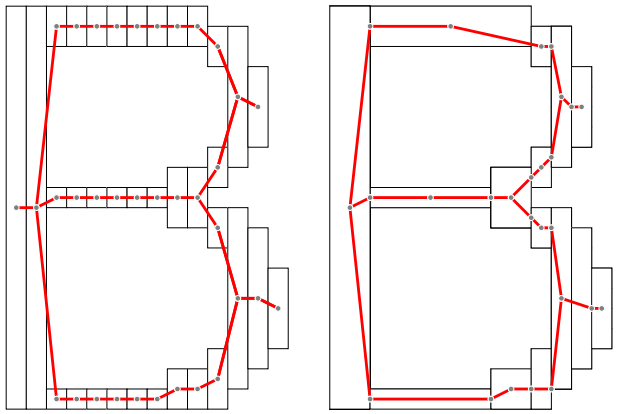
Reconstruction géométrique et topologique d'objets binaires irréguliers

3.1 Introduction

La représentation, la description et la classification de caractères et de symboles sont des tâches nécessaires dans de nombreuses applications actuelles. Elles sont appliquées sur des images généralement organisées sur des grilles régulières. Nous introduisons ici le concept de représentation de formes sur une $\mathbb{I}$-grille. Nous proposons de représenter la topologie des éléments contenus dans l'image irrégulière à deux dimensions en construisant un graphe de Reeb associé [Reeb, 1946]. Puis, nous les décrivons par une structure polygonale simple qui respecte le modèle de supercouverture discret étendu que nous avons rappelé dans la partie I. De plus, cette structure préserve la topologie que nous détaillons dans la phase précédente. Nous nous intéressons clairement au problème de la *vectorisation* sur $\mathbb{I}$-grilles, mais pas uniquement dans le cadre de l'analyse de documents ou d'images (voir la partie III chapitre 6 pour un exemple d'application où nous utilisons notre contribution). En effet, comme nous le montrons dans le chapitre 7 de la partie III, nous pouvons aussi considérer une subdivision d'une partie de $\mathbb{R}^2$ représentant les solutions d'une fonction donnée $f : \mathbb{R}^2 \to \mathbb{R}$. Les algorithmes développés en arithmétique d'intervalles sont des approches intéressantes pour aborder ces problèmes [Snyder, 1992a].

Les techniques de vectorisation développées jusqu'à maintenant sur le plan discret régulier dépendent principalement de l'application finale de la méthode [Mertzios et Karras, 1999, Wenyin et Dori, 1999, Cordella et Vento, 2000, Song *et al.*, 2002, Hilaire et Tombre, 2005]. Nous nous concentrons ici sur quelques méthodologies de vectorisation, largement développées pour des applications d'analyse de document. Les méthodes basées sur le RLE construisent d'abord une décomposition en cellules allongées suivant un axe de l'image. Dès

lors, on construit un graphe d'adjacence de droites (*line adjacency graph* ou LAG) [Burge et Kropatsch, 1999, Elgammal et Ismail, 2001]. Ces méthodes cherchent à décrire la topologie des objets rencontrés dans l'image, mais la structure géométrique qu'on en déduit doit être améliorée par de nombreux post-traitements (voir figure 3.28). Les méthodes de squelettisation et d'amincissement sont assurément



(a) (b)

Fig. 3.28: Comparaison d'une approche de vectorisation par LAG [Burge et Kropatsch, 1999] avec notre contribution. La technique de LAG implique la construction de nombreux segments, elle nécessite des post-traitements pour alléger la polygonalisation (a). Grâce à notre approche, nous construisons directement peu de segments au sein de l'objet traité (b).

les plus souvent employées en vectorisation [Attali *et al.*, 2007]. On peut remarquer que les outils définis en morphologie mathématique [Soille, 2003] forment une option fréquemment choisie pour préparer les images avant ces processus. Un état de l'art des méthodes de vectorisation basées sur le squelette peut être lu dans [Lam *et al.*, 1992], et un autre sur celles qui ne l'utilisent pas dans [Liu et Dori, 1998]. Le but est ici de calculer un axe médian de l'objet qui représente de manière minimale sa forme [Klette et Rosenfeld, 2004, Coeurjolly *et al.*, 2007] (voir la figure 3.29 pour une illustration du calcul de l'axe médian). Néanmoins, ces techniques nécessitent une phase de pré-traitement par filtrage ou par lissage pour réduire le bruit pouvant perturber l'axe médian calculé. De plus, si ces approches ne sont pas basées sur un principe de boule maximale, elles modifient la géométrie de l'objet, puisqu'elles ne sont pas réversibles. De manière plus générale, un objet peut contenir des trous, et peut être composé d'*arcs épais* [Alhalabi et Tougne, 2005]. Dans les travaux de I. Debled et al. [Debled Rennesson et Reveilles, 1995, Debled-Rennesson *et al.*, 2004, Debled-Rennesson *et al.*, 2005], la définition de *segment flou* rejoint ce concept d'arc régulier épais. Mais, au-delà de cette représentation géométrique d'arcs, la structuration globale n'est pas discutée, ainsi il n'y a pas de description de la topologie des objets reconnus. Le seule technique extensible aux $\mathbb{I}$-grilles qui a été

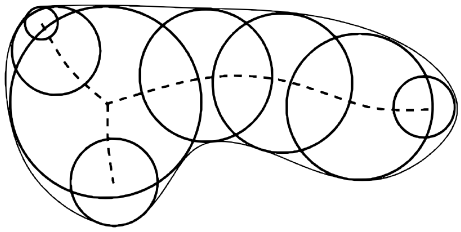


Fig. 3.29: Illustration du calcul de l'axe médian d'une forme. Dans cette figure issue de [Coeurjolly *et al.*, 2007], on voit également le lien avec le principe de boule maximale incluse dans la forme.

proposée par M. Dexet [Dexet, 2006], grâce à la définition d'une préimage généralisée [Dexet et Andres, 2006, Dexet et Andres, 2008], est détaillée dans ce chapitre. Là encore, la structure topologique globale de l'objet n'est pas discutée.

Dans ce chapitre, nous traitons de la représentation géométrique et topologique d'objets complexes appartenant à une $\mathbb{I}$-grille quelconque. Nous proposons plusieurs variantes de reconstruction rapide en segments de droite, et nous décrivons la forme des objets traités grâce au graphe de Reeb. Cette structure permet également de prendre en compte rapidement une modification de l'objet reconnu. Cet aspect dynamique, peu traité dans la littérature, nous permet de décrire un cadre très général de la représentation d'objets complexes, et de leurs évolutions par des moyens divers (de manière supervisée, dirigée par l'arithmétique d'intervalles, *etc.*). Enfin, l'ensemble des outils que nous détaillons ici sera utilisé dans deux applications dans la partie suivante.

3.2 Reconstructions d'un k-arc en segments de droite

Dans cette section, nous rappelons en premier lieu l'algorithme décrit dans [Coeurjolly et Zerarga, 2006] pour représenter un k-arc S sous forme de segments de droite, qui respectent ainsi la définition 2.11 de droite discrète irrégulière (DDI) introduite dans la partie I. Nous avons alors noté qu'un ensemble de cellules S dans une $\mathbb{I}$-grille $\mathbb{I}$ est un morceau de DDI si et seulement si il existe une droite I telle que pour tout $R \in S$, $\mathbb{B}^\infty(R) \cap I \neq \emptyset$. Pour détecter si $\mathbb{B}^\infty(R) \cap I$ est vide ou non, on teste si I intersecte l'une des diagonales de la cellule R, notées d_1 et d_2. Sans perte de généralité, on peut supposer que I est donnée par l'équation $\alpha x + \beta = 0$. Notre problème de reconnaissance d'une DDI revient alors à

$$\mathbb{B}^\infty(R) \cap I \neq \emptyset \iff ou \begin{cases} I \cap d_1 \neq \emptyset \wedge \alpha \geq 0 \\ I \cap d_2 \neq \emptyset \wedge \alpha < 0 \end{cases} . \tag{3.21}$$

Pour savoir si S représente une DDI, on peut considérer l'ensemble des droites euclidiennes dont la discrétisation contient l'ensemble S , ou *préimage* de S. Si cette préimage est vide, cela signifie que S n'est pas une DDI. En prenant en compte les notations de taille et

de position d'une cellule R de $\mathbb{I}$, les deux inégalités précédentes impliquent dans l'espace (α, β) respectivement :

$$\mathcal{E}^+(R) = \begin{cases} \alpha\left(x_R - \frac{l_R^x}{2}\right) + \beta - y_R - \frac{l_R^y}{2} \leq 0 \\ \alpha\left(x_R + \frac{l_R^x}{2}\right) + \beta - y_R + \frac{l_R^y}{2} \geq 0 \end{cases}, \tag{3.22}$$

$$\mathcal{E}^-(R) = \begin{cases} \alpha\left(x_R - \frac{l_R^x}{2}\right) + \beta - y_R + \frac{l_R^y}{2} \geq 0 \\ \alpha\left(x_R + \frac{l_R^x}{2}\right) + \beta - y_R - \frac{l_R^y}{2} \leq 0 \end{cases}. \tag{3.23}$$

Il est à noter que $\mathcal{E}^+(R)$ est défini uniquement pour $\alpha \geq 0$, et $\mathcal{E}^-(R)$ pour $\alpha < 0$. On peut ainsi définir les préimages associées à une portion de DDI :

Définition 3.14 (Préimages d'une DDI)**.** *Soit S une DDI, les deux préimages $\mathcal{P}^+$ et $\mathcal{P}^-$ de S sont définies par :*

$$\mathcal{P}^+(S) = \bigcap_{R \in S} \mathcal{E}^+(R) \text{ et } \mathcal{P}^-(S) = \bigcap_{R \in S} \mathcal{E}^-(R) \tag{3.24}$$

Par conséquent, tester si un objet irrégulier est une DDI revient à vérifier la proposition suivante :

Proposition 3.3. *Soit S un ensemble de cellules d'une $\mathbb{I}$-grille $\mathbb{I}$. S est une portion de DDI si et seulement si $\mathcal{P}^+(S) \neq \emptyset$ ou $\mathcal{P}^-(S) \neq \emptyset$.*

Dans la figure 3.30, nous présentons l'exemple du calcul de ces deux preimages pour un ensemble des trois cellules $S = \{(10, 10, 2, 2), (20, 12, 18, 10), (37, 15, 3, 4)\}$ où une cellule R est notée avec sa position et sa taille de la manière suivante : (x_R, y_R, l_x^R, l_y^R). Dans cet exemple précis, le calcul de la préimage $\mathcal{P}^-(S)$ revient à

$$\mathcal{P}^-(S) = \begin{cases} 9\alpha + \beta - 9 \geq 0 \\ 11\alpha + \beta - 11 \leq 0 \\ 11\alpha + \beta - 7 \geq 0 \\ 29\alpha + \beta - 17 \leq 0 \\ 29\alpha + \beta - 13 \geq 0 \\ 45\alpha + \beta - 17 \leq 0 \end{cases}. \tag{3.25}$$

Le calcul de $\mathcal{P}^+(S)$ peut être décrit de manière similaire. Le système de représentation de $\mathcal{P}^-(S)$ contient $2n$ inégalités, où n est le nombre de cellules de S. Par conséquent, le problème de reconnaissance d'une DDI (proposition 3.3) revient à résoudre deux systèmes de tailles $2n$. Pour calculer *complètement* la préimage de S, on peut utiliser l'algorithme de Preparata et Shamos [Preparata et Shamos, 1985] optimal en $\mathcal{O}(n \log n)$, ou une approche par programmation linéaire [Coeurjolly, 2005] qui aboutit à la même complexité. De même, nous pourrions utiliser la technique de M. Dexet [Dexet, 2006] se basant sur

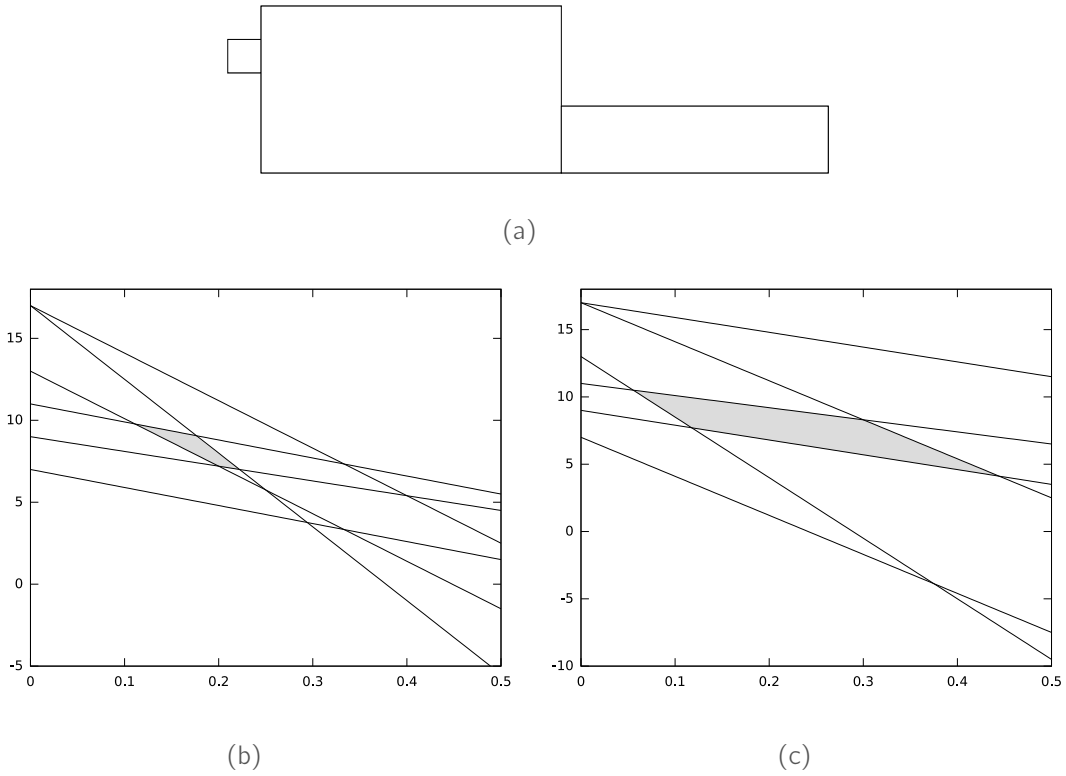


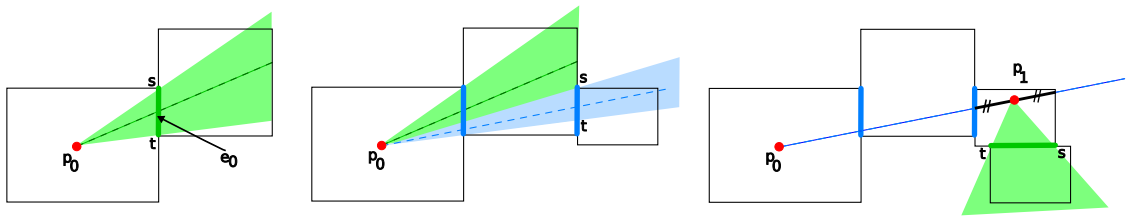
Fig. 3.30: Le calcul des préimages d'un ensemble de cellules 2-D dans l'espace (α, β). Pour les trois cellules $S = \{(10, 10, 2, 2), (20, 12, 18, 10), (37, 15, 3, 4)\}$, avec R notée (x_R, y_R, I_x^R, I_y^R), nous avons calculé $\mathcal{P}^-(S)$ (b) et $\mathcal{P}^+(S)$ (c) dans l'espace des paramètres (α, β). Elles sont représentées par la partie grisée dans chaque graphique.

un modèle de préimage généralisée. Nous obtiendrions là encore une complexité équivalente. Dans cette thèse, nous préférons calculer *partiellement* cette préimage, et décrire un algorithme linéaire en $\mathcal{O}(n)$ pour calculer un représentant euclidien associé à S. Nous pouvons donc comparer notre contribution avec les techniques d'identification de DDI [Megiddo, 1984, Buzer, 2002].

Ainsi, nous décrivons maintenant un algorithme qui permet de donner un ensemble de segments de droites à partir d'un k-arc irrégulier, dont la discrétisation est égale à l'objet initial (propriété de *réversibilité*). Cette technique linéaire permet à la fois de distinguer les portions de DDI au sein du k-arc *et* de calculer un segment euclidien pour chacune de ces portions, qui se discrétise dans celles-ci. Grâce à l'utilisation d'un *cône de visibilité* [Sivignon *et al.*, 2004, Coeurjolly, 2002b], on évite des calculs complexes en programmation linéaire, que l'on pourrait choisir pour résoudre exactement les deux tests d'égalité de la proposition précédente [Coeurjolly, 2005].

Tout d'abord, on définit le prédicat `positif`(a, b, c) qui renvoie vrai si les points $\{a, b, c\}$ sont ordonnés dans le sens trigonométrique dans le plan. Il peut être calculé grâce au déterminant $det(\vec{ab}, \vec{bc})$. Soit $S = \{R_i\}_{i=1,n}$ un k-arc, on fixe la première extrémité p_0 du premier segment telle que $p_0 \in R_0$ (ici nous avons choisi p_0 centre de R_0). On note alors

e_0 le segment euclidien partagé par R_0 et R_1. On considère le premier cône $C_0(p_0, s, t)$ tel que s et t coïncident avec les extrémités de e_0 et $\{p_0, s, t\}$ sont ordonnés dans le sens trigonométrique (*i.e.* $\texttt{positif}(p_0, s, t)$ est vrai, voir la figure 3.31). On peut remarquer



(a) (b) (c)

Fig. 3.31: Construction progressive et mise à jour du cône de visibilité avec la version 1 de construction d'un nouveau cône (bissectrice). Dans le cas présenté, la mise à jour échoue (b) et l'on construit un nouveau cône $C_1(p_1, s, t)$ en (c). Les droites en tirets représentent les bissectrices des cônes, et les traits en gras sont les interfaces e_1, e_2, e_3.

que C_0 est un cône de visibilité car pour n'importe quel point p appartenant à l'intersection de C_0 et des cellules de S, le segment $[p_0 p]$ se situe dans S. Cela signifie aussi que C_0 représente un sous-ensemble des préimages notées $\mathcal{P}^+(\{R_0, R_1\})$ et $\mathcal{P}^-(\{R_0, R_1\})$ dans l'espace des paramètres (α, β). Ensuite, pour chaque cellule suivante R_i, on considère le segment e_i partagé par R_{i-1} et R_i. Dans l'algorithme 3.5, nous indiquons une procédure simple qui permet de mettre à jour le cône actuel $C_j(p_j, s, t)$ en fonction de e_i (avec les extrémités u et l du segment $e_i = [ul]$, avec $\texttt{positif}(p_j, l, u)$ à vrai). Quand la mise à jour échoue (*i.e.* il n'existe pas de droite euclidienne qui passe par p_j et R_i), nous commençons une nouvelle reconstruction en créant un nouveau cône. Nous donnons maintenant deux algorithmes possibles pour construire ce nouveau cône.

Construction du nouveau cône de visibilité 1, par la bissectrice Nous proposons ici de construire le cône $C_{j+1}(p_{j+1}, s, t)$, où s et t sont donnés par le segment actuel e_i. Pour calculer le nouveau centre p_{j+1} du cône, on choisit l'approche de [Sivignon *et al.*, 2004]. Dans ce cas, on considère la bissectrice du cône C_j et l'on construit p_{j+1} comme le milieu du segment d'intersection entre cette bissectrice et R_{i-1} (voir figure 3.31 pour un exemple). Bien que le calcul de p_{j+1} soit simple, cette procédure de construction possède l'inconvénient de ne pas considérer le placement relatif des cellules R_{i+1}, R_i et R_{i-1}. Ainsi, le cône C_{j+1} peut être très étroit dès le début de la reconnaissance suivante. Nous proposons maintenant une nouvelle manière adaptative de calculer p_{j+1} suivant la configuration des cellules R_{i+1} et R_i, et qui prend aussi en compte le dernier cône construit.

Construction du nouveau cône de visibilité 2, par extension de cône Nous introduisons le cône étendu de C_j, qui est réalisé en prolongeant C_j à travers R_i. Supposons que

Algorithme 3.5 : Procédure de mise à jour du cône de visibilité

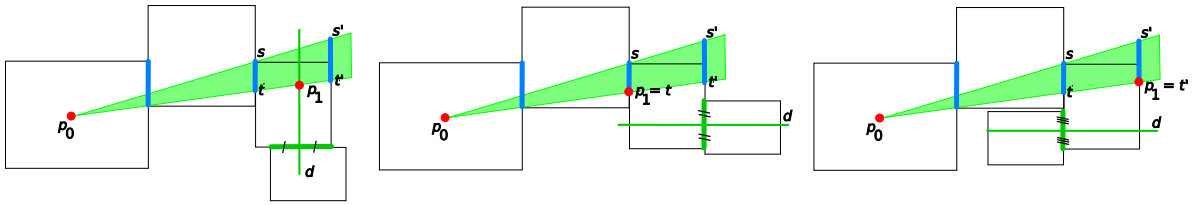
entrée : le cône actuel $C_j(p_j, s, t)$ et le segment $e_i = [ul]$ partage par R_i et R_{i-1}.
sortie : le cône $C_j(p_j, s, t)$ est mis à jour en modifiant éventuellement s ou t.

```
début
    si ¬ positif(p_j, t, u) ∨ positif(p_j, s, l) alors
        // Le test de visibilité échoue
        retourner ∅
    sinon
        si positif(p_j, t, l) alors
            t ← l ;
            retourner C_j(p_j, s, t) ;
        si ¬ positif(p_j, s, u) alors
            s ← u ;
            retourner C_j(p_j, s, t) ;
fin
```

e_i appartient au bord gauche de R_i (un raisonnement analogue peut être déroulé pour les autres bords), nous construisons deux nouveaux points s' et t' sur la droite supportée par le bord opposé (*i.e.* droit) de R_i tels que $\texttt{positif}(p_j, t', s')$ soit vrai. Considérons maintenant la cellule suivante dans le k-arc R_{i+1}, et la médiatrice d du segment e_{i+1} commun à R_i et R_{i+1}. Le point p_{j+1} est construit dans le quadrilatère $ss't't$ afin d'augmenter la taille du prochain cône C_{j+1}. Plus précisément, nous prenons en compte principalement trois cas possibles (voir la figure 3.32 pour des exemples de configurations) :

- ▷ La droite d intersecte le quadrilatère $ss't't$, ce qui se produit essentiellement lorsque $e_{i+1} \perp (st)$. p_{j+1} est alors choisi de telle sorte que le point d'intersection le plus proche de e_{i+1}.
- ▷ d n'intersecte pas $ss't't$ et e_{i+1} et la droite $(s't')$ sont confondus. Comme précédemment, nous choisissons p_{j+1} comme le point appartenant à $[st]$ le plus proche de d.
- ▷ d n'intersecte pas $ss't't$ et e_{i+1} et la droite (st) sont confondus. Dans ce cas, p_{j+1} est le point appartenant à $[s't']$ le plus proche de d.

L'avantage principal de cette version de construction du nouveau cône est de permettre de calculer des segments de droites plus en accord avec la forme du k-arc irrégulier traité. Nous présentons dans la section suivante les effets positifs de cette nouvelle version sur la reconstruction d'objets irréguliers complexes. Quelle que soit la procédure du nouveau cône choisie, l'algorithme de reconnaissance de DDI est linéaire en le nombre de cellules en entrée. En effet, à chaque étape, on opère soit une mise à jour du cône, soit le calcul d'un nouveau cône. Ces deux tâches sont réalisées en $\mathcal{O}(1)$, car elles impliquent des opérations élémentaires sur un nombre de cellules constant.



(a) (b) (c)

Fig. 3.32: Quelques exemples de construction d'un nouveau cône de visibilité par extension du cône précédent. Dans le premier cas (a), l'intersection entre la droite d et le quadrilatère $ss't't$ n'est pas vide, ce qui permet d'obtenir le nouveau point p_1. Dans les cas (b) et (c), nous choisissons ce point de telle manière à augmenter la taille du prochain cône.

Dans l'algorithme 3.6, nous donnons l'algorithme complet qui permet de reconstruire un k-arc irrégulier isothétique en utilisant l'algorithme de mise à jour d'un cône de visibilité et l'une des procédures de construction d'un nouveau cône. Nous présentons dans la

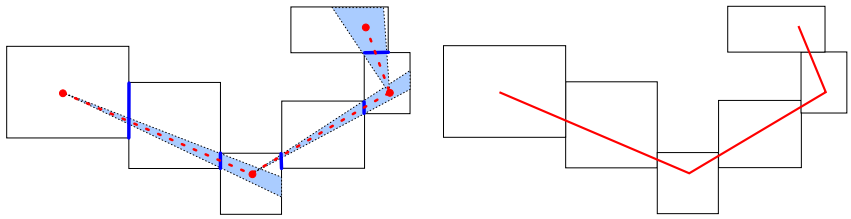
Algorithme 3.6 : Algorithme `reconstruction` de vectorisation d'un k-arc.

entrée : un k-arc $S = \{R_i\}_{i=1,n}$ et p_0 le premier point de la reconstruction.
sortie : l'ensemble $\mathcal{L}$ des segments de droites (sous forme de points) représentant S.

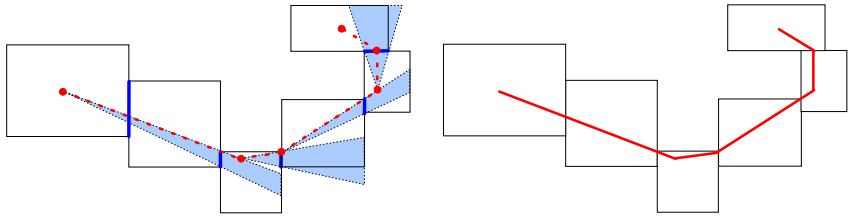
début
  $j \leftarrow 0$;
  initialisation du cône $C_j(p_j, s, t)$ avec R_0 et R_1, $\texttt{positif}(p_j, t, s) = 1$;
  $L \leftarrow \{p_j\}$;
  pour $i = 2$ **à** n **faire**
    calculer le segment e_i commun à R_i et R_{i-1} ;
    $C' \leftarrow$ mise à jour du cône C_j avec l'algorithme 3.5 ;
    si $C' = \emptyset$ **alors**
      calculer le point p_{j+1} avec la bissectrice de C_j ou par extension de C_j ;
      initialisation du cône C_{j+1} avec p_{j+1} et e_i ;
      $\mathcal{L} \leftarrow \mathcal{L} \cup \{p_j\}$;
    sinon
      $C_j \leftarrow C'$;
  retourner $\mathcal{L}$
fin

figure 3.33 deux exemples de reconstruction d'un k-arc simple, suivant ces deux versions. Dans la seconde version, nous avons choisi d'ajouter le centre de la dernière cellule traitée à $\mathcal{L}$, ce qui permettra dans la section suivante de joindre plusieurs reconstructions qui se

terminent en la même cellule. Nous étudions maintenant plusieurs propriétés intéressantes de notre nouvelle procédure de création de cônes. Dans le cas de k-arcs quelconques, on peut remarquer que notre algorithme implique la construction de plus de cônes lors du parcours du k-arc (figure 3.33). Ainsi, les segments calculés sont plus courts et plus proches de la forme des cellules traitées, comme le montre également la figure 3.34 d'un k-arc plus complexe. Lorsque l'on considère la reconnaissance d'une portion de DDI « difficile » (figure 3.35), on peut noter que choisir un point p_{j+1} de manière arbitraire (version 1) peut induire la propagation d'une erreur importante dans la reconstruction. La version 2 engendre une erreur moindre, car les cônes qu'elle construit sont plus larges, et peuvent intersecter plus facilement les prochaines interfaces e_i entre cellules. Enfin, cette dernière version de la procédure de nouveau cône est plus adaptée lorsque l'on considère des k-arcs dont les interfaces successives sont alignées suivant un axe. Par exemple, dans la section suivante, nous construisons des k-arcs tels que $e_i \parallel (OY)$ pour toute interface e_i entre deux cellules. Nous donnons un exemple de reconstruction d'un tel ensemble de cellules par les deux versions dans la figure 3.36.



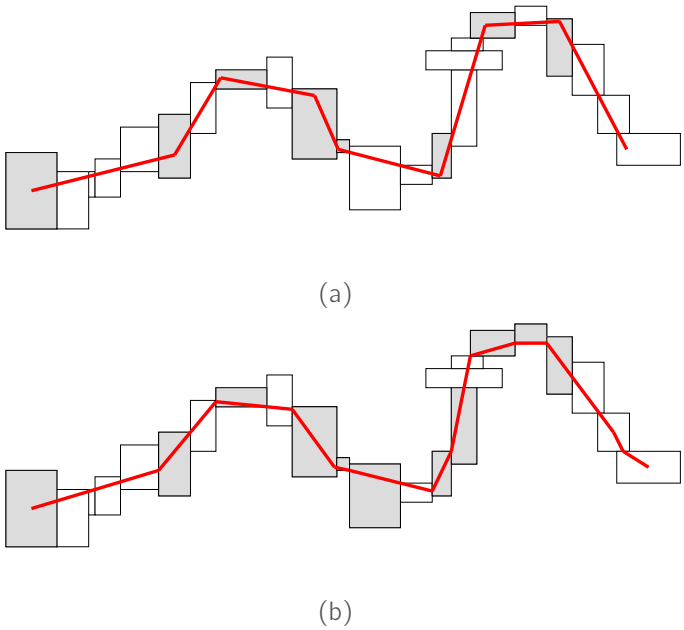
(a)



(b)

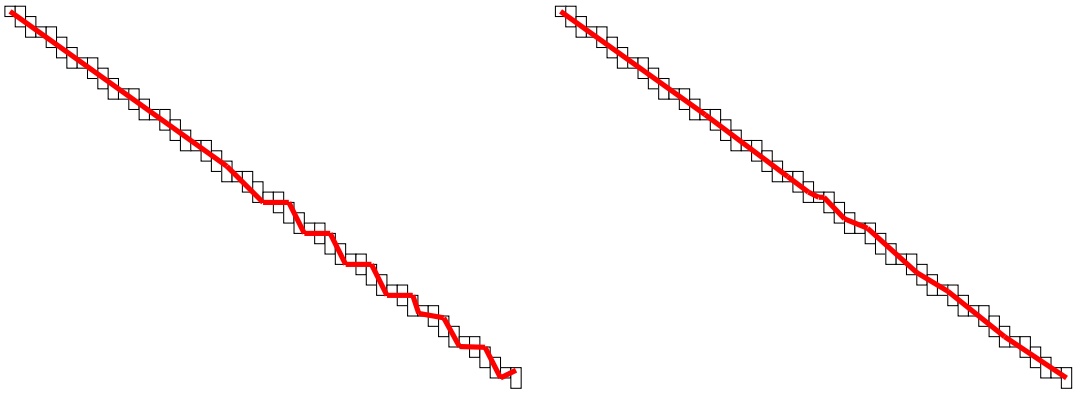
Fig. 3.33: Détails de la mise à jour des cône de visibilité grâce aux algorithmes 3.5 et 3.6. Les deux versions de nouveau cône sont présentées : par la bissectrice en (a) et par extension de cône en (b). Pour les deux cas, l'ensemble des cônes construits est illustré.

Nous pouvons remarquer que l'algorithme 3.6 est linéaire en le nombre de cellules traitées, soit en $\mathcal{O}(n)$, quelle que soit la méthode de construction de nouveau cône de visibilité. En effet, nous avons présenté ici une approche incrémentale, qui appelle à chaque étape l'algorithme 3.5, qui est en $\mathcal{O}(1)$. Nous présentons maintenant un système basé sur



(a) (b)

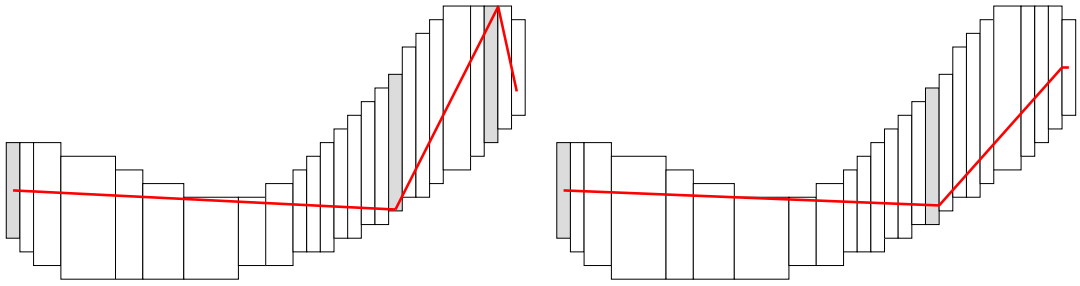
Fig. 3.34: Reconstruction d'un k-arc plus complexe avec les deux versions de notre algorithme : en (a) par la bissectrice, et en (b) par cône étendu. Les cellules grisées représentent l'endroit où un nouveau cône est nécessaire.



(a) (b)

Fig. 3.35: Reconstruction d'une DDI par les deux versions de notre algorithme : par la bissectrice en (a) et par extension de cône en (b).

les outils que nous avons présentés dans cette section, qui permet de reconstruire des objets complexes sur $\mathbb{I}$-grilles.



(a) (b)

Fig. 3.36: Reconstructions d'un k-arc où les interfaces e_i sont parallèles à l'axe des Y. Dans ce cas, la version par cône étendu (b) permet d'obtenir une reconstruction plus en accord avec la forme de départ. Au contraire, choisir un point arbitraire dans la version par bissectrice de cône altère la fin de la reconstruction (a). Les cellules grisées représentent l'endroit où un nouveau cône est nécessaire.

3.3 Représentation et reconstruction d'objets complexes sur $\mathbb{I}$-grilles

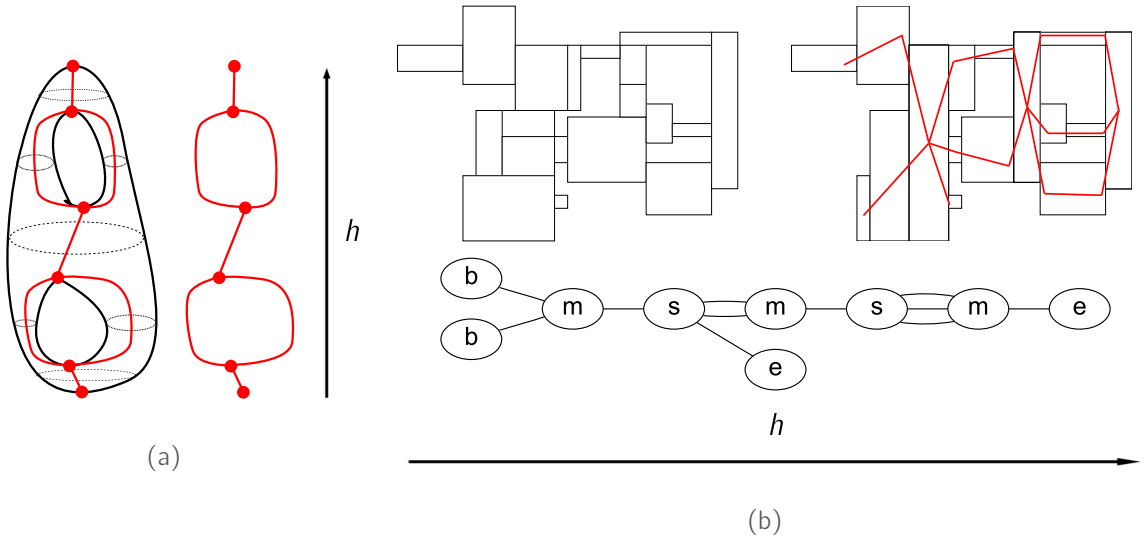
3.3.1 Représentation topologique par un graphe de Reeb discret irrégulier

Soit une $\mathbb{I}$-grille $\mathbb{I}$, pour représenter la forme d'un $k-$objet $\mathcal{E} \in \mathbb{I}$, nous avons choisi une approche incrémentale directionnelle, en accord avec un ordre lexicographique sur $\mathbb{I}$ basé sur les bords (voir partie I). Elle permet de construire son graphe de Reeb associé $\mathcal{G}$, comme dans le domaine continu (voir la figure 3.37). Ce graphe, basé sur la théorie de Morse [Milnor, 1963, Gramain, 1971, Hart, 1999, Matsumoto, 2002], est aussi utilisé dans diverses applications de description de courbes et de surfaces [Hétroy, 2003, Xiao *et al.*, 2003, Tung, 2005, Biasotti *et al.*, 2008]. Le graphe de Reeb $\mathcal{G}$ est associé à une *fonction de hauteur* (ou *fonction de Morse*) h définie sur $\mathcal{E}$, et les nœuds de $\mathcal{G}$ représentent les points critiques de h. Il peut être défini de la manière suivante :

Définition 3.15 (Graphe de Reeb)**.** *Soit h une fonction réelle définie sur une variété compacte M, $h : M \to \mathbb{R}$. Le graphe de Reeb de h est l'espace quotient de h dans $M \times \mathbb{R}$, par la relation d'équivalence $(p_1, h(p_1)) \sim (p_2, h(p_2))$ vérifiée si et seulement si :*

$$\begin{cases} h(p_1) = h(p_2), \\ p_1 \text{ et } p_2 \text{ appartiennent à la même composante connexe de } h^{-1}(h(p_1)). \end{cases}$$

De plus, nous voulons représenter un arc entre deux nœuds par un k-arc, pour avoir une information topologique minimale dans la représentation de $\mathcal{E}$. Ces arcs seront segmentés dans l'étape de description polygonale de $\mathcal{E}$.



(a) (b)

Fig. 3.37: Schéma global de la représentation d'un k-objet quelconque par le graphe de Reeb. En (a) est présenté un exemple de graphe de Reeb $\mathcal{G}$ pour un objet continu $\mathcal{E}$. Les nœuds de $\mathcal{G}$ représentent les points critiques de h (maxima, minima, point d'inflection), et une arête est une composante connexe de $\mathcal{E}$ entre deux points critiques. En (b), nous avons présenté un exemple d'objet irrégulier $\mathcal{E}$ (gauche), la structure finalement recodée avec des $k-$arcs (droite) et le graphe de Reeb associé à la fonction de hauteur h définie sur $\mathcal{E}$ (bas).

Rappelons tout d'abord que nous notons les bords gauche, droit, haut et bas d'une cellule R respectivement R^L, R^R, R^T et R^B. On a, par exemple, l'abscisse de R^L égale à $x_R - (l_R^X/2)$ (que l'on note par commodité $R^L = x_R - (l_R^X/2)$). Nous dirons également qu'un k-arc A et une cellule R sont k-adjacents s'il existe une cellule R' dans A telle que R et R' sont k-adjacentes. Soit $\mathcal{E} = \{R_i\}_{i=1,\ldots,n}$ un ensemble 2-D de cellules donné. On choisit en premier lieu une direction pour traiter les cellules de $\mathcal{E}$. Sans perte de généralité, on peut supposer que l'on prend l'orientation de gauche à droite, *i.e.* la fonction de hauteur h est définie suivant l'axe des X. Pour construire le graphe de Reeb associé à $\mathcal{E}$ suivant la fonction h, nous utilisons un ordre lexicographique basé sur les bords des cellules de $\mathbb{I}$. De la même manière que nous ordonnons les cellules d'une $\mathbb{I}$-grille, nous choisissons un ordre conforme à h sur $\mathcal{E}$. Par exemple, si nous décidons de considérer la relation d'ordre $\preceq_{lx}$, nous pouvons obtenir l'ensemble de cellules dont le bord R^L est aligné avec le plus petit bord gauche de $\mathcal{E}$, grâce à la procédure $\texttt{min}(\mathcal{E})$. Ensuite, on peut parcourir les cellules de $\mathcal{E}$ suivant les ensembles de cellules dont le bord gauche est aligné avec $\alpha \in \mathbb{R}$, avec α croissant. Un ensemble $\mathcal{E}_\alpha$ représente des groupes de cellules alignées et k-adjacentes deux à deux. En plus de ces cellules, nous souhaitons traiter les cellules intersectées par la droite d'équation $x = \alpha$. Cela permet de considérer ensemble des cellules qui font parties de la même composante connexe. Comme nous l'illustrons dans la figure 3.38, nous considérons en fait l'équivalent des *lacets* dans l'espace continu [Gramain, 1971] (voir également [Erickson et Har-Peled, 2002] pour l'utilisation de la notion de *lacet* pour

décomposer une surface 3-D). De plus, nous sommes en accord avec la définition d'un graphe de Reeb, où la fonction h peut être définie sur tous les points de $\mathcal{E}$ de même composante connexe.

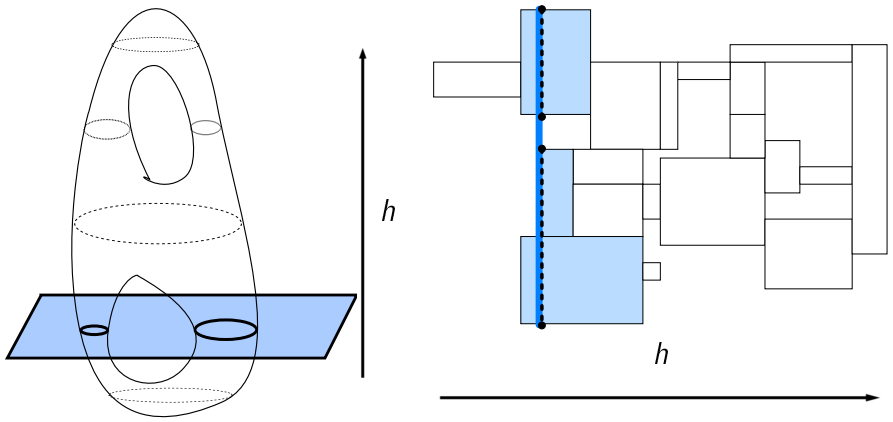


Fig. 3.38: Parcours des cellules d'un objet $\mathcal{E}$ par la relation d'ordre $\preceq_{lx}$. À chaque étape, en plus des cellules dont le bord gauche est aligné avec $\alpha \in \mathbb{R}$ croissant, nous souhaitons traiter les cellules intersectées par la droite d'équation $x = \alpha$ (b). Les groupes de cellules k-adjacentes (en pointillés) font partie d'une même composante connexe. Ainsi, on peut noter l'analogie avec la notion de lacet dans l'espace continu et la définition du graphe de Reeb (a).

Pour déterminer l'ensemble des cellules de même composante connexe, pour $\alpha \in \mathbb{R}$, nous proposons de recoder les cellules de $\mathcal{E}$. Nous décrivons maintenant notre algorithme principal, en détaillant étape par étape les opérations nécessaires pour construire un graphe de Reeb associé à E. Au temps $t = 1$, on fusionne ensemble toutes les cellules k-adjacentes R de $\mathcal{E}$ avec le plus petit bord gauche $\alpha_{t=1} = \texttt{min}(\mathcal{E})$, *e.g.* $R^L = 0$. Cette étape de fusion est réalisée par la procédure de mise à jour décrite ci-après. Ces m collections de cellules définissent les *cellules début* des k-arcs initialement reconnus $A_1, \ldots, A_m$.

Procédure de mise à jour Soit A un $k-$arc, et R_1, R_2 deux cellules adjacentes de $\mathcal{E}$ telles que $R_1 \in A$, $R_1^L < R_2^L$, et R_2 doit être ajoutée à A. Si $R_2^L = R_1^R$, on ajoute juste R_2 à A, sinon la procédure *met à jour* l'arc A avec R_2, et recode éventuellement A. Pour cela, on construit d'abord le *plus grand rectangle commun* F_2 de R_1 et R_2.

Définition 3.16 (Plus grand rectangle commun)**.** *Soit R_1 et R_2 deux rectangles adjacents. F_2 est le plus grand rectangle commun (ou PGRC) de R_1 et R_2 si et seulement si*

i) $F_2 \subseteq R_1 \cup R_2$,

ii) $F_2 \cap R_1 \neq \emptyset$,

iii) $F_2 \cap R_2 \neq \emptyset$,

iv) il n'existe pas de rectangle plus grand que F_2 par inclusion respectant i), ii) and iii).

Ensuite, nous considérons les rectangles $R_1 - F_2$ et $R_2 - F_2$. Si $R_1^R < R_2^R$, on note $R_1 - F_2 = F_1$ et $R_2 - F_2 = F_3$, sinon on préfèrera $R_1 - F_2 = \{F_1, F_3\}$. On peut remarquer que ces rectangles peuvent être vides, *e.g.* $F_3 = \emptyset$ si $R_1^R = R_2^R$, puisque dans ce cas $F_3^L = F_3^R$. La figure 3.39 présente cinq configurations générales de la procédure de mise à jour (il en existe cinq autres, obtenues par symétrie quand $R_2^T > R_1^T$), et le recodage de l'arc que l'on doit réaliser. De plus, nous proposons de réduire le nombre de cellules

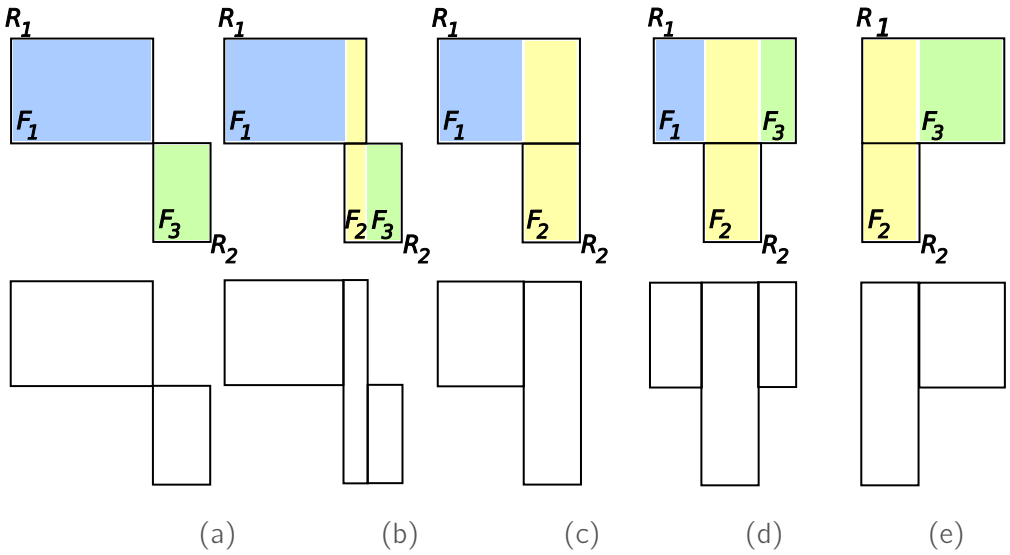


Fig. 3.39: Différentes configurations de construction du PGRC de deux cellules. Les rectangles F_1, F_2 et F_3 dans la procédure de mise à jour (haut), et les cellules associées (bas). Lorsque $R_1^R < R_2^R$ (a et b), $R_1 - F_2 = F_1$ et $R_2 - F_2 = F_3$, sinon $R_1 - F_2 = \{F_1, F_3\}$ (d et e). Si $R_1^R = R_2^L$, $F_2 = \emptyset$, quand $R_1^R = R_2^R$, $F_3 = \emptyset$ et finalement $F_1 = \emptyset$ dans le cas où $R_1^L = R_2^L$. Il existe cinq autres configurations, obtenues par symétrie quand $R_2^T > R_1^T$.

dans A en joignant les deux rectangles F_1 et F_3 si $F_1^T = F_3^T$, $F_1^B = F_3^B$ et $F_2 = \emptyset$. Cette jonction est traitée en remplaçant F_1 et F_3 par le rectangle $F_1 \cup F_3$. Enfin, la procédure se termine en supprimant R_1 de A, et en ajoutant les cellules correspondant aux rectangles F_1 et F_2 à A. F_3 est ajoutée à $\mathcal{E}$, et sera traité ultérieurement ; plus exactement au temps t tel que $\alpha_t = F_3^L$.

Au temps $t+1$, notre algorithme consiste à fusionner les cellules adjacentes avec le même bord gauche α_{t+1} en k cellules $R_1, R_2, \ldots, R_k$ (voir la procédure de mise à jour). Ces cellules candidates peuvent être ajoutées à un ou plusieurs arcs parmi $A_i,\ i \in \{1, \ldots, m\}$ si elles sont adjacentes à A_i. Il est clair que seule une cellule R construite au temps t et ayant un bord droit R^R égal à α_{t+1} peut respecter l'adjacence avec une cellule $R_j,\ j \in \{1, \ldots, k\}$. Une cellule R_j peut être traitée de plusieurs manières (voir également l'algorithme 3.7 en fin de section) :

- ▷ R_j n'est adjacente à aucun k-arc A_i. On initialise un nouveau k-arc A_{m+1} et on l'affecte avec la cellule R_j. R_j représente la *cellule begin* de A_{m+1}.

- ▷ Quand R_j est k-adjacent avec p k-arcs $A_i, A_{i+1}, \ldots, A_{i+p}$, c'est une *phase de regroupement*. Premièrement, on met à jour chaque arc avec R_j. La cellule R_j est ensuite marquée comme *cellule merge* et indique que chaque arc $A_i, \ldots, A_{i+p}$ possède un arc $A_{m+1} = \{R_j\}$ lié comme *arc suivant*.
- ▷ Le cas où p cellules $R_j, R_{j+1}, \ldots, R_{j+p}$ sont k-adjacentes avec un arc A_i est une *phase de découpe*. On met d'abord à jour A_i avec R_j par la procédure de mise à jour. Ensuite, on note R la cellule dans A_i telle que $R^R = \alpha_{t+1}$. Nous définissons aussi p nouveaux arcs suivants $A_{m+1}, \ldots, A_{m+p}$ de A_i tels que $A_{m+1} = \{R, R_j\}, \ldots, A_{m+p} = \{R, R_{j+p}\}$. Dans ces p arcs et dans A_i, R est marqué comme une *cellule split*.

Quand l'algorithme se termine, au temps t tel que α_t est le plus grand bord gauche dans $\mathcal{E}$ (*i.e.* lorsque l'on a atteint la plus grande cellule suivant $\preceq_{lx}$, notée $\mathtt{max}(\mathcal{E})$), on définit la dernière cellule ajoutée dans tous les arcs A_i comme une *cellule end*. Nous illustrons dans la figure 3.40 la construction progressive du graphe et le recodage du k-objet présenté dans la figure 3.37 dans cinq étapes de l'algorithme. De plus, nous rappelons dans la table 3.2 la signification des notations des nœuds du graphe de Reeb.

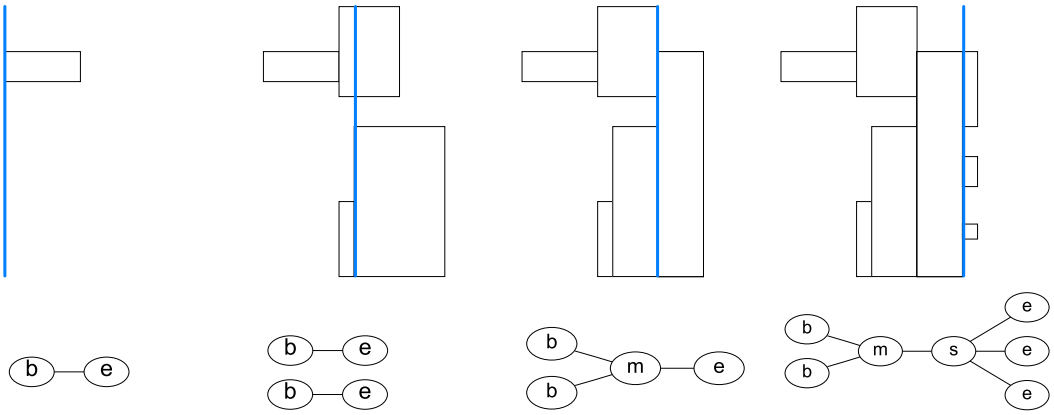


Fig. 3.40: Les arcs reconnus et le graphe de Reeb associé pour quelques itérations de notre méthode sur l'objet présenté figure 3.37. D'abord, on initialise un arc avec la cellule de plus petit bord gauche. Puis, on met à jour et on recode progressivement les arcs. La troisième et la quatrième image présentent les phases de regroupement et de découpe. On peut noter que l'étape de recodage n'est pas détaillée dans cette figure, et que les arcs $m - s$ représentent un k-arc avec une seule cellule dans cet exemple. Nous avons également ajouté la droite d'équation $x = \alpha_t$, pour chaque itération.

Notre algorithme construit finalement une représentation homotope complète de $\mathcal{E}$ grâce au graphe de Reeb $\mathcal{G}$ en reconnaissant et en liant les cellules b, m, s et e (voir également la figure 3.37 et la table 3.2). Il existe neuf configurations possibles d'arcs dans $\mathcal{G}$: $b - s$, $b - m$, $b - e$, $s - s$, $s - m$, $s - e$, $m - s$, $m - m$ et $m - e$. Le nombre de points critiques dans h peut être lié au nombre d'Euler χ de $\mathcal{E}$ [Reeb, 1946]. Considérons l'équation suivante, où $\mathcal{G}$ est noté comme le couple d'ensembles de sommets et d'arêtes

Notation	Signification
b	Début (*begin*) d'un nouveau k-arc, dans le sens de h
s	Division (*split*) d'un k-arc en plusieurs k-arcs adjacents
m	Regroupement (*merge*) de plusieurs k-arcs en un k-arc adjacent
e	Fin (*end*) d'un k-arc selon le sens de h

Tab. 3.2: Signification des notations b, s, m et e dans le graphe de Reeb.

(V, E) :

$$\chi = \sum_{n \in V, (n=b) \vee (n=e)} (deg(n)) - \sum_{n \in V, (n=s) \vee (n=m)} (deg(n) - 2)$$

où $deg(n)$ est le *degré* du nœud n dans $\mathcal{G}$, donc $deg(n) = 1$ si n est un nœud *begin* ou *end*. Le nombre d'Euler permet de décrire la topologie d'un objet par une valeur unique. Par exemple, pour un tore, $\chi = 0$, pour un disque, $\chi = 2$, et l'objet décrit dans la figure 3.37 *(b)* possède un nombre d'Euler $\chi = -4$; on peut également dire que cette forme est homéomorphe à un tore avec trois trous, où $\chi = 2 - 2 \times |trous| = -4$. Le nombre d'Euler est un exemple de l'emploi du graphe de Reeb pour la description de forme. Au-delà de ces invariants topologiques obtenus par les points critiques, la structure du graphe dépend clairement de la direction choisie pour la fonction de hauteur h. Une partie des nœuds et des arcs peut changer, mais l'information sur la topologie de $\mathcal{E}$, *i.e.* les nœuds internes de $\mathcal{G}$, n'est pas modifiée. Soit maintenant $\mathcal{E}'$ l'objet dessiné dans la quatrième image de la figure 3.40. Les trois cellules ajoutées durant la dernière itération peuvent représenter du bruit modifiant le contour de $\mathcal{E}'$. Le graphe de Reeb est changé par une phase de découpe, trois nœuds sont créés, alors que ces cellules sont peut-être nuisibles. En fait, le problème de la perturbation du contour de $\mathcal{E}'$ pourrait être certainement réduit si l'objet était d'abord filtré ou lissé. Ce genre de pré-traitement est souvent adopté, quelle que soit l'approche choisie pour la représentation de forme, *e.g.* la squelettisation. Enfin, avec la procédure de mise à jour, nous recodons les cellules de $\mathcal{E}$ afin qu'un k-arc soit toujours représenté entre deux nœuds de $\mathcal{G}$. Ce réarrangement géométrique dépend assurément de la direction de h, mais ne change ni la topologie ni le contour du k-arc reconnu. Cela implique pour la polygonalisation que nous décrivons ci-après que nous ne modifions pas l'espace des reconstructions par supercouverture étendue. La structure topologique est simple, et prépare la phase suivante de notre système global de reconstruction d'objets complexes.

3.3.2 Reconstruction polygonale d'objets complexes

Comme la reconstruction en polylignes affecte toujours le premier point p_0 comme le centre de la première cellule traitée (section précédente), nous proposons de commencer la reconstruction de tous les k-arcs calculés dans la phase précédente par les nœuds *merge* et *split* détectés dans le graphe de Reeb $\mathcal{G}$. Nous assurons ainsi que chacun de

ces nœuds particuliers de $\mathcal{G}$ sera représenté par un seul point dans la polygonalisation finale. Les segments sont reconnus de l'intersection entre plusieurs parties de l'objet $\mathcal{E}$ vers ses extrémités, *i.e.* nous considérons les arcs $m-e$, $s-e$, $b-m$ et $b-s$ de $\mathcal{G}$. De plus, puisque l'algorithme de reconnaissance est glouton, les éventuelles erreurs induites par notre approche de cône de visibilité sont propagées vers les extrémités de $\mathcal{E}$, et non pas vers ces intersections qui décrivent la forme de l'objet. Ce problème de l'erreur de reconstruction est indépendant de l'approche que l'on choisit : cône de visibilité, préimage généralisée [Dexet, 2006], *etc.* La plupart des algorithmes de reconstruction, comme le nôtre, optent pour une approche gloutonne, qui aboutit généralement à la propagation d'une erreur. Pour les arcs $s-s$, $m-m$, $m-s$ et $s-m$ de $\mathcal{G}$, nous proposons de réaliser une reconstruction bidirectionnelle qui démarre de chaque nœud de l'arc, et se termine en son centre. Par conséquent, l'éventuelle erreur de reconstruction serait concentrée au milieu de ces arcs. Cette approche se justifie par le fait que les nœuds m et s de $\mathcal{G}$ représentent les lieux où la description de sa géométrie doit être précise. Enfin, nous choisissons de traiter les arcs $b-e$ avec la même reconstruction bidirectionnelle, qui apparaît comme le moyen le plus efficace de garantir une reconstruction robuste. Dans la figure 3.41, nous montrons l'intérêt de ce principe pour un cas pathologique de la reconstruction par cône de visibilité. De plus, nous donnons le résultat de la reconstruction avec nos deux versions de mise à jour du cône de visibilité (voir section précédente pour plus de détails). Nous pouvons remarquer dans la figure 3.42 que la reconstruction que nous venons de décrire permet également de traiter efficacement les objets symétriques. On peut noter dans ces expérimentations que la structure que nous proposons respecte le modèle de supercouverture étendu aux $\mathbb{I}$-grilles. Dans cette thèse, nous ne traitons pas le problème de liaison entre deux reconstructions sur le k-arc (reconstruction avec *patch*), car une technique générale et efficace de jointure entre deux droites discrètes impliquerait que notre algorithme ne serait plus linéaire [Breton, 2003]. Par conséquent, nous ajoutons juste un segment entre les deux polylignes. Cette phase de notre système ne peut être considérée sans patch, puisque nous utilisons les points internes de la forme de $\mathcal{E}$ pour guider la reconstruction géométrique.

Bilan algorithmique de la méthode proposée Nous donnons maintenant l'algorithme global 3.7 qui regroupe le construction du graphe de Reeb discret irrégulier associé à l'objet traité $\mathcal{E}$, et la polygonalisation en considérant ce graphe et les k-arcs reconnus. Dans la première phase de l'algorithme, nous récapitulons les différentes configurations de mise à jour du graphe de Reeb $\mathcal{G}$ en fonction des cellules adjacentes avec les k-arcs déjà traités. Nous avons décrit notre algorithme en considérant les cellules $\{R_i\}_{i=1,n}$ ordonnées suivant une relation $\preceq_{lx}$ par exemple. Ainsi, nous avons utilisé implicitement la procédure $\texttt{suivant}(R_i) = R_{i+1}$, pour $i \in \{1, n-1\}$. On pourrait aussi présenter l'algorithme avec les ensembles $\mathcal{E}_{\alpha_t}$, $t \in \{1, T\}$, avec un fonctionnement similaire.

Comme un k-arc A est associé à une arête $n_1 - n_2 \in E, n_1, n_2 \in \{b, e, m, s\}$, et puisque nous avons choisi une orientation pour le parcours des cellules, nous notons ici $inf(A) = n_1$ le début du k-arc et $sup(A) = n_2$ la fin. La complexité de notre méthode

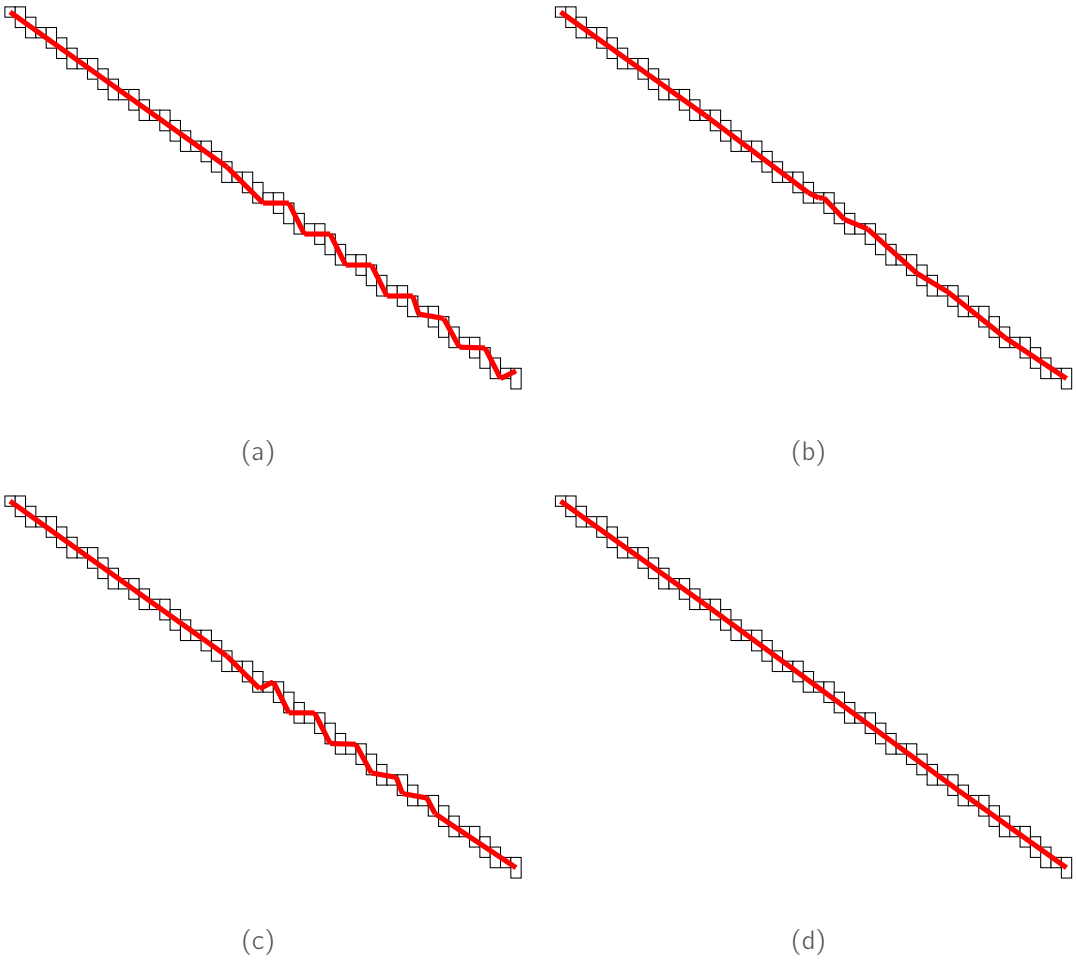
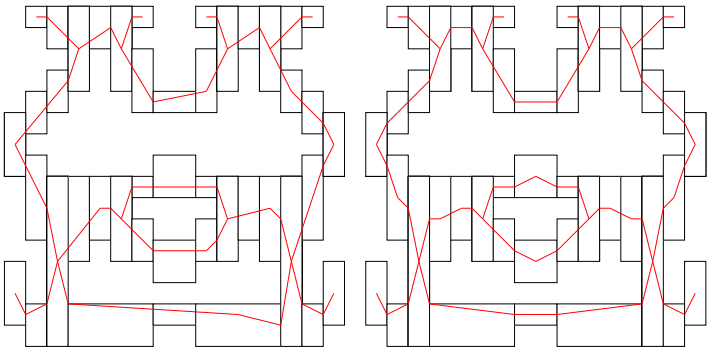


Fig. 3.41: Intérêt de notre approche sur un exemple de DDI. Nous rappelons dans cette figure la reconstruction de la gauche vers la droite avec la version 1 (a) et la version 2 (b) de la procédure de construction d'un nouveau cône. En (c), nous choisissons une reconstruction, par la bissectrice de cône, des cellules extrêmes vers l'intérieur de ce k-arc. Cela permet de centrer l'erreur commise, et de la diminuer. Enfin, nous montrons en (d) le résultat de cette reconstruction avec notre nouvelle méthode de création de cône de visibilité (par extension de cône).

est au pire cas en $\mathcal{O}(n_a \times n)$, où n est le nombre de cellules 2-D traitées, et n_a est le nombre total de k-arcs qui recodent $\mathcal{E}$. Dans la dernière section de ce chapitre, nous détaillons des expérimentations sur des objets plus complexes pour montrer la robustesse et la rapidité d'exécution de cet algorithme. De plus, on pourra se référer aux chapitres 7 et 6 de la partie III qui montrent des applications de cet algorithme pour le tracé de courbes implicites et la reconnaissance de formes.



(a) (b)

Fig. 3.42: Exemple de reconstruction respectant la symétrie de l'objet traité. En (a) chaque k-arc est polygonalisé de la gauche vers la droite, tandis qu'en (b) nous nous servons de la configuration des nœuds du graphe de Reeb pour guider la reconstruction.

3.4 Reconstruction et représentation dynamiques sur $\mathbb{I}$-grilles

Dans cette section, nous proposons de mettre à jour efficacement la reconstruction que nous avons décrite précédemment, en considérant une ou plusieurs opérations locales de raffinement et de groupement de cellules. Ces schémas peuvent être dirigés de manière supervisée, en choisissant manuellement les cellules à traiter et la mise à jour à effectuer. Généralement, un modèle de restructuration guide ces opérations, comme le quadtree ou encore l'algorithme de *split-and-merge* [Horowitz et Pavlidis, 1977] par exemple. Nous montrons dans le chapitre 7 une application où la mise à jour de l'approximation de courbes implicites est guidée par l'analyse en intervalles. Nous présentons en détails deux algorithmes pour modifier localement le graphe de Reeb et la structure polygonale d'un objet irrégulier. Rappelons que les éléments calculés par l'algorithme 3.7 sont $\mathcal{G}$, $\mathcal{L}$ et $\mathcal{A}$. De plus, pour chaque k-arc $A_i \in \mathcal{A}$, l'arête associée dans $\mathcal{G}$ est notée en minuscules a_i.

3.4.1 Schéma de raffinement local

Soit $R \in \mathcal{E}$ une cellule où nous désirons une décomposition plus précise, *e.g.* un raffinement par quadtree du niveau t au niveau $t+1$. On note par $\mathtt{RS}$ le schéma de raffinement effectué sur R. Par conséquent, $\mathtt{RS}(R)$ est un ensemble de cellules disjointes appartenant au domaine borné par R, obtenues par $\mathtt{RS}$. Avec ces notations, notre problème consiste donc à mettre à jour $\mathcal{G}$, $\mathcal{L}$ et $\mathcal{A}$, en prenant en compte $\mathtt{RS}(R)$. En premier lieu, nous considérons tous les k-arcs dans $\mathcal{A}$ qui contiennent R. Puis, nous remplaçons R par les nouvelles cellules $\mathtt{RS}(R)$, et nous utilisons l'algorithme 3.7 pour obtenir une reconstruction locale de ces cellules. Puis, nous joignons les k-arcs localement reconstruits représentés dans $\mathcal{G}_R$, $\mathcal{L}_R$ et $\mathcal{A}_R$ (associés à $\mathtt{RS}(R)$) dans $\mathcal{G}$, $\mathcal{L}$ et $\mathcal{A}$ respectivement. Dans la figure 3.43,

Algorithme 3.7 : Algorithme global de reconstruction topologique et géométrique d'objets sur $\mathbb{I}$-grilles.

entrée : $\mathcal{E} = \{R_i\}_{i=1,\ldots,n}$ un ensemble de cellules 2-D dans une $\mathbb{I}$-grille $\mathbb{I}$.
sortie : $\mathcal{G} = (V, E)$, le graphe de Reeb associé à $\mathcal{E}$,
$\mathcal{L}$, la reconstruction par segments de droites de $\mathcal{E}$,
$\mathcal{A} = \{A_i\}_{i=1,\ldots,n_a}$, l'ensemble des n_a k-arcs recodant $\mathcal{E}$.

début
ajouter une arête $b - e$ dans $\mathcal{G}$; // $V = \{b, e\}$, $E = \{b - e\}$
créer un k-arc A_1 avec $inf(A_1) \leftarrow b$ et $sup(A_1) \leftarrow e$; // $n_a = 1$
pour chaque *cellule* R_i, $2 \leq i \leq n$ **faire**
pour chaque *k-arc* A_j, $1 \leq j \leq n_a$ **faire**
si *une cellule* R_i *est adjacente avec un k-arc* A_j **alors**
mettre à jour A_j avec R_i ;
si p *cellules* $R_i, R_{i+1}, \ldots, R_{i+p}$ *sont adjacentes avec* A_j **alors**
mettre à jour A_j avec $R_i, R_{i+1}, \ldots, R_{i+p}$;
ajouter un nœud *split* s dans $\mathcal{G}$; // $deg(s) = p + 1$
$sup(A_j) = s$;
joindre p nouveaux k-arcs $A_{n_a+1} \leftarrow R_i, \ldots, A_{n_a+p} \leftarrow R_{i+p}$ avec A_j ;
$inf(A_{n_a+1}) \leftarrow s, \ldots, inf(A_{n_a+p}) \leftarrow s$;
si *une cellule* R_i *est adjacente avec* p *k-arcs* $A_j, A_{j+1}, \ldots, A_{j+p}$ **alors**
mettre à jour $A_j, A_{j+1}, \ldots, A_{j+p}$ avec R_i ;
ajouter un nœud *merge* m dans $\mathcal{G}$; // $deg(m) = p + 1$
$sup(A_j) \leftarrow m, \ldots, sup(A_{j+p}) \leftarrow m$;
joindre un nouveau k-arc $A_{n_a+1} \leftarrow R_i$ avec $A_j, A_{j+1}, \ldots, A_{j+p}$;
$inf(A_{n_a+1}) \leftarrow m$;
si *une cellule* R_i *n'est adjacente avec aucun k-arc* A_j **alors**
ajouter une arête $b - e$ dans $\mathcal{G}$;
créer un k-arc A_{n_a+1} avec $inf(A_{n_a+1}) \leftarrow b$ et $sup(A_{n_a+1}) \leftarrow e$;
procéder à la polygonalisation de l'ensemble des k-arcs $\{A_j\}_{j=1,\ldots,n_a}$ en considérant $\mathcal{G}$;
retourner $\mathcal{G}$, $\mathcal{L}$, $\mathcal{A}$
fin

nous illustrons le comportement de notre algorithme dans le cas où R appartient à un seul k-arc ($n_a = 1$ dans l'algorithme 3.8), et quand $n_a > 1$, *i.e.* R appartient à plusieurs k-arcs.

Pour trouver les n_a k-arcs dans la première opération de l'algorithme 3.8, nous utilisons un pointeur entre la cellule R et un k-arc $A_k \in \mathcal{A}$ tel que $R \in A_k$. Si R est à l'intérieur de A_k, il est clair que $n_a = 1$, et par conséquent seul un k-arc est considéré dans la suite. Lorsque R se situe sur une des extrémités de A_k, nous devons prendre en compte

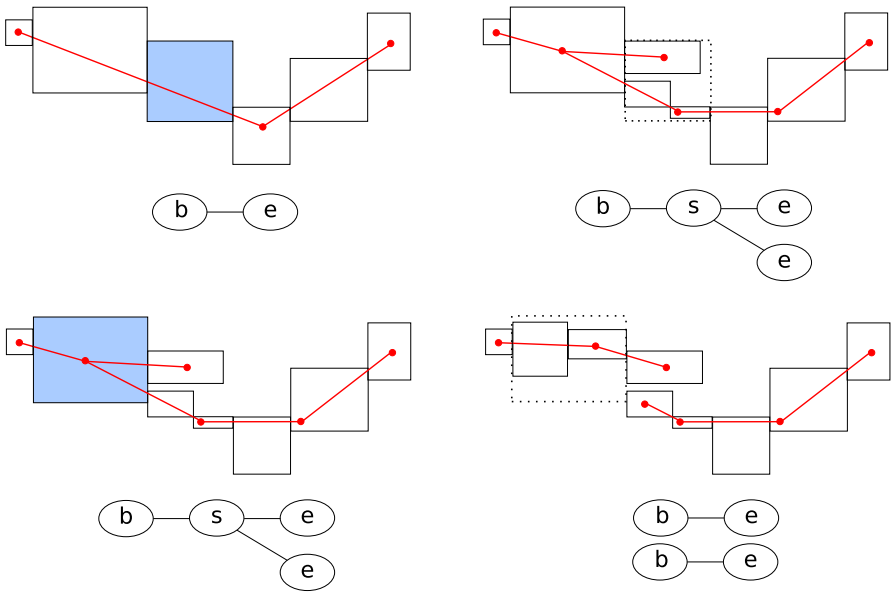


Fig. 3.43: Exemples de résultats obtenus par l'algorithme 3.8 de raffinement de cellules. La première reconstruction (haut-gauche) ne contient qu'un seul k-arc, puis on sélectionne la cellule grisée et on la décompose en deux k-arcs (haut-droite). Cette cellule n'appartient qu'à un seul k-arc, donc $n_a = 1$ et l'arête $b - e$ doit être divisée en deux avant d'ajouter le nœud s. Le raffinement suivant (bas), concerne trois k-arcs, puisque l'on choisit de diviser la cellule grisée ($n_a > 1$). Cette cellule et son nœud associé (s) sont supprimés avant d'insérer les nouvelles arêtes.

tous les k-arcs liés à A_k. Grâce à l'arête associée a_k dans $\mathcal{G}$, et l'accès direct aux arêtes liées à a_k, cette opération de recherche est réalisée en temps constant $\mathcal{O}(1)$. Ensuite, les étapes suivantes modifient les n_a arêtes $\{a_j\}_{j=k,k+n_a}$ du graphe de Reeb $\mathcal{G}$ et impactent les n_a k-arcs $\{A_j\}_{j=k,k+n_a}$ trouvés dans la première opération. La topologie de l'objet $\mathcal{E}$ est mise à jour localement. Le reste du graphe ne subit aucune modification. Dans la dernière phase de l'algorithme, la reconstruction polygonale est d'abord réalisée sur les k-arcs appartenant au raffinement local $\mathcal{A}_R$. Puis, cette reconstruction peut être propagée aux n_a k-arcs globaux dans $\mathcal{A}$, comme dans la figure 3.43. En fait, plusieurs stratégies peuvent être adoptées. On pourrait également joindre les segments de droites locaux aux globaux. Nous avons préféré propager la reconstruction aux k-arcs adjacents, pour obtenir une structure polygonale plus homogène. Dans le pire cas, la mise à jour de cette structure est effectuée de manière locale sur $\mathcal{A}_R$ et $\mathcal{A}$, et ne modifie pas le reste de la reconstruction. La complexité au pire cas de l'algorithme 3.8 est en $\mathcal{O}\big(|\mathtt{RS}(R)| + \Delta n_a\big)$, où $\Delta = |\mathcal{G}_R| + \max_{k=1,n_a} |A_k|$ représente (1) la taille du graphe de Reeb qui est impliqué dans la phase de connexion, lors de la double boucle commençant à la ligne 1 et (2) la taille maximale des k-arcs voisins, puisque nous avons choisi d'y effectuer une reconstruction polygonale (ligne 2 de l'algorithme). Finalement, on peut noter que notre algorithme impacte localement la reconstruction topologique et géométrique calculée à partir de $\mathcal{E}$.

Algorithme 3.8 : Algorithme de raffinement d'une cellule dans la reconstruction d'un objet $\mathcal{E} \in \mathbb{I}$.

entrée : $\mathcal{E} = \{R_i\}_{i=1,\ldots,n}$ un ensemble de cellules 2-D dans une $\mathbb{I}$-grille $\mathbb{I}$,
une cellule $R \in \mathcal{E}$,
$\mathtt{RS}(R)$, la décomposition locale de $R \in \mathcal{E}$ par $\mathtt{RS}$
$\mathcal{G}$, le graphe de Reeb associé à $\mathcal{E}$,
$\mathcal{L}$, l'ensemble des segments de droites reconstruites à partir de $\mathcal{E}$,
$\mathcal{A}$, l'ensemble des k-arcs recodant $\mathcal{E}$.
sortie : $\mathcal{G}$, $\mathcal{L}$ et $\mathcal{A}$ sont mis à jour.

début
  trouver les n_a k-arcs $A_k, \ldots, A_{k+n_a} \in \mathcal{A}$ tels que $R \in A_k, \ldots, A_{k+n_a}$;
  si $n_a > 1$ **alors**
    // R représente un nœud *split* ou *merge* dans $\mathcal{G}$
    supprimer le nœud $n \in \mathcal{G}$ associé à R ;
  sinon
    // R est dans un seul k-arc
    décomposer a_k en deux arêtes a_k, a_{k+1} ;
  supprimer les polylignes de $\mathcal{L}$ associés aux k-arcs $A_k, \ldots, A_{k+n_a}$;
  supprimer R de $A_k, \ldots, A_{k+n_a}$;
  utiliser l'algorithme 3.7, qui calcule $\mathcal{G}_R$, $\mathcal{L}_R$ et $\mathcal{A}_R$ associés à $\mathtt{RS}(R)$;
1 **pour chaque** *arête* $a'_i \in \mathcal{G}_R$ **faire**
    pour chaque *arête* $a_j \in \mathcal{G}$, $j = k, \ldots, k + n_a$ **faire**
      si A_j *et* A'_i *sont k-adjacents* **alors**
        joindre a'_i à a_j ;
        fusionner les k-arcs A_j et A'_i ;
  ajouter tous les polylignes de $\mathcal{L}_R$ dans $\mathcal{L}$;
2 **pour chaque** k-*arc* A_j, $j = k, \ldots, k + n_a$ **faire**
    reconstruire A_j en segments de droites ;
fin

Pour employer notre méthode sur plusieurs cellules raffinées, on calcule tout d'abord le graphe de Reeb avec le graphe local en utilisant la même approche. Puis, nous réalisons la polygonalisation finale sur les k-arcs concernés.

3.4.2 Schéma de groupement local

Soit $R_1, R_2, \ldots, R_n \in \mathcal{E}$ un ensemble de cellules que l'on souhaite grouper ensemble (*e.g.* une fusion locale de quatre nœuds d'un quadtree). Dans une application interactive, la sélection de $\{R_i\}_{i=1,n}$ pourrait être effectuée en traçant un rectangle dans le plan. Dans ce cas, nous forçons l'utilisateur à choisir des paquets de cellules adjacentes dans un ou

plusieurs k-arcs de $\mathcal{A}$. Comme dans le précédent algorithme, nous effectuons tout d'abord une recherche de tous les k-arcs dans $\mathcal{A}$ qui contiennent une cellule de $\{R_i\}_{i=1,n}$. Nous supprimons des cellules de $\{R_i\}_{i=1,n}$ dans ces k-arcs et nous mettons à jour les arêtes associées dans $\mathcal{G}$. Dans l'algorithme 3.9, nous devons considérer si une cellule R_i telle que $R_i \in A_j$ se situe ou non à l'extrémité de A_j. Puis, nous remplaçons ces cellules par la plus petite cellule englobante $B = \texttt{GS}(\{R_i\}_{i=1,n})$ telle que $R_1, R_2, \ldots, R_n \subseteq B$. D'autre méthodologies de regroupement, manuelles ou automatiques (par exemple [Horowitz et Pavlidis, 1977]), peuvent être envisagées pour définir la manière de sélectionner $\{R_i\}_{i=1,n}$ et le processus GS, mais elles n'impliquent pas de changement dans notre algorithme. Dans notre cadre, la reconstruction locale de cette cellule est une arête unique $b - e$ dans $\mathcal{G}_B$ et un seul point dans $\mathcal{L}_B$. Nous avons juste à joindre ces informations locales simples aux globales, et réaliser ensuite une reconstruction polygonale sur les k-arcs de $\mathcal{A}$ mis à jour. Nous illustrons dans la figure 3.44 les deux cas pris en compte dans notre algorithme : un seul k-arc est concerné par la mise à jour ($n_a = 1$), et plusieurs k-arcs sont impliqués ($n_a > 1$).

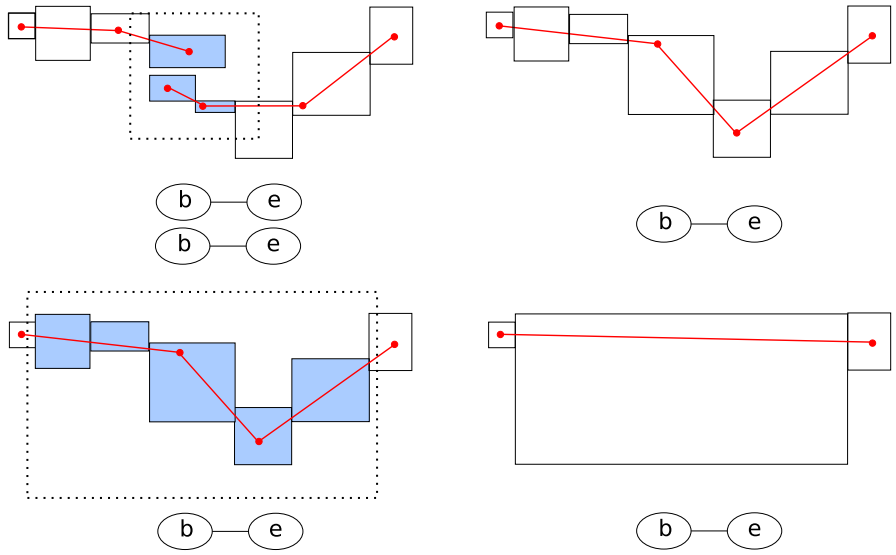


Fig. 3.44: Exemples de résultats obtenus par l'algorithme 3.9 de regroupement de cellules. La première mise à jour (haut) permet de joindre deux arêtes dans le graphe de Reeb. La seconde (bas) ne change pas la topologie de l'objet, mais engendre une reconstruction polygonale plus simple et une description en k-arcs plus grossière. La sélection de l'utilisateur est tracée en pointillés.

Dans l'algorithme 3.9, la première opération de recherche est toujours réalisée en temps constant $\mathcal{O}(1)$ pour chaque cellule R_i, comme dans l'algorithme précédent. L'opération suivante doit être réalisée pour toutes les cellules de $\{R_i\}_{i=1,n}$ et pour tous les k-arcs de $A_k, \ldots, A_{k+n_a}$. Par conséquent, elle n'implique que ces n_a k-arcs et leurs arêtes associées dans le graphe de Reeb. Puis, la reconstruction par l'algorithme 3.7

Algorithme 3.9 : Algorithme de regroupement de cellules dans la reconstruction d'un objet $\mathcal{E} \in \mathbb{I}$.

entrée : $\mathcal{E} = \{R_i\}_{i=1,\dots,n}$ un ensemble de cellules 2-D dans une $\mathbb{I}$-grille $\mathbb{I}$,
des cellules $\{R_i\}_{i=1,n} \in \mathcal{E}$,
$\mathtt{GS}(\{R_i\}_{i=1,n}) = B$, la cellule englobante de $\{R_i\}_{i=1,n}$ par $\mathtt{GS}$,
$\mathcal{G}$, le graphe de Reeb associé à $\mathcal{E}$,
$\mathcal{L}$, l'ensemble des segments de droites reconstruites à partir de $\mathcal{E}$,
$\mathcal{A}$, l'ensemble des k-arcs recodant $\mathcal{E}$.
sortie : $\mathcal{G}$, $\mathcal{L}$ et $\mathcal{A}$ sont mis à jour.

début
  trouver les n_a k-arcs $A_k, \dots, A_{k+n_a} \in \mathcal{A}$ tels que $R_i \in A_k, \dots, A_{k+n_a}$, $i = 1, \dots, n$;
  pour chaque *k-arc* A_j, $j = k, \dots, k + n_a$ **faire**
    pour chaque *cellule* R_i, $i = 1, \dots, n$ **faire**
      si $R_i \in A_j$ **alors**
        supprimer R_i de A_j ;
        mettre à jour l'arête a_j dans $\mathcal{G}$;
  utiliser l'algorithme 3.7, qui calcule $\mathcal{G}_B$, $\mathcal{L}_B$ et $\mathcal{A}_B$ associés à B ;
  pour chaque *arête* a_j, $j = k, \dots, k + n_a$ **faire**
    // $\mathcal{A}_B$ ne contient qu'un seul élément A'_1
    si A_j *et* A'_1 *sont k-adjacents* **alors**
      joindre a'_1 à a_j ;
      fusionner les k-arcs A_j et A'_1 ;
  pour chaque *k-arc* A_j, $j = k, \dots, k + n_a$ **faire**
    reconstruire A_j en segments de droites ;
fin

est très rapide (en $\mathcal{O}(1)$), puisque nous ne considérons que la cellule B. La phase de fusion impacte des k-arcs et des arêtes de manière locale, et ne modifie pas le reste de notre structure. Enfin, la reconstruction polygonale est toujours propagée localement dans $\mathcal{A}$. La complexité de l'algorithme 3.9 est donc en $\mathcal{O}\big(\Gamma \times n_a\big)$, où nous avons noté $\Gamma = |\{R_i\}_{i=1,n}| + \max_{k=1,n_a} |A_k| = n + \max_{k=1,n_a} |A_k|$. Ces opérations sont dues à : (1) nous traitons l'ensemble des cellules $\{R_i\}_{i=1,n}$ de taille n, et (2) nous réalisons la reconstruction polygonale sur les n_a k-arcs adjacents à B.

Bilan des algorithmes dynamiques proposés Dans cette section, les algorithmes de raffinement et de regroupement de cellules que nous avons proposés n'impliquent que des éléments locaux dans l'ensemble des k-arcs $\mathcal{A}$ et dans le graphe de Reeb irrégulier $\mathcal{G}$. Ces mises à jour sont efficacement et rapidement effectuées, grâce aux structures que nous avons mises en place. La reconstruction polygonale n'est opérée que sur l'ensemble des

n_a k-arcs adjacents aux nouvelles cellules créées, et permet une représentation globale par segments correcte et homogène. Enfin, on peut noter que nous respectons toujours le modèle de supercouverture étendu aux $\mathbb{I}$-grilles. Nous montrons dans le chapitre 7 de la partie III des expérimentations plus poussées sur la reconstruction dynamique d'objets irréguliers issus de la description de courbes implicites par l'arithmétique d'intervalles. Les mises à jour lors du raffinement de cellules sont également guidées par cette arithmétique.

3.5 Expérimentations et analyse des algorithmes proposés

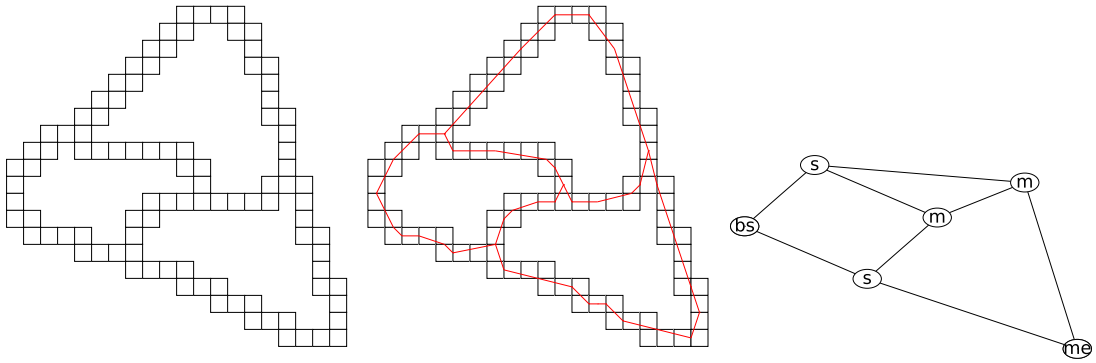
Comme nous l'avons évoqué dans l'introduction de ce chapitre, la reconstruction exacte d'objets discrets réguliers a amené au développement de nombreux algorithmes. Ici, nous montrons dans un premier temps quelques résultats obtenus avec l'algorithme 3.7 que nous avons proposé, sur ce type d'objets. Ensuite, nous détaillons quelques exemples de reconstructions d'objets irréguliers isothétiques.

3.5.1 Reconstruction d'objets discrets réguliers

Nous présentons dans la figure 3.45 un exemple de courbe régulière 4-connexe. Grâce à notre algorithme, nous proposons une reconstruction polygonale qui respecte le modèle de supercouverture étendu, et le graphe de Reeb discret de l'objet. On peut noter que cette structure contient 3 cycles, ce qui signifie que l'objet contient 3 trous. Dans la figure 3.46 sont montrés les mêmes éléments calculés à partir d'une image binaire complexe de grande taille. En observant la partie zoomée de cette figure, on peut remarquer que notre approche permet de proposer une polygonalisation pour les objets épais [Alhalabi et Tougne, 2005, Debled-Rennesson *et al.*, 2005].

3.5.2 Reconstruction d'objets discrets irréguliers isothétiques

Nous nous intéressons en premier lieu à la représentation d'objets binaires appartenant à une image compressée par une décomposition en quadtree (figure 3.47). Comme notre approche construit les k-arcs par calcul du PGRC de deux cellules adjacentes, puis par regroupement des cellules de même hauteur, on peut noter que la reconstruction polygonale et le graphe de Reeb sont les mêmes quelle que soit la $\mathbb{I}$-grille choisie pour décrire l'image binaire. En d'autres termes, cela signifie que notre algorithme n'est sensible qu'aux contours des objets irréguliers isothétiques traités. Ces remarques sont valables pour une fonction de hauteur h donnée, par exemple de la gauche vers la droite comme nous l'avons choisi ici. Lorsque l'on décrit une courbe implicite par des hyper-rectangles issus d'une analyse en intervalles, le tracé de segments de droites au sein de ces cellules permet d'approximer la courbe initiale, comme nous le montrons dans la figure 3.48. Dans cette dernière, nous avons choisi la courbe implicite de fonction f associée telle que



(a) (b) (c)

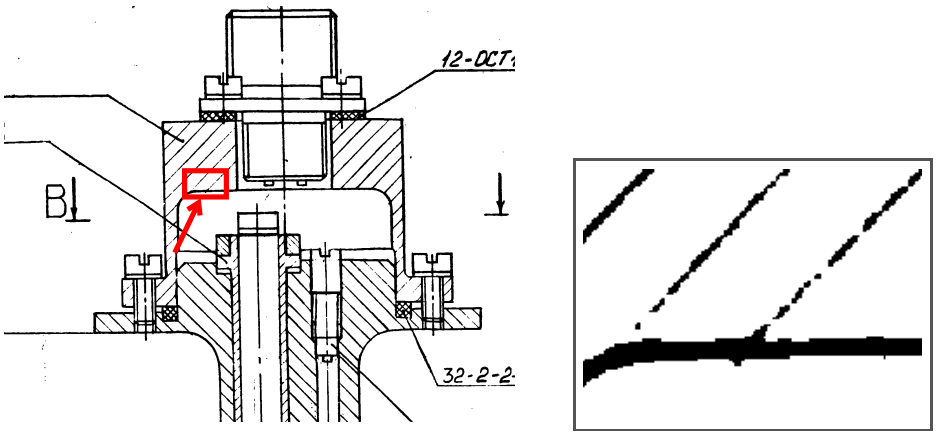
Fig. 3.45: Reconstruction géométrique et topologique d'une courbe régulière 4-connexe. La structure polygonale respecte le modèle de supercouverture (b). Pour plus de clarté, les nœuds du graphe de Reeb (c) qui sont représentés par deux lettres signifient qu'il y a deux types de points critiques au même endroit de l'objet. Par exemple, le nœud *bs* pourrait être représenté par une arête $b - s$ dans le graphe.

$f : (x, y) \rightarrow x^2 + y^2 + \cos(2\pi x) + \sin(2\pi y) + \sin(2\pi x^2)\cos(2\pi y^2) - 1$ dans le domaine $[-1.10; 1.10] \times [-1.10; 1.10]$ (issu de [Snyder, 1992b]). Nous donnons dans le chapitre 7 de plus amples explications sur le lien entre le modèle d'$\mathbb{I}$-grille et l'arithmétique d'intervalles. Nous y présentons également des expérimentations plus poussées pour montrer la robustesse et l'efficacité de nos algorithmes pour traiter ces problèmes de tracés.

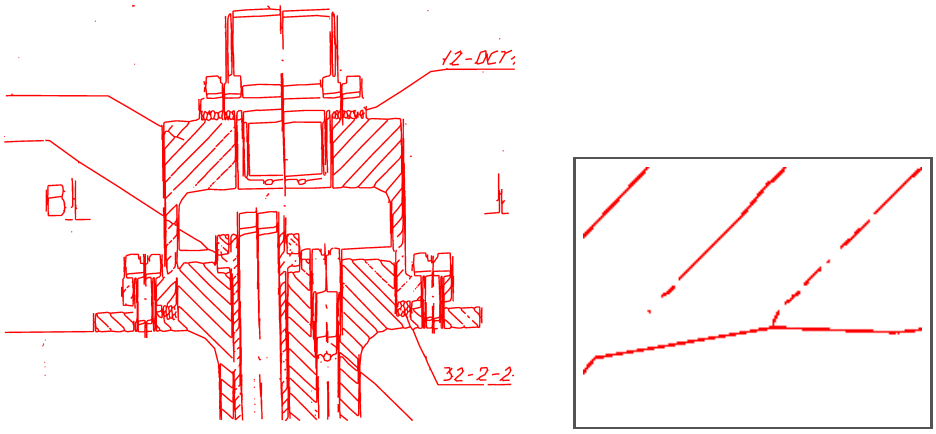
3.6 Bilan et perspectives

Nous nous sommes intéressés dans ce chapitre à la description géométrique (par des segments de droites) et topologique (grâce au graphe de Reeb) d'objets 2-D appartenant à une $\mathbb{I}$-grille. La reconstruction polygonale est exacte, et respecte le modèle de supercouverture étendu aux $\mathbb{I}$-grilles (ce qui implique la réversibilité). Nous avons aussi proposé deux algorithmes rapides pour traiter deux types de mises à jour possibles de ces objets : la raffinement et le groupement de cellules.

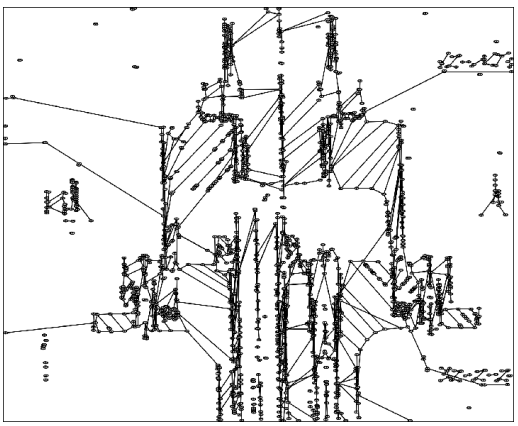
Dans ce chapitre, nous ne traitons pas de la qualité de la reconstruction polygonale. En effet, cette évaluation dépend grandement de l'application finale de notre algorithme. Dans le chapitre 7, nous proposons de comparer notre structure polygonale avec des algorithmes classiques de tracé de courbes implicites. Nous montrons que notre approche permet une approximation de qualité satisfaisante, avec un nombre de segments tracés sensiblement inférieur aux autres techniques. Nous nous sommes intéressés dans le chapitre 6 à la description de caractères dans une application de reconnaissance de plaques minéralogiques. Grâce à des expérimentations à grande échelle, nous avons montré que notre algorithme permet de décrire de manière pertinente des caractères ambigus (*e.g.* '8' et 'B'), et donc de les distinguer.



(a)

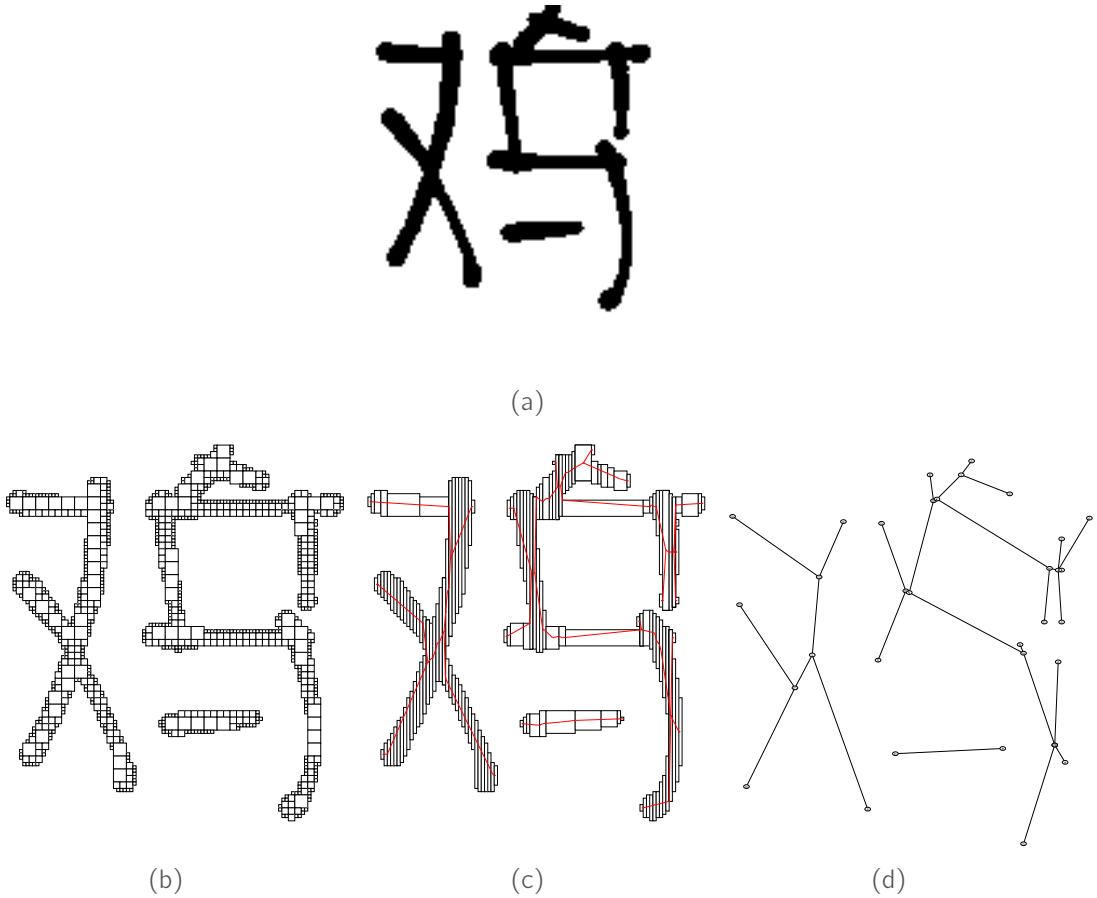


(b)



(c)

Fig. 3.46: Reconstruction d'une image de dessin technique de taille 1765 x 1437 pixels, dont une partie est zoomée, indiquée par la flèche (a). La polygonalisation que nous obtenons et le zoom associé sont présentés (b). Le graphe de Reeb complet (environ 300 nœuds) est également illustré (c).



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

Fig. 3.47: Reconstruction géométrique et topologique d'un objet obtenu par la décomposition quadtree d'une image binaire. En (c) sont représentés les k-arcs recodant l'objet traité et en (d) le graphe de Reeb associé.

La représentation topologique d'objets irréguliers isothétiques en 3-D nécessite que l'on construise le graphe de Reeb associé. Dans le cas régulier classique (voir un exemple dans [Xiao *et al.*, 2003]), cette opération est généralement effectuée en calculant tout d'abord un maillage triangulaire associé à l'objet, par l'algorithme de *marching cubes* [Lorensen et Cline, 1987]. En effet, le graphe de Reeb d'un tel maillage peut être calculé rapidement [Tierny *et al.*, 2006, Pascucci *et al.*, 2007]. Notre principal objectif serait d'éviter cette phase de triangulation pour construire le graphe, et de traiter directement l'ensemble de cellules 3-D. Nous pourrions nous inspirer des travaux sur les fonctions de Morse discrètes de R. Forman [Boissonnat *et al.*, 2003, Forman, 1998]. Une fois le graphe de Reeb construit, la reconstruction polyédrale d'objets 3-D pourrait être réalisée en considérant la structure du graphe, comme nous l'avons proposé ici. Cela reviendrait alors à polyédriser des k-arcs 3-D. Dans notre cadre, cette étape élémentaire a été prise en charge par une approche de cône de visibilité, mais on pourrait également utiliser la préimage généralisée décrite par M. Dexet [Dexet et Andres, 2006]. Une telle approche posséderait l'avantage d'être aisément extensible en 3-D.

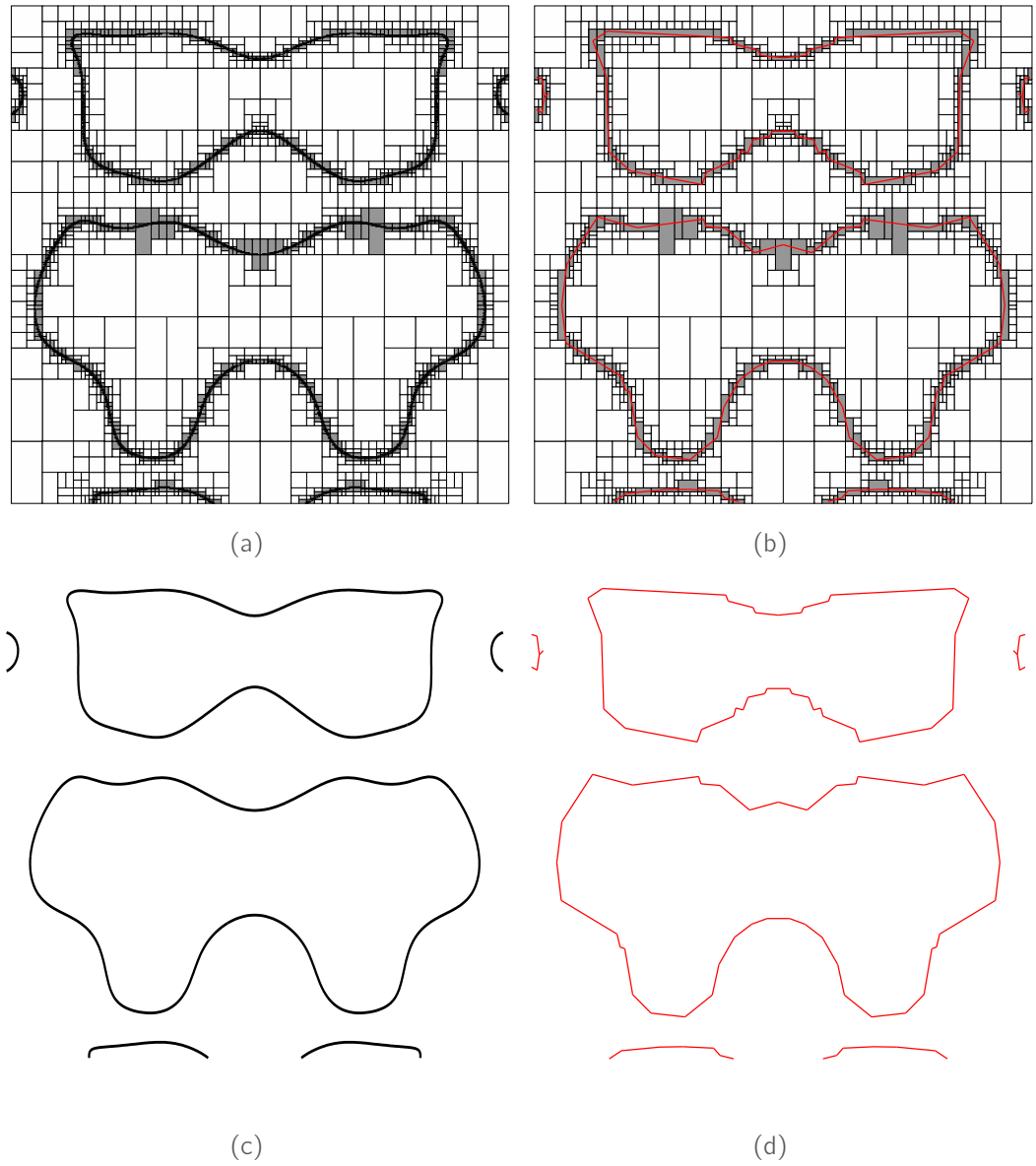


Fig. 3.48: Approximation polygonale d'une courbe implicite grâce à notre algorithme. Une analyse en intervalles permet d'encadrer la courbe avec des cellules 2-D (a). En utilisant l'algorithme 3.7, nous proposons une structure polygonale qui approxime la courbe initiale (b). Les images (c) et (d) sont tracées sans les cellules, pour plus de clarté.

CHAPITRE 4

Algorithmes de transformée en distance sur grilles irrégulières

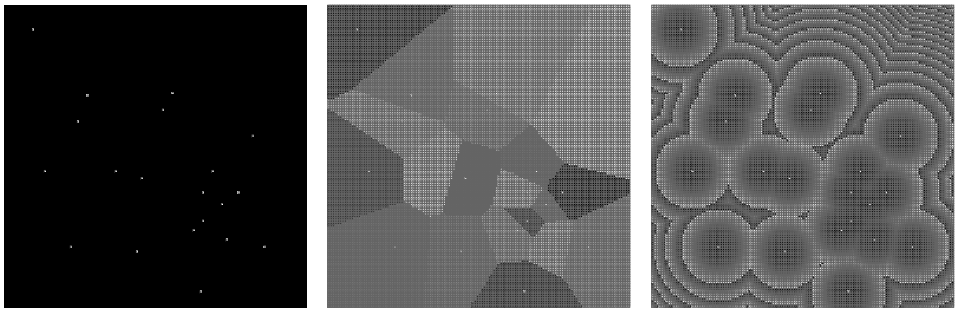
4.1 Introduction

4.1.1 Distance discrète, transformée en distance et diagramme de Voronoï dans le cas régulier

La définition de distance discrète est un concept très important en analyse d'images et en description de formes [Rosenfeld et Pfaltz, 1966, Rosenfeld et Pfaltz, 1968, Coeurjolly *et al.*, 2007]. La transformée en distance permet d'étiqueter chaque point d'un objet discret avec sa distance au complémentaire. De cette information on calcule généralement un squelette centré (ou *axe médian*) qui représente la forme globale de $\mathcal{E}$, avec un nombre minimal de points. Pour calculer la transformée en distance classique, on peut se baser sur la définition de distances approchées de chanfrein [Rosenfeld et Pfaltz, 1968, Remy et Thiel, 2000, Fouard et Malandain, 2005] ou des séquences de chanfrein [Rosenfeld et Pfaltz, 1966, Nagy, 2005]. Ces méthodes ont l'avantage d'être efficaces en terme de complexité et de proposer une représentation correcte de la carte de distance. Néanmoins, des approches optimales (linéaires en la taille d'une image) permettent aujourd'hui de proposer une carte de distances exacte (par exemple [Hirata, 1996, Maurer *et al.*, 2003]). Nous nous intéressons dans cette thèse à ces dernières méthodologies, permettant de calculer la distance euclidienne entre un point et la frontière de l'objet qui le contient.

La transformée en distance euclidienne au carré (E^2DT pour *squared euclidean distance transform*) d'une image binaire est un outil qui a été largement étudié pendant les dernières décennies. Elle représente un moyen d'analyser les formes d'objets graphiques, dans de nombreuses applications (voir [Paglieroni, 1992] et les références de [Maurer *et al.*, 2003] pour plus de détails). L'objectif de ce processus est d'étiqueter chaque cellule de *premier plan* d'un objet avec la distance au carré de la plus proche cellule du complémentaire

(cellule de *fond*). Cette opération peut être naturellement liée au calcul du diagramme de Voronoï (VD pour *Voronoi diagram*) [Preparata et Shamos, 1985, de Berg *et al.*, 2000, Coeurjolly et Montanvert, 2007]. Le VD d'un ensemble de points $\mathcal{P} = \{p_i\}$ (ou *sites*) est une partition de l'espace en cellules qui respecte la propriété suivante [Voronoi, 1908] : tout point q de l'espace appartient à la cellule $\mathcal{C}_{p_i}$ associée au site p_i si et seulement si $d_e(q, p_i) < d_e(q, p_j)$ pour tous les sites p_j tels que $j \neq i$. Clairement, si nous considérons chaque cellule du fond comme un site de Voronoï, déterminer la plus proche cellule du fond d'une cellule R revient à chercher le site de Voronoï p_i tel que $R \in \mathcal{C}_{p_i}$. Ensuite, on calcule la distance au carré entre R et p_i pour étiqueter R. De plus, on peut noter que le VD peut être extrait de la carte de distance issue de l'E^{2}DT [Saito et Toriwaki, 1994, Coeurjolly, 2002a, Hesselink *et al.*, 2005]. Dans le cadre discret régulier, le principe de la plupart des approches est de construire un VD *discret*, et d'en extraire la E^{2}DT. Dans la figure 4.49, nous montrons un exemple de calcul du VD discret régulier associé à un ensemble de sites dans le plan. Chaque cellule de Voronoï est représentée par une couleur différente. La carte de distance de cette configuration y est également présentée. Dans



(a) (b) (c)

Fig. 4.49: Transformée en distance, diagramme de Voronoï dans le cas régulier classique. En (a) est donné un ensemble de sites aléatoires dans $\mathbb{Z}^2$. On peut calculer le VD discret associé à ces sites (b) et donc la E^{2}DT (c), où chaque point de $\mathbb{Z}^2$ a la valeur de la distance au carré au plus proche site (réduite par un modulo pour plus de clarté).

ce chapitre, nous présentons des algorithmes pour calculer la E^{2}DT sur $\mathbb{I}$-grilles. Grâce à la définition de distances sur ces grilles, nous montrons également le lien entre le VD et la E^{2}DT sur $\mathbb{I}$-grille.

4.1.2 État de l'art des algorithmes de transformée en distance sur grilles non-régulières

La E^{2}DT d'une image binaire `I` considère généralement `I` comme une grille régulière. De nombreux travaux ont permis de développer des transformées en distance (DT pour *distance transform*) pour des grilles non-régulières, mais ces approches ne peuvent être étendues à toutes les $\mathbb{I}$-grilles. Les DT calculées sur un quadtree ou

un octree [Jung et Gupta, 1996, Samet, 1983, Samet, 1990b, Vörös, 2001] sont dépendantes de la structure spécifique traitée. Ces approches calculent la DT en propageant les valeurs des distances des nœuds parents aux nœuds fils. Pour traiter les images dans un cadre d'applications médicales, généralement discrétisées sur une grille anisotrope, de nombreuses méthodologies manipulant la *distance de chanfrein* ont été adaptées [Chehadeh *et al.*, 1996, Cuisenaire, 1999, Fouard et Malandain, 2005, Sintorn et Borgefors, 2004]. Pour adapter ces techniques sur une $\mathbb{I}$-grille quelconque $\mathbb{I}$, il faudrait étendre la définition des masques de chanfrein et les modifier pour chaque cellule de $\mathbb{I}$ (calcul non-stationnaire de la DT). De la même manière, les algorithmes développés pour les autres grilles non-standards [Wang et Bertrand, 1992, Strand et Borgefors, 2005, Fouard *et al.*, 2007], *e.g.* les grilles FCC et BCC (pour *face-centered cubic* et *body-centered cubic* respectivement), supposent la régularité du voisinage d'une cellule de la grille, et ne peuvent pas être aisément adaptées à toutes les $\mathbb{I}$-grilles. En effet, nous ne faisons aucune hypothèse sur la configuration des voisins d'une cellule de $\mathbb{I}$ (nombre, position, taille, *etc.*) dans la définition 2.1 d'une $\mathbb{I}$-grille. Une approche qui permet d'approximer la DT sur un échantillon de points non structurés est la FMM (ou *fast marching method*) [Sethian, 1999, Sethian, 2001]. Le principe est de déterminer par un processus itératif une ligne de front à partir de l'ensemble des points en entrée, qui suit une fonction de vitesse donnée. Plus précisément, cette méthode permet de résoudre l'équation eikonale

$$|\nabla T| F(x, y, z) = 1, \ T = 0 \ sur \ \Gamma, \tag{4.26}$$

où Γ représente l'ensemble des points de départ, et F la fonction de vitesse (ici définie sur $\mathbb{R}^3$). Cette équation trouve de nombreuses applications : planification de trajectoires, segmentation d'images, *etc.* (voir [Sethian, 2001] pour plus de détails sur les FMM). Les FMM ont, dans la version originale, une complexité en $\mathcal{O}(n \log n)$, où n est le nombre de points en entrée. Dans certaines configurations (*e.g.* les points sont définis sur une courbe paramétrée), calculer une carte de distance grâce aux FMM peut être réalisé en temps linéaire [Hassouna et Farag, 2007, Bronstein *et al.*, 2007]. Bien que, dans cette thèse, nous cherchons à proposer un modèle global qui puisse être appliqué à n'importe quel type de $\mathbb{I}$-grille et qui calcule la *DT exacte* associée, adapter les FMM sur $\mathbb{I}$-grille pourrait être une approche intéressante pour calculer rapidement la DT (voir figure 4.50 pour une illustration de cette méthodologie).

Dans le cas discret régulier, la plupart des algorithmes basés sur le VD [Saito et Toriwaki, 1994, Breu *et al.*, 1995, Maurer *et al.*, 2003, Schouten et van den Broek, 2004] construisent un VD partiel pour calculer la E^2DT et y parviennent avec un temps optimal linéaire en le nombre de cellules de la grille (soit en $\mathcal{O}(n \times n)$ pour une image carrée de côté n). Comme certaines de ces méthodes sont séparables, *i.e.* elles effectuent des opérations indépendamment le long des deux axes de l'image 2-D, elles sont naturellement extensibles pour traiter des images d-dimensionnelles et les grilles anisotropes. Ici, nous proposons d'étendre trois de ces algorithmes pour calculer la E^2DT sur $\mathbb{I}$-grille (ou $\mathbb{I}$-DT). Après une comparaison de leur complexité et de

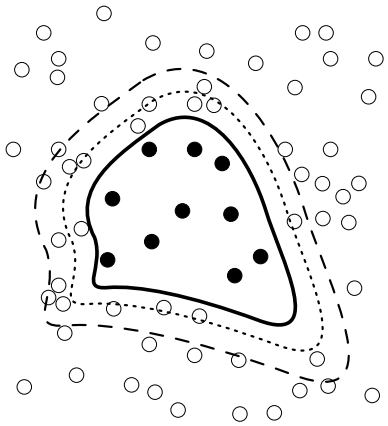


Fig. 4.50: Propagation de la ligne de front dans une approche FMM sur un échantillonnage de points irréguliers. L'ensemble des points en entrée Γ est décrit par la courbe pleine. L'initialisation est alors $T = 0$. Deux itérations de la progression de la ligne de front, $T = 1$ et $T = 2$, sont montrées en pointillés et en tirets respectivement.

leur vitesse d'exécution sur des expérimentations variées, nous dressons un bilan sur le comportement de chaque algorithme en fonction des grilles traitées.

4.2 Notion de distance sur $\mathbb{I}$-grilles et algorithmes de calcul de la $\mathbb{I}$-DT en 2-D

Nous donnons ici quelques notions supplémentaires sur les distances entre cellules. Dans notre cadre, nous considérons des $\mathbb{I}$-grilles *étiquetées* , *i.e.* chaque cellule d'une telle grille $\mathbb{I}$ a une valeur « fond » ou « premier plan » (respectivement 0 ou 1 par exemple). Dans ce chapitre, on note par $\mathbb{I}_B$ les cellules de fond, et $\mathbb{I}_F$ celles de premier plan. La taille de ces ensembles est notée n_B et n_F respectivement. On peut tout d'abord considérer que la distance entre deux cellules R et R' de $\mathbb{I}$ est la distance entre leur centre. Si l'on note p et p' les centres associés à $R = (x_R, y_R)$ et $R' = (x_{R'}, y_{R'})$, on a alors

$$d_e^2(R, R') = d_e^2(p, p') = (x_R - x_{R'})^2 + (y_R - y_{R'})^2 \tag{4.27}$$

pour la distance euclidienne au carré d_e^2. Pour une cellule $R \in \mathbb{I}$, la $\mathbb{I}$-CDT (pour *center-based DT*) est définie par :

$$\mathbb{I}-\mathrm{CDT}(R) \;=\; \min_{R'} \left\{ d_e^2(R, R');\ R' \in \mathbb{I}_B \right\} \tag{4.28}$$

ce qui est exactement la $\mathrm{E}^2\mathrm{DT}$ si l'on considère $\mathbb{I}$ comme une grille régulière. Cette distance discrète irrégulière calculée entre les centres de cellules peut être appliquée dans les systèmes d'information géographique (SIG), ou une structure spatiale hiérarchique permet de localiser rapidement les points et de calculer la distance entre

eux [Bentley, 1975, Samet, 1990a] (voir également partie I, chapitre 1 pour plus de détails). La $\mathbb{I}$-CDT écrite ainsi peut être liée, comme dans le cas régulier, au calcul du diagramme de Voronoï des centres de cellules de fond (voir plus loin).

On peut également étendre la distance proposée par H. Samet [Samet, 1983] (utilisée à l'origine dans un cadre de transformée en distance *chessboard* en référence à la grille régulière sous-jacente). Dans ce cas, on doit calculer la distance entre le centre p d'une cellule de premier plan R et la plus proche frontière entre l'objet contenant R et le fond. Pour décrire cette frontière, nous nous basons sur le fait que l'intersection entre deux cellules R et R' d'une $\mathbb{I}$-grille est un *lignel* [Herman, 1998]. Comme dans le cas discret classique, bien que la notion de pavage implique que $R \cap R' = \emptyset$, nous pouvons considérer l'élément de dimension 1 entre R et R'. Nous parlerons par la suite de segment entre R et R' pour plus de clarté. Soit $\mathcal{S}$ l'ensemble des segments tels que l'intersection entre une cellule $R \in \mathbb{I}_F$ et une cellule $R' \in \mathbb{I}_B$ adjacente à R appartienne à $\mathcal{S}$. Alors, la $\mathbb{I}$-BDT (pour *border-based DT*) est définie de la manière suivante :

$$\mathbb{I}\text{-}\mathrm{BDT}(R) = \min_{\alpha} \left\{ d_e^2(p, \alpha);\ \alpha \in \mathcal{S} \right\}. \tag{4.29}$$

Contrairement à la $\mathbb{I}$-CDT donnée dans l'équation 4.28, ce processus ne prend pas en compte la représentation du fond de la grille. Plus précisément, on ne considère pas les centres des cellules de fond de $\mathbb{I}$. Comme dans le cas discret classique, ces deux versions de la $\mathbb{I}$-DT peuvent être liées à un calcul de VD. Lorsque l'on considère les centres de cellules, la $\mathbb{I}$-CDT peut être calculée en considérant un VD (partiel ou non) des centres de cellules de fond. A l'inverse, si l'on prend en compte les frontières fond / premier plan, notre définition impose de traiter les cellules de fond par leurs bords. Cela implique que l'on calcule un VD de *segments*, et non un VD de sites classique (voir la figure 4.51 pour un exemple de ces deux diagrammes appliqués sur une $\mathbb{I}$-grille simple).

4.2.1 Approche basée sur le calcul du diagramme de Voronoï complet en 2-D

Nous proposons maintenant un premier algorithme basé sur le VD où les sites sont les centres des cellules de fond dans $\mathbb{I}_B$. Puis, on calcule la $\mathbb{I}$-CDT de chaque cellule R de $\mathbb{I}_F$ en localisant son centre p dans le VD. On cherche s, l'un des sites de Voronoï les plus proches de p, pour finalement calculer la $\mathbb{I}$-CDT de R (voir l'algorithme 4.10 pour plus de détails). On peut remarquer que dans cette approche nous n'exploitons pas la notion d'ordre sur $\mathbb{I}$-grille, ni la géométrie des cellules. Dans la figure 4.52 sont présentés les résultats de cet algorithme sur des grilles construites à partir d'une image binaire simple.

Le VD $\mathcal{V}$ est créé grâce à la librairie `CGAL` [CGAL, 2008, Karavelas, 2004, Fabri, 2007]. Cette construction est réalisée en temps optimal $\mathcal{O}(n_B \log n_B)$, et requiert $\mathcal{O}(n_B)$ en espace [Devillers, 1998]. Puis, la boucle principale consiste à localiser la cellule de premier plan R dans $\mathcal{V}$ (avec son centre p), et à chercher un des plus proches sites de Voronoï

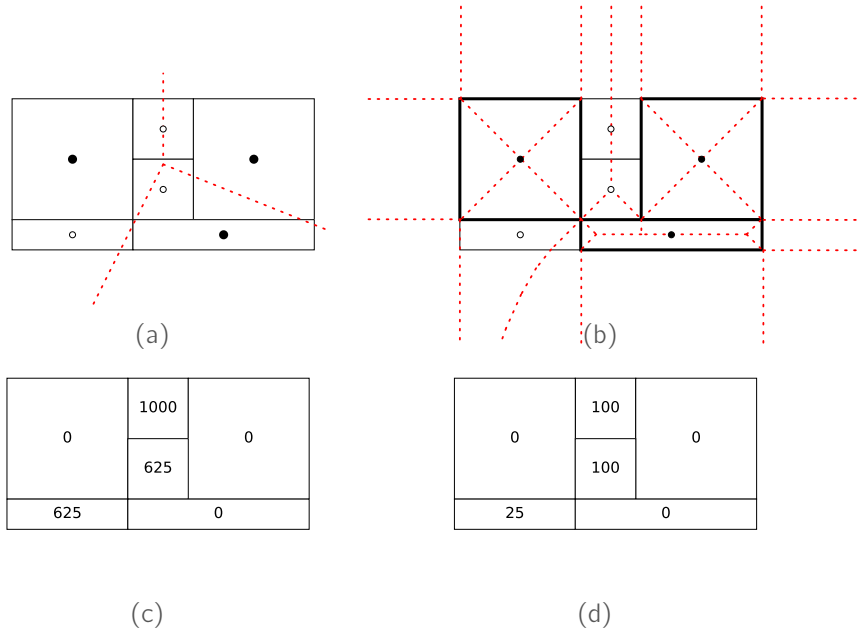


Fig. 4.51: Illustration des diagrammes de Voronoï de sites et de segments et leurs liens avec la $\mathbb{I}$-DT. A partir d'une $\mathbb{I}$-grille étiquetée simple, la transformée en distance diffère si l'on considère les centres de cellules (a) ou la frontière premier plan / fond (b). Dans ce dernier cas, on remarquera que les arêtes à l'intérieur des cellules de fond peuvent être ignorées, car elles ne contiennent aucun autre centre de cellule. En (c) et (d) sont exposés respectivement les résultats de l'$\mathbb{I}$-CDT et de l'$\mathbb{I}$-BDT de la grille.

(noté s) dans $\mathcal{V}$. La requête de localisation est en $\mathcal{O}(\log n_B)$ [Devillers, 1998]. Par conséquent, on réalise cette boucle en $\mathcal{O}(n_F \log n_B)$. En fait, chercher le site le plus proche est en temps constant dans les trois cas étudiés : p se situe dans une cellule de Voronoï, sur une arête de Voronoï, ou sur un sommet de Voronoï. Quand p est sur un sommet ou une arête, le choix de la cellule de Voronoï contenant s est arbitraire. Ainsi, l'algorithme 4.10 est exécuté en $\mathcal{O}(n \log n_B)$, avec une complexité en espace en $\mathcal{O}(n)$, où $n = n_B + n_F$ est le nombre total de cellules dans $\mathbb{I}$.

On peut également remarquer que notre démarche est applicable pour calculer la $\mathbb{I}$-BDT basée sur la frontière. En utilisant les mêmes outils [Fabri, 2007], il est possible de représenter efficacement le diagramme de Voronoï des segments correspondant aux bords des cellules de fond. Dans ce cas, cette construction est réalisable en $\mathcal{O}(n_S \log^2 n_S)$, où $n_S = 4n_B$ est le nombre total de segments. L'augmentation de la complexité s'explique par le fait que l'on doit gérer l'intersection entre segments [1]. Ainsi, la complexité globale de l'algorithme 4.10 devient $\mathcal{O}(n \log^2 n_B)$ car la requête de recherche du plus proche site de Voronoï peut être optimisée en $\mathcal{O}(\log^2 n_B)$ [Karavelas, 2004].

Étendre cette méthode aux $\mathbb{I}$-grilles d-dimensionnelles (ou d-D) est une tâche diffi-

[1] Dans notre cadre, nous n'avons que des intersections aux extrémités des segments, dites *faibles* [Karavelas, 2004].

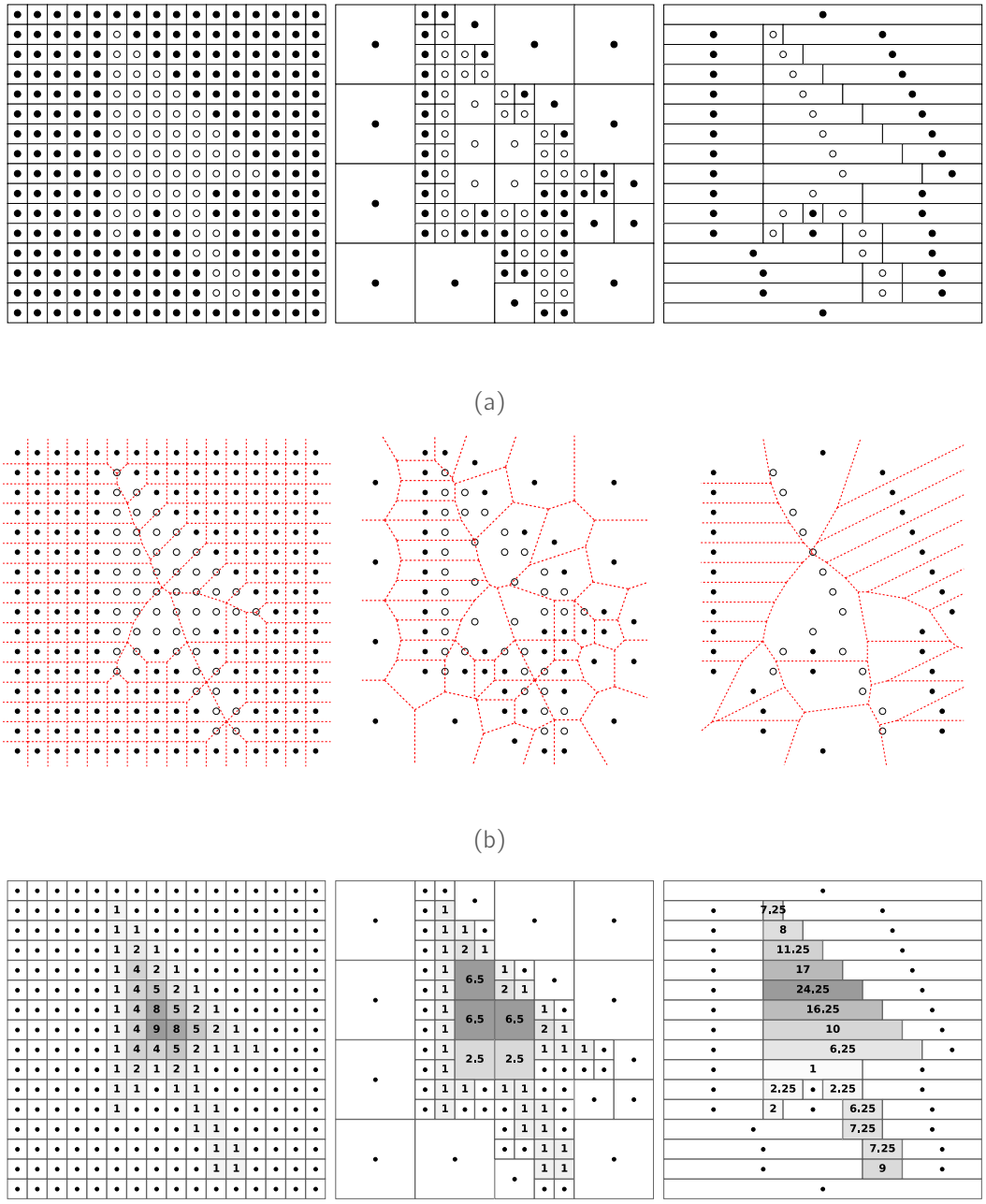


Fig. 4.52: Exemples de résultats de $\mathbb{I}$-CDT par l'algorithme 4.10 sur une image simple de taille 16 x 16. L'image *cursor* est discrétisée sur une grille régulière, par une décomposition en quadtree, et par un encodage RLE suivant l'axe des X (a). Les centres de cellules de fond sont tracés en noir, et ceux de premier plan en blanc. Le VD de chaque cas est illustré en pointillés (b) et la carte de distance pour la distance d_e^2 est donnée en (c).

Algorithme 4.10 : $\mathbb{I}$-CDT basée sur le VD complet

entrée : une $\mathbb{I}$-grille étiquetée $\mathbb{I}$.
sortie : la $\mathbb{I}$-CDT de $\mathbb{I}$.

```
début
    calculer le VD V des points {p ; R ∈ I_B};
    pour chaque cellule R ∈ I_F faire
        localiser p dans V;
        si p est confondu à un sommet de Voronoï v dans V alors
            s ← site de Voronoï d'une cellule adjacente de v;
        sinon si p appartient à une arête de Voronoï e dans V alors
            s ← site de Voronoï d'une cellule adjacente à e;
        sinon
            // p appartient à une cellule de Voronoï c dans V
            s ← site de Voronoi de c;
        I-CDT(R) ← d²_ê(s, p);
    pour chaque cellule R ∈ I_B faire
        I-CDT(R) ← 0;
fin
```

cile. Il faut remarquer tout d'abord que le VD est calculable en d-D grâce à l'enveloppe convexe de dimension $d+1$, ou par le *gift wrapping* [de Berg *et al.*, 2000]. Quelle que soit la technique choisie, l'espace requis est alors au pire cas en $\mathcal{O}(n_B^{\lceil d/2 \rceil})$ et le temps optimal pour le construire est en $\mathcal{O}(n_B \log n_B + n_B^{\lceil d/2 \rceil})$. Malheureusement, ces méthodes ne proposent pas une structure de localisation associée au VD construit et ne permettent donc pas d'adapter le synopsis de l'algorithme 4.10 au cas d-D. Pour obtenir une structure permettant de localiser un point dans le VD, on peut choisir par exemple de l'approximer [Arya *et al.*, 2002]. Dans le cadre du calcul de la $\mathbb{I}$-CDT sur un ensemble de points dans l'espace d-D, une approche qui pourrait être développée serait de se baser sur une structure de grille de subdivision, comme dans [Park *et al.*, 2005].

On peut conclure de cette section que l'approche par la construction du VD complet que nous proposons souffre de plusieurs inconvénients majeurs :

- ▷ L'extension $\mathbb{I}$-BDT, où la distance sur $\mathbb{I}$-grilles est basée sur la frontière, requiert plus d'espace et de temps de calcul.
- ▷ Elle ne permet pas de construire la $\mathbb{I}$-CDT pour les $\mathbb{I}$-grilles d-dimensionnelles.
- ▷ Puisque l'on utilise le VD complet pour chercher le site le plus proche d'un point p, une telle méthodologie est inefficace dans le cadre d'une grille très dense. Dans ce cas, une structure de données plus adéquate à l'architecture des $\mathbb{I}$-grilles que l'on manipule serait nécessaire.

Dans les sections suivantes, nous proposons de résoudre ces problématiques grâce à trois algorithmes inspirés de la littérature.

4.2.2 L'extension de l'algorithme de [Breu et al., 1995] pour une $\mathbb{I}$-CDT linéaire en 2-D

Le principe de cette technique est de construire une structure de données basée sur des listes en balayant l'image binaire traitée. Grâce à un prédicat simple, il est possible de traiter l'ensemble des points de $\mathbb{Z}^2$ en entrée en temps linéaire, et de construire le VD discret pour obtenir la E^2DT finale. Nous présentons ici l'adaptation de cette approche pour calculer la $\mathbb{I}$-CDT, qui traite les centres des cellules d'une $\mathbb{I}$-grille appartenant à $\mathbb{R}^2$. Rappelons que l'on note par p le centre de chaque cellule $R \in \mathbb{I}$. Nous considérons ici l'ordre des cellules d'une $\mathbb{I}$-grille pour les parcourir, par une relation d'ordre sur les centres des cellules. Ainsi, nous décrivons une approche de type *sweep line* [de Berg *et al.*, 2000] très commune en géométrie algorithmique. Rappelons que l'approche par VD complet que nous avons décrite dans la section précédente n'exploite pas cette information pour calculer la $\mathbb{I}$-CDT.

À chaque étape, on considère les points de coordonnées inférieures ou égales à $r \in \mathbb{R}$. Pour respecter les notations données dans [Breu *et al.*, 1995], on note $\mathcal{P}_r$ l'ensemble de ces points, et $\mathcal{V}_{\mathcal{P}_r}$ le VD discret associé. Un ensemble de listes $\mathcal{L}_r$ est construit pendant le balayage, et chacune d'entre elles correspond aux points de fond (ou sites) dont la cellule de Voronoï dans $\mathcal{V}_{\mathcal{P}_r}$ intersecte la droite d'équation $y = r \in \mathbb{R}$. Grâce à la notion d'ordre total $\preceq_y$ sur $\mathbb{I}$-grille que nous avons donnée dans la définition 2.2 de la partie I, nous sommes capables de parcourir la $\mathbb{I}$-grille $\mathbb{I}$ en considérant successivement les cellules dont l'ordonnée du centre est alignée avec la droite d'équation $y = r \in \mathbb{R}$, avec r croissant (voir figure 4.53). Cela signifie que nous connaissons *a priori* les différentes ordonnées possibles des cellules de $\mathbb{I}$. Ainsi, nous notons par la suite $\{r_i\}_{i=1,n_l}$, $r_i \in \mathbb{R}$, $i = 1, \ldots, n_l$ cet ensemble de n_l valeurs.

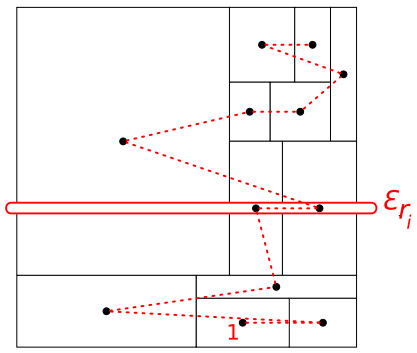


Fig. 4.53: Parcours des cellules d'une $\mathbb{I}$-grille grâce à l'ordre $\preceq_y$. Les cellules dont le centre a pour ordonnée r_i sont notées par la suite $\mathcal{E}_{r_i}$. On peut remarquer que le balayage de la grille dans l'autre sens suit le même chemin, à l'envers (la cellule « 1 » devient la dernière cellule parcourue).

Le but de l'algorithme que nous adaptons de [Breu *et al.*, 1995] est de déduire $\mathcal{L}_{r_i}$ à partir de $\mathcal{L}_{r_{i-1}}$. Pour cela, les auteurs de ces travaux se basent tout d'abord sur deux observations, que nous pouvons aisément étendre à notre étude :

- ▷ Soient $u \in \mathcal{L}_{r_i}$ et $v \in \mathcal{L}_{r_{i-1}}$ deux sites tels que $u_x = v_x$ (et donc $u_y > v_y$). Alors la cellule de Voronoï centrée en v n'intersecte pas la droite $y = r_i$. Ce qui signifie qu'il ne peut y avoir qu'un seul site par abscisse dans la liste.
- ▷ Si les sites de $\mathcal{P}_{r_i}$ sont triés selon leur abscisse, alors les cellules en intersection avec la droite $y = r_i$ le seront également.

Ensuite, nous définissons le prédicat `caché_par()`, qui permet de supprimer les sites de $\mathcal{L}_{r_{i-1}}$ pour obtenir $\mathcal{L}_{r_i}$. Ce prédicat prend en entrée trois points $u, v, w \in \mathbb{R}^2$, et indique que v est *caché par* u et w le long de la droite $y = r_i$ si la cellule de Voronoï associée à v n'intersecte pas cette droite, alors que c'est le cas pour u et w (voir la figure 4.54 pour un exemple).

Algorithme 4.11 : Prédicat `caché_par()` réel

entrée : les points u, v et w de $\mathbb{R}^2$ et la droite d'équation $y = r \in \mathbb{R}$.
sortie : v est-il caché par u et w sur la droite $y = r$?

début
  retourner $(w_x - v_x) \times (v_x^2 - u_x^2 - 2r(v_y - u_y) + v_y^2 - u_y^2) \geq (v_x - u_x) \times (w_x^2 - v_x^2 - 2r(w_y - v_y) + w_y^2 - v_y^2)$
fin

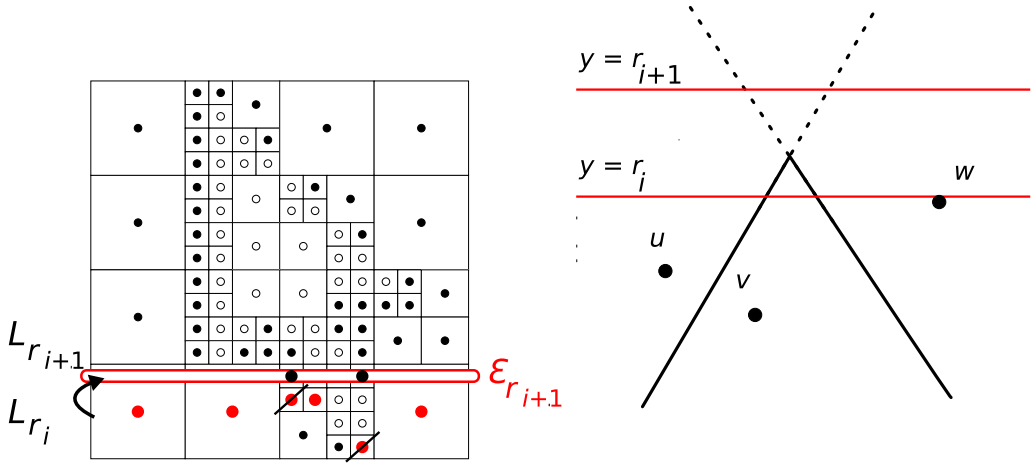
A partir de ces éléments, l'algorithme consiste principalement à supprimer d'une liste de sites *candidats* $\mathcal{C}_{r_i}$, d'ordonnées maximales $y \leq r_i$ triées par abscisse croissante, en considérant la proposition suivante :

Proposition 4.4 (Preuve dans [Coeurjolly, 2002a])**.** *Etant donné un site v d'un ensemble de sites $\mathcal{P}$ et une droite l, v n'appartient pas à la liste $\mathcal{L}_l$ si et seulement si il existe au moins une paire de sites (u, w) de $\mathcal{P}\backslash\{v\}$ tels que u et w cachent v le long de l.*

En respectant ce même ordre, pour déterminer la $\mathbb{I}$-CDT de chaque point p de premier plan, il suffit de vérifier quel est le site le plus proche, en considérant seulement l'abscisse de p (*i.e.* trouver dans quelle cellule du VD discret se trouve p). Nous avons décrit de manière plus détaillée le déroulement de notre algorithme ci-après. L'algorithme 4.13 se base sur un double balayage de l'ensemble des cellules de la $\mathbb{I}$-grille $\mathbb{I}$ en entrée. Ces balayages sont réalisés en considérant toutes les cellules R dont le centre a pour ordonnée r_i (ligne 1). Nous pouvons obtenir en temps linéaire ces cellules en utilisant notre structure de listes que nous avons décrite dans la partie I. Cette liste de cellules alignées est notée $\mathcal{E}_{r_i}$ (voir figure 4.54). De plus, grâce à l'ordre lexicographique sur $\mathbb{I}$, ces cellules sont triées dans l'ordre croissant des abscisses. Rappelons que nous connaissons également le nombre d'ordonnées différentes dans $\mathbb{I}$, noté n_l. Si une cellule $R \in \mathcal{E}_{r_i}$ est une cellule de fond, alors on met à jour la liste de candidats associée, ce qui revient à

$$\mathcal{C}_{r_i} = \mathcal{L}_{r_{i-1}} \cup \{p \text{ centre d'une cellule } R \in \mathbb{I}_B;\ y_R = r_i\}, i \geq 2, \quad (4.30)$$

$$= \mathcal{L}_{r_{i-1}} \cup \{R \in \mathcal{E}_{r_i} \wedge R \in \mathbb{I}_B;\ y_R = r_i\}, i \geq 2, \quad (4.31)$$



(a) (b)

Fig. 4.54: La mise à jour de la structure de listes pour calculer la $\mathbb{I}$-CDT en temps linéaire. A chaque ordonnée est associée une liste de sites ayant même hauteur (a). Elles permettent d'appliquer le prédicat `caché_par()` et de supprimer les sites inutiles en considérant deux listes successives. Par exemple, les points barrés seront supprimés dans $\mathcal{L}_{r_{i+1}}$ et seront remplacés par des points de même abscisse. Dans (b), nous avons illustré un cas plus général de suppression de sites avec le prédicat `caché_par()`. On a ici $\mathcal{L}_{r_i} = \{u, v, w\}$ et $\mathcal{L}_{r_{i+1}} = \{u, w\}$.

Algorithme 4.12 : Fonction `supprimer_sites()` de construction de $\mathcal{L}_{r_i}$ à partir de $\mathcal{C}_{r_i}$.

entrée : la liste de sites candidats $\mathcal{C}_{r_i}$.
sortie : la liste de sites valides $\mathcal{L}_{r_i}$.

début
  $\mathcal{L}_{r_i}[1] \leftarrow \mathcal{C}_{r_i}[1]$;
  $\mathcal{L}_{r_i}[2] \leftarrow \mathcal{C}_{r_i}[2]$;
  $k \leftarrow 2,\ l \leftarrow 3,\ c \leftarrow |\mathcal{C}_{r_i}|$;
  tant que $l \leq c$ **faire**
    $w = \mathcal{C}_{r_i}[l]$;
    tant que $k \geq 2 \wedge$ `caché_par`$(\mathcal{L}_{r_i}[k-1], \mathcal{L}_{r_i}[k], w, r)$ **faire**
      $k \leftarrow k - 1$;
    $k \leftarrow k + 1$;
    $l \leftarrow l + 1$;
    $\mathcal{L}_{r_i}[k] \leftarrow w$;
  retourner $\mathcal{L}_{r_i}$
fin

Algorithme 4.13 : $\mathbb{I}$-CDT linéaire inspirée de [Breu *et al.*, 1995]

entrée : une $\mathbb{I}$-grille étiquetée $\mathbb{I}$.
sortie : la $\mathbb{I}$-CDT de $\mathbb{I}$.

```
début
    // Première passe de bas en haut, suivant l'ordre ≼_y
    pour i = 1 à n_l faire
        F ← ∅;
        si i ≥ 2 alors
            C_{r_i} ← L_{r_{i-1}};
        sinon
            C_{r_i} ← ∅;
        // E_{r_i} est l'ensemble des cellules dont
        // le centre est aligné à y = r_i
1       pour chaque centre p d'une cellule R ∈ E_{r_i} faire
            si R ∈ I_B alors
                C_{r_i} ← C_{r_i} ∪ {p};
                I-CDT(R) ← 0;
            sinon
                F ← F ∪ {p};
                I-CDT(R) ← ∞;
        L_{r_i} ← supprimer_sites(C_{r_i});
        k ← 1, l ← |L_{r_i}|;
        pour chaque point p ∈ F faire
            d_1 ← d_e(p, L_{r_i}[k]);
            d_2 ← d_e(p, L_{r_i}[k + 1]);
            tant que d_1 ≥ d_2 ∧ k < l faire
                k ← k + 1;
            si d_1 < I-CDT(R) alors
2               I-CDT(R) ← d_1^2;
    // Deuxième passe de haut en bas, suivant l'ordre ≽_y
    pour i = n_l à 1 faire
        ⋮
fin
```

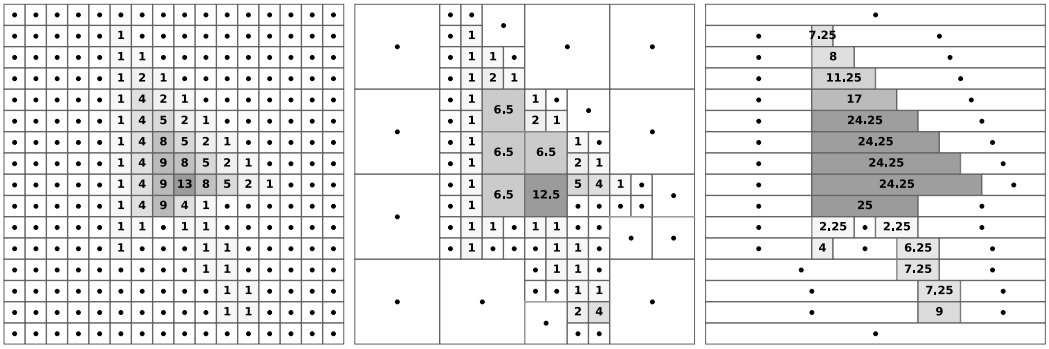
en notant que lorsque $i = 1$, le membre de gauche de l'union n'existe pas. Nous insistons sur le fait que l'invariant de notre algorithme est que $\mathcal{L}_{r_{i-1}}$ est triée suivant l'axe des X, et comme $\mathcal{E}_{r_i}$ l'est également (par l'ordre lexicographique induit par $\preceq_y$), le calcul de $\mathcal{C}_{r_i}$ est linéaire. De plus, cette liste de candidats est, elle aussi, triée suivant l'ordre croissant des abscisses. Si R est une cellule de premier plan, nous ajoutons son centre dans une

liste $\mathcal{F}$ qui représente les centres de cellules de premier plan de même ordonnée r_i. Nous initialisons également la valeur de la distance pour chaque cellule à 0 ou à l'infini, suivant sa valeur. Une fois ces deux listes construites, on utilise `supprimer_sites()` pour éliminer les sites candidats cachés par deux autres sites (proposition 4.4). Cette fonction, ainsi que le prédicat `caché_par()`, ne sont pas modifiés par rapport à la version régulière classique. Comme l'algorithme original de [Breu *et al.*, 1995], nous utilisons une structure de pile pour parcourir les sites de $\mathcal{C}_{r_i}$ et tester `caché_par()`. Rappelons que cette fonction est opérée en temps linéaire $\mathcal{O}(|\mathcal{C}_{r_i}|)$ en le nombre de points candidats. Une fois la liste $\mathcal{L}_{r_i}$ construite, nous parcourons l'ensemble des points de premier plan ($\mathcal{F}$) et nous assignons à chacun d'entre eux la distance au carré du point de fond le plus proche dans $\mathcal{L}_{r_i}$ (ligne 2). On peut noter que la boucle que nous venons de décrire implique un double parcours linéaire des centres de cellules de même ordonnée r_i, car elles sont triées par abscisse croissante (grâce à l'accès à l'ensemble $\mathcal{E}_{r_i}$). Une deuxième passe est également réalisée dans le sens inverse (fin de l'algorithme), similaire à la première. Dans ce cas, nous devons considérer un nouvel ordre sur $\mathbb{I}$ pour parcourir les cellules. Brièvement, on considère la relation d'ordre total $\succeq_y$ à la place de $\preceq_y$, qui est définie de manière analogue (un exemple est illustré dans la figure 4.53).

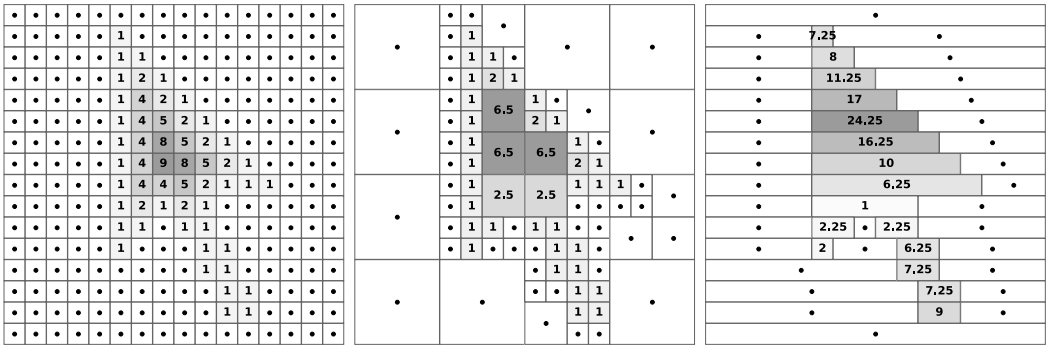
Par conséquent, si l'on note n le nombre total de cellules dans la $\mathbb{I}$-grille $\mathbb{I}$, l'algorithme 4.13 est exécuté en temps linéaire $\mathcal{O}(n)$. En effet, à chaque étape ($y = r_i$), nous construisons l'ensemble $\mathcal{E}_{r_i}$ en $\mathcal{O}(|\mathcal{E}_{r_i}|)$. Puis, comme il est trié suivant l'axe des X, les phases de construction de $\mathcal{C}_{r_i}$ et de $\mathcal{L}_{r_i}$ sont réalisées respectivement en $\mathcal{O}(|\mathcal{C}_{r_i}|)$ et $\mathcal{O}(|\mathcal{L}_{r_i}|)$, et la taille de ces ensembles est bornée par celle de $\mathcal{E}_{r_i}$. Grâce à cette propriété de tri des abscisses, l'affectation de la valeur de $\mathbb{I}$-CDT(R) pour chaque cellule $R \in \mathcal{E}_{r_i}$ de premier plan est réalisée en temps linéaire suivant le nombre de cellules de $\mathcal{E}_{r_i}$. Ainsi, pour chaque passe de l'algorithme, la complexité est clairement en $\mathcal{O}(\sum_{i=1,n_l} |\mathcal{E}_{r_i}|) = \mathcal{O}(n)$. Nous avons illustré dans la figure 4.55 le résultat de chacune de ces deux passes dans notre méthode sur les $\mathbb{I}$-grilles issues de l'image *cursor*.

L'approche que nous venons de décrire permet de construire le VD discret des centres de cellules de fond d'une $\mathbb{I}$-grille étiquetée. Cela implique que pour faire de même avec les frontières premier plan / fond (*i.e.* calculer $\mathbb{I}$-BDT), il faut considérer les bords de cellules de fond dans notre algorithme. On pourrait développer un algorithme qui se baserait sur une structure de listes triées de segments, réorganisées grâce à un prédicat similaire à `caché_par()`. Ainsi, calculer la $\mathbb{I}$-BDT basée sur la frontière d'une cellule R reviendrait à parcourir ces listes pour trouver le plus proche segment du centre de R.

Enfin, nous pourrions essayer d'optimiser notre technique par une approche par listes, comme les auteurs de [Guan et Ma, 1998] le proposent dans le cas régulier, en espérant que la complexité de l'algorithme soit réduite à $\mathcal{O}(\sqrt{n \times n_B})$. Il faut noter qu'une phase d'initialisation sera toujours nécessaire, avec une complexité en $\mathcal{O}(n)$. Par ailleurs, bien que notre approche soit linéaire en le nombre de cellules de la $\mathbb{I}$-grille, elle n'est pas directement extensible aux dimensions supérieures. Nous développerons dans la section suivante une méthode que les auteurs présentent comme une extension de [Breu *et al.*, 1995] à d dimensions.



(a)



(b)

Fig. 4.55: Calcul de la $\mathbb{I}$-CDT par l'algorithme 4.13 sur des $\mathbb{I}$-grilles simples issues de l'image *cursor*. En (a) nous avons représenté le résultat de la première passe de l'algorithme, et en (b) le résultat de la $\mathbb{I}$-CDT.

4.3 Algorithmes d-dimensionnels pour calculer la $\mathbb{I}$-DT

4.3.1 Principe d'un algorithme d-dimensionnel dans le cas régulier et son adaptation sur $\mathbb{I}$-grille

Nous rappelons tout d'abord le principe de l'algorithme décrit dans [Saito et Toriwaki, 1994] pour calculer la E^2DT d'une image binaire 2-D, aisément extensible aux dimensions supérieures. La méthode que nous présentons maintenant consiste à traiter séparément les deux axes de l'image. On parle ainsi d'une approche *séparable*. Plus précisément, les valeurs de distances sont d'abord calculées suivant les lignes de l'image ; puis ces valeurs sont utilisées pour calculer les distances minimales suivant les colonnes. Soit $B = \{b(i,j)\}_{i,j\in\{1,n\}}$ une image binaire de taille $n \times n$. Nous notons dans cette section $b(i,j) = 0$ pour un pixel de fond, et $b(i,j) = 1$ pour un pixel de premier plan. La première passe consiste à calculer l'image $G = \{g(i,j\}$ suivant

un processus de minimisation suivant l'axe des X :

$$g(i,j) = \min_{x} \{|i-x|;\ 1 \leq x \leq n \wedge (x,j) = 0\}\,. \tag{4.32}$$

Ensuite, on construit l'image $H = \{h(i,j)\}$ par un processus de minimisation suivant l'axe des Y :

$$h(i,j) = \min_{y} \left\{g(i,j)^2 + (j-y)^2;\ 1 \leq y \leq n\right\}. \tag{4.33}$$

Dans la figure 4.56, nous présentons un exemple de calcul de la E^{2}DT sur une image binaire de petite taille. Les différentes étapes de l'algorithme et leur implémentation sont

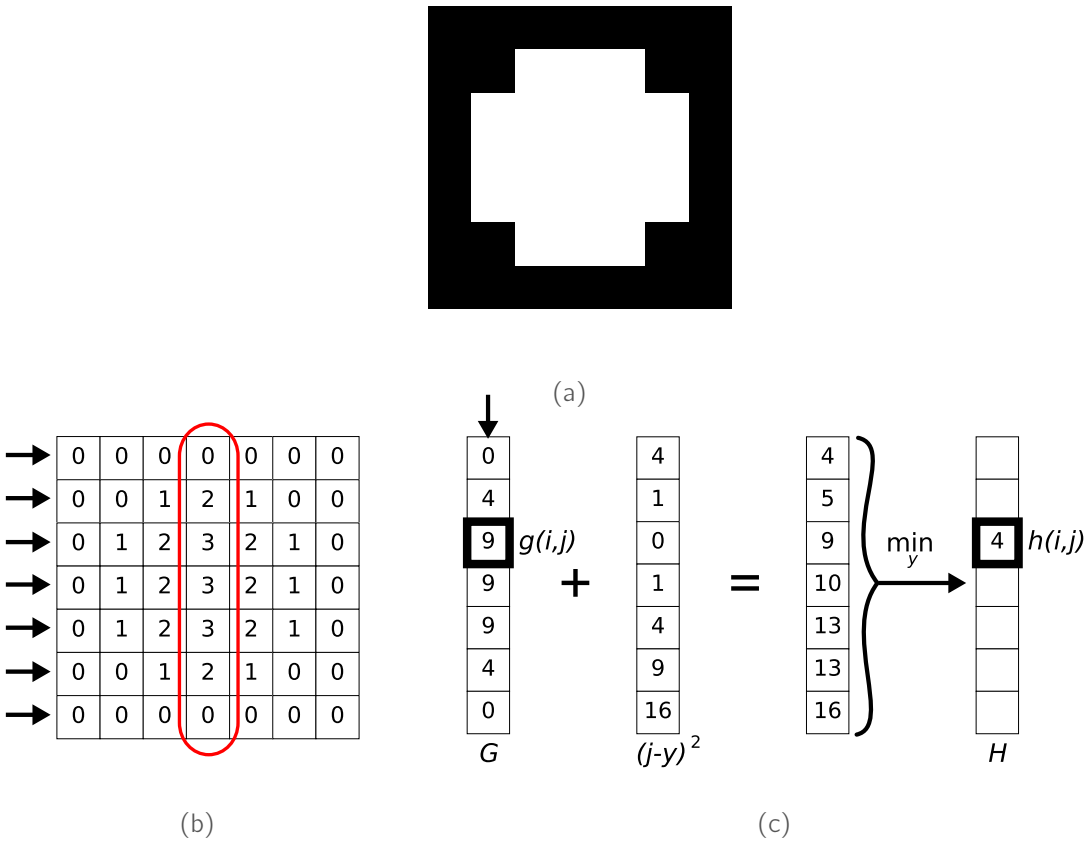


Fig. 4.56: Processus de double minimisation de [Saito et Toriwaki, 1994] sur une image binaire simple. À partir de l'image binaire B (a), la première passe de l'algorithme calcule G, qui représente les distances minimales suivant l'axe des X (b). Si l'on choisit une des colonnes de G, la seconde phase consiste à calculer $g(i,j)^2 + (j-y)^2$ pour chaque pixel de cette colonne. Nous avons détaillé ce calcul pour un pixel $g(i,j)$. Ensuite, la minimisation sur cette nouvelle colonne permet de calculer la E^{2}DT représentée en $h(i,j)$.

détaillés par la suite, lorsque nous l'adaptons sur $\mathbb{I}$-grilles.

Nous devons tout d'abord prendre soin de la structure de données employée pour étendre ce type d'approche sur $\mathbb{I}$-grilles. Une première idée pour adapter l'approche

de [Saito et Toriwaki, 1994] serait de réaliser un parcours suivant l'ordre lexicographique engendré par $\preceq_y$ (minimisation des cellules alignées selon l'abscisse de leur centre), puis de faire de même en considérant $\preceq_x$. Mais ce double parcours ne permet pas toujours d'obtenir, pour une cellule de premier plan $R \in \mathbb{I}$, la cellule de fond la plus proche. Dans la figure 4.57, nous présentons un exemple où la $\mathbb{I}$-CDT ne serait pas correctement calculée en utilisant une approche basée sur l'algorithme 2.3 `parcourir`(). Pour palier à ce pro-

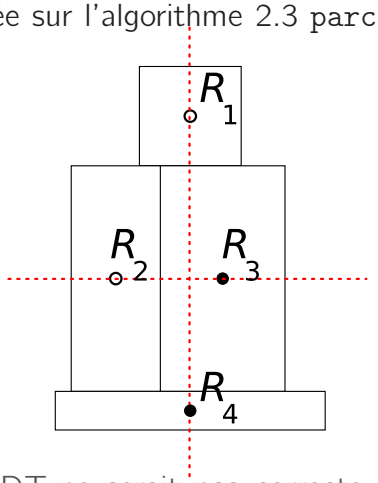


Fig. 4.57: Exemple où la $\mathbb{I}$-CDT ne serait pas correcte, ceci est dû à la structure de données engendrée par l'ordre lexicographique. Les cellules R_2 et R_3 sont parcourues lors de la première phase (ordre $\preceq_y$), et dans ce cas R_3 est la plus proche cellule de fond de R_2. Cette information ne peut pas être utilisée lors du parcours des cellules R_1 et R_4 (suivant $\preceq_x$), car ni R_2 ni R_3 ne sont alignées avec elles. Ainsi, la cellule de fond la plus proche de R_1 ne serait pas R_3, mais R_4, ce qui serait incorrect.

blème du calcul de la $\mathbb{I}$-CDT erroné à cause des cellules non alignées par leur centre, nous proposons maintenant une structure de données pour adapter ensuite deux algorithmes séparables.

La *matrice irrégulière* $\mathbf{A}$ associée à une $\mathbb{I}$-grille 2-D étiquetée $\mathbb{I}$ est donc construite en organisant les cellules suivant les axes X et Y (voir la figure 4.58 pour plus de détails). La valeur d'un nœud de $\mathbf{A}$ est fixée selon deux cas : (1) $\mathbf{A}(i,j)$ est le centre d'une cellule dans $\mathbb{I}$, et ce nœud peut être un nœud « premier plan » ($\mathbf{A}(i,j) = 1$) ou un nœud « fond » ($\mathbf{A}(i,j) = 0$) ; (2) il ne correspond pas à un centre de cellule dans $\mathbb{I}$, et on le crée en tant que *extra nœud* de premier plan ($\mathbf{A}(i,j) = 1$). Plus précisément, $\mathbf{A}$ possède autant de colonnes que de coordonnées en X dans l'ensemble des centres de cellules de $\mathbb{I}$. De même, nous construisons les lignes de $\mathbf{A}$ avec les coordonnées en Y de ces points. Ces coordonnées en X et en Y sont enregistrées dans deux tables T_X et T_Y. On note $n_X = |T_X|$ et $n_Y = |T_Y|$ le nombre de colonnes et de lignes de $\mathbf{A}$. Ainsi, pour connaître la coordonnée en X de $\mathbf{A}(i,j)$, il suffit de consulter $T_X(i)$. Construire la matrice irrégulière d'une $\mathbb{I}$-grille $\mathbb{I}$ peut être réalisée en $\mathcal{O}(n_X n_Y)$ en considérant un ordre basé sur les centres que nous avons défini. En effet, on parcourt d'abord toutes les cellules de $\mathbb{I}$ pour connaître les n_Y lignes et les n_X colonnes de $\mathbf{A}$. Puis, on considère chaque nœud de $\mathbf{A}$ et on affecte sa valeur en vérifiant s'il coïncide ou non avec un centre de cellule de $\mathbb{I}$. Nous proposons maintenant d'étudier l'adaptation de deux algorithmes destinés initialement à calculer la E^2DT d'une

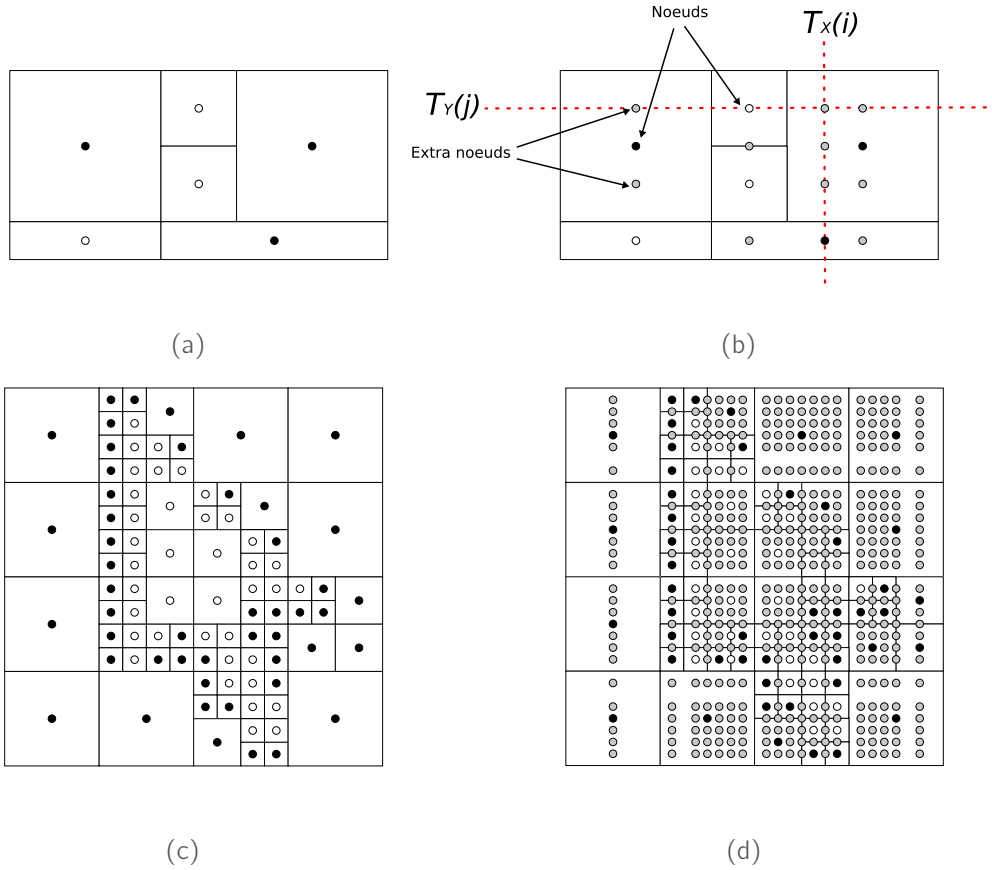


Fig. 4.58: Construction de la matrice irrégulière associée à une grille simple (a). Nous avons distingué un nœud $\mathbf{A}(i,j)$ de la matrice par l'intersection des droites en pointillés, et les extra nœuds sont en gris clair (b). Nous avons également présenté la matrice irrégulière associée à la décomposition quadtree de l'image *cursor* (bas).

image binaire. Grâce aux deux approches que nous développons, nous sommes capables de calculer la $\mathbb{I}$-CDT et $\mathbb{I}$-BDT d'une $\mathbb{I}$-grille 2-D étiquetée quelconque, et de l'étendre à d dimensions.

4.3.2 Un algorithme séparable pour calculer la $\mathbb{I}$-DT basé sur [Saito et Toriwaki, 1994]

Calcul de la $\mathbb{I}$-CDT, basée sur les centres de cellules

Dans nos recherches, nous avons d'abord proposé d'adapter l'algorithme séparable de [Saito et Toriwaki, 1994] sur la matrice irrégulière. Dans le reste de ce chapitre, pour nommer notre algorithme, nous ne citons que l'article de T. Saito et J. I. Toriwaki, bien que cette technique soit optimale grâce à d'autres travaux (elle devient alors linéaire en la taille de l'image) [Hirata, 1996, Meijster *et al.*, 2000]. La $\mathbb{I}$-CDT d'une $\mathbb{I}$-grille $\mathbb{I}$ donnée

par l'équation 4.28 est représentée par les valeurs de la matrice de la forme :

$$\mathbf{C}(i,j) = \min_{x,y} \quad \big\{(T_X(i) - T_X(x))^2 + (T_Y(j) - T_Y(y))^2; \\ x \in \{0,\dots,n_X-1\},\ y \in \{0,\dots,n_Y-1\},\ \mathbf{C}(x,y) = 0\big\}. \quad (4.34)$$

L'opérateur *min* est calculé sur tous les éléments de **C** (nœuds et extra nœuds). Pour obtenir la $\mathbb{I}$-CDT, notre algorithme se décompose en deux étapes :

1. Soit **A** la matrice irrégulière construite à partir d'une $\mathbb{I}$-grille $\mathbb{I}$. On réalise d'abord une $\mathbb{I}$-DT monodimensionnelle suivant l'axe des X, stockée dans une matrice irrégulière **B** telle que :

$$\mathbf{B}(i,j) = \min_{x} \big\{|T_X(i) - T_X(x)|;\ x \in \{0,\dots,n_X-1\},\ \mathbf{A}(x,j) = 0\big\}. \quad (4.35)$$

2. On traite ensuite suivant l'axe des Y pour construire la matrice irrégulière finale **C** :

$$\mathbf{C}(i,j) = \min_{y} \big\{\mathbf{B}(i,y)^2 + (T_Y(j) - T_Y(y))^2;\ y \in \{0,\dots,n_Y-1\}\big\}. \quad (4.36)$$

Nous présentons en détails les deux phases de notre approche dans l'algorithme 4.14. Dans la première étape, on peut remarquer que la seule différence avec le cas discret régulier [Saito et Toriwaki, 1994, Coeurjolly et Montanvert, 2007] est le calcul de la distance, lignes 1 et 2. En fait, nous devons considérer dans ces opérations la distance entre le point $\mathbf{B}(i,j)$ et son voisin (*e.g.* $|T_X(i) - T_X(i-1)|$ ligne 1). La deuxième phase de notre algorithme est en fait le calcul de l'enveloppe inférieure d'un ensemble de paraboles [Hirata, 1996]. Après l'étape 1, on peut étudier l'ensemble des paraboles $\mathcal{F}_y^i(j) = \mathbf{B}(i,y)^2 + (T_Y(j) - T_Y(y))^2$ sur la colonne $\{\mathbf{B}(i,y)\}_{0\le y\le n_Y}$. Avec l'étape 2, la colonne $\{\mathbf{C}(i,y)\}_{0\le y\le n_Y}$ est l'enveloppe inférieure de l'ensemble $\{\mathcal{F}_y^i\}_{0\le y\le n_Y}$. De plus, la fonction $CSep^i(u,v)$ est la coordonnée exacte du point d'intersection entre deux paraboles [Meijster *et al.*, 2000, Coeurjolly et Montanvert, 2007] (on notera simplement $\mathcal{F}_y(j) = \mathcal{F}_y^i(j)$ et $Sep(u,v) = Sep^i(u,v)$ quand le paramètre i est fixé) :

$$CSep^i(u,v) = \big(T_Y(v)^2 - T_Y(u)^2 + \mathbf{B}(i,v)^2 - \mathbf{B}(i,u)^2\big)/\big(2(T_Y(v) - T_Y(u))\big). \quad (4.37)$$

Dans la figure 4.59 sont présentées les deux phases de notre approche, à partir d'une $\mathbb{I}$-grille simple. On peut remarquer que, à l'issue de l'étape 1, les nœuds *inf* signifient qu'il n'existe pas de nœud de fond sur la ligne contenant ce point. Donc, la dernière boucle *pour* dans l'algorithme 4.14 ligne 2 ne change pas $\mathbf{B}(i,j) = \infty$. Si l'on choisit une colonne de la matrice irrégulière **B** (b), la phase 2 revient à calculer l'intersection entre l'enveloppe inférieure de l'ensemble des deux paraboles données en (d). Elles sont associées au nœuds de **B** qui n'ont pas de valeur *inf*. Dans ce graphique, nous avons illustré par des points l'intersection entre les nœuds d'ordonnées $T_Y(y)$ et cette enveloppe (pleins pour les nœuds et vides pour les extras nœuds). Une fois la distance calculée en chacun des points de la matrice (c), on peut en déduire la $\mathbb{I}$-CDT totale de la grille $\mathbb{I}$ (e).

Algorithme 4.14 : Calcul de la $\mathbb{I}$-CDT par une approche séparable [Saito et Toriwaki, 1994]

entrée : une $\mathbb{I}$-grille étiquetée $\mathbb{I}$.
sortie : la $\mathbb{I}$-CDT de $\mathbb{I}$, stockée dans la matrice irrégulière $\mathbf{C}$.

début
construire la matrice irrégulière $\mathbf{A}$ associée à $\mathbb{I}$;
// Étape 1 suivant l'axe des X
pour $j = 0$ **à** $n_Y - 1$ **faire**
si $\mathbf{A}(0, j) = 0$ **alors**
$\mathbf{B}(0, j) \leftarrow 0$;
sinon
$\mathbf{B}(0, j) \leftarrow \infty$;
pour $i = 1$ **à** $n_X - 1$ **faire**
si $\mathbf{A}(i, j) = 0$ **alors**
$\mathbf{B}(i, j) \leftarrow 0$;
sinon
1 $\mathbf{B}(i, j) \leftarrow |T_X(i) - T_X(i-1)| + \mathbf{B}(i-1, j)$;
pour $i = n_X - 2$ **à** 0 **faire**
si $\mathbf{B}(i+1, j) < \mathbf{B}(i, j)$ **alors**
2 $\mathbf{B}(i, j) \leftarrow |T_X(i) - T_X(i+1)| + \mathbf{B}(i+1, j)$;
// Étape 2 suivant l'axe des Y
pour $i = 0$ **à** $n_X - 1$ **faire**
$q \leftarrow 0$; $s[0] \leftarrow 0$; $t[0] \leftarrow 0$;
pour $j = 1$ **à** $n_Y - 1$ **faire**
tant que $q \geq 0 \wedge \mathcal{F}_{s[q]}(t[q]) > \mathcal{F}_j(t[q])$ **faire**
$q \leftarrow q - 1$;
si $q < 0$ **alors**
$q \leftarrow 0$; $s[q] \leftarrow j$;
sinon
3 $w \leftarrow CSep(s[q], j)$;
si $w \leq T_Y(n_Y - 1)$ **alors**
4 trouver le nœud $\mathbf{B}(i, k)$, $k \in \{0, \ldots, n_Y - 1\}$ tel que $T_Y(k) > w$;
$q \leftarrow q + 1$; $s[q] \leftarrow k$; $t[q] \leftarrow w$;
pour $j = n_Y - 1$ **à** 0 **faire**
$\mathbf{C}(i, j) \leftarrow \mathcal{F}_{s[q]}(j)$;
si $T_Y(j) = t[q]$ **alors**
$q \leftarrow q - 1$;
fin

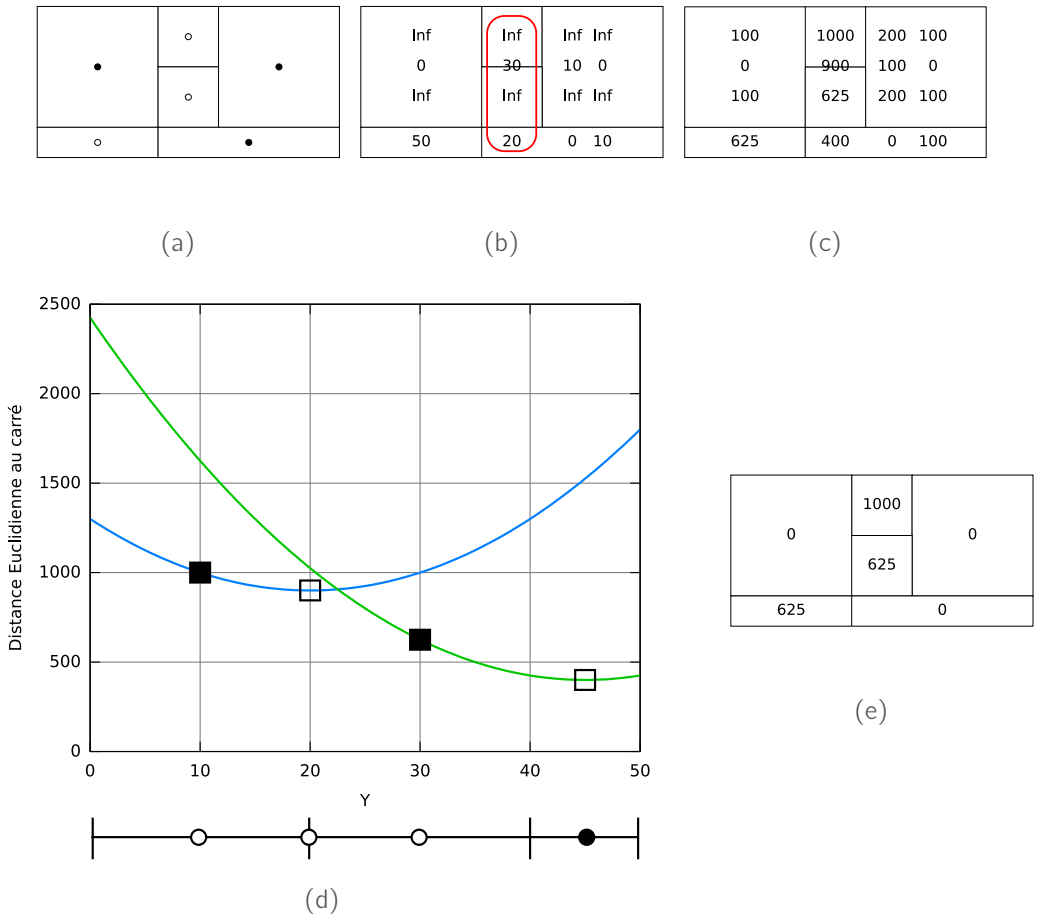
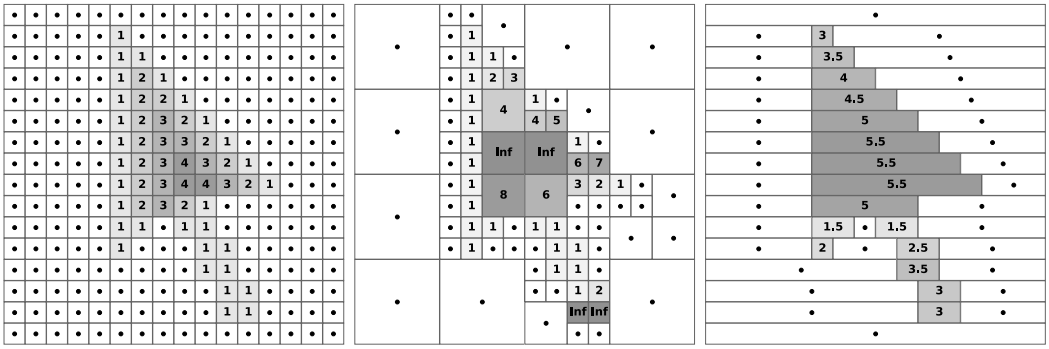


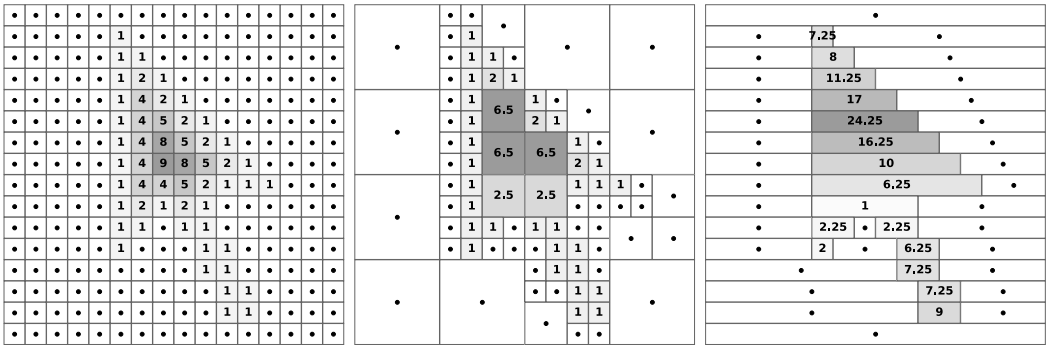
Fig. 4.59: Illustration du comportement de l'algorithme 4.14 sur un exemple simple. On construit d'abord la matrice irrégulière à partir de la $\mathbb{I}$-grille de taille 100x50 en entrée (a), puis on exécute l'étape 1 (b). Si l'on prend l'exemple de la colonne sélectionnée, l'ensemble des paraboles correspondant est donné en (d). Sous ce graphique, nous rappelons la position des nœuds de la matrice et les bords des cellules qui les contiennent. L'enveloppe inférieure nous donne la valeur de la $\mathbb{I}$-CDT de chaque nœud de cette colonne (c), et donc la $\mathbb{I}$-CDT de la grille en entrée (e).

En comparaison avec les grilles régulières, on peut voir que l'opérateur div a été remplacé par l'opérateur en arithmétique flottante / dans l'équation 4.37 afin de calculer le point d'intersection exact. Pour les $\mathbb{I}$-grilles issues d'un processus de subdivision (*e.g.* quadtree) ou d'un processus de groupement de cellules (*e.g.* RLE), les coordonnées peuvent être des demi-entiers, ce qui implique que nous avons juste à multiplier les coordonnées des cellules de la grille par 2 pour retrouver un calcul entier de $CSep()$. Le calcul de w (ligne 3) ne dépend que de la fonction $CSep(u, v)$ et permet donc de trouver le point d'intersection dans $\mathbf{B}$ (ligne 4.15). Ici, cette opération est réalisée par une recherche dichotomique au sein des nœuds ordonnés $\{\mathbf{B}(i, k)\}_{s[q] \leq k \leq n_Y - 1}$, et a une complexité en $\mathcal{O}(\log n_Y)$ au pire cas. Lors des expérimentations, nous avons observé qu'il s'agit d'une opération rapide.

On peut voir dans la figure 4.60 quelques résultats de notre algorithme sur l'image binaire *cursor* utilisée dans la figure 4.52.



(a)



(b)

Fig. 4.60: Calcul de la $\mathbb{I}$-CDT par l'algorithme 4.14 sur des $\mathbb{I}$-grilles simples issues de l'image *cursor*. La carte de distance temporaire obtenue par la phase 1 est illustrée en (a). La $\mathbb{I}$-CDT finale est rappelée en (b).

Préliminaires au calcul de la $\mathbb{I}$-BDT par notre approche

Dans la section suivante, nous détaillons l'algorithme qui permet de calculer la $\mathbb{I}$-BDT par une technique séparable inspirée de [Saito et Toriwaki, 1994, Hirata, 1996, Meijster *et al.*, 2000], comme nous l'avons proposé pour la $\mathbb{I}$-CDT. Dans le cas régulier, l'algorithme original est linéaire et correct, car l'intersection entre deux paraboles $\mathcal{F}_\alpha(y)$ et $\mathcal{F}_\beta(y)$ est clairement déterminée :

Propriété 4.1 (Intersection entre deux paraboles classiques [Hirata, 1996])**.** *Soient deux paraboles* $\mathcal{F}_\alpha : \mathbb{R} \to \mathbb{R}, y \to g_\alpha^2 + (\alpha - y)^2$ *et* $\mathcal{F}_\beta : \mathbb{R} \to \mathbb{R}, y \to g_\beta^2 + (\beta - y)^2$*. Le nombre d'intersections entre ces deux paraboles est soit un, soit une infinité si elles sont confondues, i.e. lorsque* $g_\alpha = g_\beta$ *et* $\alpha = \beta$*.*

Cette propriété vient de la totale monotonicité des paraboles [Hirata, 1996]. Elle implique que l'on peut calculer l'enveloppe inférieure de l'ensemble des paraboles car nous sommes capables d'ordonner correctement cet ensemble. Pour que notre approche adaptée sur $\mathbb{I}$-grilles soit correcte, nous devons vérifier que nous conservons cette propriété.

Par la suite, nous appelons *branche gauche* d'une parabole $\mathcal{F}_\alpha$ l'ensemble des points $(y, \mathcal{F}_\alpha(y))$ tels que $y \leq \alpha$. De même, la *branche droite* est l'ensemble des points tels que $y \geq \alpha$. Nous pouvons énoncer maintenant une autre propriété :

Propriété 4.2 (Test entre deux parties d'une parabole)**.** *Soient deux paraboles $\mathcal{F}_\alpha$ et $\mathcal{F}_\beta$, avec $\beta \neq \alpha$. S'il existe une intersection entre $\mathcal{F}_\alpha$ et la branche gauche (respectivement droite) de $\mathcal{F}_\beta$, alors il ne peut y avoir croisement entre $\mathcal{F}_\alpha$ et la branche droite (gauche) de $\mathcal{F}_\alpha$.*

Nous pouvons énoncer cette proposition car les deux paraboles sont monotones et il ne peut exister qu'une intersection au plus entre elles (propriété 4.1 précédente). Dans ce cas, nous devons déterminer comment s'intersectent deux paraboles *aplaties* de la forme (un exemple est donné dans la figure 4.61) :

$$\mathcal{G}_u(y) = \begin{cases} g_u^2 + (u - y - lu_1)^2 & si\ u - y > lu_1 \\ g_u^2 + (y - u - lu_2)^2 & si\ y - u > lu_2 \\ g_v^2 & sinon \end{cases} \tag{4.38}$$

et

$$\mathcal{G}_v(y) = \begin{cases} g_v^2 + (v - y - lv_1)^2 & si\ v - y > lv_1 \\ g_v^2 + (y - v - lv_2)^2 & si\ y - v > lv_2 \\ g_v^2 & sinon \end{cases} \tag{4.39}$$

où :

- ▷ le repère considéré est noté OYZ, ce qui signifie qu'un point du plan est noté (y, z) ;
- ▷ g_u, g_v, lu_1, lu_2, lv_1, lv_2, u et v sont des réels positifs ou nuls ;
- ▷ si $u \geq v \geq 0$, alors $u - lu_1 \geq v + lv_2$;
- ▷ si $0 \leq u \leq v$, alors $u + lu_2 \leq v - lu_1$.

Nous verrons dans la section suivante d'où provient la définition d'une telle fonction, et pourquoi nous bornons les valeurs des paramètres. Les bornes que nous imposons à u, v, et les *paramètres d'écarts* (lu_1, lu_2, lv_1 et lv_2) correspondront alors à des coordonnées et des tailles de cellules disjointes d'une $\mathbb{I}$-grille. On peut remarquer qu'une parabole aplatie $\mathcal{G}_u$ est composée de trois parties :

- ▷ Une fonction constante de valeur g_u^2 au centre, bornée au sens large par $u - lu_1$ et $u + lu_2$. Elle est nommée la *base* et notée $\mathcal{B}_u$;
- ▷ la branche gauche de la parabole de forme $g_u^2 + (y - u + lu_1)^2$ centrée au point $(g_u^2, u - lu_1)$. On note $\mathcal{F}_{u-lu_1}(y) = \mathcal{F}_u^1(y)$ la parabole entière, et $\widehat{\mathcal{F}}_u^1(y)$ sa branche gauche (son centre est exclu) ;
- ▷ la branche droite de la parabole de forme $(g_u^2 + (u - y - lu_2)^2$ centrée au point $(g_u^2, u + lu_2)$. On note $\mathcal{F}_{u+lu_2}(y) = \mathcal{F}_u^2(y)$ la parabole entière, et $\widehat{\mathcal{F}}_u^2(y)$ sa branche droite (son centre est exclu).

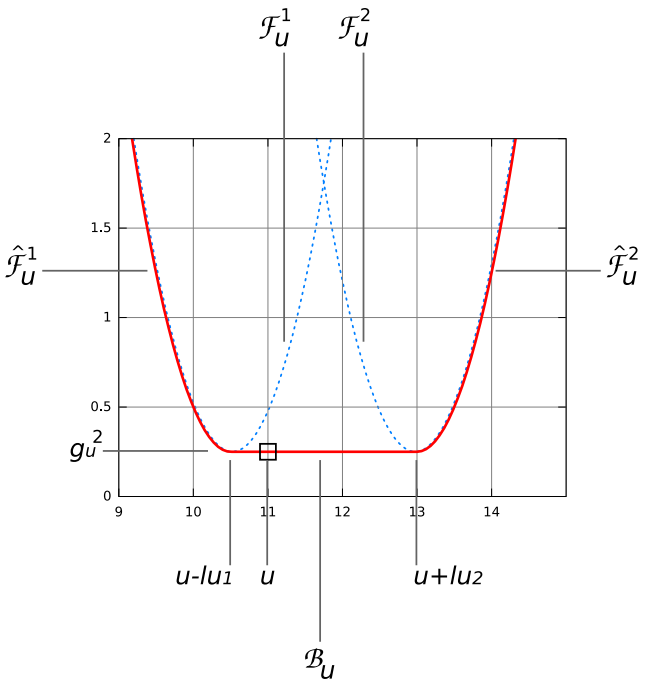


Fig. 4.61: Exemple d'une parabole aplatie $\mathcal{G}_u(y)$. On a ici $u = 11$, $g_u = 0.5$, $lu_1 = 0.5$, et $lu_2 = 2$. On peut remarquer que cela revient à considérer une fonction constante et deux branches de paraboles classiques (en pointillés).

On peut indiquer qu'avec nos notations, on a $\mathcal{G}_u(y) = \mathcal{B}_u \cup \widehat{\mathcal{F}}_u^1(y) \cup \widehat{\mathcal{F}}_u^2(y)$, car les centres des branches sont exclus. Nous démontrons maintenant que le nombre d'intersections possibles est également un ou l'infinité pour les paraboles aplaties que nous avons définies, pour des raisons de monotonicité. Nous exposons également les conditions pour que l'un de ces événements se produise, et comment calculer l'intersection de manière générale en définissant une fonction $BSep()$.

Nous considérons par la suite, sans perte de généralité, que $g_u \geq g_v$ et $v \geq u$ (des raisonnements analogues peuvent être déroulés dans les cas contraires). Nous devons étudier quatre cas distincts (pour chaque cas, une illustration est tracée pour faciliter la lecture).

Cas 1 (adjacence) : $\mathbf{g_u = g_v}$, $\mathbf{u + lu_2 = v - lv_1}$, $\mathbf{lu_1 \neq 0}$, $\mathbf{lu_2 \neq 0}$, $\mathbf{lv_1 \neq 0}$, $\mathbf{lv_2 \neq 0}$
Ici, les deux parties constantes ont même valeur, et s'intersectent en une extrémité (car $u + lu_2 = v - lv_1$). Par monotonicité, la partie droite de $\mathcal{G}_v$ ne peut intersecter $\mathcal{G}_u$. En effet, comme les écarts lv_1, lv_2 sont non nuls, elle ne peut atteindre ni $\widehat{\mathcal{F}}_u^1$ ni $\widehat{\mathcal{F}}_u^2$. Il en est de même pour la branche gauche de $\mathcal{F}_u^1$ avec $\mathcal{G}_v$. Ainsi, s'il existe une intersection, elle est réalisée soit entre les portions constantes des deux paraboles, soit entre $\widehat{\mathcal{F}}_u^2$ et $\widehat{\mathcal{F}}_v^1$. Mais

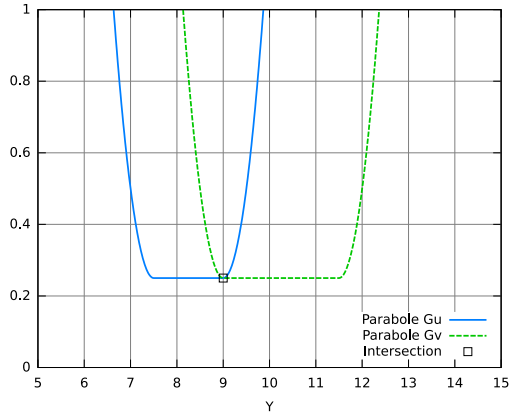


Fig. 4.62: Illustration du cas 1 de notre preuve. La partie droite de $\mathcal{G}_u$, $\widehat{\mathcal{F}}^2_u$, ne peut intersecter $\mathcal{G}_v$ car $lu_1, lu_2 \neq 0$.

cette dernière affirmation reviendrait à

$$\begin{aligned} & g_u^2 + (y - u - lu_2)^2 = g_v^2 + (v - y - lv_1)^2 \\ \Leftrightarrow \quad & g_u^2 + (y - u - lu_2)^2 = g_u^2 + (-y + v - lv_1)^2 \\ \Leftrightarrow \quad & (y - u - lu_2)^2 = (-y + v - lv_1)^2 \\ \Leftrightarrow \quad & y = u + lu_2. \end{aligned}$$

car nous ne manipulons que des valeurs positives. Cela implique que l'intersection entre les deux branches est le point $(u + lu_2, g_u^2)$. Clairement, ce point est également l'intersection entre les deux portions constantes des paraboles $\mathcal{G}_u$ et $\mathcal{G}_v$. Ainsi, dans ce cas numéro 1, l'intersection entre les deux paraboles aplaties est un unique point.

Cas 2 (chevauchement) : $\mathbf{g_u = g_v}$, et soit $\mathbf{v = u + lu_2}$, $\mathbf{lv_1 = lv_2 = 0}$, soit $\mathbf{u = v - lv_1}$, $\mathbf{lu_1 = lu_2 = 0}$ Nous sommes ici dans un cas particulier du cas 1, où l'une des deux paraboles aplaties n'a pas de paramètres d'écarts. Ainsi, on a, par exemple, $\mathcal{G}_v = \mathcal{F}^2_v = \mathcal{F}^1_v = \mathcal{F}_v$ si $v = u + lu_2$ et $lv_1 = lv_2 = 0$. Nous travaillons sur ce cas précis, sachant que le cas où $\mathcal{G}_u = \mathcal{F}_u$ se base sur des éléments similaires. Comme $\mathcal{G}_v = \mathcal{F}_v$, et comme $v = u + lu_2$, alors la parabole droite de $\mathcal{G}_u$ coïncide avec $\mathcal{G}_v$, car

$$\begin{aligned} & \mathcal{F}^2_u(y) = g_u^2 + (y - u - lu_2)^2 \\ \Leftrightarrow \quad & \mathcal{F}^2_u(y) = g_v^2 + (y - v)^2 \\ \Leftrightarrow \quad & \mathcal{F}^2_u(y) = \mathcal{F}_v(y) = \mathcal{G}_v(y). \end{aligned}$$

Cela implique naturellement qu'il existe une infinité d'intersections entre les deux paraboles $\widehat{\mathcal{F}}^2_u(y)$, la branche droite de $\mathcal{G}_u$, et $\mathcal{G}_v$. Pour borner inférieurement cette infinité de points, nous ne pouvons pas considérer d'intersection entre $\mathcal{G}_v$ et $\widehat{\mathcal{F}}^1_u(y)$ (branche gauche) car, vu que $lu_1, lu_2 \neq 0$, alors $\mathcal{G}_v$ n'intersecte que la branche droite de $\mathcal{F}^1_u(y)$. Par la propriété 4.2, on en conclut que $\mathcal{G}_v$ ne croise pas la branche gauche de $\mathcal{F}^1_u(y)$, *i.e.* $\widehat{\mathcal{F}}^1_u(y)$. Ainsi,le plus

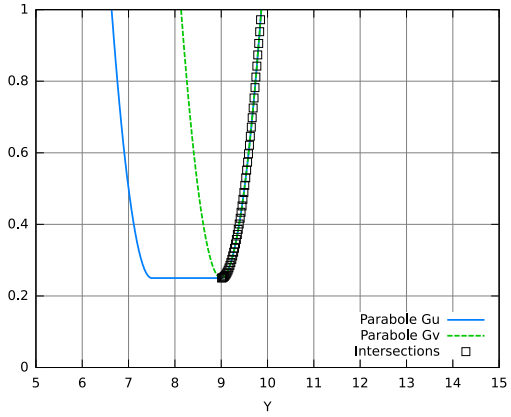


Fig. 4.63: Illustration du cas 2 de notre preuve. Comme la parabole droite de $\mathcal{G}_u$ coïncide avec $\mathcal{G}_v$, il existe une infinité de solutions.

petit point appartient à $\mathcal{B}_u$, et a pour coordonnées (v, g_u^2). Nous arrivons à la même conclusion si nous supposons que $\mathcal{G}_u = \mathcal{F}_u$ ($Iu_1 = Iu_2 = 0$), où l'on obtient que $\mathcal{F}_v^1(y) = \mathcal{G}_u(y)$. De même, on peut aussi supposer que $u = v$ et $Iu_1 = Iu_2 = Iv_1 = Iv_2 = 0$. Sous ces conditions, nous considérons $\mathcal{G}_u = \mathcal{F}_u$ et $\mathcal{G}_v = \mathcal{F}_v = \mathcal{F}_u$ comme deux paraboles classiques, et confondues. Ainsi, comme l'indique la propriété 4.1, elles ont en commun une infinité de points. Par conséquent, l'intersection entre les paraboles aplaties décrites dans ce cas 2 est une infinité de points, bornés inférieurement.

Cas 3 (général) : $\mathbf{g_u > g_v}$ Comme nous supposons maintenant que $v > u$, nous pouvons affirmer que la branche droite de $\mathcal{G}_v$ ($\widehat{\mathcal{F}}_v^2$) ne peut intersecter la parabole $\mathcal{G}_u$, par monotonicité (propriété 4.1). Supposons par exemple que $\widehat{\mathcal{F}}_v^2$ croise la branche droite de $\mathcal{G}_u$ (*i.e.* $\widehat{\mathcal{F}}_u^2$). On a alors :

$$\begin{aligned} & g_u^2 + (y - u - Iu_2)^2 = g_v^2 + (y - v - Iv_2)^2 \\ \Leftrightarrow \quad & g_u^2 - g_v^2 = (y - v - Iv_2)^2 - (y - u - Iu_2)^2, \end{aligned}$$

et comme $g_u > g_v$, nous aboutissons à

$$\begin{aligned} & (y - v - Iv_2)^2 - (y - u - Iu_2)^2 > 0 \\ \Leftrightarrow \quad & y - v - Iv_2 > y - u - Iu_2 \\ \Leftrightarrow \quad & v + Iv_2 \leq u + Iu_2. \end{aligned}$$

Or, dans notre définition d'une parabole aplatie, nous avons $v \geq u \Rightarrow u + Iu_2 \leq u - Iv_1$, et de plus, par construction, $u - Iv_1 \leq v + Iv_2$. Ainsi, nous avons à la fois $u + Iu_2 \leq v + Iv_2$ et $v + Iv_2 \leq u + Iu_2$, d'où une contradiction car $v > u$. Nous pouvons démontrer de la même manière que $\widehat{\mathcal{F}}_v^2$ ne peut pas intersecter les autres parties de $\mathcal{G}_u$, à savoir sa base $\mathcal{B}_u$ et sa branche gauche $\widehat{\mathcal{F}}_u^1$. Nous savons aussi que la base de $\mathcal{G}_v$ (d'équation $z = g_v^2$) ne peut croiser $\mathcal{G}_u$, car $g_u > g_v$. En effet, les ordonnées des points de $\mathcal{G}_u$ sont toutes supérieures à $g_u^2 > g_v^2$.

Par conséquent, nous pouvons considérer maintenant l'intersection entre la branche gauche de $\mathcal{G}_v$ (*i.e.* $\widehat{\mathcal{F}}_v^1(y)$) et la parabole $\mathcal{G}_u$. Pour déterminer à quel endroit de $\mathcal{G}_v$ coupe cette branche, nous calculons en premier lieu l'intersection entre $\widehat{\mathcal{F}}_v^1(y)$ et la droite d'ordonnée $z = g_u^2$:

$$\begin{aligned} & g_v^2 + (v - y - Iv_1)^2 = g_u^2 \\ \Leftrightarrow \quad & (v - y - Iv_1)^2 = g_u^2 - g_v^2 \\ \Leftrightarrow \quad & v - y - Iv_1 = \sqrt{g_u^2 - g_v^2} \\ \Leftrightarrow \quad & y = v - Iv_1 - \sqrt{g_u^2 - g_v^2}. \end{aligned}$$

Le passage à la racine n'est pas ambigu car nous traitons des valeurs positives. Par la suite, nous considérons les différentes valeurs de y que nous notons $\overline{y}$ pour ne pas confondre avec la variable globale y. Nous étudions également les deux points suivants :

▷ L'intersection entre $\widehat{\mathcal{F}}_v^1(y)$ et la droite d'équation $y = u - Iu_1$;

▷ L'intersection entre $\widehat{\mathcal{F}}_v^1(y)$ et la droite d'équation $y = u + Iu_2$.

Nous pouvons calculer ces croisements car on a $v - Iv_1 \geq u + Iu_2 \geq u - Iu_1$. Ainsi, par monotonicité, nous avons clairement :

$$\begin{aligned} & 0 \leq \widehat{\mathcal{F}}_v^1(u + Iu_2) \leq \widehat{\mathcal{F}}_v^1(u - Iu_1) \\ \Leftrightarrow \quad & 0 \leq \widehat{\mathcal{F}}_v^1(u + Iu_2) - g_v^2 \leq \widehat{\mathcal{F}}_v^1(u - Iu_1) - g_v^2, \end{aligned}$$

La dernière double inégalité est vraie, par construction de la branche gauche de $\mathcal{G}_v$. Nous notons par la suite

$$\begin{aligned} & F_1 = \widehat{\mathcal{F}}_v^1(u - Iu_1) - g_v^2, \\ & F_2 = \widehat{\mathcal{F}}_v^1(u + Iu_2) - g_v^2, \text{ avec} \\ & 0 \leq F_2 \leq F_1. \end{aligned}$$

Comme $g_u > g_v$, nous comparons maintenant ces deux valeurs avec $g_u^2 - g_v^2 = G > 0$ (pour un exemple de ces valeurs et voir l'intérêt de notre comparaison, on peut se référer à la figure 4.64). Pour chaque cas (suivant la place de G dans l'équation précédente), nous montrons une représentation de ces trois éléments, pour faciliter la lecture. Nous établissons également le lien avec $\overline{y}$, avant de déterminer exactement l'intersection entre les paraboles aplaties que nous considérons.

Cas 3a : $\mathbf{F_2 \leq G \leq F_1}$ Cette double inégalité implique que

$$\begin{aligned} \widehat{\mathcal{F}}_v^1(u + Iu_2) - g_v^2 \leq & \quad g_u^2 - g_v^2 \quad & \leq \widehat{\mathcal{F}}_v^1(u - Iu_1) - g_v^2 \\ \Leftrightarrow (v - u - Iu_2 - Iv_1)^2 \leq & \quad g_u^2 - g_v^2 \quad & \leq (v - u + Iu_1 - Iv_1)^2 \\ \Leftrightarrow v - u - Iu_2 - Iv_1 \leq & \quad \sqrt{g_u^2 - g_v^2} \quad & \leq v - u + Iu_1 - Iv_1 \\ \Leftrightarrow v - u - Iu_2 - Iv_1 \leq & \quad \sqrt{g_u^2 - g_v^2} \quad & \leq v - u + Iu_1 - Iv_1 \\ \Leftrightarrow -u - Iu_2 \leq & \quad -v + Iv_1 + \sqrt{g_u^2 - g_v^2} \quad & \leq -u + Iu_1 \\ \Leftrightarrow u - Iu_1 \leq & \quad \overline{y} \quad & \leq u + Iu_2. \end{aligned}$$

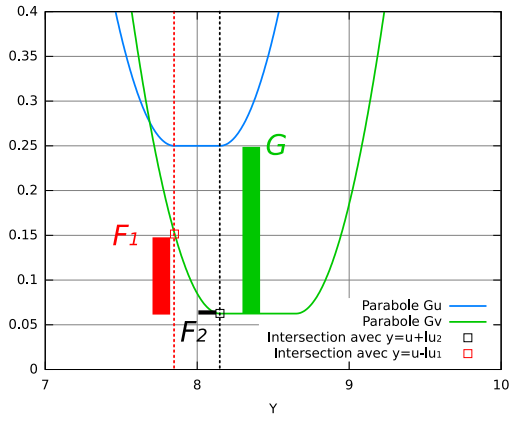


Fig. 4.64: Comparaison entre les valeurs F_1, F_2 et G de notre analyse. On a dans cet exemple $g_u^2 - g_v^2 > \widehat{\mathcal{F}}_v^1(u - Iu_1) - g_v^2 > \widehat{\mathcal{F}}_v^1(u + Iu_2) - g_v^2 \Leftrightarrow G > F_1 > F_2 > 0$, avec F_2 très proche de 0. Cela nous permet de caractériser l'intersection entre $\mathcal{G}_u$ et $\mathcal{G}_v$. Le cas qui est illustré ici (la branche gauche de $\mathcal{G}_v$ croise la branche gauche de $\mathcal{G}_u$) est traité dans le paragraphe « Cas 3b ».

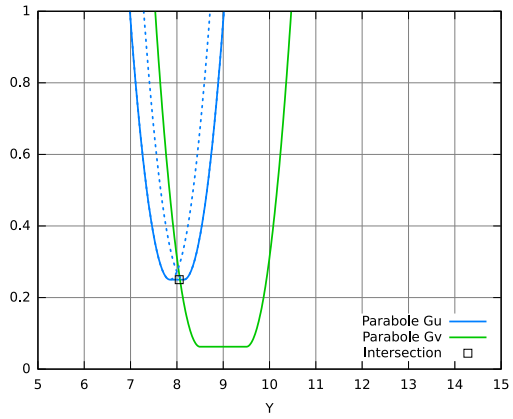


Fig. 4.65: Illustration du cas 3a de notre preuve. $\mathcal{G}_v$ intersecte obligatoirement la branche droite de $\mathcal{F}_u^1(y)$ (parabole classique de gauche en pointillés). De même, elle croise la branche gauche de $\mathcal{F}_u^2(y)$ (parabole classique de droite en pointillés).

Ce qui signifie clairement que la parabole $\mathcal{G}_v$ intersecte $\mathcal{G}_u$ en sa base, $\mathcal{B}_u$, et plus exactement au point $(\overline{y}, g_u^2)$. Ainsi, $\mathcal{G}_v$ intersecte obligatoirement la branche droite de $\mathcal{F}_u^1(y)$, car $u - Iu_1 \leq \overline{y}$. Par monotonicité (et par la propriété 4.2), elle ne peut donc pas atteindre l'autre branche de cette parabole. De la même manière, $\mathcal{G}_v$ croise la branche gauche de $\mathcal{F}_u^2(y)$, puisque $\overline{y} \leq u + Iu_2$, et ne peut pas intersecter sa branche droite. Par conséquent, $\mathcal{G}_v$ croise en un seul point la parabole $\mathcal{G}_u$ en $\mathcal{B}_u$, et il n'existe pas d'autres points d'intersections entre elles. Ces remarques peuvent être énoncées aussi quand $Iu_1 = Iu_2 = 0$. Dans ce cas, le point d'intersection est (u, g_u^2), et $\mathcal{G}_v$ ne peut pas croiser les deux branches de $\mathcal{G}_u$ (qui excluent ce point).

Cas 3b : $\mathbf{F_2 \leq F_1 \leq G}$ Nous pouvons énoncer une double inégalité de la même manière que précédemment :

$$\begin{array}{rcl} \widehat{\mathcal{F}}_v^1(u+Iu_2) - g_v^2 \leq & \widehat{\mathcal{F}}_v^1(u-Iu_1) - g_v^2 & \leq g_u^2 - g_v^2 \\ \Leftrightarrow \overline{y} \leq & u - Iu_1 & \leq u + Iu_2. \end{array}$$

Nous pouvons de suite traiter le cas $\overline{y} = u - Iu_1$, qui engendre que la parabole $\mathcal{G}_v$ passe par

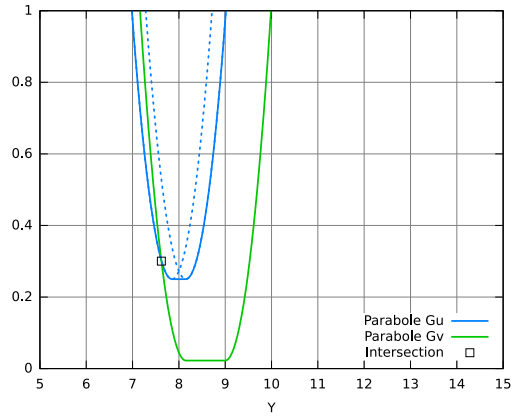


Fig. 4.66: Illustration du cas 3b de notre preuve. Si $\overline{y} < u - Iu_1$, il est clair que l'intersection a lieu entre $\mathcal{G}_v$ et la branche gauche de $\mathcal{G}_u$. Ainsi, $\mathcal{G}_v$ ne peut intersecter aucune autre partie de $\mathcal{G}_u$.

la base de $\mathcal{G}_u$. On peut se référer au cas précédent 3a pour calculer le point d'intersection. Si $\overline{y} < u - Iu_1$, et par construction d'une parabole aplatie, il est clair que l'intersection entre $\mathcal{G}_u$ et $\mathcal{G}_v$ ne peut avoir lieu que dans la branche gauche de $\mathcal{G}_u$ (*i.e.* $\widehat{\mathcal{F}}_u^2(y)$). En effet, comme $\overline{y} < u - Iu_1$, tout croisement avec la base $\mathcal{B}_u$ est impossible (c'est une fonction constante définie pour $u + Iu_2 \geq y \geq u - Iu_1$). Ensuite, supposons que $\mathcal{G}_v$ (et plus précisément sa branche gauche $\widehat{\mathcal{F}}_v^1(y)$) croise la branche droite de $\mathcal{G}_u$ ($\widehat{\mathcal{F}}_u^1(y)$), on a :

$$\begin{array}{rcl} \widehat{\mathcal{F}}_u^2(y) & = & \widehat{\mathcal{F}}_v^1(y) \\ \Leftrightarrow g_u^2 + (u - y - Iu_2)^2 & = & g_v^2 + (v - y - Iv_1)^2 \\ \Leftrightarrow g_u^2 - g_v^2 & = & (v - y - Iv_1)^2 - (u - y - Iu_2)^2. \end{array}$$

Or, nous avons supposé que $F_2 \leq F_1 \leq G = g_u^2 - g_v^2$. On en déduit :

$$\begin{array}{rcl} F_2 = \mathcal{F}_v^1(u - Iu_2) - g_v^2 & \leq & (v - y - Iv_1)^2 - (u - y - Iu_2)^2 \\ \Leftrightarrow (v - Iv_1 - u - Iu_2)^2 & \leq & (-2y + v - Iv_1 + u + Iu_2)(v - Iv_1 - u - Iu_2) \\ \Leftrightarrow v - Iv_1 - u - Iu_2 & \leq & -2y + v - Iv_1 + u + Iu_2 \\ \Leftrightarrow y & \leq & u + Iu_2, \end{array}$$

ce qui contredit notre hypothèse. En effet, il est impossible que le point d'intersection appartienne à la branche droite de $\mathcal{G}_u$, définie sur $y > u + Iu_2$, alors que nous avons

$y \leq u + Iu_2$. Ces raisonnements sont valables si $Iu_1 = Iu_2 = 0$ (dans ce cas, $\mathcal{G}_u$ est une parabole classique, et nous pouvons aussi utiliser la propriété 4.2). Nous en concluons qu'il existe un unique point d'intersection entre les deux paraboles $\mathcal{G}_u$ et $\mathcal{G}_v$, qui a pour coordonnées $(\overline{y}, \widehat{\mathcal{F}}_u^1(\overline{y}))$.

Cas 3c : $\mathbf{G \leq F_2 \leq F_1}$ Comme précédemment, nous bornons $\overline{y}$ grâce à la double inégalité suivante :

$$\begin{array}{rcll} g_u^2 - g_v^2 \leq & \widehat{\mathcal{F}}_v^1(u + Iu_2) - g_v^2 & \leq \widehat{\mathcal{F}}_v^1(u - Iu_1) - g_v^2 \\ \Leftrightarrow u - Iu_1 \leq & u + Iu_2 & \leq \overline{y}. \end{array}$$

De la même manière que pour le cas précédent, lorsque $y = u + Iu_2$, nous pouvons

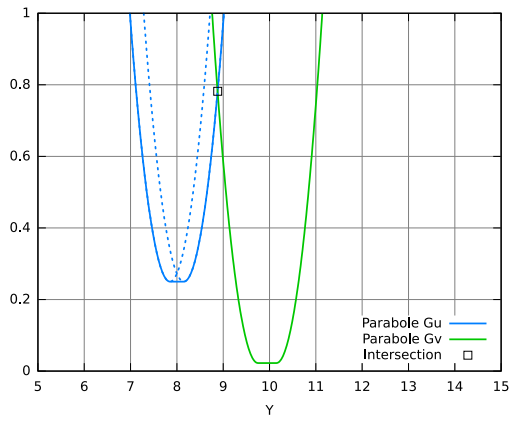


Fig. 4.67: Illustration du cas 3c de notre preuve. Si $\overline{y} > u + Iu_2$, il est clair que l'intersection a lieu entre $\mathcal{G}_v$ et la branche droite de $\mathcal{G}_u$. Ainsi, $\mathcal{G}_v$ ne peut intersecter aucune autre partie de $\mathcal{G}_u$.

appliquer la démonstration du cas 3a pour calculer le point d'intersection entre la parabole $\mathcal{G}_v$ et la base de $\mathcal{G}_u$, soit $\mathcal{B}_u$. En supposant maintenant que $u > u + Iu_2$, nous pouvons remarquer que l'intersection est impossible avec $\mathcal{B}_u$, car cette fonction constante est définie pour $u - Iu_1 \leq y \leq u + Iu_2$. Supposons maintenant que la branche gauche de $\mathcal{G}_v$ (*i.e.* $\widehat{\mathcal{F}}_v^1(y)$) croise la branche gauche de $\mathcal{G}_u$ (*i.e.* $\widehat{\mathcal{F}}_u^1(y)$). On a :

$$\begin{array}{rcl} \widehat{\mathcal{F}}_u^1(y) & = & \widehat{\mathcal{F}}_v^1(y) \\ \Leftrightarrow g_u^2 + (u - y - Iu_1)^2 & = & g_v^2 + (v - y - Iv_1)^2 \\ \Leftrightarrow g_u^2 - g_v^2 & = & (v - y - Iv_1)^2 - (u - y - Iu_1)^2. \end{array}$$

Comme nous avons posé l'hypothèse que $G \leq F_2 \leq F_1$, nous poursuivons ainsi :

$$\begin{array}{rcl} (v - y - Iv_1)^2 - (u - y - Iu_1)^2 \leq F_1 = \mathcal{F}_v^1(u - Iu_1) - g_v^2 & & \\ \Leftrightarrow (v - Iv_1 - u + Iu_1)(-2y + v - Iv_1 + u - Iu_1) & \leq & (v - Iv_1 - u + Iu_1)^2 \\ \Leftrightarrow -2y + v - Iv_1 + u - Iu_1 & \leq & v - Iv_1 - u + Iu_1 \\ \Leftrightarrow y & \geq & u - Iu_1, \end{array}$$

ce qui est en contradiction avec notre hypothèse, car nous avons supposé qu'une intersection existait sur la branche gauche de $\mathcal{G}_u$, définie pour $y < u - lu_1$, et nous avons obtenu $y \geq u + lu_1$. Par conséquent, il n'existe qu'une seule intersection entre $\mathcal{G}_v$ et $\mathcal{G}_v$, qui se situe en $(\overline{y}, \widehat{\mathcal{F}}_u^2(\overline{y}))$. De plus, nous arrivons à la même conclusion si $lu_1 = lu_2 = 0$ (comme dans le cas 3b).

Cas de non-intersection entre deux paraboles aplaties Dans l'ensemble des cas que nous avons étudiés jusqu'à présent, il reste la configuration où $u = v$, $lu_1 = lu_2 = lv_1 = lv_2 = 0$ (ce qui engendre que ce sont deux paraboles classiques $\mathcal{F}_u$ et $\mathcal{F}_v$ alignées selon l'axe des Y), avec $g_u > g_v$. Ce cas est similaire au cas 2, où les paraboles ont des valeurs g_u, g_v strictement différentes. Supposons qu'il existe un point commun aux deux paraboles, on a :

$$\begin{aligned} \mathcal{F}_u &= \mathcal{F}_v \\ \Leftrightarrow g_u^2 + (y-u)^2 &= g_v^2 + (y-v)^2 \\ \Leftrightarrow 0 < g_u^2 - g_v^2 &= (y-v)^2 - (y-u)^2 = 0, \end{aligned}$$

et nous aboutissons à cette contradiction puisque $g_u > g_v$ et $u = v$. Nous avons illustré ici le seul cas d'intersection vide entre deux paraboles aplaties.

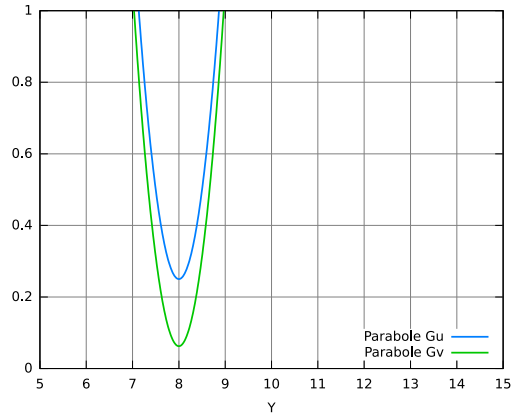


Fig. 4.68: Illustration du cas non-intersection entre deux paraboles aplaties. Si $u = v$, $lu_1 = lu_2 = lv_1 = lv_2 = 0$, et $g_u > g_v$, alors il ne peut y avoir croisement entre les deux paraboles aplaties.

Bilan et énoncé du lemme d'intersection entre deux paraboles aplaties Dans les paragraphes précédents, nous avons énuméré les différentes configurations de positions de deux paraboles aplaties $\mathcal{G}_u$ et $\mathcal{G}_v$, avec $u \leq v$ et $g_v \leq g_u$. Pour chacune d'entre elles, nous avons déterminé comment et où elles pouvaient s'intersecter. Il est clair qu'une étude similaire peut être établie pour les autres cas de comparaison entre u, v et entre g_u, g_v. Nous énonçons maintenant le lemme suivant :

Lemme 4.2 (Intersection entre deux paraboles aplaties)**.** *Soient deux paraboles aplaties $\mathcal{G}_u$ et $\mathcal{G}_v$ définies de la manière suivante*

$$\begin{array}{rcl} \mathcal{G}_u : \mathbb{R} & \to \mathbb{R} & \\ y & \to \mathcal{G}_u(y) = \left\{ \begin{array}{ll} g_u^2 + (u - y - Iu_1)^2 & si\ u - y > Iu_1 \\ g_u^2 + (y - u - Iu_2)^2 & si\ y - u > Iu_2 \\ g_v^2 & sinon \end{array} \right. \end{array}$$

et

$$\begin{array}{rcl} \mathcal{G}_v : \mathbb{R} & \to \mathbb{R} & \\ y & \to \mathcal{G}_v(y) = \left\{ \begin{array}{ll} g_v^2 + (v - y - Iv_1)^2 & si\ v - y > Iv_1 \\ g_v^2 + (y - v - Iv_2)^2 & si\ y - v > Iv_2 \\ g_v^2 & sinon \end{array} \right. \end{array}$$

où :

- ▷ *g_u, g_v, Iu_1, Iu_2, Iv_1, Iv_2, u et v sont des réels positifs ou nuls ;*
- ▷ *si $u \geq v \geq 0$, alors $u - Iu_1 \geq v + Iv_2$;*
- ▷ *si $0 \leq u \leq v$, alors $u + Iu_2 \leq v - Iu_1$.*

Le nombre d'intersections entre ces deux paraboles est soit un, soit une infinité. Pour déterminer le point d'intersection unique ou le plus petit point d'intersection (s'il y en a une infinité), nous définissons la procédure suivante :

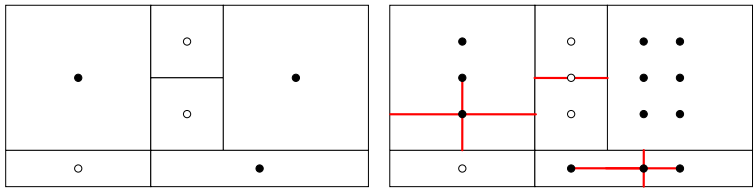
1. *Calculer le point d'intersection, d'abscisse $\overline{y}$, entre $\mathcal{G}_v$ et la droite d'équation $z = g_u^2$;*
2. *Si $u - Iu_1 \leq \overline{y} \leq u + Iu_2$, retourner le point $(\overline{y}, g_u^2)$;*
3. *Si $\overline{y} \leq u - Iu_1 \leq u + Iu_2$, retourner le point d'intersection entre $\widehat{\mathcal{F}}_v^1$ et $\widehat{\mathcal{F}}_u^1$;*
4. *Si $u - Iu_1 \leq u + Iu_2 \leq \overline{y}$, retourner le point d'intersection entre $\widehat{\mathcal{F}}_v^1$ et $\widehat{\mathcal{F}}_u^2$.*

Dans ce lemme, le calcul du point d'intersection se base tout d'abord sur le calcul de l'intersection entre $\mathcal{G}_v$ et $z = g_u^2$ par $\overline{y} = v - Iv_1 - \sqrt{g_u^2 - g_v^2}$, si $u \leq v$ et $g_u \geq g_v$. Ensuite, le premier test permet de vérifier les cas 1, 2 (les bases ont même ordonnée $g_u = g_v$) et 3a. Dans les deux autres configurations (cas 3b et 3c), nous proposons de calculer de manière générale le croisement entre les deux branches de chaque parabole, effectivement concernées par notre procédure. Nous voyons dans la section suivante pourquoi et comment appliquer ces principes dans le cadre du calcul de la $\mathbb{I}$-BDT par une approche séparable. Les paraboles aplaties permettent de représenter finalement les distances entre les bords de cellules.

Calcul de la $\mathbb{I}$-BDT, basée sur les bords des cellules

Pour appliquer la $\mathbb{I}$-BDT donnée dans l'équation 4.29 (entre une cellule de premier plan et la frontière premier plan / fond), nous devons considérer la forme des cellules que nous traitons. Dans les deux phases de notre algorithme, il est nécessaire d'intégrer la taille des cellules dans les processus de minimisation. Nous ajoutons pour chaque nœud

d'indice (i, j) de la matrice irrégulière les valeurs de « tailles » suivant les deux axes X et Y, et dans les deux directions notées $H_L(i, j)$, $H_R(i, j)$, $H_T(i, j)$ et $H_B(i, j)$ respectivement (voir la figure 4.69). Un nœud $\mathbf{A}(i, j)$ de la matrice est modifié en fonction de la cellule



(a) (b)

Fig. 4.69: Exemple de matrice irrégulière initialisée pour traiter la $\mathbb{I}$-BDT. En rouge sont illustrées les tailles associées à deux nœuds de la matrice. On peut noter que celui du bas possède une taille de gauche $H_L(i, j)$ différente de celle de droite ($H_R(i, j)$). De plus, un extra nœud a la même valeur que la cellule qui le contient, contrairement à la $\mathbb{I}$-CDT où il est toujours initialisé à 1. Le nœud premier plan de cet exemple possède une hauteur nulle, car il coïncide avec le bord d'une cellule.

R de la $\mathbb{I}$-grille $\mathbb{I}$ qui le contient, et de la position des nœuds voisins. Ainsi, $H_L(i, j)$ et $H_R(i, j)$ représentent respectivement le minimum entre

- ▷ la distance au bord gauche (droit) de R suivant X, et
- ▷ la distance au nœud à sa gauche (à sa droite) dans la matrice $\mathbf{A}$.

Il en est de même pour les valeurs de tailles suivant l'axe des Y. Pour un nœud qui coïncide avec le bord gauche de R, on a naturellement $H_R(i, j) = 0$ (voir figure 4.69). De plus, si une cellule ne contient aucun extra nœud, alors le nœud associé à son centre a pour taille $H_R(i, j) = H_L(i, j) = \frac{l^X_R}{2}$ par exemple. Cela se produit notamment lorsque l'on considère une grille régulière en entrée. La valeur du nœud est la même que celle de R (et non pas toujours de 1 pour les extras nœuds comme dans le cas précédent). Grâce à ces éléments, nous pouvons réécrire l'équation 4.35 de la manière suivante :

$$\mathbf{B}(i, j) = \min_x \Big\{ \min\big(T_X(i) - T_X(x) - H_R(x, j), T_X(x) - T_X(i) - H_L(i, j)\big) \; ; \; x \in \{0, \ldots, n_X - 1\}, \; \mathbf{A}(x, j) = 0 \Big\}. \tag{4.40}$$

Ce qui se traduit dans notre algorithme 4.14, ligne 1 par :

si $\mathbf{B}(i-1, j) = 0$ **alors**
  $\mathbf{B}(i, j) \leftarrow T_X(i) - T_X(i-1) - H_R(i-1, j)$;
sinon
  $\mathbf{B}(i, j) \leftarrow T_X(i) - T_X(i-1) + \mathbf{B}(i-1, j)$;

Cette modification est aussi réalisée de manière similaire ligne 2, où l'on teste si le nœud d'indices $(i + 1, j)$ est de fond ou non. Dans la seconde étape, l'équation 4.36

peut être écrite en changeant l'écriture de la parabole $\mathcal{F}_y(j)$ par celle de $\mathcal{G}_y(j)$:

$$\mathbf{C}(i,j) = \min_y \{\mathcal{G}_y(j); y \in \{0,\dots,n_Y-1\}\}, \tag{4.41}$$

$$\mathcal{G}_y(j) = \begin{cases} \mathbf{B}(i,y)^2 + (T_Y(j) - T_Y(y) - H_T(i,y))^2 & si\ T_Y(j) - T_Y(y) > H_T(i,y) \\ \mathbf{B}(i,y)^2 + (T_Y(y) - T_Y(j) - H_B(i,y))^2 & si\ T_Y(y) - T_Y(j) > H_B(i,y) \\ \mathbf{B}(i,y)^2 & sinon \end{cases} \tag{4.42}$$

ce qui revient à « aplatir » la parabole précédente $\mathcal{F}_y(j)$ vers les frontières de la cellule qui contient le nœud $\mathbf{B}(i,y)$ ou vers les nœuds voisins (selon la distance minimale entre les deux). Comme dans la section précédente, on peut également remarquer que ces paraboles *aplaties* sont composées de trois parties (figure 4.70) :

- ▷ une fonction constante de valeur $\mathbf{B}(i,y)^2$ au centre, bornée par les tailles du nœud,
- ▷ une branche de parabole de la forme $\mathbf{B}(i,y)^2 + (T_Y(j) - T_Y(y) - H_T(i,y))^2$ centrée au point $(\mathbf{B}(i,y)^2, T_Y(y) + H_T(i,y))$,
- ▷ une branche de parabole de la forme $\mathbf{B}(i,y)^2 + (T_Y(y) - T_Y(j) - H_B(i,y))^2$ centrée au point $(\mathbf{B}(i,y)^2, T_Y(y) - H_B(i,y))$.

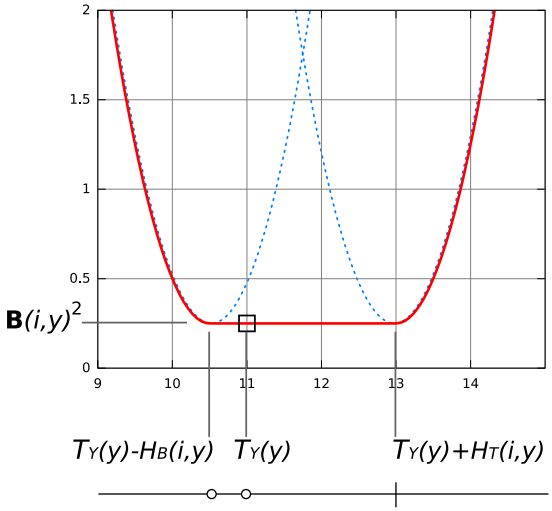


Fig. 4.70: Un exemple de parabole aplatie nécessaire au calcul de la $\mathbb{I}$-BDT dans notre approche. Le nœud centré en $y = 11$ est encadré par un nœud ($y = 10.5$) et la frontière d'une cellule avec $y = 13$. Ainsi, la parabole aplatie de cet exemple contient les paraboles d'équations $z = 0.25 + (x-11)^2$ et $z = 0.25 + (x-13)^2$ (en pointillés).

On peut enfin noter que la valeur de la distance vaut alors $\mathbf{B}(i,y)^2$ lorsque $j = y$ (à l'ordonnée du nœud $\mathbf{B}(i,y)$). Dans la figure 4.71, nous avons illustré un exemple de calcul de la $\mathbb{I}$-BDT avec notre approche. Un graphique des paraboles aplaties permet d'apprécier le comportement de l'algorithme. Notre algorithme est correct, car nous avons démontré

que l'intersection entre deux paraboles aplaties est soit un seul point, soit une infinité (lemme 4.2), comme dans le cas régulier [Hirata, 1996].

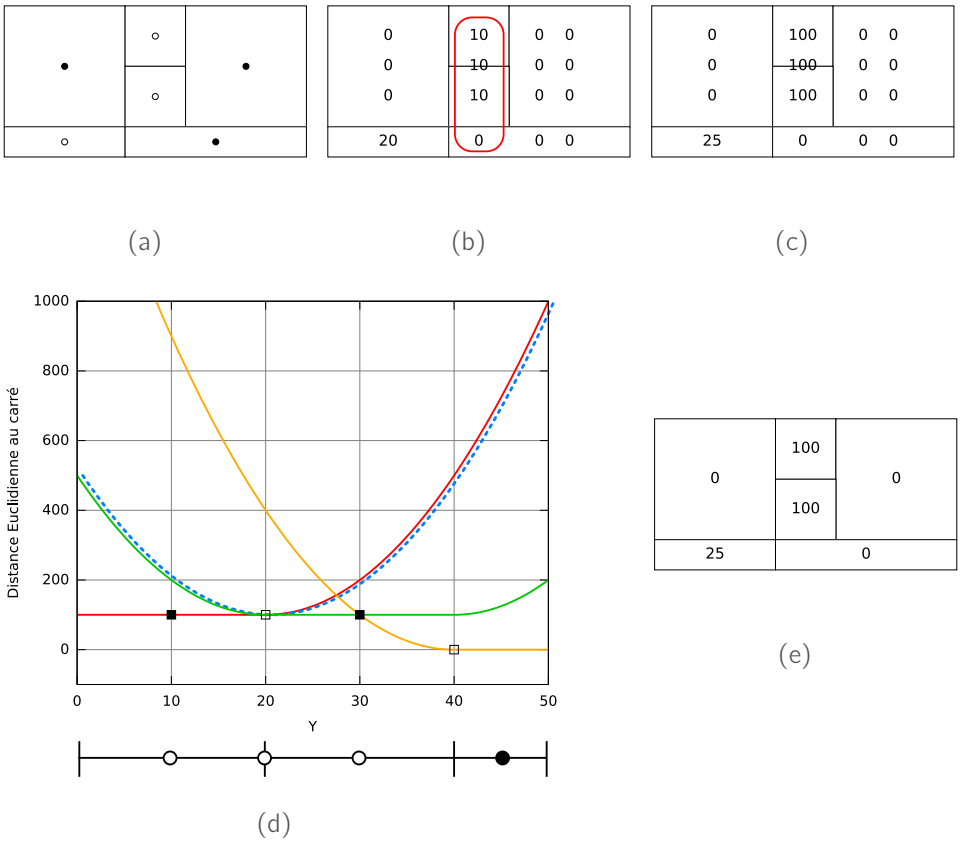


Fig. 4.71: Illustration de l'algorithme pour calculer la $\mathbb{I}$-BDT à la frontière. Comme précédemment, nous avons choisi de montrer l'ensemble des paraboles aplaties (d) calculées sur une colonne sélectionnée (b). Dans ce graphique, les carrés pleins représentent des nœuds de la matrice, et les vides des extras nœuds, qui peuvent être étiquetés comme fond. On peut noter que l'une des paraboles (pointillés) est associée à un extra nœud qui a une taille nulle dans cet axe (il coïncide avec un bord de cellule).

Pour calculer la minimisation dans l'axe des Y, nous stockons les tailles de cellules suivant Y dans deux piles h_T et h_B, initialisées avec $h_T[0] \leftarrow H_T(i,0)$ et $h_B[0] \leftarrow H_B(i,0)$. Dès que l'on met à jour les piles s et t, on modifie aussi h_T et h_B :

si $q < 0$ **alors**
  $q \leftarrow 0$; $s[q] \leftarrow j$; $h_T[q] \leftarrow H_T(i,j)$; $h_B[q] \leftarrow H_B(i,j)$;
sinon
  // Calcul de w avec $BSep()$
  // Opération de recherche dans la colonne
  $q \leftarrow q+1$; $s[q] \leftarrow k$; $t[q] \leftarrow w$; $h_T[q] \leftarrow H_T(i,j)$; $h_B[q] \leftarrow H_B(i,j)$;

Le calcul de w est réalisé grâce à la fonction $BSep()$ qui se base sur les éléments

énoncés dans notre lemme 4.2. Nous supposons ici, sans perte de généralité, que $T_Y(u) \leq T_Y(v)$ et $\mathbf{B}(i,u) \leq \mathbf{B}(i,v)$. Les autres configurations pourraient être écrites sous des formes similaires.

$$\overline{y} = T_Y(v) - H_B(i,v) - \sqrt{\mathbf{B}(i,u)^2 - \mathbf{B}(i,v)^2} \tag{4.43}$$

$$BSep(u,v) = \begin{cases} \overline{y} & si\ i) \\ CSep\big(T_Y(u) - H_B(i,u), T_Y(v) - H_B(i,v)\big) & si\ ii) \\ CSep\big(T_Y(u) + H_T(i,u), T_Y(v) - H_B(i,v)\big) & si\ iii) \end{cases} \tag{4.44}$$

$$avec \begin{cases} i) & T_Y(u) - H_B(i,u) \leq \overline{y} \leq T_Y(u) + H_T(i,u) \\ ii) & \overline{y} \leq T_Y(u) - H_B(i,u) \leq T_Y(u) + H_T(i,u) \\ iii) & T_Y(u) - H_B(i,u) \leq T_Y(u) + H_T(i,u) \leq \overline{y} \end{cases}$$

Nous présentons l'algorithme 4.15 qui regroupe l'ensemble de ces modifications pour calculer l'$\mathbb{I}$-BDT d'une $\mathbb{I}$-grille. Dans la figure 4.71 sont montrées les principales étapes de cette version de l'algorithme sur une $\mathbb{I}$-grille simple. Comme précédemment, nous détaillons l'ensemble des paraboles calculées sur une colonne de la matrice. Dans la figure 4.72, nous présentons les résultats de cette version de l'algorithme avec les $\mathbb{I}$-grilles construites avec l'image *cursor*.

Pour valider notre algorithme, nous énonçons finalement le lemme suivant :

Lemme 4.2. *L'algorithme 4.15 de calcul de la $\mathbb{I}$-BDT, par une approche séparable, est correct. Plus précisément, la phase de minimisation suivant l'axe des Y calcule effectivement l'enveloppe inférieure d'un ensemble de paraboles aplaties, aboutissant correctement à une carte de distances basées sur les bords des cellules.*

Comme l'algorithme original de E^{2}DT dans le cas régulier [Saito et Toriwaki, 1994, Hirata, 1996, Meijster *et al.*, 2000], nous calculons correctement la $\mathbb{I}$-BDT par notre approche séparable. En effet, la seconde étape de calcul de l'enveloppe inférieure d'un ensemble de paraboles est valide, car le nombre d'intersections entre deux paraboles de ce type est soit un, soit une infinité (lemme 4.2). Autrement dit, les propriétés de monotonicité et d'intersection entre les paraboles impliquent qu'il est possible de les ordonner sans erreur, et d'en déduire une carte de distances correcte.

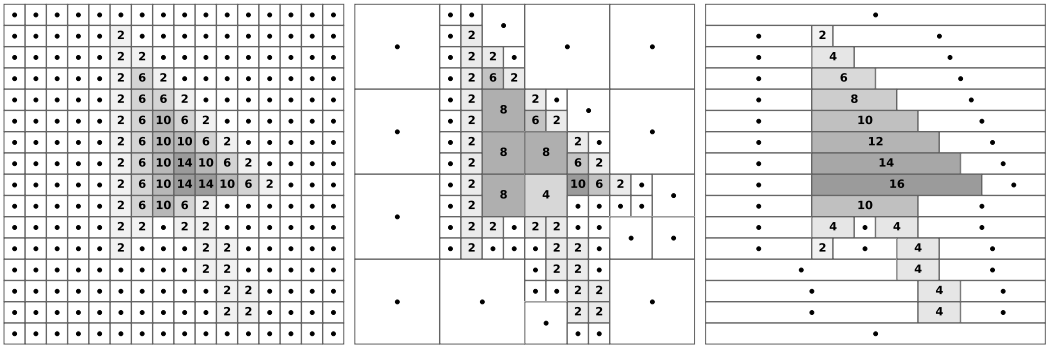
Bilan algorithmique

Rappelons que la première opération de cet algorithme (construire la matrice irrégulière) peut être réalisée en $\mathcal{O}(n_X n_Y)$ grâce à l'ordre lexicographique des cellules sur $\mathbb{I}$. L'algorithme 4.14 de calcul de la $\mathbb{I}$-CDT possède une complexité en $\mathcal{O}(n_X n_Y \log n_Y)$. Il peut être facilement étendu à des dimensions supérieures : l'étape 1 reste une initialisation, et pour chaque dimension $d > 1$, on combine les résultats obtenus dans la dimension $d-1$, comme dans l'étape 2. Si l'on considère une $\mathbb{I}$-grille étiquetée d-dimensionnelle, le coût des phases successives est en $\mathcal{O}(n^d \log^{d-1} n)$, où la taille de la matrice irrégulière $\mathbf{A}$ associée à $\mathbb{I}$ est n^d. La taille de $\mathbf{A}$ dépend de l'organisation des cellules de $\mathbb{I}$; une matrice $\mathbf{A}$ construite sur une grille régulière $\mathbb{I}$ possède la même taille que $\mathbb{I}$. Plus la grille a une

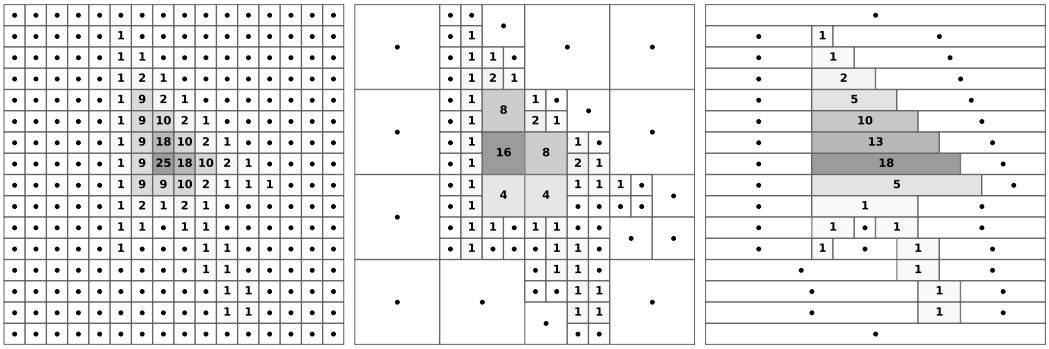
Algorithme 4.15 : Calcul de la $\mathbb{I}$-BDT par une approche séparable [Saito et Toriwaki, 1994]

entrée : une $\mathbb{I}$-grille étiquetée $\mathbb{I}$.
sortie : la $\mathbb{I}$-BDT de $\mathbb{I}$, stockée dans la matrice irrégulière $\mathbf{C}$.

début
construire la matrice irrégulière $\mathbf{A}$ associée à $\mathbb{I}$;
// Étape 1 suivant l'axe des X
pour $j = 0$ **à** $n_Y - 1$ **faire**
si $\mathbf{A}(0, j) = 0$ **alors**
$\mathbf{B}(0, j) \leftarrow 0$;
sinon
$\mathbf{B}(0, j) \leftarrow \infty$;
pour $i = 1$ **à** $n_X - 1$ **faire**
si $\mathbf{B}(i-1, j) = 0$ **alors**
$\mathbf{B}(i, j) \leftarrow T_X(i) - T_X(i-1) - H_R(i-1, j)$;
sinon
$\mathbf{B}(i, j) \leftarrow |T_X(i) - T_X(i-1)| + \mathbf{B}(i-1, j)$;
pour $i = n_X - 2$ **à** 0 **faire**
si $\mathbf{B}(i+1, j) < \mathbf{B}(i, j)$ **alors**
si $\mathbf{B}(i+1, j) = 0$ **alors**
$\mathbf{B}(i, j) \leftarrow T_X(i+1) - T_X(i) - H_L(i, j)$;
sinon
$\mathbf{B}(i, j) \leftarrow |T_X(i) - T_X(i+1)| + \mathbf{B}(i-1, j)$;
// Étape 2 suivant l'axe des Y
pour $i = 0$ **à** $n_X - 1$ **faire**
$q \leftarrow 0$; $s[0] \leftarrow 0$; $t[0] \leftarrow 0$; $h_T[0] \leftarrow H_T(i, 0)$; $h_B[0] \leftarrow H_B(i, 0)$;
pour $j = 1$ **à** $n_Y - 1$ **faire**
tant que $q \geq 0 \wedge \mathcal{G}_{s[q]}(t[q]) > \mathcal{G}_j(t[q])$ **faire**
$q \leftarrow q - 1$;
si $q < 0$ **alors**
$q \leftarrow 0$; $s[q] \leftarrow j$; $h_T[q] \leftarrow H_T(i, j)$; $h_B[q] \leftarrow H_B(i, j)$;
sinon
$w \leftarrow BSep(s[q], j)$;
si $w \leq T_Y(n_Y - 1)$ **alors**
trouver le nœud $\mathbf{B}(i, k)$, $k \in \{0, \ldots, n_Y - 1\}$ tel que $T_Y(k) > w$;
$q \leftarrow q + 1$; $s[q] \leftarrow k$; $t[q] \leftarrow w$; $h_T[q] \leftarrow H_T(i, j)$; $h_B[q] \leftarrow H_B(i, j)$;
pour $j = n_Y - 1$ **à** 0 **faire**
$\mathbf{C}(i, j) \leftarrow \mathcal{G}_{s[q]}(j)$;
si $T_Y(j) = t[q]$ **alors**
$q \leftarrow q - 1$;
fin



(a)



(b)

Fig. 4.72: Calcul de la $\mathbb{I}$-BDT basée sur la frontière par l'algorithme 4.14 sur des $\mathbb{I}$-grilles simples issues de l'image *cursor*. Comme pour la $\mathbb{I}$-CDT, nous présentons les deux étapes de notre méthode. Pour toutes les cartes, la distance au carré a été multipliée par quatre pour plus de clarté.

structure irrégulière, plus la différence entre $n_X n_Y$ et n, le nombre de cellules de $\mathbb{I}$, est importante. L'espace requis, en $\mathcal{O}(n_X n_Y)$, est principalement occupé par les matrices irrégulière **A**, **B** et **C**. De plus, lors de l'implémentation de l'algorithme, nous n'avons utilisé qu'une seule matrice qui stocke les valeurs de distance initiales et temporaires. Lorsque l'on considère la $\mathbb{I}$-BDT basée sur la frontière, la structure de données n'est pas modifiée, et les différentes étapes de l'algorithme sont très similaires à la version précédente. L'ajout des piles h_T et h_B n'engendre qu'une augmentation légère de la mémoire. Comme nous avons représenté cette minimisation par le calcul de l'enveloppe inférieure d'un ensemble de paraboles aplaties, nous avons ajouté un nombre non négligeable d'opérations nécessaires aux comparaisons de paraboles, et aux calculs d'intersections (fonction $BSep()$). Nous avons mesuré que ces opérations impliquent un accroissement du temps de calcul dépendant principalement de la taille de la grille traitée, comme nous le montrons dans la section 4.4 d'expérimentations.

Comme nous l'avons mis en évidence dans son bilan algorithmique, l'algorithme 4.14 souffre d'une opération de recherche dans sa deuxième étape qui alourdit son exécution (en $\mathcal{O}(n_X n_Y \log n_Y)$). Dans la littérature, d'autres techniques se basent sur un double balayage d'une image binaire pour calculer la E^2DT classique. Nous avons décidé d'adapter l'algorithme de R. Maurer [Maurer *et al.*, 2003] sur notre matrice irrégulière. En effet, grâce à des expérimentations poussées, R. Fabbri [Fabbri *et al.*, 2008] a montré que cette méthode peut être plus rapide que celle de [Saito et Toriwaki, 1994] pour calculer la E^2DT dans de nombreuses configurations d'images.

4.3.3 Méthode basée sur une réduction de dimension [Maurer et al., 2003]

Calcul de la $\mathbb{I}$-CDT, basée sur les centres des cellules

L'approche proposée dans [Maurer *et al.*, 2003] est une extension de celle de [Breu *et al.*, 1995] sur deux aspects : (1) elle permet de calculer la E^2DT pour des images de dimension $d \geq 2$ et (2) elle peut intégrer plusieurs distances (euclidienne, chessboard, *etc.*). Le principe de l'algorithme est de parcourir l'image régulière initiale ligne par ligne, et de construire des VD partiels minimaux (au sens du nombre de sites) pour calculer la E^2DT de cette ligne. Nous présentons ici directement la version irrégulière de cette méthode. Comme nous devons parcourir les cellules d'une $\mathbb{I}$-grille étiquetée $\mathbb{I}$ alignées suivant un axe, nous avons choisi de considérer la matrice irrégulière associée à $\mathbb{I}$, ce qui nous permet également d'étendre aisément les mesures de distance suivant un axe décrites dans [Maurer *et al.*, 2003]. En plus des propriétés des distances que nous avons énoncées dans la partie I, nous devons donner deux propriétés que respectent la plupart des distances que l'on manipule en géométrie discrète. Dans cette section, nous considérons une distance Δ, d-dimensionnelle, $\Delta : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ qui respecte les propriétés suivantes :

Propriété 4.3 (Monoticité)**.** *Soient p et q deux points de $\mathbb{R}^d$ qui diffèrent uniquement dans leur valeur à la $k^{ième}$ coordonnée (i.e. $p_i = q_i,\ i \neq k$). Par simplicité, supposons que $p_k < q_k$. Pour n'importe quels points u et v de $\mathbb{R}^d$ tels que*

$$ou \begin{cases} \Delta(p,u) \leq \Delta(p,v) \ et \ \Delta(q,v) < d(q,u) \\ \Delta(p,u) < \Delta(p,v) \ et \ \Delta(q,v) \leq d(q,u) \end{cases}$$

on a $u_k < v_k$.

Propriété 4.4. *Soient p et q deux points de $\mathbb{R}^d$ qui diffèrent uniquement dans leur valeur à la $k^{ième}$ coordonnée (i.e. $p_i = q_i,\ i \neq k$). Soient deux points u et v de $\mathbb{R}^d$ avec des valeurs identiques en leur $k^{ième}$ coordonnée (i.e. $u_k = v_k$). Si $\Delta(p,u) \leq \Delta(p,v)$, alors $\Delta(q,u) \leq \Delta(q,v)$.*

Les distances euclidiennes, chessboard, *etc.* et certaines distances de chanfrein respectent la propriété 4.3 [Maurer *et al.*, 2003]. La propriété 4.4 est sa contraposée, et sera

utile à la compréhension de l'algorithme. Nous nous concentrons ici sur la distance euclidienne en dimension 2 d_e, mais l'approche présentée ici peut être employée avec d'autres distances qui respectent cette propriété.

L'idée principale de la méthode de R. Maurer [Maurer *et al.*, 2003, Fabbri *et al.*, 2008] est que l'intersection entre le VD complet des nœuds de fond (ou sites) et une ligne de la matrice irrégulière traitée **A** peut être facilement calculé et consulté. De plus, pour une ligne j de **A**, les sites du VD sont supprimés en considérant trois remarques (nous avons donné un exemple de l'application de ces remarques dans la figure 4.73) :

- ▷ Soit un nœud p appartenant à la ligne j de **A**, et f un site de Voronoï dont la cellule intersecte la droite $y = T_Y(j)$ associée à la ligne j. Soit un autre site g sur la même colonne que f, ce qui implique que $d_e(p, f) \leq d_e(p, g)$. Alors, pour tout autre point q de la ligne j, par la propriété 4.4, on a $d_e(q, f) \leq d_e(q, g)$. Cela signifie qu'il est inutile de considérer le site g, dont la cellule associée ne peut pas intersecter la droite $y = T_Y(j)$ (figure 4.73-b).
- ▷ Si l'on considère deux points p et q sur la ligne j de **A**, et deux sites de Voronoï f et g tels que leurs cellules respectives contiennent ces points, *i.e.* $p \in \mathcal{C}_f$ et $q \in \mathcal{C}_g$. Par la propriété 4.3, si $p_x < q_x$, alors $f_x < g_x$. Comme dans notre algorithme inspiré de [Breu *et al.*, 1995], grâce à l'ordre des sites selon l'axe des abscisses, il est inutile de calculer le diagramme complet, mais seulement son intersection avec la droite $y = T_Y(j)$.
- ▷ Pour trois sites u, v et w tels que $u_x < v_x < w_x$. La cellule associée à v, $\mathcal{C}_v$, n'intersectent pas la droite $y = T_Y(j)$ si elle est cachée par u et w (figure 4.73-c). Nous utilisons ici le prédicat `caché_par()` expliqué dans la section précédente.

Dans l'algorithme 4.16, nous montrons l'adaptation de la méthode de [Maurer *et al.*, 2003] aux $\mathbb{I}$-grilles. Nous avons gardé la même structure de présentation de l'algorithme que dans ces travaux, en détaillant les fonctions `Voronoi_ICDT()` (algorithme 4.17) et `suppression_IDT()` (algorithme 4.18). La première étape de l'algorithme consiste à initialiser la distance en chaque nœud de la matrice irrégulière grâce à un parcours le long de l'axe des X. Comme dans l'algorithme inspiré de [Saito et Toriwaki, 1994], nous réalisons un double parcours linéaire pour chaque ligne. Par contre, à la fin de cette étape, chaque nœud enregistre une distance au carré (ligne 1 de l'algorithme principal 4.16), ce qui est équivalent à la DT monodimensionnelle décrite dans [de A. Lotufo et Zampirolli, 2001] [2]. Dans la seconde partie de notre méthode, nous faisons appel à la fonction `Voronoi_ICDT()` pour construire le VD partiel en intersection avec chaque colonne d'indice i. Comme dans l'algorithme original, nous utilisons deux piles g et h qui stockent les coordonnées réelles successives des sites traités (h), et leurs distances au carré (g). La première boucle de cette fonction correspond à la suppression des sites cachés, comme dans notre approche précédente inspirée de [Breu *et al.*, 1995] (algorithme 4.13). Elle fait appel au prédicat `suppression_IDT()`, qui est équivalent au prédicat `caché_par()` donné dans la section précédente (ligne 1) en projetant sur l'axe

[2] Voir la partie 5 de [Maurer *et al.*, 2003] et le code de la librairie `ANIMAL` [ANIMAL, 2008]

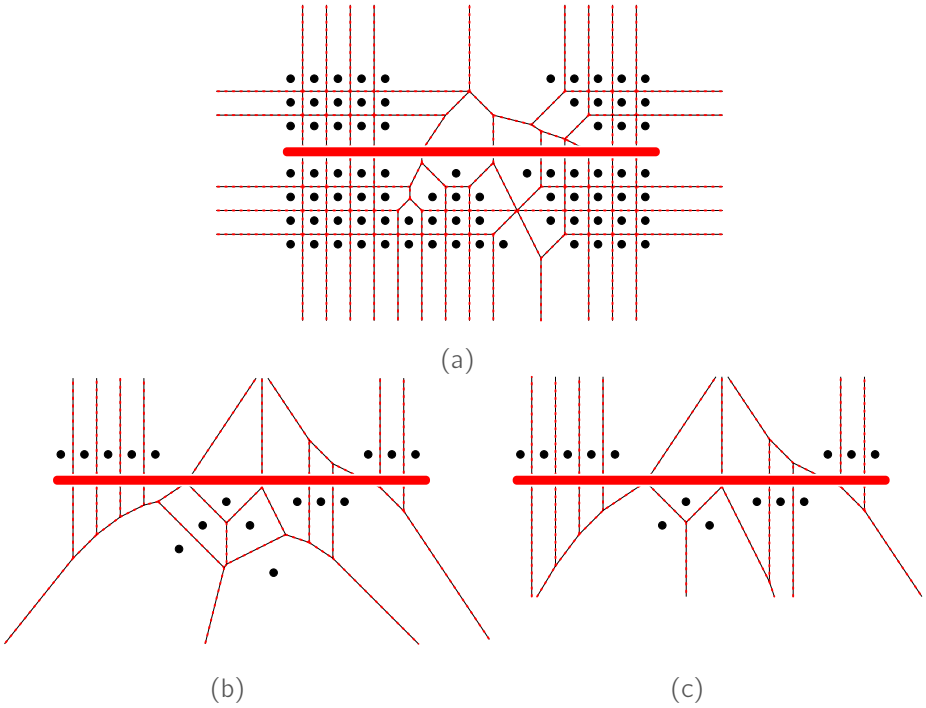


Fig. 4.73: Principe général de l'algorithme inspiré de [Maurer *et al.*, 2003] en 2-D. Lorsque l'on traite une ligne j de la matrice irrégulière, on considère le VD des cellules de fond comme dans (a). On ne garde que les cellules de Voronoï en intersection avec la ligne traitée (b). En appliquant le prédicat de suppression de sites `caché_par()`, on obtient finalement (c). Les cellules de Voronoï ainsi obtenues sont les seules qui ont une intersection avec la droite associée à la ligne j.

des Y. En effet, il suffit de développer la relation

$$c \times d_e^2(v, L) - b \times d_e^2(u, L) - a \times d_e^2(w, L) - abc > 0 \tag{4.45}$$

en considérant que la distance du point u à la droite L est en fait $d_e^2(u, L) = (u_y - r)^2$ (et de faire de même avec les points v et w). Cette inégalité est codée grâce à 11 opérations flottantes (contre 11 opérations entières dans [Maurer *et al.*, 2003]). Grâce à une structure de pile, cette étape est exécutée en temps linéaire. En testant la valeur de I (ligne 2), nous savons s'il est nécessaire ou non de parcourir à nouveau la pile pour mettre à jour les valeurs de distance. Enfin, cette dernière phase est très similaire à la fin de l'algorithme 4.13, où l'on parcourt la pile afin de trouver le site de Voronoï le plus proche du nœud courant $\mathbf{A}(i, j)$. Par conséquent, nous avons proposé un algorithme linéaire en la taille de la matrice irrégulière $\mathbf{A}$ associée à une $\mathbb{I}$-grille étiquetée $\mathbb{I}$, soit en $\mathcal{O}(n_X n_Y)$ avec les notations que nous avons adoptées. Il requiert peu de modifications par rapport à la version régulière décrite dans [Maurer *et al.*, 2003].

Algorithme 4.16 : Calcul de la $\mathbb{I}$-CDT par réduction de dimension [Maurer *et al.*, 2003]

entrée : une $\mathbb{I}$-grille étiquetée $\mathbb{I}$.
sortie : la $\mathbb{I}$-CDT de $\mathbb{I}$, stockée dans la matrice irrégulière **C**.

début
  construire la matrice irrégulière **A** associée à $\mathbb{I}$;
  // Étape 1 suivant l'axe des X
  pour $j = 0$ **à** $n_Y - 1$ **faire**
    si $\mathbf{A}(0, j) = 0$ **alors**
      $\mathbf{B}(0, j) \leftarrow 0$;
    sinon
      $\mathbf{B}(0, j) \leftarrow \infty$;
    pour $i = 1$ **à** $n_X - 1$ **faire**
      si $\mathbf{A}(i, j) = 0$ **alors**
        $\mathbf{B}(i, j) \leftarrow 0$;
      sinon
        $\mathbf{B}(i, j) \leftarrow |T_X(i) - T_X(i-1)| + \mathbf{B}(i-1, j)$;
    pour $i = n_X - 2$ **à** 0 **faire**
      si $\mathbf{B}(i+1, j) < \mathbf{B}(i, j)$ **alors**
        $\mathbf{B}(i, j) \leftarrow |T_X(i) - T_X(i+1)| + \mathbf{B}(i+1, j)$;
1     $\mathbf{B}(i, j) \leftarrow \mathbf{B}(i, j)^2$;
  // Étape 2 suivant l'axe des Y
  pour $i = 0$ **à** $n_X - 1$ **faire**
    `Voronoi_ICDT`(i);
fin

Calcul de la $\mathbb{I}$-BDT, basée sur les bords des cellules

Nous considérons ici la même matrice irrégulière que dans la $\mathbb{I}$-CDT basée sur la frontière de la section précédente. Les modifications apportées à la première phase de l'algorithme sont similaires à celles vues précédemment. On propage suivant l'axe des X la distance au carré à la frontière :

si $\mathbf{B}(i-1, j) = 0$ **alors**
  $\mathbf{B}(i, j) \leftarrow T_X(i) - T_X(i-1) - H_R(i-1, j)$;
sinon
  $\mathbf{B}(i, j) \leftarrow |T_X(i) - T_X(i-1)| + \mathbf{B}(i-1, j)$;
$\mathbf{B}(i, j) \leftarrow \mathbf{B}(i, j)^2$;

Dans la fonction `Voronoi_IBDT()`, nous utilisons deux piles supplémentaires, notées f_T et f_B, pour stocker les largeurs de cellules. Elles sont les analogues des

Algorithme 4.17 : Fonction `Voronoi_ICDT()` de construction du VD partiel suivant Y.

entrée : une colonne i de la matrice irrégulière $\mathbf{B}$.
sortie : la $\mathbb{I}$-CDT en chaque nœud de la colonne i stockée dans la matrice $\mathbf{C}$.

début
  $l \leftarrow 0$;
  $g \leftarrow \emptyset$, $h \leftarrow \emptyset$;
  pour $j = 0$ **à** $n_Y - 1$ **faire**
    si $\mathbf{B}(i,j) \neq \infty$ **alors**
1   **tant que** $l \geq 2 \wedge$ `suppression_IDT`$\big(g[l-1], g[l], \mathbf{B}(i,j), h[l-1], h[l], T_Y(j)\big)$ **faire**
        $l \leftarrow l - 1$;
      $l \leftarrow l + 1$;
      $g[l] \leftarrow \mathbf{B}(i,j)$, $h[l] \leftarrow T_Y(j)$;
2   **si** $(n_s \leftarrow l) = 0$ **alors**
      retour ;
  $l \leftarrow 1$;
  pour $j = 0$ **à** $n_Y - 1$ **faire**
    tant que $l < n_s \ \wedge \ g[l] + (h[l] - T_Y(j))^2 > g[l+1] + (h[l+1] - T_Y(j))^2$ **faire**
      $l \leftarrow l + 1$;
    $\mathbf{C}(i,j) \leftarrow g[l] + (h[l] - T_Y(j))^2$;
fin

Algorithme 4.18 : Prédicat `suppression_IDT()`.

entrée : les coordonnées en Y de trois points u, v, w de $\mathbb{R}^2$ et leurs distances au carré à la droite L d'équation $y = r$ notées $d_e^2(u, L)$, $d_e^2(v, L)$ et $d_e^2(w, L)$.
sortie : v est-il caché par u et v ?

début
  $a \leftarrow v_y - u_y$;
  $b \leftarrow w_y - v_y$;
  $c \leftarrow a + b$;
  retourner $c \times d_e^2(v, L) - b \times d_e^2(u, L) - a \times d_e^2(w, L) - abc > 0$
fin

piles h_T et h_B de notre algorithme précédent basé sur [Saito et Toriwaki, 1994]. La mise à jour de ces piles s'effectue en même temps que celles de g et h :

si $\mathbf{A}(i,j) \neq \infty$ **alors**
  `// Suppression des nœuds de fond`
  $g[l] \leftarrow \mathbf{A}(i,j)$, $h[l] \leftarrow T_Y(j)$, $f_T[l] \leftarrow H_T(i,j)$, $f_B \leftarrow H_B(i,j)$;

Avant de calculer la $\mathbb{I}$-BDT en chaque nœud de cette colonne, on la parcourt afin d'indiquer où sont les nœuds adjacents aux frontières. Nous affectons leur valeur avec la distance au carré au bord de la cellule R adjacente. Ainsi on a $\mathbf{B}(i,j) \leftarrow (L_Y(i,j-1))^2$ si les nœuds impliqués vérifient $\mathbf{A}(i,j) = 1 \wedge \mathbf{A}(i,j-1) = 0$. À la fin de cette fonction, nous changeons l'inégalité initiale en prenant en compte la parabole $\mathcal{G}_y(j)$ suivante :

$$\mathcal{G}_l(j) = \begin{cases} g[l] + (T_Y(j) - h[l] - f_T[l])^2 & si\ T_Y(j) - h[l] > f_T[l] \\ g[l] + (T_Y(y) - h[l] - f_B[l])^2 & si\ T_Y(y) - h[l] > f_B[l] \\ \mathbf{B}(i,y)^2 & sinon \end{cases} \tag{4.46}$$

Dans notre algorithme, cela se manifeste de la manière suivante :

pour $j = 0$ **à** $n_Y - 1$ **faire**
  tant que $l < n_s \ \wedge\ \mathcal{G}_l(j) > \mathcal{G}_{l+1}(j)$ **faire**
    $l \leftarrow l + 1$;
  $\mathbf{A}(i,j) \leftarrow \mathcal{G}_l(j)$;

Il paraît naturel que les modifications apportées ici soient très proches de celles

Algorithme 4.19 : Fonction `Voronoi_IBDT()` de construction du VD partiel suivant Y.

entrée : une colonne i de la matrice irrégulière $\mathbf{B}$.
sortie : la $\mathbb{I}$-BDT en chaque nœud de la colonne i stockée dans la matrice $\mathbf{C}$.

début
  $l \leftarrow 0$;
  $g \leftarrow \emptyset,\ h \leftarrow \emptyset,\ f \leftarrow \emptyset$;
  pour $j = 0$ **à** $n_Y - 1$ **faire**
    si $\mathbf{B}(i,j) \neq \infty$ **alors**
      tant que $l \geq 2 \wedge$
      `suppression_IDT`$\big(g[l-1], g[l], \mathbf{B}(i,j), h[l-1], h[l], T_Y(j)\big)$ **faire**
        $l \leftarrow l - 1$;
      $l \leftarrow l + 1$;
      $g[l] \leftarrow \mathbf{A}(i,j),\ h[l] \leftarrow T_Y(j),\ f_T[l] \leftarrow H_T(i,j),\ f_B \leftarrow H_B(i,j)$;
  si $(n_s \leftarrow l) = 0$ **alors**
    retour ;
  $l \leftarrow 1$;
  pour $j = 0$ **à** $n_Y - 1$ **faire**
    tant que $l < n_s \ \wedge\ \mathcal{G}_l(j) > \mathcal{G}_{l+1}(j)$ **faire**
      $l \leftarrow l + 1$;
    $\mathbf{A}(i,j) \leftarrow \mathcal{G}_l(j)$;
fin

que nous avons évoquées pour l'approche basée sur [Saito et Toriwaki, 1994]. En effet, les outils employés sont très semblables (structures de piles, calcul de distance par

additivité dans les deux dimensions, *etc.*). Nous avons détaillé les algorithmes nécessaires au calcul de la $\mathbb{I}$-BDT (algorithme 4.20 principal et la fonction `Voronoi_IBDT()` dans l'algorithme 4.19).

Algorithme 4.20 : Calcul de la $\mathbb{I}$-BDT par réduction de dimension [Maurer *et al.*, 2003]

entrée : une $\mathbb{I}$-grille étiquetée $\mathbb{I}$.
sortie : la $\mathbb{I}$-BDT de $\mathbb{I}$, stockée dans la matrice irrégulière $\mathbf{C}$.

début
construire la matrice irrégulière $\mathbf{A}$ associée à $\mathbb{I}$;
// Étape 1 suivant l'axe des X
pour $j = 0$ **à** $n_Y - 1$ **faire**
si $\mathbf{A}(0, j) = 0$ **alors**
$\mathbf{B}(0, j) \leftarrow 0$;
sinon
$\mathbf{B}(0, j) \leftarrow \infty$;
pour $i = 1$ **à** $n_X - 1$ **faire**
si $\mathbf{A}(i, j) = 0$ **alors**
$\mathbf{B}(i, j) \leftarrow 0$;
sinon
si $\mathbf{B}(i-1, j) = 0$ **alors**
$\mathbf{B}(i, j) \leftarrow T_X(i) - T_X(i-1) - H_R(i-1, j)$;
sinon
$\mathbf{B}(i, j) \leftarrow T_X(i) - T_X(i-1) + \mathbf{B}(i-1, j)$;
pour $i = n_X - 2$ **à** 0 **faire**
si $\mathbf{B}(i+1, j) < \mathbf{B}(i, j)$ **alors**
si $\mathbf{B}(i+1, j) = 0$ **alors**
$\mathbf{B}(i, j) \leftarrow T_X(i+1) - T_X(i) - H_L(i, j)$;
sinon
$\mathbf{B}(i, j) \leftarrow T_X(i+1) - T_X(i) + \mathbf{B}(i-1, j)$;
$\mathbf{B}(i, j) \leftarrow \mathbf{B}(i, j)^2$;
// Étape 2 suivant l'axe des Y
pour $i = 0$ **à** $n_X - 1$ **faire**
`Voronoi_IBDT`(i);
fin

Comme dans notre précédente approche séparable inspirée de [Saito et Toriwaki, 1994] (algorithme 4.14), cette technique de calcul de la $\mathbb{I}$-DT est facilement extensible aux dimensions supérieures. En effet, la phase 1 reste une initialisation de la distance au carré en chaque nœud de la matrice d-dimensionelle $\mathbf{A}$. Ensuite, pour chaque dimension

supérieure $d > 1$, on calcule le VD partiel associé, pour finalement mettre à jour chaque nœud grâce à un parcours des cellules de Voronoï ainsi construites. Dans ce cas, notre approche possède donc une complexité en $\mathcal{O}(n^d)$, où n^d est la dimension de la matrice $\mathbf{A}$. Par conséquent, nous avons proposé un algorithme linéaire en la taille de $\mathbf{A}$, contrairement à l'algorithme 4.14, qui souffre d'une complexité globale en $\mathcal{O}(n^d \log^{d-1} n)$. Enfin, nous avons mesuré une augmentation du temps d'exécution modérée lors du calcul de la $\mathbb{I}$-BDT à la frontière (voir la section suivante pour plus de détails).

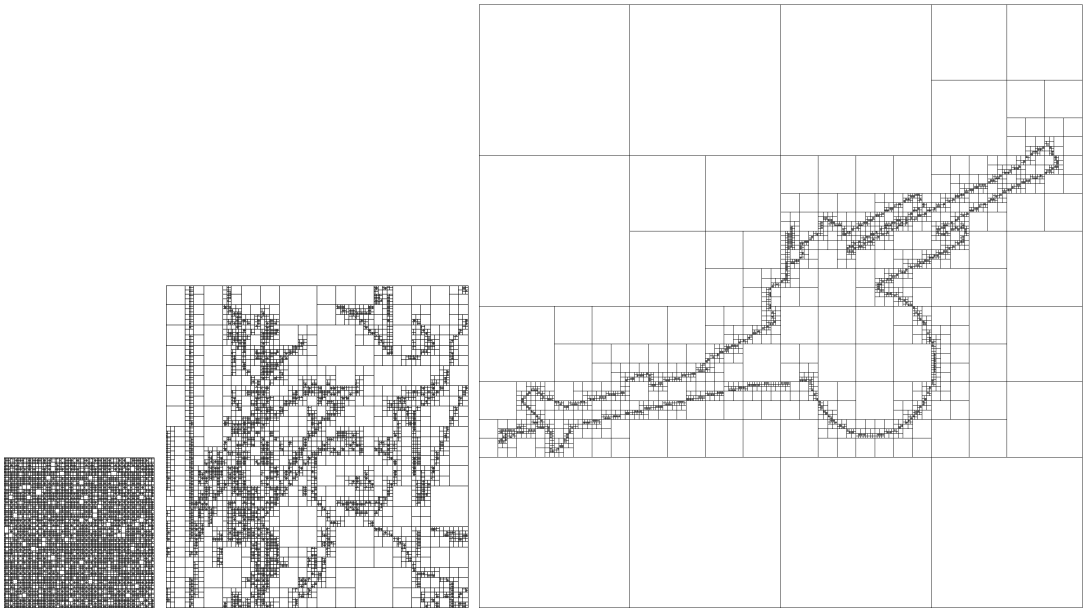
4.4 Expérimentations et comparaisons entre toutes les approches proposées

4.4.1 Tests préliminaires de performance sur des $\mathbb{I}$-grilles créées par des outils classiques en image

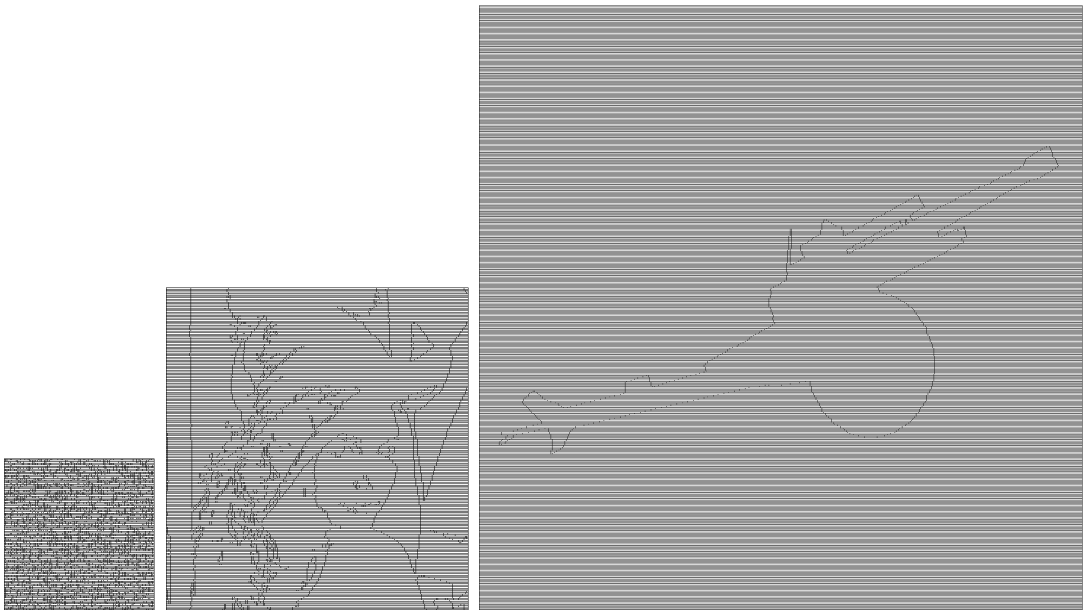
Dans un premier temps, nous avons choisi de tester ces algorithmes sur des $\mathbb{I}$-grilles communes dans les domaines liées à l'image. Dans la figure 4.74, nous proposons des grilles créées à partir de trois images binaires différentes (voir figure 4.75) : *canon* est une grande image ne contenant qu'un seul objet binaire, *lena* une image classique plus complexe et finalement une image générée par un bruit gaussien, nommée *noise*. Dans la figure 4.76,

(a) (b) (c)

Fig. 4.74: Les images que nous avons choisies pour nos expérimentations : (a) *noise*, (b) *lena*, (c) *canon*.



(a)



(b)

Fig. 4.75: Les $\mathbb{I}$-grilles de test en entrée des algorithmes de $\mathbb{I}$-DT. A partir des trois images *noise*, *lena*, et *canon*, on construit des grilles basées sur un quadtree (a) et un RLE suivant l'axe des X (b).

nous donnons les cartes de distances $\mathbb{I}$-CDT obtenues par nos algorithmes, exécutés sur ces $\mathbb{I}$-grilles associées aux images de test. Le niveau de gris c de la cellule correspond à la valeur de la distance d par une simple opération de modulo ($c = d \mod 255$). Nous présentons dans la table 4.5 les temps d'exécution des quatre approches que nous avons proposées pour calculer la $\mathbb{I}$-CDT sur les $\mathbb{I}$-grilles de test, et dans la table 4.6 les temps d'exécution pour calculer la $\mathbb{I}$-BDT par les approches d-dimensionnelles. Nous avons résumé dans la table 4.3 les caractéristiques importantes pour chaque $\mathbb{I}$-grille (*e.g.* le nombre de cellules de fond ou de premier plan, la taille de la matrice irrégulière, *etc.*). Les tests ont été réalisés avec un processeur Intel Pentium M© cadencé à 1.5 Ghz, et 1 Go de RAM. Dans la table 4.4, nous rappelons également les complexités en temps et en espace de chaque méthode, en fonction du nombre de cellules de départ.

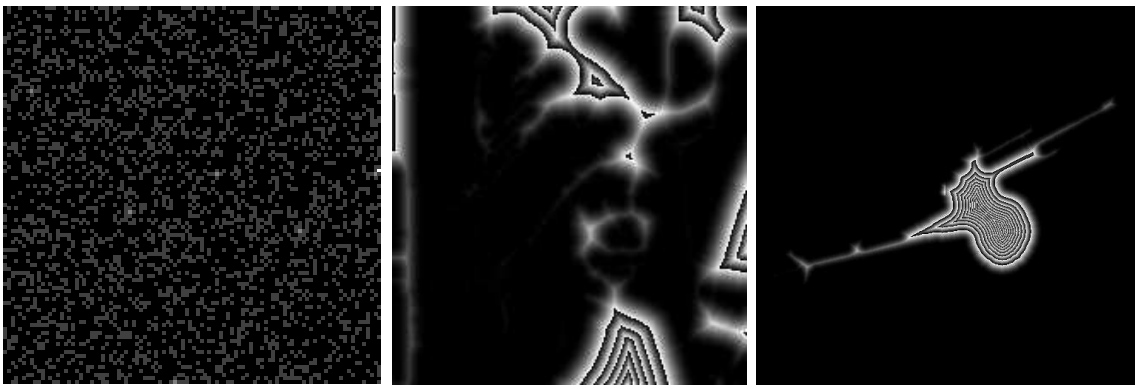
Image	Taille	$\mathbb{I}$-grille	n	n_B	n_F	$n_X \times n_Y$	$\lvert n_X \times n_Y \rvert$	$\frac{n}{\lvert n_X \times n_Y \rvert}$
noise	100×100	$\mathbb{D}$	10 000	7 414	2 586	100×100	10 000	1.000
		$q\mathbb{T}$	7 457	5 003	2 454	138×138	19 044	0.390
		$\mathbb{L}$	3 903	1 973	1 930	199×100	19 900	0.200
lena	208×222	$\mathbb{D}$	46 176	18 789	27 387	208×222	46 176	1.000
		$q\mathbb{T}$	8 171	4 143	4 028	308×365	112 420	0.070
		$\mathbb{L}$	3 462	1 695	1 767	392×222	87 024	0.040
canon	512×512	$\mathbb{D}$	262 144	234 302	27 842	512×512	226 144	1.000
		$q\mathbb{T}$	4 177	2 162	2 015	920×527	484 840	0.010
		$\mathbb{L}$	1 432	972	460	577×512	295 424	0.005

Tab. 4.3: Caractéristiques des $\mathbb{I}$-grilles testées. Pour chaque $\mathbb{I}$-grille est indiquée la taille de l'image binaire, le nombre de cellules (total, de fond et premier plan), et la taille de la matrice irrégulière associée.

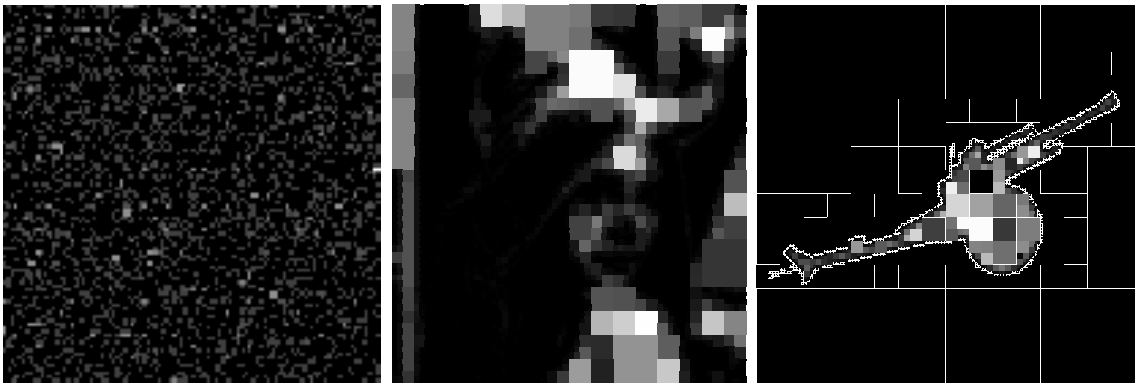
Id.	Algorithme	Temps	Espace	d-D	$\mathbb{I}$-CDT	$\mathbb{I}$-BDT
1	VD complet	$\mathcal{O}(n \log n_B)$	$\mathcal{O}(n)$		✓	
2	Basé sur [Breu *et al.*, 1995]	$\mathcal{O}(n)$	$\mathcal{O}(n)$		✓	
3	Basé sur [Saito et Toriwaki, 1994]	$\mathcal{O}(n_X n_Y \log n_Y)$	$\mathcal{O}(n_X n_Y)$	✓	✓	✓
4	Basé sur [Maurer *et al.*, 2003]	$\mathcal{O}(n_X n_Y)$	$\mathcal{O}(n_X n_Y)$	✓	✓	✓

Tab. 4.4: Complexité en temps et en espace pour chaque approche que nous avons proposée. n_B désigne le nombre de cellules de fond dans la $\mathbb{I}$-grille $\mathbb{I}$, n est le nombre total de cellules de $\mathbb{I}$, enfin n_Y et n_Y représentent les dimensions de la matrice irrégulière associée à $\mathbb{I}$.

On peut remarquer en premier lieu que la complexité linéaire de l'algorithme 2 est vérifiée par nos expérimentations. En effet, il s'avère être le plus rapide pour toutes les $\mathbb{I}$-grilles testées. Par contre, il n'est pas extensible en d-D. Dans ce cas, nous devons nous référer aux temps réalisés par les autres approches. Dans la table 4.5, nous avons marqué en bleu les deuxièmes meilleurs temps. On peut noter que pour les $\mathbb{I}$-grilles denses,



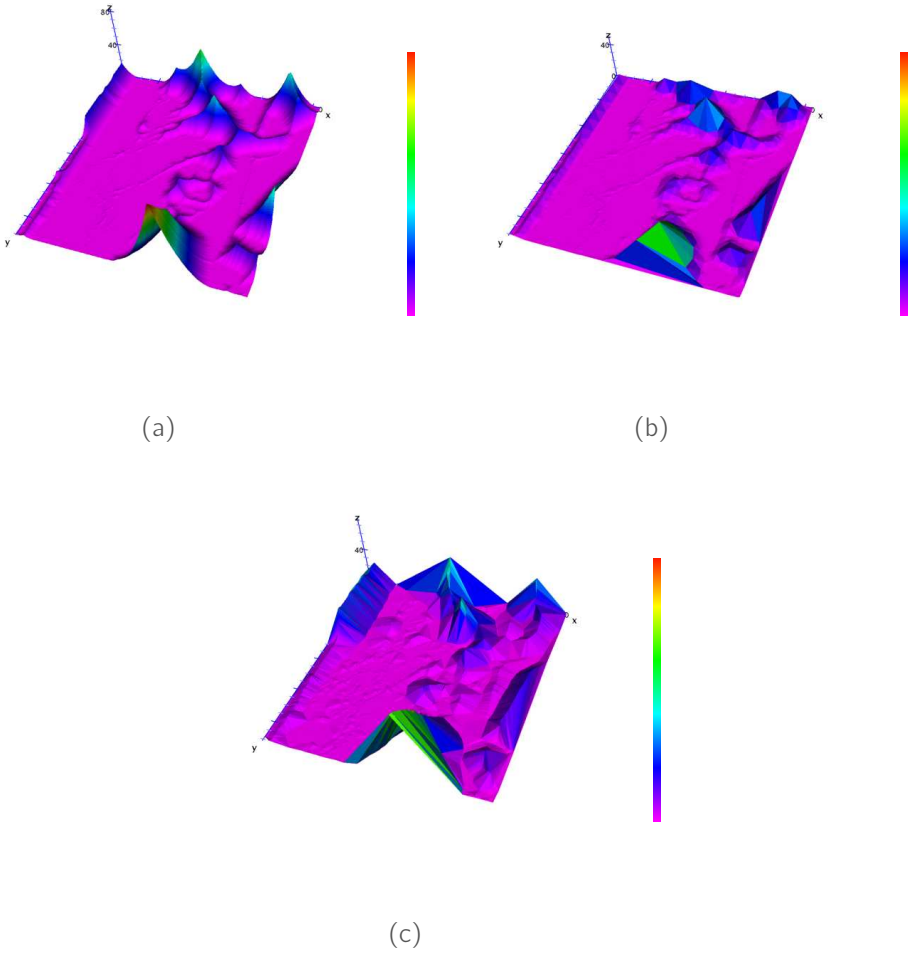
(a)



(b)

(c)

Fig. 4.76: Cartes de distance $\mathbb{I}$-CDT obtenues pour les $\mathbb{I}$-grilles de test : grille régulière (a), quadtree (b), et RLE suivant X (c). La couleur c de chaque cellule est obtenue à partir de la valeur de distance d par un modulo : $c = d \mod 255$.



(a) (b)

(c)

Fig. 4.77: Cartes de distances $\mathbb{I}$-CDT représentées en 3-D pour les $\mathbb{I}$-grilles issues de l'image *lena* : régulière en (a), quadtree en (b) et RLE en (c). La palette de couleurs choisie, illustrée à droite, est définie de 0 à 1400 environ.

l'algorithme 4 basé sur [Maurer *et al.*, 2003] est globalement plus rapide que la méthode 1 (qui calcule le VD complet des sites de fond), et légèrement plus rapide que l'approche séparable inspirée de [Saito et Toriwaki, 1994]. Au contraire, la méthode 1 affiche des temps plus courts, pour les $\mathbb{I}$-grilles éparses, que les méthodes qui sont supportées par une matrice irrégulière. Pour résumer, l'approche inspirée de [Breu *et al.*, 1995] semble la plus recommandée pour calculer la $\mathbb{I}$-CDT en 2-D. Lorsque l'on veut passer aux dimensions supérieures, plusieurs options sont possibles en fonction des contraintes imposées par l'application finale de nos outils. Si l'on veut utiliser peu de mémoire, ou que l'on sait *a priori* que les $\mathbb{I}$-grilles traitées seront très éparses, on s'orienterait vers la technique basée sur le VD complet. Dans tous les autres cas de figures, l'approche basée sur [Maurer *et al.*, 2003] est la plus polyvalente et serait plus adaptée. En ce qui concerne le calcul de la $\mathbb{I}$-BDT par nos méthodes d-D, la table 4.6 nous confirme que l'algorithme 4 est globalement

Image	Algorithme 1 VD complet		
	$\mathbb{D}$	$q\mathbb{T}$	$\mathbb{L}$
noise	0.255	0.104	0.053
lena	1.413	0.145	0.081
canon	36.236	0.273	0.234

Image	Algorithme 2 [Breu *et al.*, 1995]		
	$\mathbb{D}$	$q\mathbb{T}$	$\mathbb{L}$
noise	0.035	0.057	0.025
lena	0.118	0.097	0.053
canon	0.734	0.272	0.208

Image	Algorithme 3 [Saito et Toriwaki, 1994]		
	$\mathbb{D}$	$q\mathbb{T}$	$\mathbb{L}$
noise	0.037	0.077	0.044
lena	0.192	0.376	0.245
canon	1.678	1.322	1.316

Image	Algorithme 4 [Maurer *et al.*, 2003]		
	$\mathbb{D}$	$q\mathbb{T}$	$\mathbb{L}$
noise	0.046	0.065	0.038
lena	0.185	0.166	0.135
canon	1.134	0.485	0.585

Tab. 4.5: Le temps d'exécution en secondes de la $\mathbb{I}$-CDT pour les quatre approches, et pour chaque $\mathbb{I}$-grille de test. En rouge sont illustrés les meilleurs temps et en bleu les deuxièmes meilleurs temps.

Image	Algorithme 3 [Saito et Toriwaki, 1994]		
	$\mathbb{D}$	$q\mathbb{T}$	$\mathbb{L}$
noise	0.047 (27)	0.100 (29)	0.054 (23)
lena	0.256 (33)	0.491 (31)	0.320 (31)
canon	2.248 (34)	2.107 (59)	2.020 (53)

Image	Algorithme 4 [Maurer *et al.*, 2003]		
	$\mathbb{D}$	$q\mathbb{T}$	$\mathbb{L}$
noise	0.065 (42)	0.085 (31)	0.049 (29)
lena	0.258 (40)	0.209 (26)	0.170 (26)
canon	1.507 (33)	0.563 (16)	0.718 (23)

Tab. 4.6: Le temps d'exécution en secondes de la $\mathbb{I}$-BDT pour les algorithmes séparables, et pour chaque $\mathbb{I}$-grille de test. Entre parenthèses est indiqué l'augmentation en % par rapport au calcul de la $\mathbb{I}$-CDT.

plus rapide que l'approche inspirée de [Saito et Toriwaki, 1994] (et lorsqu'il est plus lent, la différence entre les temps des deux approches est très faible). Lorsque l'on étudie le taux d'augmentation du temps de calcul des deux algorithmes, on peut remarquer que l'algorithme 3 est moins pénalisé dans le cas des $\mathbb{I}$-grilles denses (entre 23 et 33%). Dans

les autres cas, l'accroissement du temps d'exécution peut être très important (jusqu'à près de 60%) et augmente lorsque le rapport $\frac{n}{|n_X \times n_Y|}$ diminue. On peut expliquer ce phénomène par le fait que la complexité initiale de cet algorithme n'est pas linéaire en la taille de la matrice irrégulière, et les opérations supplémentaires nécessaires au calcul de la $\mathbb{I}$-BDT empirent ce défaut. A l'inverse, l'approche inspirée de [Maurer *et al.*, 2003] voit son taux d'augmentation diminuer quand le rapport $\frac{n}{|n_X \times n_Y|}$ décroît. Comme cet algorithme est linéaire en $n_X \times n_Y$, le calcul de la $\mathbb{I}$-BDT n'implique qu'un surcroît de temps de calcul maîtrisé (de l'ordre de 25 à 45% environ dans la majorité de nos expérimentations).

Dans cette première section, nous avons mis en évidence plusieurs résultats sur les méthodes que nous avons développées pour calculer la $\mathbb{I}$-DT sur des $\mathbb{I}$-grilles issues de méthodologies classiques en image. Tout d'abord, l'approche inspirée de [Breu *et al.*, 1995] est la plus rapide pour calculer la $\mathbb{I}$-CDT en 2-D. Pour travailler en dimensions supérieures, on peut sans aucun doute s'orienter vers l'algorithme basé sur [Maurer *et al.*, 2003], qui est globalement le plus rapide. De plus, il est le plus adapté pour calculer la $\mathbb{I}$-BDT. Dans la section suivante, nous cherchons à confirmer ces phénomènes, en utilisant des $\mathbb{I}$-grilles générées de manière aléatoire. Ces tests nous permettrons de tester également la robustesse de nos méthodes.

4.4.2 Tests de performance et de robustesse grâce à une génération aléatoire de $\mathbb{I}$-grilles binaires

Pour tester la robustesse de nos algorithmes, et apporter une comparaison complémentaire de leur temps d'exécution, nous avons choisi de réaliser des tests basés sur des $\mathbb{I}$-grilles synthétiques. Ces grilles sont générées grâce à un algorithme basé sur la construction de kd-tree que nous avons décrite dans le chapitre 1, partie I. La création des cellules filles R_1 et R_2 au sein de R est régie par deux probabilités p_{valeur} et $p_{feuille}$, et le placement de l'axe de découpe est contraint par un facteur de perturbation $0 \leq t_{coupe} \leq 1$. Le modèle que nous proposons consiste à effectuer de nombreux tirages de Bernouilli suivant les éléments que nous venons de décrire. L'objectif de ce processus est de générer des $\mathbb{I}$-grilles aléatoirement très différentes respectant p_{valeur}, $p_{feuille}$, et t_{coupe}. Plus précisément, lorsque l'on choisit un nombre de niveaux maximum l_{max} dans l'arbre, voici la description de ces trois éléments :

- ▷ Une cellule fille R' a une probabilité p_{valeur} d'avoir la même valeur que la cellule mère R (fond ou premier plan). A $p_{valeur} \to 0$, cela signifie que les cellules filles ont peu de chance d'avoir la même valeur que la cellule mère. On note par `génère_valeur`(R', R) la fonction d'attribution de la valeur de R' suivant cette probabilité, et la valeur de R.
- ▷ Les deux cellules filles ont une probabilité $p_{feuille}$ de ne pas être créées. Cela implique que si $p_{feuille} \to 0$, alors la hauteur de l'arbre de récursivité tend vers l_{max}. Le prédicat `feuille`(R) permet de vérifier si la cellule R sera feuille ou non suivant $p_{feuille}$.
- ▷ Le placement de l'axe de découpe suivant X (un raisonnement similaire peut être

fait suivant Y) est choisi dans l'intervalle $[x - \frac{t_{coupe} \times l_R^x}{2}; x + \frac{t_{coupe} \times l_R^x}{2}]$. Ainsi, plus t_{coupe} est grand, et plus le placement de l'axe est variable dans la cellule mère R. La procédure `génère_plan`(R_1, R_2, R, a) résume la création de l'axe de découpe paramétré (suivant l'axe $a = X$ ou Y) par t_{coupe} dans la cellule R, et la mise en place des cellules filles R_1 et R_2 au sein de celle-ci.

Pour résumer, on peut obtenir une image binaire régulière et bruitée comme *noise* en fixant $p_{valeur} = 0.5$, $p_{feuille} = 0.0$ et $t_{coupe} = 0.0$. L'algorithme 4.21 est une adaptation de la construction de kd-tree pour notre génération aléatoire de $\mathbb{I}$-grilles. Il prend en compte les paramètres et les fonctions associées que nous venons de décrire.

Algorithme 4.21 : Procédure récursive `génère_cellules()` de génération aléatoire d'une $\mathbb{I}$-grille.

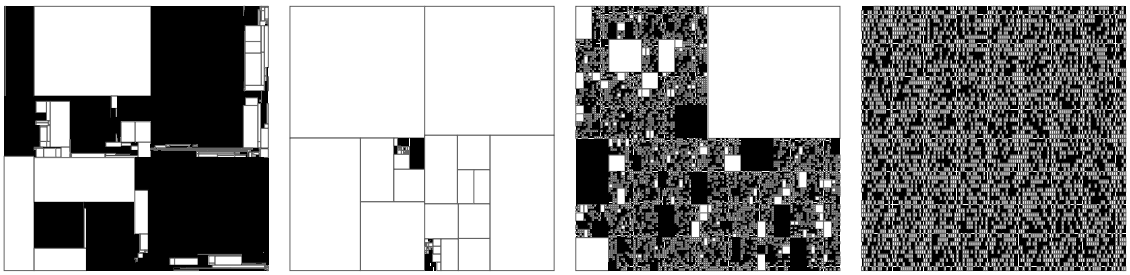
entrée : une cellule R de la $\mathbb{I}$-grille, de niveau l et l'axe de découpe actuel a
les probabilités $p_{feuille}$ et p_{valeur} et le paramètre t_{coupe} sont fixés
sortie : la cellule R est éventuellement subdivisée et on appelle `génère_cellules` sur ses filles

```
début
    // Test de fin de génération pour R suivant p_feuille
    si ¬feuille(R) et l < l_max alors
        cellule R_1 ← R;
        cellule R_2 ← R;
        // Attribution de la valeur des cellules filles suivant p_valeur
        génère_valeur(R_1, R);
        génère_valeur(R_2, R);
        si a = X alors
            // Création des cellules filles suivant le paramètre t_coupe
            génère_plan(R_1, R_2, R, a);
            prochain axe de subdivision a' ← Y;
        sinon si a = Y alors // idem que pour X
        génère_cellules(R_1,l + 1,a');
        génère_cellules(R_2,l + 1,a');
fin
```

Le protocole de tests que nous avons entrepris se base sur plusieurs configurations de paramètres nous permettant de générer des $\mathbb{I}$-grilles diverses. La liste de ces configurations est récapitulée dans le tableau de la figure 4.78. Pour chaque configuration de $\mathbb{I}$-grille, nous avons généré 1000 grilles de niveaux $l_{max} = 5, 6, \ldots, 14, 15$, puis nous avons exécuté et chronométré nos quatre approches de calcul de $\mathbb{I}$-CDT pour chacune d'entre elles. Enfin, pour chaque niveau l_{max} et pour chaque méthode i, le temps moyen $\mu_i(l_{max})$ et l'écart type $\sigma_i(l_{max})$ sont calculés. Dans les figures 4.78, nous avons illustré des $\mathbb{I}$-grilles générées suivant les configurations que nous avons choisies. Pour chaque configuration et pour chaque méthode, nous donnons également le graphique résumant les temps moyens

par la courbe $y_i(l_{max}) = \mu_i(l_{max})$ dans la figure 4.79. Nous repérons aussi la courbe $y_i(l_{max}) = \mu_i(l_{max}) - \sqrt{\sigma_i(l_{max})/1000}$, en tirets, qui représente les écarts-types.

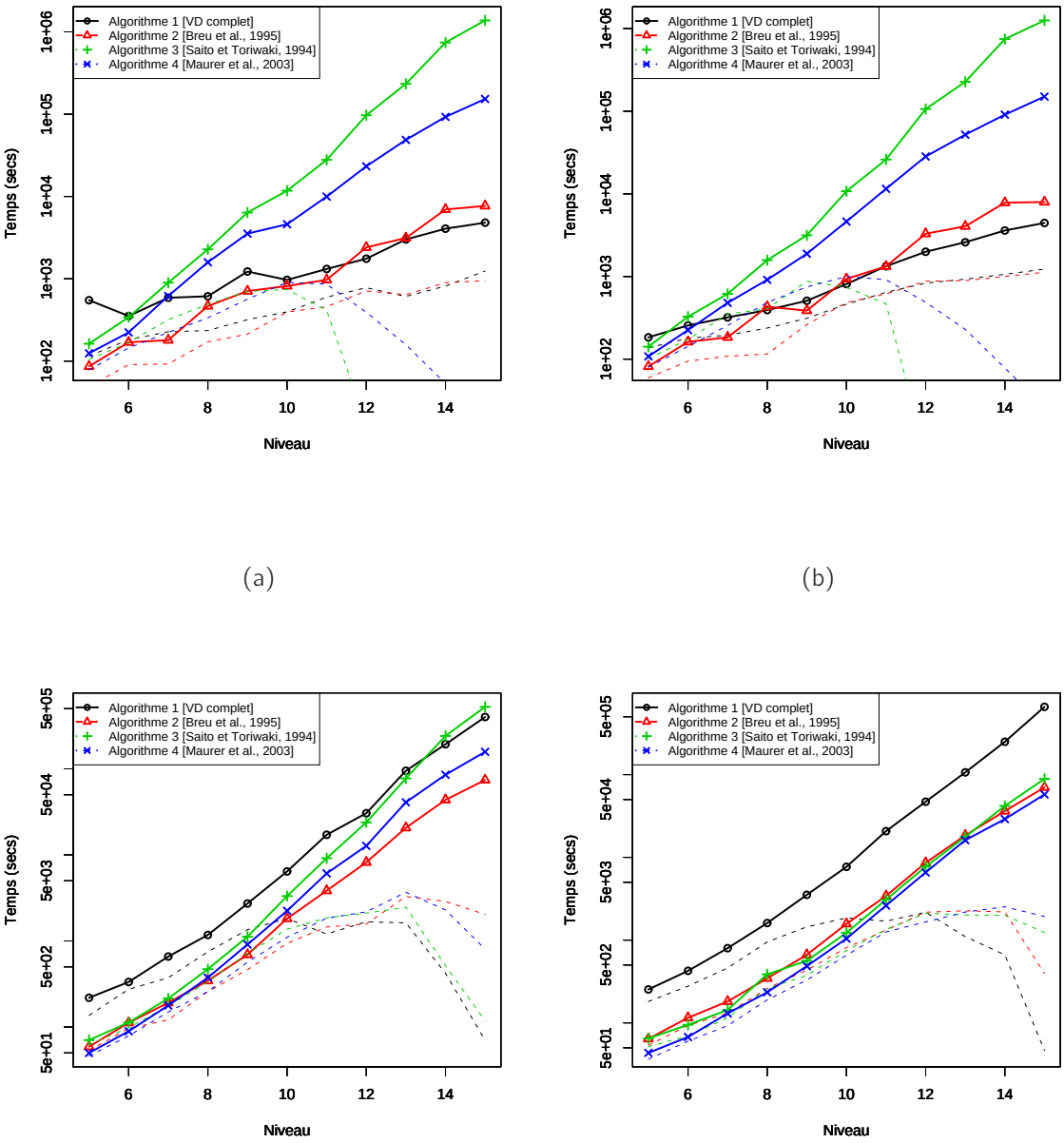
Id.	p_{valeur}	$p_{feuille}$	t_{coupe}
1	0.05	0.30	1.00
2	0.90	0.10	0.10
3	0.25	0.05	0.00
4	0.50	0.00	0.00



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

Fig. 4.78: $\mathbb{I}$-grilles générées suivant les paramètres que nous avons choisis. Pour chaque configuration (voir table en haut), une $\mathbb{I}$-grille de niveau élevé est représentée.

Nous pouvons noter que les configurations 1 et 2 impliquent des comportements similaires pour chaque méthode testée. En particulier, les approches 1 et 2 sont les plus rapides et affichent des variations de temps de calculs équivalentes (courbe des écarts-types en tirets). L'algorithme basé sur [Breu *et al.*, 1995] est légèrement plus lent que l'approche basée sur le VD complet lorsque l_{max} est élevé. Pour le niveau 15, il faut compter en moyenne 0.08 seconde pour le premier algorithme contre 0.05 pour le second (soit 1.6 fois plus lent environ). Cela peut s'expliquer par le fait que les $\mathbb{I}$-grilles sont très chaotiques et que la structure de listes nécessite un premier tri des cellules suivant leurs centres (ce qui n'est pas nécessaire pour l'algorithme 1). Les deux autres approches affichent des temps et des écarts-types bien plus importants : pour le même niveau et en moyenne, l'algorithme 3 dure 1.4 secondes et l'algorithme 4 dure 0.15 secondes. L'ensemble de ces commentaires peuvent être remarqués dans la majorité des jeux de paramètres encadrés par les configurations 1 et 2 : p_{valeur} quelconque, $p_{feuille} > 0$, et $t_{coupe} > 0$. Les $\mathbb{I}$-grilles générées par les paramètres de la configuration 3 sont des kd-trees où l'axe de découpe est médian à la cellule. Dans ce cadre, l'algorithme 2 est le plus performant, et l'algorithme 4 affiche les deuxièmes meilleurs temps. Dans le hauts niveaux de hiérarchies, l'algorithme basé sur [Saito et Toriwaki, 1994] est le plus lent de tous. Pour le niveau $l_{max} = 15$, les temps moyens dans l'ordre croissant sont de 0.07, 0.16, 0.4, et 0.5 secondes. Enfin, la dernière configuration est une génération d'images régulières (niveaux pairs) et anisotropes (niveaux impairs) de résolution croissante. On peut remarquer que l'algorithme basé sur le VD complet est largement le moins rapide (au niveau le plus élevé,



(a) (b)

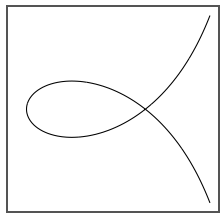
(c) (d)

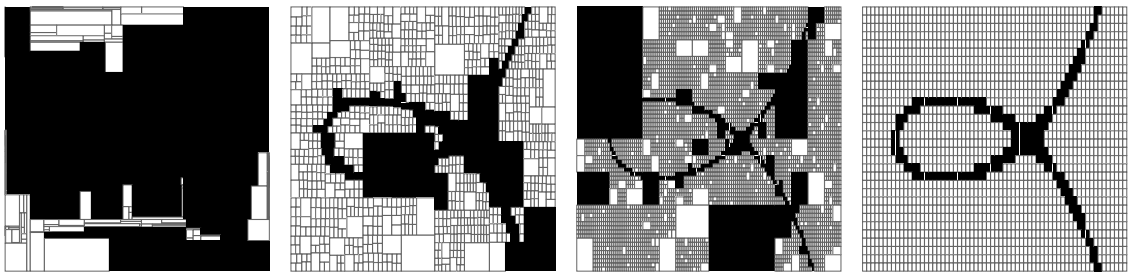
Fig. 4.79: Résultats avec les $\mathbb{I}$-grilles générées suivant les paramètres que nous avons choisis. Pour chaque configuration (voir table de la figure 4.78), le graphique résumant les temps moyens (trait plein) et l'écart-type (tirets) de chaque approche est tracé.

0.7 secondes), tandis que les trois autres ont des comportements similaires (pour le même niveau, environ 0.08 secondes). Parmi eux, l'algorithme 4 se détache, d'abord parce qu'il est le plus rapide, ensuite parce qu'il est le plus stable dans ses résultats (écart-type le plus faible des trois). Nous avons donc des résultats équivalents à ceux énoncés par R. Fabbri [Fabbri *et al.*, 2008] pour des images binaires régulières, et nous avons également montré que l'algorithme de R. Maurer est le plus performant pour les grilles anisotropes.

Nous proposons maintenant un protocole où l'on discrétise un objet réel sur la $\mathbb{I}$-grille générée. Nous considérons alors uniquement les paramètres $p_{feuille}$ et t_{coupe} énoncés précédemment. En effet, dans nos précédentes expérimentations, nous n'avons pas noté de changement notable de temps d'exécution des algorithmes lorsque p_{valeur} varie. Ainsi, nous avons choisi de vérifier le comportement de nos contributions en considérant des valeurs de cellules représentant un objet réel. Nous avons choisi de discrétiser la courbe implicite $\mathcal{C}$ donnée par son équation $f(x,y) = O$ avec $f(x,y) = -\frac{169}{64} + \frac{51}{8}x - 11x^2 + 8x^3 + 9y - 8xy - 9y^2 + 8xy^2$ définie sur le domaine $\Omega = [0,1] \times [0,1]$ (issue de [Martin *et al.*, 2002]). Grâce à l'arithmétique d'intervalles (voir chapitre 7, partie III), nous pouvons savoir si une cellule 2-D dans le domaine Ω est intersectée ou non par la courbe $\mathcal{C}$. Le schéma global de nos expérimentations est le même que le précédent : nous générons 1000 $\mathbb{I}$-grilles suivant les mêmes configurations (en ignorant p_{valeur}), et la valeur de chaque cellule est attribuée suivant son intersection avec la courbe $\mathcal{C}$. Nous itérons cette opération pour des niveaux de générations s'étalant de 5 à 15. Les configurations choisies et une $\mathbb{I}$-grille de haut niveau ,associée à chacune d'entre elles, sont rappelées dans la figure 4.80. Dans la figure 4.81 sont illustrées , pour chaque configuration, les courbes de temps moyens et d'écarts-types. De ces nouvelles expérimentations, nous pouvons tout d'abord noter que

Id.	$p_{feuille}$	t_{coupe}
1	0.30	1.00
2	0.10	0.10
3	0.05	0.00
4	0.00	0.00





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

Fig. 4.80: $\mathbb{I}$-grilles générées, représentant une courbe réelle (voir en haut à droite). Pour chaque configuration (voir table en haut), une $\mathbb{I}$-grille de niveau élevé est représentée.

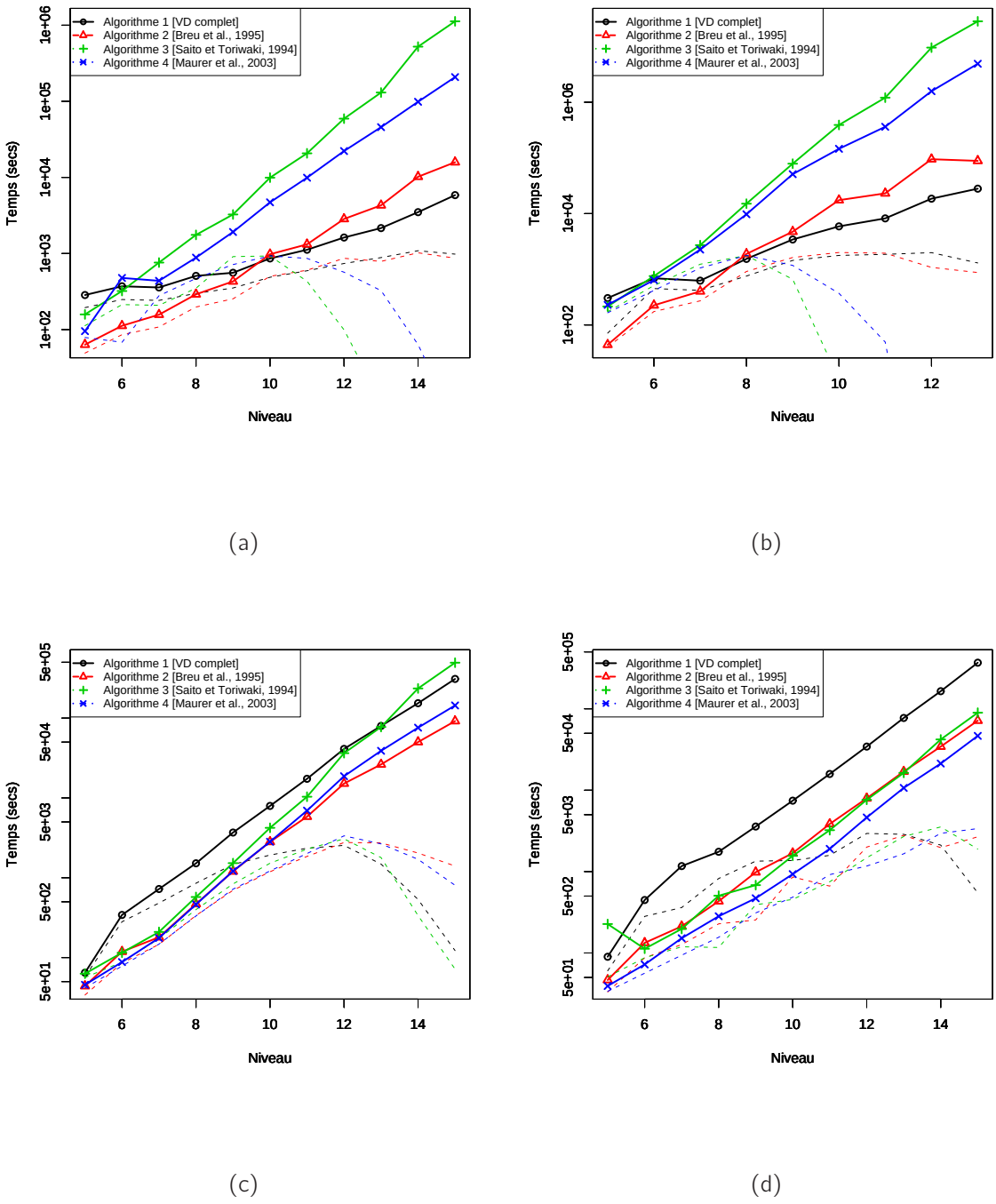


Fig. 4.81: Résultats avec les $\mathbb{I}$-grilles générées, représentant une courbe réelle. Pour chaque configuration (voir table de la figure 4.80), le graphique résumant les temps moyens (trait plein) et l'écart-type (tirets) de chaque approche est tracé à droite pour chaque configuration.

les configurations 1 et 2 impliquent le même comportement des différents algorithmes

que précédemment : les méthodes 1 et 2 restent les plus compétitives, et la technique issue de [Breu *et al.*, 1995] est plus lente avec un facteur d'environ 1.7 lorsque $l_{max} = 15$. Il en est de même pour la configuration 3, où cette dernière se détache nettement des trois autres, et l'approche de R. Maurer arrive en seconde place. Enfin, pour le dernier jeu de paramètre, cette méthode se révèle plus efficace que dans les expérimentations antérieures. Elle est environ 1.6 fois plus rapide que les algorithmes 2 et 3 dans le dernier niveau de génération.

4.5 Bilan et perspectives

Nous avons présenté dans ce chapitre des algorithmes pour calculer la $\mathbb{I}$-DT, généralisation de la E^2DT sur $\mathbb{I}$-grilles. La définition de la $\mathbb{I}$-DT est dérivée suivant deux modèles de distance sur $\mathbb{I}$-grilles : l'un prend en compte le centre des cellules de fond ($\mathbb{I}$-CDT) et l'autre ($\mathbb{I}$-BDT) considère les bords des cellules qui sont à l'interface premier plan / fond. Nous avons proposé en premier lieu d'étudier le cas bidimensionnel avec l'algorithme basé sur le VD complet des cellules de fond de la grille, et l'extension de la méthode décrite dans [Breu *et al.*, 1995]. La première ne suppose aucune structuration des cellules de la grille, tandis que la seconde suppose un ordre lexicographique des cellules (et se base donc sur notre structure de listes). Ces approches n'étant ni extensibles à d dimensions ni aisément implémentable pour la $\mathbb{I}$-BDT, nous avons décrit ensuite deux techniques de calcul de la $\mathbb{I}$-CDT et la $\mathbb{I}$-BDT basées sur des travaux classiques de transformée en distance [Saito et Toriwaki, 1994, Maurer *et al.*, 2003]. Comme la distance entre deux cellules peut être définie de deux manières différentes, nous avons exposé le calcul de la $\mathbb{I}$-DT suivant ces deux versions. Comme nous l'avons expliqué dans le cadre de l'algorithme de [Saito et Toriwaki, 1994], nous ne pouvons plus employer une structure de listes pour développer ces approches séparables. Nous avons donc implémenté la matrice irrégulière, qui permet de parcourir une $\mathbb{I}$-grille suivant une première dimension, puis de fusionner les résultats obtenus pendant le parcours de la seconde, pour obtenir la $\mathbb{I}$-DT finale. Grâce aux différentes expérimentations que nous avons menées, nous avons montré que l'algorithme 2 inspiré des travaux de H. Breu est le plus adapté pour le cadre 2-D. Pour le calcul en d-D, l'approche inspirée de l'algorithme de R. Maurer est la plus polyvalente, mais on peut aussi considérer l'approche basée sur le VD complet si les grilles traitées sont très éparses.

Comme nous l'avons évoqué lors de la présentation de l'approche décrite dans [Maurer *et al.*, 2003], différentes métriques peuvent être utilisées dans le calcul d'une DT, *e.g.* les distances chessboard, de chanfrein, *etc.* Ainsi nous pourrions envisager d'autres métriques dans nos modèles de $\mathbb{I}$-DT. Nous voudrions implémenter ces techniques dans le cadre tridimensionnel. Il serait intéressant de montrer notamment qu'elles peuvent être efficaces dans le cas des grilles les plus étudiées en 3-D, à savoir les grilles anisotropes, très répandues en imagerie médicale et les grilles FCC et BCC. La technique illustrée dans [Park *et al.*, 2005] pourrait être une approche originale pour calculer la carte de distance d'un ensemble de points dans l'espace d-D. Il serait également inté-

ressant de comparer nos contributions avec les FMM [Sethian, 2001], qui paraissent être une excellente alternative pour approximer la $\mathbb{I}$-DT rapidement.

Une application naturelle de la DT est de calculer l'axe médian, représentant dans un objet un ensemble minimal de points les plus éloignés de son bord. Nous souhaiterions développer des outils adaptés aux $\mathbb{I}$-grilles se basant sur la $\mathbb{I}$-DT pour calculer un axe médian d'un objet irrégulier. La principale problématique est de proposer une représentation *réversible*, *i.e.* qui permette de retrouver la forme initiale. La notion de boule ouverte sur $\mathbb{I}$-grille que nous avons décrite dans la partie I pourrait être un support de la construction d'un axe médian réversible sur $\mathbb{I}$-grilles. Enfin, nous avons aussi mis en évidence l'application de nos travaux dans le cadre des SIG, et nous aimerions travailler également sur la planification de trajectoires, où les $\mathbb{I}$-grilles sont fréquemment utilisées pour calculer un plus court chemin dans un environnement 3-D par exemple.

Troisième partie

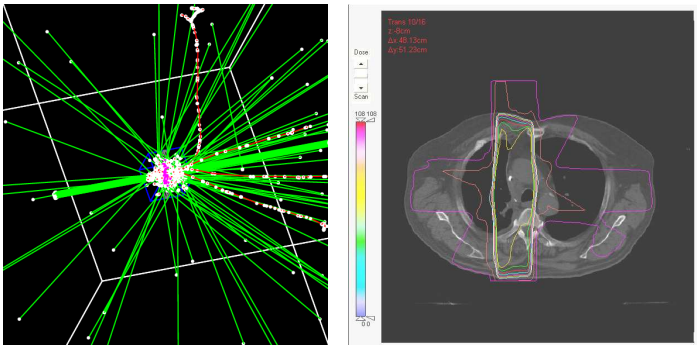
Applications des grilles irrégulières et des outils discrets irréguliers en imagerie

SOMMAIRE DE LA PARTIE

CHAPITRE 5

Accélération des simulations de radiothérapie grâce à une modélisation par RLE

La radiothérapie est une technique de traitement du cancer dont l'objectif consiste à délivrer des rayons ionisants de haute energie sur une zone cancéreuse afin de la stériliser, tout en préservant au maximum les tissus sains. Dans ce type de traitement, la planification requiert une simulation précise des séances d'irradiation. Dans un premier temps, un physicien médical contoure la tumeur, puis intègre la région sélectionnée avec le positionnement du patient et les paramètres du faisceau (energie, position, largeur, *etc.*) dans le simulateur. Les simulations permettent ensuite d'observer quels sont les tissus (sains ou non) atteints par les rayons incidents. Pour corriger la balistique calculée, on modifie les paramètres d'irradiation (positionnement et faisceau) pour viser au mieux la tumeur. Les codes de simulation dits de Monte Carlo (utilisés dans des domaines divers, tel que suivi en vision par ordinateur [Vacavant et Chateau, 2005] ou le recalage d'images 3-D [Ma et Ellis, 2004]) permettent de simuler la trajectoire des particules sortant de l'accélérateur en décrivant pas à pas leurs caractéristiques (energie, vitesse, masse, *etc.*), et l'energie qu'elles déposent éventuellement dans les tissus du patient. La modélisation physique est très poussée et permet de simuler avec précision les interactions des particules avec leur environnement. L'inconvénient majeur de ces codes est de prendre un temps important pour exécuter des simulations complexes. `Geant4`, réalisé par un consortium d'équipes internationales (notamment le CERN à Genève, Suisse), est une bibliothèque `C++` de simulations de hautes energies qui est facilement adaptable à des applications complexes en radioprotection, en calorimétrie, et en physique médicale. Ce code de simulation est actuellement un des plus puissants disponibles pour la communauté scientifique, car il modélise la plupart des connaissances en physique des particules. L'objectif de ces travaux a été de modéliser un patient, décrit initialement par un scan CT au format DICOM, en une image 3-D transmise ensuite à `Geant4`. L'image est décrite par des voxels (de dimen-



(a) (b)

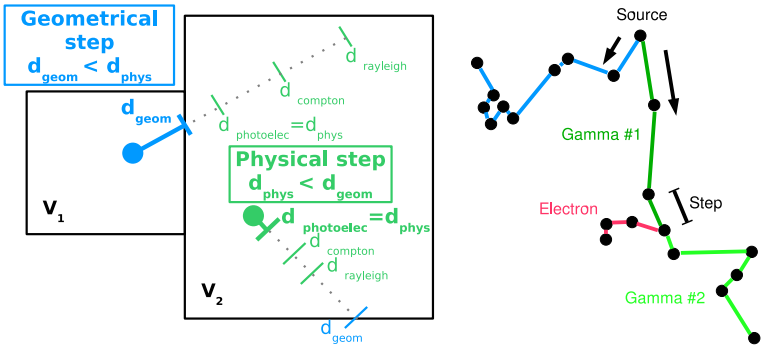
Fig. 5.82: La modélisation d'une simulation de radiothérapie grâce à la librairie `Geant4` dans une scène 3-D et la dose déposée pendant la séance représentée sur un scan CT, à droite.

sion $2 \times 2 \times 2$ mm^3 par exemple), au sein desquels on veut connaître le dépôt d'energie. Cependant, le nombre de voxels est très important (dans la même résolution, environ neuf millions de voxels) et ralentit le calcul des trajectoires des particules au sein du patient. De plus, l'occupation mémoire est lourde. Mon travail a donc consisté à proposer des méthodes d'une part pour adapter le code pour qu'il prenne en compte des images de plusieurs millions de voxels, d'autre part pour accélérer le temps de calcul (pour plus de détails sur ces recherches, se référer à [Vacavant, 2005]).

5.1 Les simulations de Monte Carlo

La simulation des particules γ (photon gamma) est étudiée pas à pas. Ainsi le parcours de la particule en un laps de temps dt est appelé *step*. A l'issue de chaque *step*, on cherche à connaître le comportement de la particule et ses caractéristiques (figure 5.83). L'algorithme de simulation peut se décomposer ainsi [Agostinelli et al., 2003] :

1. Au départ d'un *step*, on détermine quel est le processus physique qui va intervenir (photo-électrique, Compton, production de paire, *etc.*). Pour ce faire :
   - ▷ On se réfère d'abord aux caractéristiques de la particule. On calcule alors pour chaque processus le NMFP (*number of mean free path*) grâce à une loi exponentielle d'interaction. C'est ici que l'on génère des nombres aléatoires pour connaître les NMFP.
   - ▷ Suivant le matériau dans lequel se situe la particule (eau, air, etc.), on convertit chacun de ces NMFP en longueurs physiques (PL). La PL représente la distance que parcourt la particule après le processus associé. On sélectionne alors celui de plus petite PL, notée $\min(PL)$.



(a) (b)

Fig. 5.83: En (a), la différence entre un step géométrique (bloqué par la modélisation de la scène) et un step physique où un processus physique peut se produire. En (b) est illustré un schéma global de la simulation temps par temps du trajet des particules.

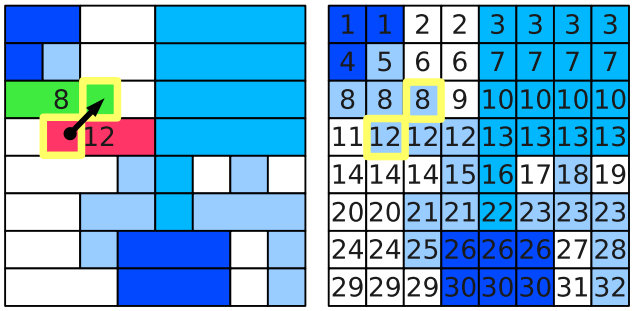
2. (*Geometrical step*) Si la particule appartient à un objet de la scène et que la distance au bord de cet objet (GL) est inférieure à $min(PL)$, alors on ne considère pas le phénomène choisi précédemment. On transporte la particule au bord de l'objet, puis on passe à l'étape 1.
3. (*Physical step*) Sinon, on applique le processus et on transporte la particule sur la distance $\min(PL)$ déterminée à l'étape 1.
4. Si la particule est toujours en vie, on boucle à l'étape 1.
5. Sinon, on arrête.

Il faut remarquer que l'étape 1 de cet algorithme est le coeur de la simulation physique, où l'on détermine le processus qui doit intervenir grâce à des nombres aléatoires. L'étape 2 est très dépendante de la structure géométrique de la scène ; la modélisation du patient choisie influence donc considérablement cette phase de l'algorithme. Comme nous l'avons présenté dans la partie I de ce manuscrit (simulation de fluides dans une structure d'octree [Losasso *et al.*, 2004]), la principale problématique est de simuler le trajet des particules dans une représentation irrégulière de la scène.

5.2 Modélisation du patient dans une simulation de Monte Carlo

La simulation de radiothérapie par des codes de Monte Carlo, et plus particulièrement `Geant4`, est un domaine de recherche récent sur lequel on dénombre encore peu de parutions d'articles. Certaines études se concentrent sur la segmentation de l'image en classes de matériaux (os, graisse, *etc.*) afin de la représenter précisément ensuite dans un simulateur [Schneider *et al.*, 2000, Rodrigues et al., 2004, Verhaegen et Devic, 2005]. D'autres

décrivent la modélisation 3-D ou 4-D (3-D et la dimension temporelle) sur laquelle repose la simulation [Paganetti, 2004, Jiang et Paganetti, 2004]. Dans cette étude, nous avons choisi de segmenter l'image 3-D représentant le patient par seuillage simple. Ainsi, on passe des 3000 valeurs possibles du scan CT initial à une dizaine de valeurs, comme dans la plupart des recherches équivalentes. Pour accélérer les simulations, nous avons ensuite choisi de coder notre image 3-D grâce à un RLE. Cet encodage est régulièrement employé pour simplifier sensiblement la géométrie d'une scène complexe dans les simulations [Cai et Sakas, 2000, Losasso *et al.*, 2006]. Soit G la matrice 3-D en entrée, et $\overline{G}$ la grille finalement codée, suivant l'axe des X (il est clair qu'un raisonnement similaire serait présenté pour un autre axe d'encodage). Par analogie à [Coeurjolly, 2005], on peut affirmer que les cellules de $\overline{G}$ ont pour coordonnées $(x, y = \lambda j, z = \mu k)$ et pour taille $(l^X, l^Y = \lambda, l^Z = \mu)$, $j, k \in \mathbb{Z}$. Dans ces formules, λ et μ sont les dimensions des voxels en entrée, appartenant à G. Comme le codage est effectué suivant l'axe des X, seule la composante l^X peut changer. On a donc l^Y et l^Z constants égaux à λ et μ respectivement. De même, les coordonnées y et z sont alignées sur la grille G, et seule la composante en X varie. Pour décrire le fonctionnement de la navigation dans `Geant4`, nous gardons



(a) (b)

Fig. 5.84: La grille régulière de départ G est conservée comme table de référence (b) pour parcourir la grille irrégulière $\overline{G}$ (a) quand une particule doit changer de volume. G permet également d'enregistrer le dépôt de dose.

en mémoire la grille initiale G comme d'une *table de référence* (voir aussi la figure 5.84). Lors de la recherche de la prochaine cellule (x, j, k) visitée par une particule p dans $\overline{G}$, on se réfère à G pour connaître le voxel (i, j, k) tel que p visite (i, j, k). On cherche d'abord le voxel qui contient actuellement p, puis on parcourt ses six voisins (comme dans la partie précédente) pour connaître le voxel (i, j, k) vers lequel se dirige la particule. Enfin, on en déduit la cellule de $\overline{G}$ qui contient le voxel (i, j, k) grâce à la fonction f_i. De même, pour déposer l'energie d'une particule p, nous considérons la grille G, en cherchant le voxel (i, j, k) où réside p. En effet, le calcul de dose doit rester précis (résolution de G), malgré l'usage de la grille irrégulière.

5.3 Expérimentations et résultats

Ces expérimentations ont pour but de comparer la méthode que nous avons développé avec la références de simulations de Monte Carlo en radiothérapie [Jiang et Paganetti, 2004] :

- ▷ Méthode 1 (GR). Cette approche permet une navigation optimisée dans les voxels [Jiang et Paganetti, 2004]. Plus exactement, on passe d'un voxel à l'un de ses voisins en $\mathcal{O}(1)$. Il peut sembler étrange que cette recherche dans le voisinage d'une matrice 3-D ne soit pas déjà prise en compte dans `Geant4`. Mais la plupart des simulations en physique des particules n'impliquent pas la gestion d'une structure aussi complexe que celle d'un patient décrit par un examen scanner de plusieurs millions de voxels.
- ▷ Méthode 2 (GI). Dans cette version l'algorithme de simulation, on peut prendre en compte n'importe grille irrégulière. Nous avons choisi le RLE pour faciliter le transport des particules, mais d'autres représentations peuvent être envisagées comme l'octree (voi chapitre suivant pour plus de détails).

Nous proposons ici d'exposer quelques résultats issus de simulations effectuées grâce à des images créées par échantillonnages d'une image de thorax d'un patient, $\mathtt{I}_2$ (voir figure 5.85 et table 5.7). L'objectif est de comparer les deux approches GR (de référence) et notre contribution, GI pour des images de résolution différentes encodées avec un RLE. Dans la table 5.7 est représenté tout d'abord le nombre de cellules encodées lors d'un RLE suivant l'axe Z pour chacune des cinq images de test ($\mathtt{I}_2$ et quatre images ré-échantillonnées à partir d'elle). Le rapport entre le nombre de voxels de l'image initiale et le nombre de cellules est également illustré. Chaque simulation a été lancée sur un Intel Pentium © P4

Nom	Résolution en mm^3	Nombre de voxels	Cellules RLE 4 matériaux	Cellules RLE 10 matériaux
$\mathtt{I}_1$	$1 \times 1 \times 1$	33 207	621 (53.5)	1 808 (13.4)
$\mathtt{I}_2$	$0.94 \times 0.94 \times 5.0$	7 548	625 (12.3)	1 322 (05.7)
$\mathtt{I}_3$	$2 \times 2 \times 2$	4 137	161 (25.7)	408 (10.1)
$\mathtt{I}_4$	$5 \times 5 \times 5$	266	24 (11.1)	49 (05.4)
$\mathtt{I}_5$	$10 \times 10 \times 10$	33	5 (06.6)	8 (04.1)

Tab. 5.7: Nombre d'éléments dans une grille encodée par un RLE. Le rapport entre le nombre de cellules et le nombre de voxels initial est indiqué entre parenthèses en %. Les nombres de voxels et de cellules sont en milliers, et le RLE est codé suivant l'axe des Z.

2.8 GHz, avec 2 Go de RAM, et elle génère un million de particules. Dans la figure 5.86, nous présentons d'abord les temps de simulation pour chacune des méthodes, avec les cinq images test. J'expose en parallèle le nombre de steps géométriques et physiques comptabilisés pendant la simulation pour chaque méthode.

De ces expérimentations, différentes remarques peuvent être énumérées. Tout d'abord, l'utilisation de grilles irrégulières permet de réduire sensiblement le temps de simulation,

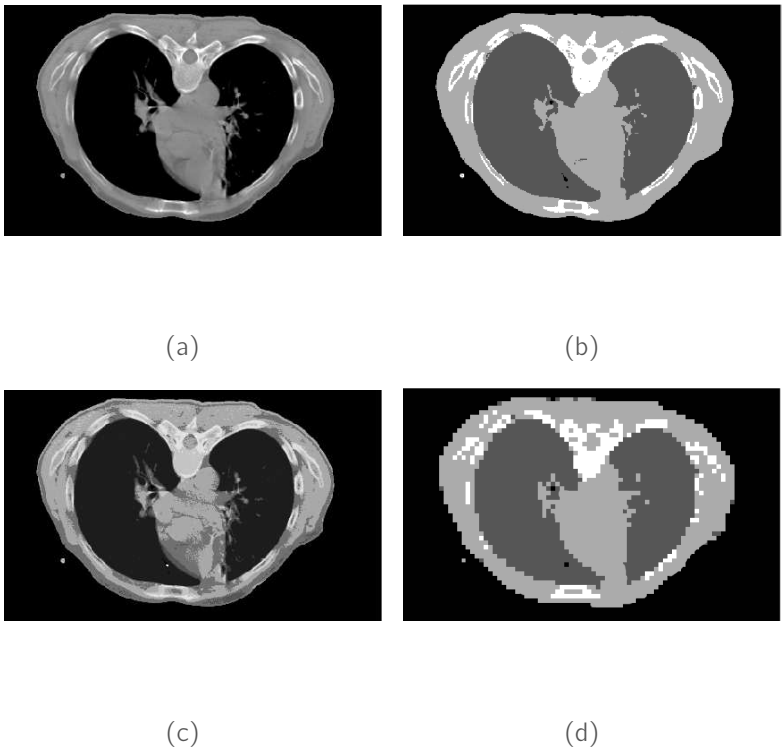
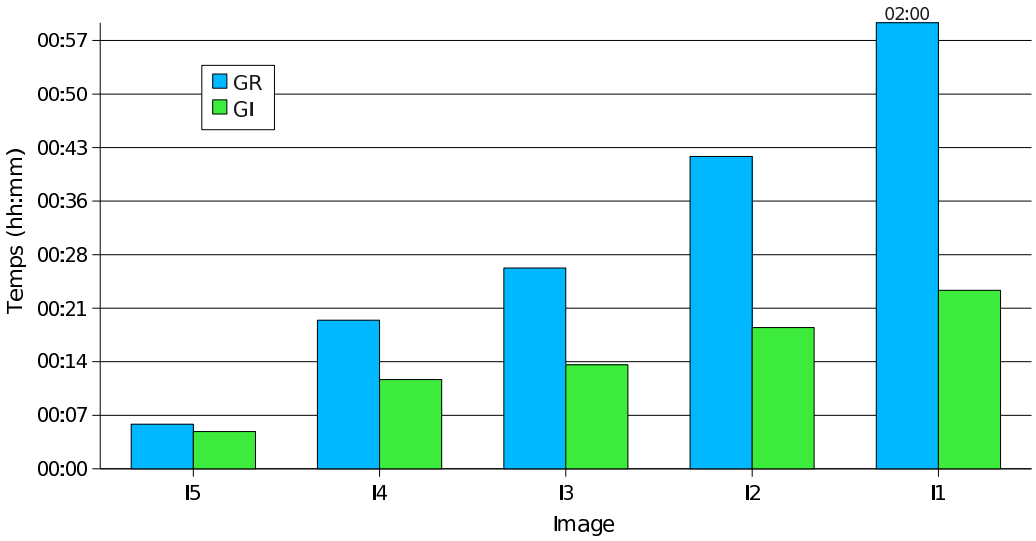


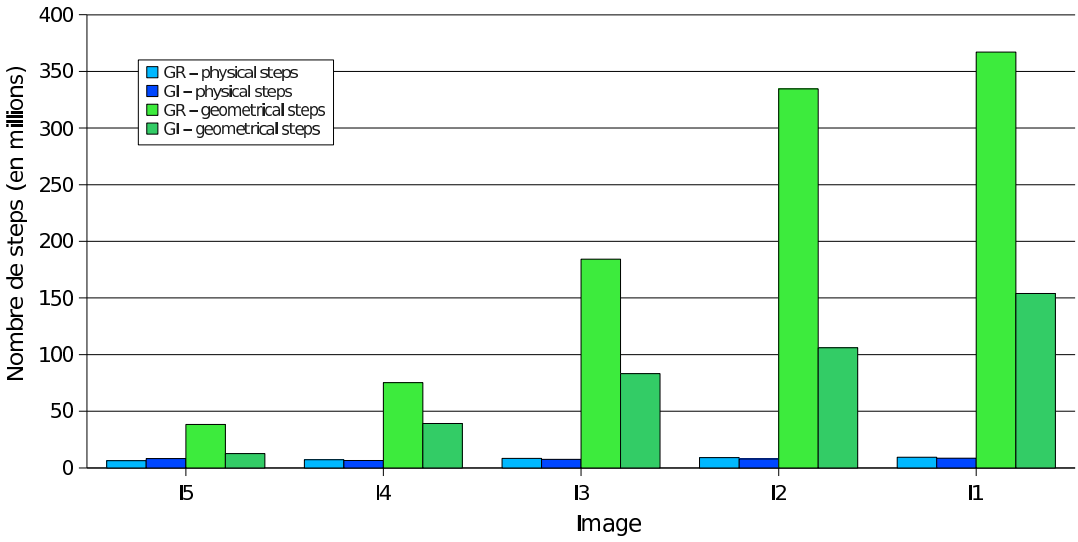
Fig. 5.85: (a) : une coupe de l'image originale du thorax d'un patient ($\mathtt{I}_2$). (b) : la même coupe étiquetée avec quatre matériaux différents. L'air est en noir, les poumons en gris foncé, les tissus mous en gris clair, et les os en blanc. (c) : la même coupe étiquetée avec dix matériaux. Enfin, (d) est un ré-échantillonnage de l'image (c) avec une résolution inférieure ($5 \times 5 \times 5$ mm^3).

de deux heures à 25 minutes avec 30 millions de voxels (et 1 million de cellules). Comme on pouvait le prévoir, cela est dû à la diminution du nombre de volumes dans la scène : le nombre de steps géométriques décroît tandis que le nombre de steps physiques reste constant. De plus, nous avons calculé que l'erreur moyenne (basée sur une somme moyenne des différences absolues) de notre contribution par rapport à la méthode de référence GR est de 3.8% sur toute la carte de dose, si l'on considère l'image $\mathtt{I}_2$ avec une segmentation en quatre matériaux. Pour valider complètement un tel résultat, il est clair qu'une comparaison avec la dose déposée lors d'une séance d'irradiation *réelle* serait nécessaire.

A la suite de ces travaux menés avec David Sarrut, d'autres études ont été menées pour améliorer et valider les simulations de Monte Carlo par `Geant4`. Dans [Sarrut et Guigues, 2008], les auteurs proposent d'abord de segmenter l'image grâce à un octree, puis de la représenter par des régions de formes arbitraires. Cela permet de réduire le temps de simulation d'un rapport 15, tout en garantissant une erreur inférieure à 1% par rapport aux méthodes classiques. On peut également lire dans [Hubert-Tremblay *et al.*, 2006] des recherches sur l'accélération de simulations par un encodage en octree.



(a)



(b)

Fig. 5.86: Nombre de steps simulés et temps d'exécution en fonction de l'image en entrée. En (a) est illustré le temps de simulation en fonction de l'image 3-D choisie, et en (b) la proportion de steps physiques et géométriques pour ces mêmes simulations.

CHAPITRE 6

Distinction de caractères ambigus dans une plateforme de reconnaissance de plaques minéralogiques

6.1 Introduction

L'identification des véhicules par leur plaque minéralogique est un sujet de recherche très étudié depuis une vingtaine d'années. Les principales applications de cette opération correspondent au contrôle d'accès (parking par exemple), à l'identification de voitures volées, à l'automatisation du contrôle de vitesse, *etc.* La reconnaissance de plaques d'immatriculation est considérée aujourd'hui comme un outils maîtrisé, puisque des logiciels « clé en main » sont commercialisés. En particulier, des OCR (pour *optical character recognition*) commerciaux très abordables existent, et présentent des taux de reconnaissance impressionnants (98-99%). Pourtant, ces solutions se basent généralement sur des hypothèses contraignantes : conditions d'éclairage contrôlées, angle de vue et distance entre l'appareil d'acquisition et le véhicule limités, *etc.* Surtout, la plupart des algorithmes implémentés sont optimisés pour reconnaître une région géographique et une nationalité de la plaque spécifiques.

Ainsi, la lecture de plaques minéralogiques ouvre des problématiques de recherche encore largement non résolues, et des solutions innovantes doivent être proposées pour que les logiciels finalement développés puissent être utilisés dans des conditions réalistes, et que les contraintes précédentes puissent être relâchées. Par exemple, les plaques peuvent être abîmées, l'image acquise de qualité moyenne, *etc.* Ainsi, nous avons collaboré avec Nicolas Thome [Thome, 2007], membre de l'entreprise Foxstream, spécialisée dans la surveillance vidéo. Nous avons développé un système qui surmonte les différentes limitations que nous avons évoquées. Il suit le schéma montré dans la figure 6.87, qui récapitule les étapes nécessaires à la reconnaissance et au suivi d'une plaque minéralogique dans une séquence vidéo. Plus précisément, nous avons utilisé l'algorithme de reconstruction d'ob-

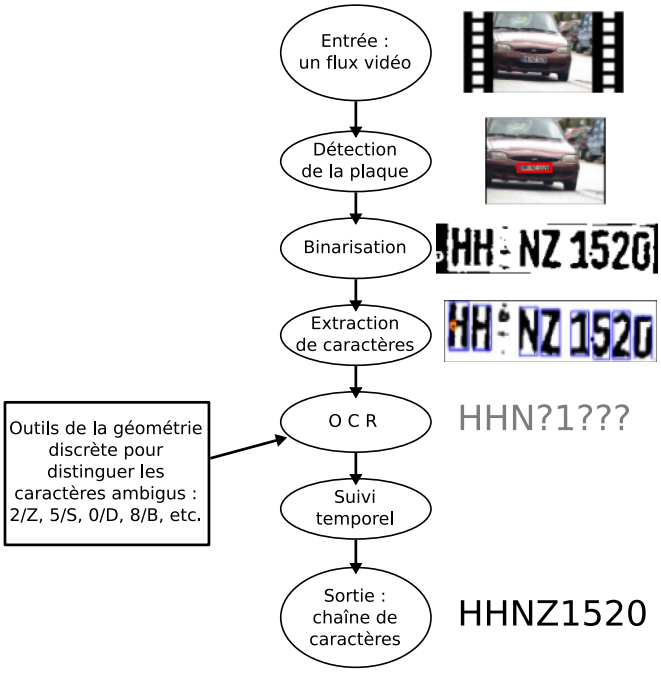


Fig. 6.87: Le système global d'identification de plaques minéralogiques que nous avons développé. Nous avons contribué à améliorer la phase de reconnaissance de caractères, en distinguant ceux qui sont ambigus. Dans ce chapitre, nous évoquons principalement la distinction entre '8' et 'B'.

jets complexes (chapitre 3 de la partie II) pour distinguer les caractères ambigus, lors de l'étape de reconnaissance de caractères (*e.g.* les caractères '8' et 'B').

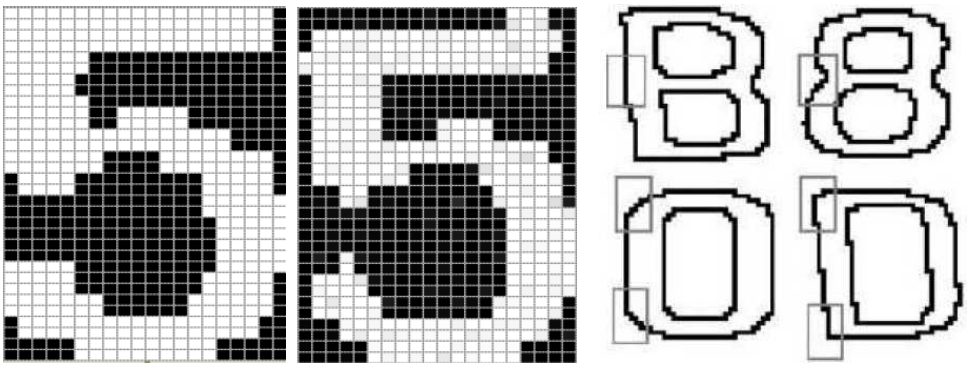
Dans ce chapitre, nous allons tout d'abord tracer un état de l'art des méthodes classiques de reconnaissance de caractères. Puis, nous présentons brièvement la première partie de l'outil de reconnaissance développé dans l'entreprise Foxstream, basé sur un réseau de neurones hiérarchique, et une analyse statistique de l'image en entrée. Nous enchaînons ensuite par notre contribution dans le projet, en présentant en détail l'analyse structurelle que nous avons mise en place. Enfin, nous présentons quelques expérimentations qui permettent de valider notre approche.

6.2 État de l'art des méthodes de reconnaissance de plaques minéralogiques

Nous présentons ici une classification des méthodes de reconnaissance de caractères existantes, et les approches qui ont été retenues dans le contexte de la lecture de plaques minéralogiques. Une fois que l'image est binarisée et les caractères isolés, nous avons en entrée de cette étape un ensemble de blocs de l'image qui représentent des caractères. La reconnaissance de caractères au sein de ces blocs est généralement décomposée en

deux étapes : l'*extraction* de caractéristiques et la *classification*. En ce qui concerne la méthodologie employée, on peut aussi distinguer deux types de stratégies : l'approche *statistique* et l'approche *structurelle*.

Dans les méthodes statistiques, le motif en entrée est composé d'un ensemble de N caractéristiques, et l'étape de classification peut être vue comme un processus de décision statistique. On peut lire dans [Zhang et Lu, 2004, Veltkamp et Latecki, 2006] une liste complète des descripteurs de forme les plus utilisés dans les applications liées à l'image, et nous nous concentrons ici sur certaines méthodologies utilisées pour l'identification de plaques minéralogiques. Les caractéristiques les plus basiques que l'on puisse considérer sont les pixels de l'image binaire. De nombreuses approches (par exemple [Duan *et al.*, 2005, Arth *et al.*, 2007]) utilisent cette stratégie dans leur système de reconnaissance de plaques. Malheureusement, comme nous le montrons dans la figure 6.88-a, cette technique n'est pas robuste lorsque l'on traite des images réelles (en basse résolution par exemple), car le vecteur en entrée n'est pas invariant par translation. Ainsi, des représentations plus riches sémantiquement ont été étudiées pour traiter ces



(a) (b)

Fig. 6.88: L'impact des différences locales dans des images binaires. En (a) les deux images ont des séquences de pixels sensiblement différentes, alors qu'elles appartiennent à la même classe. Au contraire, en (b), des caractères différents ne se distinguent que localement (figure issue de [Chang *et al.*, 2004]).

images réelles : les descripteurs de Fourier [Shridhar et Badreldin, 1984], les moments de Zernike [Broumandnia et Shanbehzadeh, 2007], *etc.* L'étape de reconnaissance consiste alors à étiqueter un ensemble de descripteurs en entrée avec un modèle. Cela peut être réalisé par exemple avec une approche par *template matching* [Anagnostopoulos *et al.*, 2005], en utilisant les cartes auto-organisatrices de Kohonen [Chang *et al.*, 2004], *etc.*

Pour la reconnaissance de formes, les méthodes statistiques ont l'avantage d'être robustes au bruit. Néanmoins, de petites différences entre des motifs similaires peuvent être difficiles à distinguer. Comme nous l'avons montré dans la figure 6.88-b, issue de [Chang *et al.*, 2004], les approches statistiques sont très efficaces pour discriminer

la majorité des classes, mais le sont beaucoup moins pour les caractères « proches » (comme 'B'/'8' ou '2'/'Z'), et surtout dans le cas d'images traitées avec une résolution faible. Les approches structurelles décomposent les motifs en primitives plus simples et obtiennent des propriétés sur ces primitives et/ou des relations entre elles. Les méthodes structurelles sont plus proches du système de reconnaissance humain [Pavlidis, 2003], et sont plus à même de distinguer des motifs différents. Par contre, ce type de méthodologie est plus sensible au bruit. Il existe de nombreuses manière de décomposer une image en parties élémentaires. Nous nous concentrons dans cette thèse sur les techniques de vectorisation [Mertzios et Karras, 1999, Wenyin et Dori, 1999, Cordella et Vento, 2000, Song *et al.*, 2002, Hilaire et Tombre, 2005] dont nous avons fait un état de l'art dans le chapitre 3 de la partie II.

Finalement, il est important de rappeler que beaucoup d'approches intègrent un ensemble de connaissances *a priori*, spécifiques à la nationalité de la plaque traitée (citons [Naito *et al.*, 2000] ou [Duan *et al.*, 2005]). Elles permettent d'améliorer significativement les étapes d'extraction et de reconnaissance de caractères dans les applications présentées. Par exemple, dans [Naito *et al.*, 2000], les auteurs proposent un système qui est optimisé pour les plaques japonaises, qui sont composées d'un nom de quartier, un nombre de classification, un préfixe et un numéro d'enregistrement. Grâce à ces informations, il n'est plus nécessaire de distinguer les caractères similaires les plus répandus (par exemple, pour notre étude, ce serait 'B'/'8'). Par contre, ces méthodes sont uniquement adaptées pour reconnaître une nationalité de plaques très spécifique, et elles échouent dès que le type de plaque diffère légèrement de l'*a priori* supposé. Les méthodes qui tentent d'être efficaces pour une grande variété de plaques minéralogiques sont très rares, et ne sont développées que récemment [Chang *et al.*, 2004, Bremananth *et al.*, 2005, Anagnostopoulos *et al.*, 2005].

Dans la section suivante, nous détaillons brièvement l'analyse statistique des caractères qu'a développé N. Thome pour l'entreprise Foxstream. Cette phase, basée sur un réseau de neurones hiérarchique, permet de reconnaître la forme globale de ces caractères alphanumériques. Cette étape est très efficace, atteignant un taux de reconnaissance global de 96%. De plus, si on ne considère que les caractères « non ambigus », qui correspondent à la majorité des classes, le taux d'erreur est quasiment nul (moins de 0.5%). Cependant, la reconnaissance est nettement détériorée pour les classes de caractères de forme globalement proche, *i.e.* '8'/'B', '2'/'Z', '0'/'D'/'Q', 'V'/'Y'.

Pour distinguer ces caractères similaires, nous procédons à l'analyse structurelle que nous avons développée. Elle est basée sur notre outil de reconstruction topologique et géométrique d'objets binaires complexes (chapitre 3). Nous utilisons le graphe de Reeb et la structure polygonale construite à partir d'une image binaire pour la décrire, et mettre en évidence les différences locales des motifs traités. Nous présentons également comment optimiser le taux de reconnaissance en construisant une base d'apprentissage cohérente et robuste pour la phase de reconnaissance.

6.3 Une stratégie hybride statistique et structurelle pour la reconnaissance de caractères alphanumériques

Nous exposons dans cette section l'OCR que nous avons développé avec N. Thome qui combine les avantages des méthodes de reconnaissance statistiques et structurelles. Nous rappelons le schéma global de notre application dans la figure 6.89.

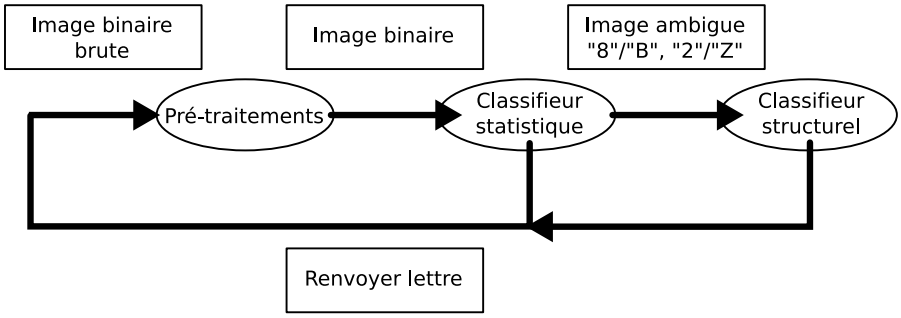


Fig. 6.89: Le schéma global de l'OCR que nous avons développé.

6.3.1 Analyse statistique

Nous décrivons ici brièvement l'étape de reconnaissance statistique, basée sur les *descripteurs de Fourier* (FD). Ils permettent d'obtenir une signature discriminante du contour d'un objet [Persoon et Fu, 1986]. Les N points du contour $X(i) = \{x(i), y(i)\}, i \in \{1, N\}$ peuvent être considérés comme des nombres complexes. Les FD $A(k)$ sont déterminés en calculant la DFT, ou transformée de Fourier discrète de la fonction X (voir également chapitre 2 de la partie I pour plus de détails) :

$$A(k) = \frac{1}{N} \sum_{i=0}^{N-1} X(i) e^{-j2\pi kn/N}, \;\; k \in \{-\frac{N}{2}, \frac{N}{2}\}. \tag{6.47}$$

Les coefficients $A(k)$ représentent le contour discret de la forme dans le domaine fréquentiel. La forme générale de l'objet est représentée par les descripteurs de basse fréquence. Par conséquent, il est possible de ne garder qu'un sous-ensemble des coefficients des plus basses fréquences pour extraire une signature de forme robuste.

L'étape de classification est gérée par une hiérarchie de réseaux de neurones (ou HNN pour *hierarchical neural network*). La structure globale du HNN développé est présentée dans la figure 6.90. Au premier niveau, trois perceptrons différents, notés HC_1, HC_2 et HC_3, sont utilisés. Ils ont pour objectif de reconnaître le contour externe d'une forme, et éventuellement les deux contours internes. Plus précisément, le classifieur HC_1 inclut les 35 différents symboles alphanumériques, avec une même classe pour '0' (zéro) et 'O'. HC_2 permet d'identifier un unique contour pour les symboles '0', '4', 'A', 'D', 'Q', et le contour supérieur de '8', '9', 'B', 'R' et 'P'. Enfin, le classifieur HC_3 analyse la forme du

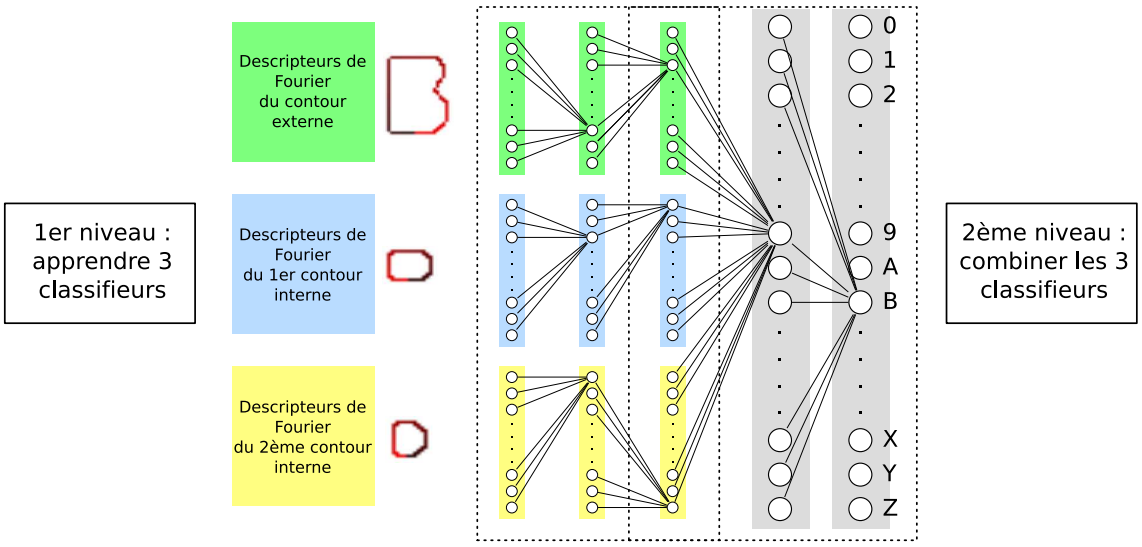


Fig. 6.90: Architecture du classifieur basé sur un HNN.

contour interne inférieur de '8', 'B', et '6'. Le second niveau du HNN prend en entrée le résultat des trois précédents réseaux et réalise la reconnaissance finale. Le résultat est représenté sous la forme d'un vecteur représentant la probabilité de chaque caractère.

Grâce à cette première étape de classification, notre système obtient des taux de reconnaissance élevés dans la majorité des classes (voir la section d'expérimentations de ce chapitre). En particulier, l'architecture hiérarchique et le découplage des deux niveaux logiques du classifieur permettent un apprentissage fiable qui limite au maximum les phénomènes de sur-apprentissage. Ils rendent également possible la reconnaissance de caractères altérés : contours internes bouchés, topologie inexacte, *etc.*. Néanmoins, pour lever l'ambiguïté entre les caractères similaires, nous avons développé un classifieur structurel, basé sur les outils de la géométrie discrète développés dans le chapitre 3.

6.3.2 Analyse structurelle

Dans cette section, nous proposons de construire un graphe de Reeb [Reeb, 1946] associé à une image binaire `I`. Puis, nous calculons une structure polygonale à l'intérieur de `I`. Ces deux opérations sont réalisées grâce à l'algorithme 3.7 `reconstruction()`. Par la suite, nous traitons la classification des symboles '8' et 'B', mais nous pourrions dérouler un raisonnement similaire pour d'autres cas (*e.g.* '2'/'Z', où l'on prend en compte la partie haute de l'image). Tout d'abord, on peut noter que ces deux caractères diffèrent principalement dans la partie gauche de l'image binaire `I`. Par conséquent, nous avons décidé de ne prendre en compte que cette partie de `I`, notée $\mathtt{I}_L$. Comme la lettre est centrée dans l'image, seule la moitié de `I` est considérée. Nous rappelons ici les éléments que nous calculons avec `reconstruction()` (voir figure 6.91) :

- ▷ Les pixels de la grille régulière associée à `I`, notée $\mathbb{I}$, sont regroupés sous forme de k-arcs (ensemble $\mathcal{A}_{\mathtt{I}_L}$). Les n_c cellules contenues dans les k-arcs sont notées

$\{R_i\}_{i=1,n_c}$. Dans une image représentant un 'B', ces cellules sont globalement plus larges que celles qui sont issues d'un '8'.

- ▷ Un graphe de Reeb $\mathcal{G}_{\mathtt{I}_L}$ associé à $\mathtt{I}_L$ est calculé, suivant une fonction de hauteur h donnée. Ainsi, nous choisissons de définir h suivant l'axe des X. Cela signifie également que l'on considère un ordre lexicographique $\preceq_{lx}$ sur la grille régulière associée à `I`. Avec cette orientation, le graphe de Reeb d'une image 'B' commence par une composante (noeud b), et termine avec trois noeuds e. Dans une image '8', ce graphe possède généralement deux noeuds b.
- ▷ Une reconstruction polygonale $\mathcal{L}_{\mathtt{I}_L}$ est calculée au sein des k-arcs $\mathcal{A}_{\mathtt{I}_L}$. Comme on peut le constater dans la figure 6.91, cette reconstruction aboutit à un nombre de points plus important dans une configuration '8' que dans le cas 'B'.

Dans la figure 6.91, nous avons montré le résultat de l'algorithme `reconstruction` pour des images synthétiques et pour des images issues de l'étape d'extraction de caractères de l'application développé au sein de l'entreprise Foxstream. On peut noter que les caractéristiques que nous avons énoncées sur les sorties de `reconstruction` sont valables dans les deux cas. Ainsi, nous allons nous baser par la suite sur ces propriétés pour classifier les images binaires ambiguës.

6.3.3 Classification basée sur un algorithme de boosting

Nous avons développé un classifieur structurel basé sur l'algorithme `AdaBoost` (pour *adaptive boosting*) [Freund et Schapire, 1995, Schapire, 2003, Meir et Rätsch, 2003] (on peut lire également [Thome, 2007]). Cet algorithme, implémenté dans la librairie `OpenCV` [OpenCV, 2008], est très adapté à notre problématique, puisque nous considérons uniquement deux classes C_1 et C_2 (*e.g.* $C_1 = 8$ et $C_2 = B$). En considérant les remarques que nous avons énoncées précédemment au sujet de l'algorithme `reconstruction()`, nous construisons $\mathbf{x}$, le vecteur de caractéristiques suivant :

$$\mathbf{x}^T = \begin{pmatrix} \max_{i=1,n_c}\left(S(R_i)\right), R_i \in \mathcal{A}_{\mathtt{I}_L} \\ \left|\left\{n = b, n \in \mathcal{G}_{\mathtt{I}_L}\right\}\right| \\ \left|\left\{(x,y) \in \mathcal{L}_{\mathtt{I}_L}\right\}\right| \end{pmatrix}, \tag{6.48}$$

où $S(R_i)$ représente la surface de la cellule R_i. Rappelons également que la notation $n = b$ signifie que l'on considère les noeuds du graphe de Reeb n tels que n est un noeud de début de composante connexe, noté b. En considérant un ensemble (de référence) d'exemples d'images, on entraîne tout d'abord le classifieur `AdaBoost` avec les N vecteurs de caractéristiques $\mathbf{x}_i$ (soit $N/2$ vecteurs par classe), concaténés dans une seule matrice $\mathbf{X}$. Nous donnons dans l'algorithme 6.22 le déroulement du classifieur `AdaBoost` avec les éléments que nous avons calculés. Nous nous sommes basés sur les outils implémentés dans la librairie `OpenCV` [OpenCV, 2008]. Le principe de cet algorithme est de déterminer une règle forte qui permette de faire correspondre l'ensemble des échantillons $\mathbf{X}$ avec leurs classes associées, représentées dans le vecteur $\mathbf{y}$ [1]. À chaque itération $t = 1, \ldots, T$, on

[1] On cherche donc à établir une relation de la forme $H(\mathbf{X}) = \mathbf{y}$.

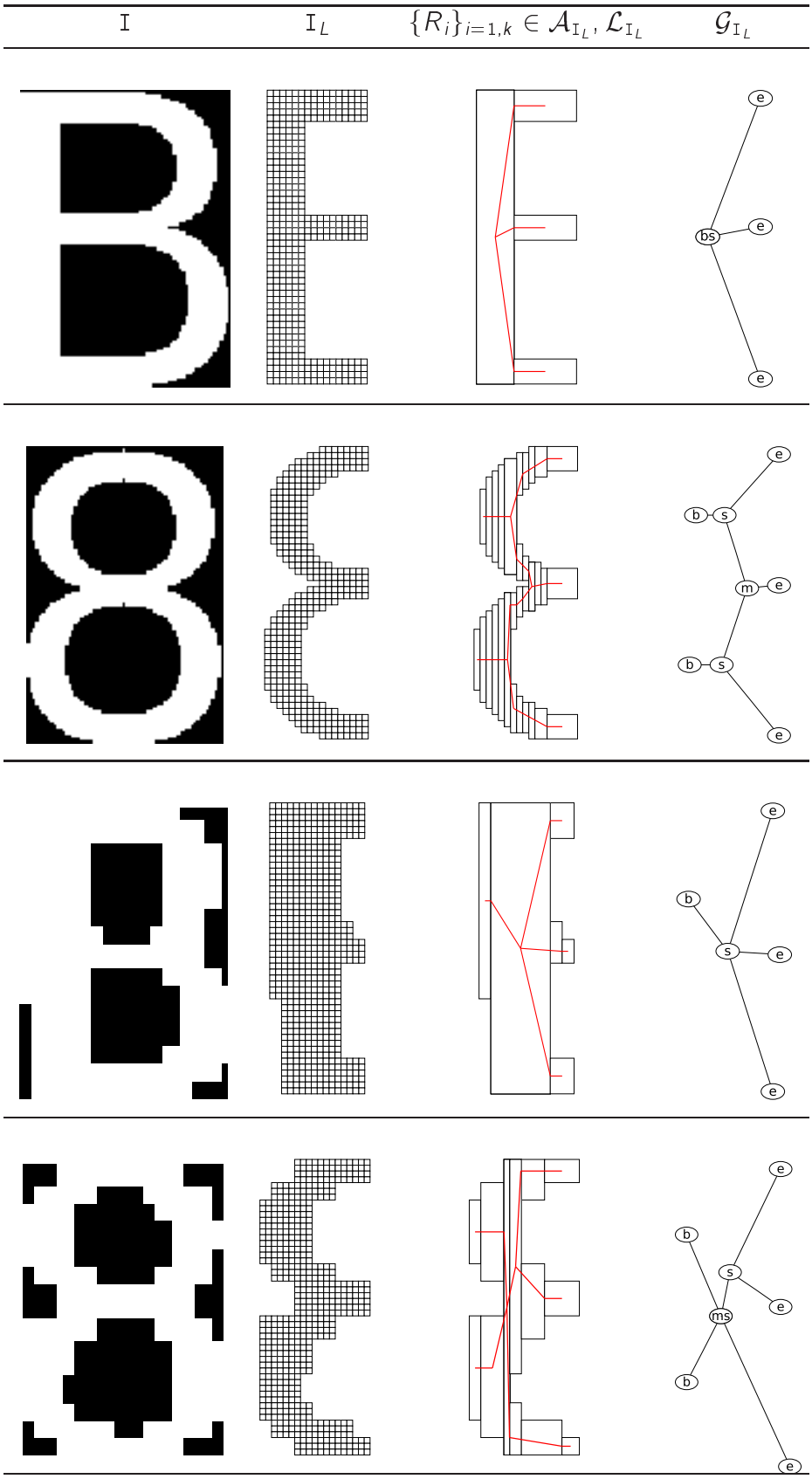


Fig. 6.91: Les éléments calculés par l'algorithme `reconstruction()` dans notre application. Ils sont illustrés pour des images représentant les caractères 'B' et '8' de synthèse (deux premières lignes), et pour des caractères issus de l'étape d'extraction de caractères (deux lignes du bas).

Algorithme 6.22 : Algorithme `AdaBoost()` [Freund et Schapire, 1995].

entrée : un ensemble de N échantillons $\mathbf{X} = \{\mathbf{x}_i\}_{i=1,N}$,
la classe associée à chaque échantillon $\mathbf{y} = \{y_i\}_{i=1,N}$, $y_i \in \{-1, +1\}$,
(on notera dans cet algorithme le couple $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$,
T un nombre d'itérations.

sortie : une règle de classification de la forme $H(\mathbf{X})$ combinant des classifieurs faibles.

début

initialiser $d_i^{(1)} = \frac{1}{N},\ i = 1, N$;

pour $t = 1$ **à** T **faire**

entraîner un classifieur faible h_t en considérant l'ensemble pondéré $(S, \mathbf{d}^{(t)})$;

`// Calcul de l'erreur de` h_t

$\epsilon_t = \sum_{i=1}^{N} d_i^{(t)}[y_i \neq h_t(\mathbf{x}_i)]$;

$\alpha_t \leftarrow \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$;

`// Mettre à jour les poids`

$d_i^{(t+1)} = d_i^{(t)} \exp\big(-\alpha_t y_i h_t(\mathbf{x}_i)\big)$;

`// Normalisation des poids afin que` $\sum_i d_i^{(t+1)} = 1$

retourner $H(\mathbf{X}) = signe\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{X})\right)$

fin

calcule une distribution de probabilités sur les N échantillons en fonction des résultats de l'itération précédente ; le poids $d_i^{(t)}$ est associé à l'exemple $(\mathbf{x}_i, y_i)$. Au départ, tous les poids ont la même valeur $1/N$, et les poids des échantillons mal classés sont augmentés. Cela force le classifieur à se concentrer sur les exemples difficiles pour construire la règle H. Àu temps t, on détermine le classifieur faible (ou l'hypothèse) $h_t : \mathbf{X} \rightarrow \{-1, +1\}$ qui représente au mieux $\mathbf{d}^{(t)}$ sur $\mathbf{X}$. Ces classifieurs faibles peuvent être de simples arbres de décisions [OpenCV, 2008], et font la particularité de `AdaBoost` (« un ensemble de classifieurs faibles pour construire enfin une règle forte »). Chaque hypothèse est affectée d'un poids α_t qui indique la force de cette hypothèse dans la combinaison finale. Plus l'erreur ϵ_t est faible, plus h_t a un poids élevé. La classification d'un nouvel exemple (que l'on notera par la suite `AdaBoost`($\mathbf{x}$)) est réalisée par la règle

$$H(\mathbf{x}) = signe\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right) \tag{6.49}$$

On peut finalement noter que ce classifieur possède des propriétés intéressantes [Freund et Schapire, 1995]. L'erreur de l'hypothèse forte H est bornée, et elle diminue exponentiellement avec t, si l'on choisit des classifieurs faibles plus performants qu'un tirage aléatoire. Maintenant que nous avons les outils nécessaires pour construire une règle forte à partir d'une base d'images échantillons, nous allons nous intéresser à la

manière de calculer une base optimale pour la distinction de caractères ambigus.

6.3.4 Optimisation de la base d'apprentissage

Grâce à l'expérience de l'entreprise Foxstream dans la vidéo surveillance, nous avons obtenu une base importante d'images représentant des caractères 'B' et '8'. Nous cherchons maintenant à proposer une base d'apprentissage suffisamment pertinente pour classifier par la suite les images correctement. Nous posons pour le moment cette base complète d'images $\mathcal{C}$, de taille N. Ainsi, nous avons développé un algorithme itératif qui permet de tester une base de $K \leq N$ images, de calculer un taux de reconnaissance sur un jeu de M images de test, et de recommencer avec une nouvelle base jusqu'à ce que l'on arrête le processus, ou que le taux de reconnaissance soit suffisamment élevé. Ainsi, nous définissons implicitement une *fonction de coût* grâce à l'algorithme `AdaBoost()` que nous avons détaillé précédemment. À chaque combinaison de K images binaires $\{\mathtt{I}_{i_1}, \ldots, \mathtt{I}_{i_K}\}$, $i_1, \ldots, i_K \in \{1, \ldots, N\}$ appartenant à la base, nous pouvons définir un ensemble de vecteurs de caractéristiques $\{\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_K}\}$. Cette base est ensuite l'entrée de l'apprentissage de l'algorithme `AdaBoost()`. Une fois la règle de classification déterminée, nous l'utilisons pour reconnaître une base de M images test. Ainsi, nous définissons la fonction de coût

$$\begin{aligned} F : \quad \mathcal{P}_K(\mathcal{C}) \quad &\longrightarrow \mathbb{R} \\ \{\mathtt{I}_{i_1}, \ldots, \mathtt{I}_{i_K}\} \quad &\longrightarrow r, \ 0 \leq r \leq 100 \end{aligned} \tag{6.50}$$

où $\mathcal{P}_K(\mathcal{C})$ représente l'ensemble des K-combinaisons possibles au sein de $\mathcal{C}$, et r est le taux de reconnaissance associé à la base d'apprentissage testée. Pour trouver un meilleur arrangement de l'ensemble $\mathcal{P}_K(\mathcal{C})$ sans le parcourir de manière exhaustive, nous nous basons sur la technique d'optimisation combinatoire *hill climbing* [Selman *et al.*, 1992] (voir l'algorithme 6.23). Au début de l'algorithme 6.23, nous générons aléatoirement une première base $\{\mathtt{I}_{i_1}, \ldots, \mathtt{I}_{i_K}\}$ dans $\mathcal{P}_K(\mathcal{C})$. Ensuite, le processus itératif principal (T itérations au maximum) considère tous les voisins possibles $\{\mathtt{I}_{j_1}, \ldots, \mathtt{I}_{j_K}\}$ (*i.e.* $j_1 = i_1 + r_1$, $r_1 \in \{-1, 0, +1\}$ par exemple). À partir de chaque combinaison voisine, une règle de classification avec `AdaBoost`, puis on calcule un taux de reconnaissance associé. Une fois le voisinage complètement parcouru, on obtient r_{loc}, meilleur taux de reconnaissance des voisins de $\{\mathtt{I}_{i_1}, \ldots, \mathtt{I}_{i_K}\}$. Lorsque r_{loc} est également meilleur que le taux maximal actuel (r_{max}), nous considérons $\{\mathtt{I}_{j_1}, \ldots, \mathtt{I}_{j_K}\}_{loc}$ comme la nouvelle combinaison actuelle, et nous passons à l'itération suivante. Dans le cas contraire, cela signifie qu'aucune combinaison du voisinage n'aboutit à une meilleure reconnaissance. Néanmoins, nous avons choisi de prendre un de ses voisins au hasard pour continuer la recherche locale. Il est important de noter que nous vérifions que cette recherche locale ne dure pas un nombre d'itérations trop élevé (T_{loc}). Dès que $t_{loc} > T_{loc}$, nous générons complètement une nouvelle base de manière aléatoire, comme à l'étape d'initialisation. À la fin de l'algorithme, nous renvoyons la base optimale $\{\mathtt{I}_{i_1}, \ldots, \mathtt{I}_{i_K}\}_{opt}$ dont le taux de reconnaissance associé r_{max} est

Algorithme 6.23 : Optimisation combinatoire pour calculer une base d'apprentissage pertinente.

entrée : une base $\mathcal{C}$ de N images binaires,
K est le nombre d'images considérées dans la base d'apprentissage,
M est le nombre d'images de la base de test $\mathcal{B}$,
T est le nombre d'itérations maximal de l'algorithme,
T_{loc} est le nombre d'itérations maximal pour la recherche locale.

sortie : une combinaison de K images de $\mathcal{C}$, aboutissant à un haut taux de reconnaissance.

début
- $\{\mathtt{I}_{i_1}, \ldots, \mathtt{I}_{i_K}\} \leftarrow$ `générer_base()` ;
- $\{\mathtt{I}_{i_1}, \ldots, \mathtt{I}_{i_K}\}_{opt} \leftarrow \emptyset$;
- $r_{max} \leftarrow 0$;
- $t_{loc} \leftarrow 0$;
- **pour** $t = 1$ **à** T **faire**
  - $r_{loc} \leftarrow 0$;
  - **pour chaque** *combinaison* $\{\mathtt{I}_{j_1}, \ldots, \mathtt{I}_{j_K}\}$ *voisine de* $\{\mathtt{I}_{i_1}, \ldots, \mathtt{I}_{i_K}\}$ **faire**
    - `AdaBoost`$(\{\mathbf{x}_{j_1}, \ldots, \mathbf{x}_{j_K}\})$;
    - **pour chaque** *image de test* $\mathtt{I}_m \in \mathcal{B}$ **faire**
      - `AdaBoost`$(\mathbf{x}_m)$;
    - calculer le taux de reconnaissance r associé à $\{\mathtt{I}_{j_1}, \ldots, \mathtt{I}_{j_K}\}$;
    - **si** $r > r_{loc}$ **alors**
      - $r_{loc} \leftarrow r$;
      - $\{\mathtt{I}_{j_1}, \ldots, \mathtt{I}_{j_K}\}_{loc} \leftarrow \{\mathtt{I}_{j_1}, \ldots, \mathtt{I}_{j_K}\}$;
  - `// Le meilleur voisin devient la combinaison actuelle`
  - **si** $r_{loc} > r_{max}$ **alors**
    - $r_{max} \leftarrow r_{loc}$;
    - $\{\mathtt{I}_{i_1}, \ldots, \mathtt{I}_{i_K}\} \leftarrow \{\mathtt{I}_{j_1}, \ldots, \mathtt{I}_{j_K}\}_{loc}$;
    - $\{\mathtt{I}_{i_1}, \ldots, \mathtt{I}_{i_K}\}_{opt} \leftarrow \{\mathtt{I}_{j_1}, \ldots, \mathtt{I}_{j_K}\}_{loc}$;
    - $t_{loc} \leftarrow 0$;
  - `// La combinaison actuelle devient un voisin au hasard`
  - **sinon**
    - $\{\mathtt{I}_{i_1}, \ldots, \mathtt{I}_{i_K}\} \leftarrow$ `générer_voisin()` ;
    - $t_{loc} \leftarrow t_{loc} + 1$;
  - `// On génère aléatoirement une nouvelle combinaison`
  - **si** $t_{loc} > T_{loc}$ **alors**
    - $\{\mathtt{I}_{i_1}, \ldots, \mathtt{I}_{i_K}\} \leftarrow$ `générer_base()` ;
    - $t_{loc} \leftarrow 0$;
- **retourner** $\{\mathtt{I}_{i_1}, \ldots, \mathtt{I}_{i_K}\}_{opt}$

fin

le meilleur trouvé pendant l'optimisation. Notre technique d'optimisation de la base d'apprentissage se basant sur une approche de recherche locale, il est difficile d'affirmer avec certitude que cette base optimale est « la » meilleure K-combinaison dans $\mathcal{C}$. Néanmoins, nous présentons, dans la section suivante des évaluations des performances de notre système. Nous montrons alors que les choix d'utiliser les outils de la géométrie discrète, puis de construire une base d'images pertinentes avec notre approche aboutissent à des résultats très satisfaisants (plus de 93% de taux de reconnaissance contre 85% avec le classifieur statistique). Finalement, si l'on veut s'assurer que la base construite représente un maximum global dans notre fonction de coût F, on pourrait s'inspirer des algorithmes de classification active [Kotsiantis *et al.*, 2006, Huang *et al.*, 2007], ou envisager d'autres approches d'optimisation combinatoire : le recuit simulé [Kirkpatrick *et al.*, 1983], la recherche tabou [Glover, 1989a, Glover, 1989b], l'optimisation par colonies de fourmis [Dorigo *et al.*, 1996], ou encore des approches développées plus récemment comme l'algorithme des abeilles [Pham *et al.*, 2005].

6.4 Expérimentations sur la performance du classifieur

Nous donnons tout d'abord les résultats de tests de performances du classifieur statistique (basé sur des descripteurs de Fourier et une HNN). Dans la table 6.8, nous listons l'ensemble de ces performances par classe de caractères. Dans ces tests, on peut noter qu'il existe une grande disparité entre les différentes classes, et le taux d'erreur pour les caractères similaires ('8'/'B', '2'/'Z', '0'/'D'/'Q', 'V'/'Y') est sensiblement plus grand que pour les autres classes. Le couple '8'/'B' est celui qui aboutit à l'erreur la plus importante. Si nous ne prenons pas en compte ces caractères, le taux de reconnaissance de ce classifieur est de 99.5%. Ces résultats justifient la validité du HNN à discriminer efficacement la majorité des classes. Néanmoins, nous remarquons également la limite des descripteurs de Fourier pour distinguer les symboles similaires. En effet, d'une part ils capturent une information fréquentielle trop globale pour discriminer efficacement ces classes. D'autre part, les caractères ambigus se différencient principalement par la présence d'angles droits qui sont perdus (ou du moins atténués) lors de l'extraction des descripteurs de Fourier (il faudrait un nombre infini de coefficients pour reconstruire parfaitement un angle droit).

Pour traiter ces cas difficiles, l'analyse structurelle permet de détecter les différences locales entre ces symboles. Dans ces expérimentations, nous nous sommes concentrés sur la distinction '8'/'B' qui est la plus pénalisante dans notre système. Dans la figure 6.92, nous présentons un exemple du comportement de notre algorithme d'optimisation combinatoire. Nous avons choisi un temps d'optimisation de 2 heures environ ($T = 10000$ itérations). L'expérience a été réalisée sur une base complète $\mathcal{C}$ avec N =446 images de '8' et de 'B'. Nous avons fixé $K = 100$, et pour une combinaison d'images parcourue $\{\mathtt{I}_{j_1}, \ldots, \mathtt{I}_{j_K}\}$ pendant l'algorithme, nous construisons la base de test de $M = N - K = 336$ images $\mathcal{C} \backslash \{\mathtt{I}_{j_1}, \ldots, \mathtt{I}_{j_K}\}$. Les bases testées dans l'algorithme ont $K/2 = 50$ images représentant des '8' et autant des 'B'. Àprès quelques itérations, le taux de reconnaissance globale est meilleur que celui obtenu par l'approche statistique (environ 85%) et aug-

Classe	Nombre d'erreurs	Nombre d'exemples	Taux d'erreur	Classe	Nombre d'erreurs	Nombre d'exemples	Taux d'erreur
0	25	248	10.08	I	1	56	1.79
1	0	241	0.60	J	0	78	0.00
2	4	263	1.58	K	0	134	0.00
3	3	290	1.03	L	0	85	0.00
4	3	175	1.71	M	0	103	0.00
5	0	269	0.00	N	0	88	0.00
6	6	381	1.56	P	0	183	0.00
7	1	261	0.38	Q	5	59	8.47
8	98	410	23.90	R	0	158	0.00
9	1	335	0.30	S	0	131	0.00
A	1	191	0.52	T	0	53	0.00
B	35	276	12.68	U	0	53	0.00
C	0	95	0.00	V	6	76	7.89
D	15	156	9.62	W	0	72	0.00
E	0	63	0.00	X	0	78	0.00
F	0	55	0.00	Y	0	61	0.00
G	4	140	2.86	Z	0	123	0.00
H	0	76	0.00	Total	208	5592	3.72

Tab. 6.8: Les performances du classifieur statistique par classe de caractères. Les classes similaires sont illustrés avec la même couleur.

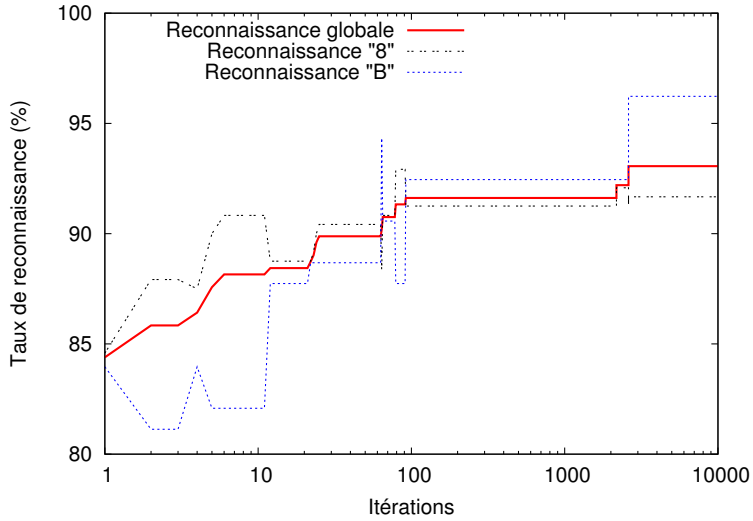


Fig. 6.92: Un exemple de l'optimisation de la base d'apprentissage pour le classifieur structurel. Nous avons choisi 10 000 itérations (environ 2 heures). Les taux de reconnaissance individuels pour les classes '8' et 'B' sont tracés en pointillés, et le taux global est en trait plein.

mente jusqu'à plus de 90% après une centaine d'itérations (1.5 minutes). Le meilleur score (plus de 93%, voir table 6.9) est obtenu en 25 minutes dans cet exemple. Dans la table 6.9, nous indiquons les performances du classifieur structurel pour la même base d'images. Son taux de reconnaissance dépasse de 13% celui du classifieur statistique, et atteint plus de 93% sur l'ensemble de la base.

Ensemble	Classification statistique	Classification structurelle
Images '8'	76.1 %	92.9 %
Images 'B'	87.0 %	95.3 %
Total	80.6 %	93.6 %

Tab. 6.9: Taux de reconnaissance des deux classifieurs sur une même base d'images.

6.5 Bilan et perspectives

Dans ce chapitre, nous avons présenté une application de l'algorithme de reconstruction d'objets complexes pour la reconnaissance de caractères. Grâce au graphe de Reeb et à la structure polygonale associés à une image binaire, nous sommes capables de distinguer des symboles similaires (et en particulier '8'/'B'). Notre implémentation est très spécifique à l'application que nous avons décrite ici, mais elle reflète parfaitement l'intérêt de nos représentations pour caractériser des objets discrets. Nous avons développé un classifieur structurel basé sur l'algorithme `AdaBoost` et une base d'apprentissage optimale est calculée grâce à une technique d'optimisation combinatoire par recherche locale. Nous avons montré que notre approche hybride aboutissait à des taux de reconnaissance élevés.

Plusieurs points de notre méthode pourraient être approfondis. Tout d'abord, nous utilisons le nombre de points de la reconstruction polygonale pour distinguer un '8' d'un 'B'. Ainsi, nous pourrions opter pour une autre approche de reconstruction d'un k-arc et étudier son impact sur notre système. Ensuite, nous avons choisi une approche par recherche locale de *hill climbing* pour calculer une base optimale. Il serait intéressant de tester d'autres méthodologies, comme les optimisations par colonies de fourmis [Dorigo *et al.*, 1996], ou les techniques de classification active [Huang *et al.*, 2007]. Enfin, on pourrait étudier la distinction entre d'autres classes similaires (*e.g.* '2'/'Z'), bien qu'elles soient moins critiques que celles que nous avons traitées.

CHAPITRE 7

Approximation polygonale de courbes implicites par une analyse en intervalles

7.1 Introduction

En modélisation géométrique, l'approximation polygonale de courbes implicites planaires est un problème classique. En effet, puisqu'une telle courbe est un objet graphique très complexe, il devient difficile de la tracer, ou d'appliquer des opérations non-analytiques sur son intérieur, comme les opérations booléennes (union, intersection, *etc.*). L'approximation par des segments de droites permet de prendre en charge rapidement ces tâches. Dans notre cadre, le problème se résume à calculer un objet polygonal $\mathcal{L}$ approximant une courbe $\mathcal{C}$, donnée par les solutions de la fonction associée f :

$$\mathcal{C} = \{(x_1, x_2) \in \Omega : f(x_1, x_2) = 0\} \text{ telle que } f \ : \Omega \subseteq \mathbb{R}^2 \longrightarrow \mathbb{R}. \tag{7.51}$$

Le problème du tracé de cette courbe revient à calculer et tracer les zéros de l'équation $f(x_1, x_2) = 0, x = (x_1, x_2) \in \mathbb{R}^2$. Cela peut être traité par plusieurs approches [Boissonnat *et al.*, 2006] (voir aussi les travaux de B. Mourrain, par exemple [Mourrain, 2002]) :

- ▷ Les *modeleurs analytiques* exploitent les valeurs de f et ses dérivées. On peut citer par exemples les approches par minimisation, ou les méthodes de parcours continu basées sur l'algorithme itératif de Newton/Raphson :

$$x_{n+1} = x_n - f(x_n).J_f(x_n)^{-1} \tag{7.52}$$

$$\iff \text{résoudre } J_f(x_n)(x_{n+1} - x_n) = -f(x_n), \tag{7.53}$$

  où $J_f(x_n)$ représente le Jacobien de f en x_n.
- ▷ Les *méthodes par subdivision* utilisent un critère pour isoler les racines, *e.g.* la loi de Descartes ou l'arithmétique d'intervalles (voir plus loin).

▷ On peut enfin citer sans être exhaustif d'autres méthodes employées de manière très sporadique telles que les approches par transformation (*e.g.* projection), les méthodes algébriques (*e.g.* calcul des valeurs propres), *etc.*

La méthode de Newton converge, si l'on initialise x_0 assez près d'une solution réelle. Cela implique qu'il faut déterminer cette première valeur de manière suffisamment précise pour que le tracé final soit correct. De plus, le parcours continu par saut des modeleurs analytiques peut entraîner une erreur dans la topologie du tracé [Faudot et Michelucci, 2007] (voir figure 7.93). Pour traiter le problème de l'approximation polygonale et éviter ces

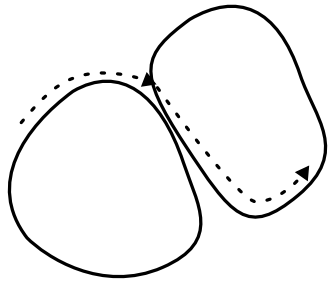


Fig. 7.93: Le non-respect de la topologie lors du parcours itératif par saut des modeleurs analytiques. Alors que l'algorithme suit une composante connexe de la courbe, il peut "sauter" sur vers une autre composante si elle est suffisamment proche. Dans ce cas, la tracé n'aura qu'une seule composante au lieu d'en décrire deux.

problèmes, on peut tout d'abord choisir de discrétiser la courbe sur une grille régulière afin de construire une approximation à partir d'un ensemble de pixels [Taubin, 1994, Sloboda et Zat'ko, 1995, Sloboda et Zat'ko, 2001]. Néanmoins, ces méthodes *énumératives* requièrent une résolution globale très fine, et de très nombreux pixels représentent une partie du plan qui n'intersecte pas la courbe. Plutôt que d'utiliser inutilement de l'espace pour ces parties sans intérêt, de nombreux algorithmes *adaptatifs* [Lopes *et al.*, 2001, Plantinga et Vegter, 2004, Plantinga et Vegter, 2007, Snyder, 1992b, Yu *et al.*, 2006] utilisent l'*arithmétique d'intervalles* (ou *analyse en intervalles*) pour approximer $\mathcal{C}$ avec un ensemble de cellules 2-D (ou *intervalles* pour plus de clarté, au lieu d'*hyper-intervalles*). Ces cellules ont des tailles variables, et permettent une représentation adaptative de la courbe. L'arithmétique d'intervalles, introduite par R.E. Moore dans les années 60-70 [Moore et Yang, 1959, Moore, 1966], définit les opérateurs de l'arithmétique classique et des fonctions plus complexes en considérant des intervalles contenant la valeur réelle calculée. Ces intervalles sont encadrés par deux nombres représentables en virgule flottante. Ce principe permet d'éviter les erreurs de l'arithmétique en virgule flottante (*floating point arithmetic* dans la littérature), en encadrant par des intervalles les quantités réelles que l'on manipule. Puis, en combinant la redéfinition de ces opérateurs et des fonctions mathématiques de bases (cosinus, exponentielle, *etc.*), une *fonction d'inclusion* associée

à f peut être construite. Avec une telle fonction, on peut tester si un intervalle 2-D intersecte ou non la courbe planaire $\mathcal{C}$. En général, pour calculer l'ensemble des intervalles $\mathcal{E}$, les algorithmes de tracé subdivisent récursivement un intervalle initial I de grande taille en intervalles qui intersectent $\mathcal{C}$. Cette boucle s'arrête lorsque tous les intervalles obtenus sont suffisamment petits pour approximer au mieux la courbe. De plus, ce schéma de raffinement peut être topologiquement contrôlé. Par exemple, J. M. Snyder [Snyder, 1992b] propose de tester si un intervalle est *globalement paramétrisable* pour tracer correctement les courbes et surfaces implicites. S. Plantinga et G. Vegter [Plantinga et Vegter, 2007] ont choisi de maintenir un quadtree équilibré et un critère de *petite variation de la normale* (voir également [Boissonnat *et al.*, 2006] pour un état de l'art global). Enfin, l'approximation polygonale $\mathcal{L}$ de $\mathcal{C}$ est réalisée en traçant un ou plusieurs segments dans chaque intervalle I. Plus précisément, on calcule les points où $\mathcal{C}$ intersecte les bords de la cellule associée à I.

Dans ce chapitre, nous proposons un système qui permet donc d'approximer par une structure polygonale la courbe $\mathcal{C}$ et de représenter la topologie de $\mathcal{C}$ grâce à des outils de la géométrie discrète (chapitre 3). Plus précisément, nous considérons les cellules 2-D permettant d'encadrer $\mathcal{C}$ comme appartenant à une $\mathbb{I}$-grille. Ainsi, nous pouvons proposer de mettre à jour les structures que nous obtenons en prenant en charge des raffinements et des groupements locaux de cellules. Grâce à ces algorithmes, nous pouvons élaborer une technique en deux passes qui (1) raffine dans un premier temps les intervalles tant qu'un certain critère géométrique ou topologique n'est pas respecté, et (2) les regroupe afin de simplifier la structure calculée. On peut faire un parallèle entre cette approche et l'algorithme de segmentation d'images *split-and-merge* [Horowitz et Pavlidis, 1977], où l'image est d'abord subdivisée en petites régions, qui sont ensuite fusionnées ensemble si elles sont suffisamment homogènes. Nous décrivons tout d'abord de manière détaillée l'arithmétique d'intervalles (et quelques éléments sur une de ses extensions) et nous établissons le lien entre cette théorie et la géométrie discrète sur $\mathbb{I}$-grilles. Ensuite, nous montrons comment les algorithmes que nous avons développés précédemment peuvent être utilisés dans notre cadre. Pour ce faire, nous détaillons des expérimentations qui présentent un exemple d'applications de notre système pour l'approximation automatique ou manuelle de courbes implicites. Enfin, nous comparons la rapidité et la qualité de notre technique avec deux méthodes classiques [Lopes *et al.*, 2001, Snyder, 1992b] d'approximation de courbes, afin d'en valider la robustesse et l'utilité dans cette optique.

7.2 Tracé classique de courbes implicites par analyse en intervalles

7.2.1 Les arithmétiques d'intervalles et affines

L'arithmétique d'intervalles a été introduite par R. E. Moore dans les années 60 [Moore et Yang, 1959, Moore, 1966] (voir aussi [de Figueiredo et Stolfi, 1997]). Dans

cette théorie, une quantité réelle x est représentée par un intervalle $\overline{x} = [\overline{x}.inf, \overline{x}.sup]$ de nombres représentables en virgule flottante. Les opérateurs de l'arithmétique classique sont définis de telle manière que les résultats obtenus contiennent la quantité réelle x. Nous pouvons résumer ainsi :

$$\begin{aligned}
\overline{x} \oplus \overline{y} &= \Big[\overline{x}.inf + \overline{y}.inf, \overline{x}.sup + \overline{y}.sup\Big] \\
\overline{x} \ominus \overline{y} &= \Big[\overline{x}.inf - \overline{y}.inf, \overline{x}.sup - \overline{y}.sup\Big] \\
\overline{x} \otimes \overline{y} &= \Big[\min\big(\overline{x}.inf \times \overline{y}.inf, \overline{x}.inf \times \overline{y}.sup, \overline{x}.sup \times \overline{y}.inf, \overline{x}.sup \times \overline{y}.sup\big), \\
&\qquad \max\big(\overline{x}.inf \times \overline{y}.inf, \overline{x}.inf \times \overline{y}.sup, \overline{x}.sup \times \overline{y}.inf, \overline{x}.sup \times \overline{y}.sup\big)\Big] \\
\overline{x} \oslash \overline{y} &= \Big[\min\big(\frac{\overline{x}.inf}{\overline{y}.inf}, \frac{\overline{x}.inf}{\overline{y}.sup}, \frac{\overline{x}.sup}{\overline{y}.sup}, \frac{\overline{x}.sup}{\overline{y}.sup}\big), \\
&\qquad \max\big(\frac{\overline{x}.inf}{\overline{y}.inf}, \frac{\overline{x}.inf}{\overline{y}.sup}, \frac{\overline{x}.sup}{\overline{y}.sup}, \frac{\overline{x}.sup}{\overline{y}.sup}\big)\Big], 0 \notin \overline{y}
\end{aligned} \tag{7.54}$$

Les opérateurs $\oplus, \ominus, \otimes$ et $\oslash$ sont aussi appelés *fonctions d'inclusion* des opérateurs arithmétiques $+, -, \times, /$. On peut définir de la même façon des fonctions mathématiques, comme $\sin\overline{x}$, $\cos\overline{x}$, $\log\overline{x}$, $\exp\overline{x}$, avec des intervalles. D'une manière plus générale, une fonction d'inclusion $\square f$, associée à une fonction planaire f, permet d'estimer l'ensemble des valeurs prises par f dans un domaine 2-D I. En 2-D, I est le produit cartésien entre deux intervalles monodimensionnels (ou 1-D), *i.e.* $I = \overline{x} \times \overline{y} = [\overline{x}.inf, \overline{x}.sup] \times [\overline{y}.inf, \overline{y}.sup]$.

Définition 7.17 (Fonction d'inclusion, cas général)**.** *Soit $f : \mathbb{R}^m \to \mathbb{R}^n$ une fonction donnée. $\square f$ est dite fonction d'inclusion de f si elle calcule, pour un intervalle m-dimensionnel I, l'intervalle n-dimensionnel $\square f(I)$ tel que :*

$$x \in I \Rightarrow f(x) \in \square f(I) \tag{7.55}$$

Pour construire une fonction d'inclusion complexe $\square f$, nous pouvons, par exemple, combiner des fonctions d'inclusion connues, puisque $\square f \circ g = \square f(\square g))$. Une autre approche pourrait être de développer $\square f$ grâce à des séries de Taylor, afin de réduire l'erreur induite par la fonction d'inclusion [Moore, 1966]. Un exemple du problème de la propagation de l'erreur peut être illustré par un exemple issu de [Hu *et al.*, 2002] : Soient $a = 77617.0$ et $b = 33096.0$, et l'on cherche à calculer $c = 333.75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2) + 5.5b^8 + \frac{a}{2b}$. Avec l'arithmétique en virgule flottante, on obtient $c = 1.172603\ldots$ en simple précision. En réalité, on a $c \simeq -0.8274$. Lorsque l'on calcule avec l'arithmétique d'intervalles, c est encadré par l'intervalle $\overline{c} = [-0.566.10^{23}, 0.555.10^{23}]$. On voit ici que l'intervalle résultat $\overline{c}$ encadre la valeur réelle de c, mais avec une erreur très importante. Une autre manière d'optimiser cette erreur est de réécrire les expressions arithmétiques de la fonction d'inclusion. En effet, on peut chercher à rendre $\square f$ *optimale*, *i.e.* qu'elle renvoie toujours l'intervalle le plus petit (au sens de l'inclusion) pour une évaluation donnée. Nous pouvons énoncer la propriété suivante qui permet de caractériser une telle fonction d'inclusion :

Propriété 7.5 ([Moore, 1966]). *Soit $f : \mathbb{R}^n \to \mathbb{R}$ et $\square f$ la fonction d'inclusion associée à f. Si toutes les variables n'apparaissent qu'une seule fois dans l'expression arithmétique de f, alors $\square f$ est optimale.*

Enfin, l'*arithmétique affine* [Comba et Stolfi, 1993, de Figueiredo et Stolfi, 2004] est une extension de l'arithmétique d'intervalles, où l'on représente un réel x sous forme affine : $\widehat{x} = x_0 + x_1\epsilon_1 + x_2\epsilon_2 + \ldots$, où $x_0, x_1, x_2, \ldots$ sont des réels et $\epsilon_1, \epsilon_2, \ldots$ sont des symboles de bruit. On a ainsi $\widehat{x} \pm \widehat{y} = (x_0 \pm y_0) + (x_1 \pm y_1)\epsilon_1 + \ldots + (x_n \pm y_n)\epsilon_n$. Les algorithmes d'encadrement de courbes implicites que nous décrivons ci-après sont basés sur cette arithmétique, et l'on obtient les intervalles calculés par une conversion simple : $\overline{x} = [x_0 - r, x_0 + r]$ avec $r = \sum_{i=1}^{n} |x_i|$.

7.2.2 Tracé de courbes implicites grâce à un encadrement par un ensemble d'intervalles 2-D

Grâce aux notions d'arithmétiques que nous avons décrites précédemment, il est maintenant possible d'implémenter un *oracle d'absence*. Nous traitons de techniques d'encadrement de fonction basées sur l'arithmétique d'intervalles, mais nous pourrions utiliser d'autres outils [Martin *et al.*, 2002], comme les coefficients de Bernstein [Alberti et Mourrain, 2007, Farin, 1993], les fonctions spline [Elberand et Kim, 2001], *etc.* Soit $\mathcal{C}$ une courbe implicite donnée par sa fonction planaire $f : \mathbb{R}^2 \to \mathbb{R}$, telle que $f(x, y) = 0$. Si $0 \notin \square f(R)$, alors $0 \notin f(R)$, ce qui signifie que $f(x, y) \neq 0$ pour tout point (x, y) de R. Clairement, R n'intersecte pas $\mathcal{C}$. J. M. Snyder [Snyder, 1992b] a proposé un algorithme récursif simple pour construire un ensemble d'intervalles 2-D $\mathcal{E}$ qui encadrent $\mathcal{C}$, basé sur cet oracle d'absence (voir algorithme 7.24).

Dans cet algorithme, on utilise la fonction d'inclusion $\square F$ basée sur des opérateurs logiques. Cette fonction peut avoir trois valeurs possibles, suivant la configuration de l'intervalle R testé : $\square F(R) = [0, 0]$ implique que R n'est pas solution (région *impossible*), $\square F(R) = [1, 1]$ atteste que R est une région *possible* [1], et enfin $\square F(R) = [0, 1]$ signifie que R est une région indéterminée. Ainsi, une fois un intervalle de départ $R_0 \subset \mathbb{R}^2$ choisi tel que $\mathcal{C} \cap R_0 \neq \emptyset$, on évalue tout d'abord $\square F$ sur R_0. Puis, en appliquant l'oracle d'absence, on peut déterminer si R_0 est une région possible, *i.e.* $0 \in \square f(R_0)$, ou non. On applique ainsi l'oracle d'absence sur R_0 et ses sous-intervalles. Par conséquent, nous devons choisir une *contrainte d'acceptation de solution*, qui spécifie si une région est finalement acceptée. Dans l'algorithme 7.24, elle est notée $\square A$. Par exemple, $\square A$ peut renvoyer vrai ($\square A(R) = [1, 1]$) si R satisfait $w(\square f(R)) < \delta$, où $\delta \in \mathbb{R}$ est une valeur très petite, et $w(R)$ est la largeur de R. Cette condition permet de raffiner la solution en subdivisant R en plusieurs sous-intervalles. L'algorithme continue de traiter récursivement les intervalles en les subdivisant par une approche de quadtree, jusqu'à ce que tous les intervalles solution respectent la contrainte $\square A$. H. Lopes *et al.* [Lopes *et al.*, 2001] proposent de prendre en compte la première dérivée de f dans le critère d'arrêt. En construisant un

[1] *infeasible* et *feasible region* respectivement dans [Snyder, 1992b]

Algorithme 7.24 : Algorithme `solve` [Snyder, 1992b]

entrée : une fonction implicite $f : \mathbb{R}^2 \to \mathbb{R}$,
$\mathcal{C}$ la courbe associée à $f(x, y) = 1$,
R_0 est le domaine de départ
sortie : une représentation par un ensemble d'intervalles de la courbe $\mathcal{C}$

début
soit $\square F$ une fonction d'inclusion de f, basée sur des opérations logiques;
soit $\square A$ une contrainte d'acceptation de la solution, basée sur des opérations logiques;
placer R_0 dans la pile L;
tant que $|L| \neq \emptyset$ **faire**
dépiler R de L;
évaluer $\square F$ sur R;
si $\square F(R) = [1, 1]$ **alors**
ajouter R à la solution;
sinon si $\square F(R) = [0, 0]$ **alors**
ignorer R;
`// Dans les autres cas,` $\square F(R) = [0, 1]$
sinon si $\square A(R) = [1, 1]$ **alors**
ajouter R à la solution;
sinon
diviser R en deux sous régions R_1 et R_2 et les insérer dans L;
fin

intervalle $\square f'(R)$ avec les deux composantes du gradient normalisé de f, on peut tester si R satisfait $w(\square f(I)) < \delta \vee w(\square f'(I)) < \beta$ pour un réel β donné. Il existe d'autres critères qui permettent de prendre en compte efficacement la topologie de la courbe dans un intervalle. Dans le tableau 7.10, nous exposons une liste non exhaustive de tels critères. Quelque soit le critère choisi pour stopper l'algorithme récursif, il assure que l'ensemble des intervalles $\mathcal{E}$ contient la courbe $\mathcal{C}$ et l'approxime géométriquement. En effet, l'arithmétique d'intervalles garantit que la décomposition en cellules que nous construisons converge vers $\mathcal{C}$

Proposition 7.5 (Convergence de l'arithmétique d'intervalles [Boissonnat *et al.*, 2006])**.** *La taille des intervalles* $\square f(\overline{x_1} \times \overline{x_2})$ *tend vers 0 si la taille des intervalles 1-D paramètres* $\overline{x_1}$ *et* $\overline{x_2}$ *tendent vers 0.*

Cette proposition est clairement vérifiée dans notre cadre, puisque nous subdivisons les intervalles solutions, impliquant que leur taille tend vers 0.

Une fois cette étape achevée, les algorithmes de tracé de courbes implicites construisent l'ensemble des segments de droites approximant la courbe en considérant les cellules calculées [Snyder, 1992b, Boissonnat *et al.*, 2006]. Pour ce faire, on considère

Critère	Formulation	Référence
DSA	$w(\square f(R)) < \delta,\ \delta \in \mathbb{R}$	[Snyder, 1992b]
PG en X	$\frac{\partial}{\partial y} f(x,y) \neq 0\ \forall (x,y) \in R$	[Snyder, 1992b]
DGA	$w(\square f(I)) < \delta \vee w(\square f'(I)) < \beta\ ,\ \delta, \beta \in \mathbb{R}$	[Lopes *et al.*, 2001]
PVN	$\langle \nabla f(p_1), \nabla f(p_2) \rangle \geq 0,\ \forall p_1, p_2 \in R$	[Plantinga et Vegter, 2004]

Tab. 7.10: Liste des principaux critères d'arrêt d'un processus de décomposition en intervalles. On considère ici une fonction f en un intervalle R. La décomposition spatialement adaptative et la paramétrisabilité globale sont notées respectivement DSA et PG. La décomposition géométriquement adaptative est notée DGA, et enfin la petite variation de la normale PVN.

chaque cellule $R \in \mathcal{E}$, et on réalise les opérations suivantes :

- ▷ On calcule les intersections de $\mathcal{C}$ avec les bords de R. On peut utiliser dans cette étape une méthode itérative de Newton/Rhapson, mais nous avons choisi une analyse en intervalles 1-D très précise, comme dans [Snyder, 1992b] (voir la figure 7.94).
- ▷ Cette liste I_R d'intervalles 1-D est triée suivant l'axe des X (on peut également choisir Y avec un raisonnement analogue). Dans cette étape, on considère la borne inférieure d'un intervalle pour déterminer sa place dans I_R, de même pour les intervalles alignés suivant Y.
- ▷ On parcourt l'ensemble I_R et on lie deux intersections successives par un segment si $\mathcal{C}$ les relie. Là encore, une analyse en intervalles 1-D réalisée sur le segment médian peut résoudre ce problème.

Enfin, on regroupe ensemble les segments de droites qui sont dans la même composante connexe. On peut remarquer que cette phase de l'algorithme de tracé classique implique la construction d'un ou plusieurs segments dans chaque cellule de $\mathcal{E}$. Dans la section suivante, nous utilisons les algorithmes de reconstruction géométrique et topologique sur $\mathbb{I}$-grilles (présentés dans le chapitre 3 de la partie II) afin de représenter une courbe implicite, à partir d'un encadrement en intervalles 2-D. L'approximation polygonale que nous proposons contient naturellement moins de segments que l'approche que nous venons de décrire.

7.3 Approximation polygonale de courbes implicites par des outils discrets sur $\mathbb{I}$-grilles

Dans cette section, nous nous intéressons tout d'abord aux liens que l'on peut établir entre le modèle de discrétisation par supercouverture que nous avons décrit dans le chapitre 2 de la partie I et l'arithmétique d'intervalles. Ensuite, nous traiterons de la reconstruction de courbes implicites grâce aux algorithmes du chapitre 3 de la partie II qui respectent le modèle de supercouverture. Tout d'abord, rappelons que les modèles de discrétisation permettent de garantir qu'un objet discret représente au mieux la topologie de

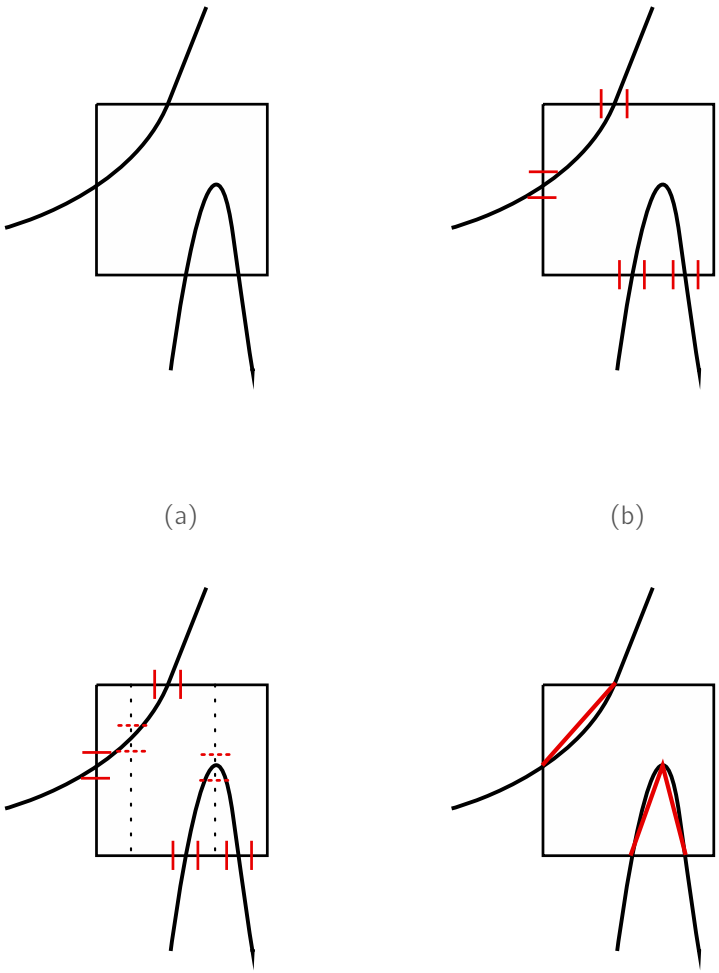


Fig. 7.94: Construction de l'approximation polygonale à partir d'un ensemble d'intervalles 2-D. A partir d'un intervalle R (a), on calcule l'intersection entre la courbe $\mathcal{C}$ et les bords de R grâce à des intervalles 1-D (b). On lie les points d'intersection par un segment si $\mathcal{C}$ les relie (d). Là encore, une analyse en intervalles 1-D peut être réalisée entre deux intersections successives (c).

l'objet Euclidien original, si la résolution de la grille est suffisamment fine. Ce phénomène de convergence vers l'objet continu en fonction de l'augmentation de la résolution de la grille est très similaire à la convergence du processus d'encadrement de courbes implicites que nous avons décrit dans la section précédente. Ainsi, il est clair que la propriété d'inclusion 2.1 que nous avons donnée pour ce modèle de discrétisation est à mettre en relation avec la propriété d'inclusion que nous avons décrite précédemment. Considérons mainte-

nant l'arithmétique $\mathbb{I}_{\mathbb{Z}+\frac{1}{2}}$ définissant les intervalles sur les demi-entiers (*i.e.* les nombres $k+\frac{1}{2}$, pour $k \in \mathbb{Z}$). Les intervalles K se construisent de la manière suivante : Pour un nombre $k \in \mathbb{Z}$, on a

$$K = \left[k - \frac{1}{2}, k + \frac{1}{2}\right], \tag{7.56}$$

et pour un réel $x \in \mathbb{R}$, on peut définir les opérateurs suivants

$$\uparrow x \uparrow = \left\lceil x + \frac{1}{2} \right\rceil - \frac{1}{2}, \tag{7.57}$$

$$\downarrow x \downarrow = \left\lfloor x + \frac{1}{2} \right\rfloor - \frac{1}{2}. \tag{7.58}$$

Grâce à ces éléments, nous pouvons énoncer le lemme suivant qui permet de formaliser le lien entre supercouverture et arithmétique d'intervalles.

Lemme 7.5 (Preuve dans [Coeurjolly, 2007])**.** *Supposons l'arithmétique sur* $\mathbb{I}_{\mathbb{Z}+\frac{1}{2}}$ *et* $f : \mathbb{R} \to \mathbb{R}$ *une application continue, alors :*

$$\mathbb{S}(f) \subseteq \bigcup_{k \in \mathbb{Z}} [K \times \square f(K)], \tag{7.59}$$

où K *est l'intervalle* $\left[k - \frac{1}{2}, k + \frac{1}{2}\right]$, $k \in \mathbb{Z}$*. De plus, si* $\square f$ *est optimale sur* $\mathbb{I}_{\mathbb{Z}+\frac{1}{2}}$*, alors :*

$$\mathbb{S}(f) = \bigcup_{k \in \mathbb{Z}} [K \times \square f(K)]. \tag{7.60}$$

Nous pouvons noter que ce lemme est aisément extensible aux dimensions supérieures, et qu'il ajoute une caractérisation arithmétique au modèle de supercouverture étendu aux $\mathbb{I}$-grilles. Ainsi, les algorithmes de reconstruction polygonale que nous avons décrit précédemment, qui respectent ce modèle, semblent complètement appropriés pour trouver un représentant correct d'un objet irrégulier issu d'une décomposition en intervalles comme l'algorithme `solve()`. De plus, les schémas de raffinement et de groupement que nous avons développés respectent aussi la propriété d'inclusion. En effet, les nouvelles cellules construites au sein d'une cellule subdivisée R sont incluses dans R, par définition. De plus, la cellule englobante que nous avons choisie pour décrire le processus de groupement inclut les cellules traitées par construction. Nous parlons donc de *schéma de raffinement local par inclusion* et de *schéma de groupement local par inclusion*. Dans la figure 7.95, nous illustrons le principe de convergence du processus de décomposition en intervalles de l'algorithme `solve()`. Nous avons également utilisé l'algorithme 3.7 pour construire le graphe de Reeb $\mathcal{G}$ associé à $\mathcal{E}$, ensemble des rectangles issus de `solve()` pour une courbe cubique $(\mathcal{C})$. L'ensemble des k-arcs qui recodent $\mathcal{E}$ est noté $\mathcal{A}$, et l'ensemble des segments de droite $\mathcal{L}$. Dans cette figure, nous pouvons remarquer que la topologie de l'objet irrégulier $\mathcal{E}$ est toujours correctement représentée. Par contre, $\mathcal{E}$ n'est pas forcément isotope à la courbe $\mathcal{C}$. Par exemple, pour $\delta = 0.5$ (second cas), le graphe de Reeb

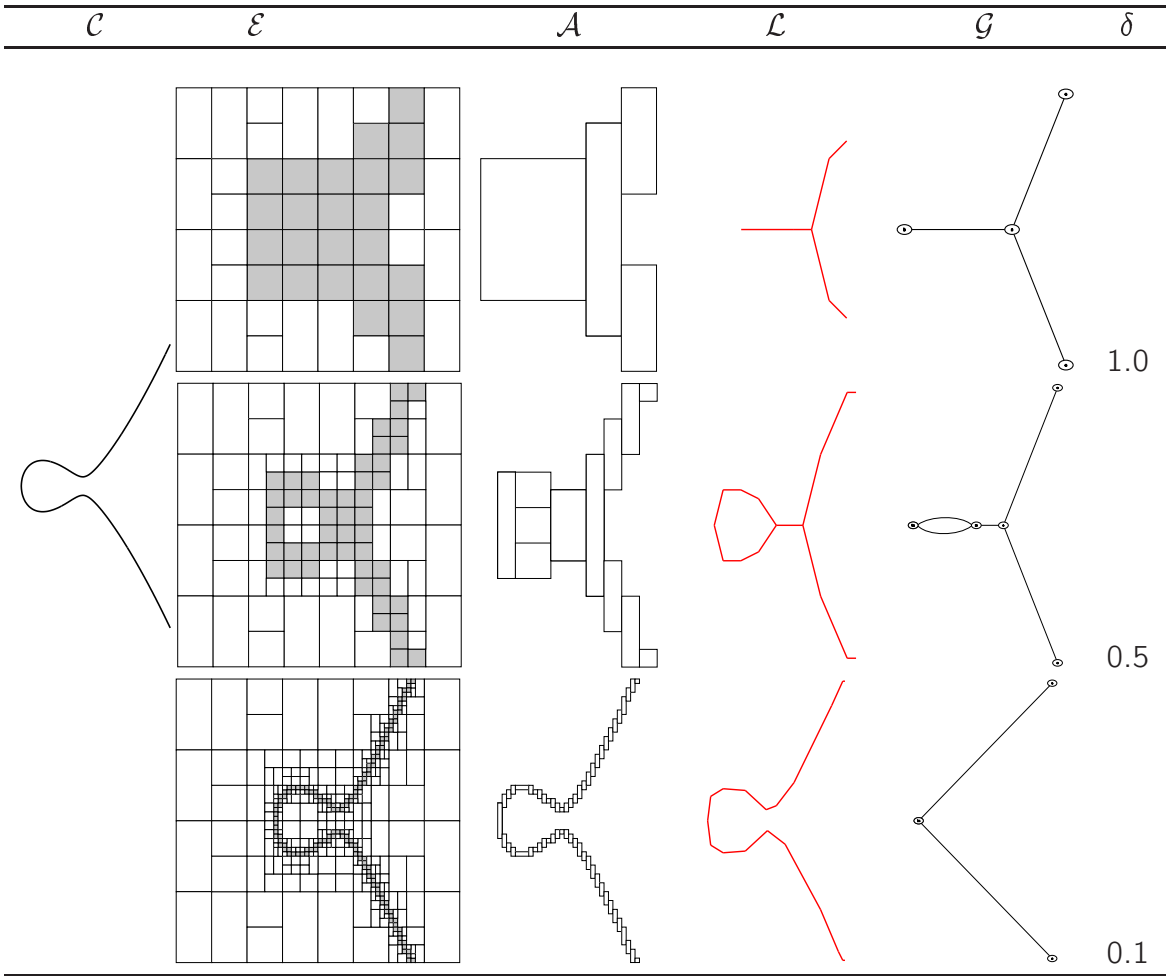


Fig. 7.95: Illustration de la convergence du processus d'encadrement par analyse en intervalles. Nous avons choisi la fonction cubique $f(x, y) = y^2 - x^3 + x - 0.5$ dans le domaine $[-4.0; 4.0] \times [-4.0; 4.0]$ (de courbe associée $\mathcal{C}$). Dans les trois cas, δ, le critère d'arrêt pour l'analyse en intervalles, représente la largeur des rectangles dans $\mathcal{E}$. Il est fixé à 1.0, 0.5 et 0.1 respectivement.

contient un trou. Cette "erreur" pourrait également survenir pour un autre algorithme de tracé de courbes implicites, puisque l'analyse d'intervalles implique que $\mathcal{E}$ contient un trou. Comme en géométrie discrète, la qualité des représentations géométrique et topologique dépend grandement du choix du critère d'arrêt [2] pendant cette étape de la reconstruction. Grâce à l'algorithme de polygonalisation de k-arcs que nous avons développé, le nombre de segments de droites est très compétitif, en comparaison des méthodes classiques d'approximation de courbes implicites, où un à plusieurs segments sont tracés dans chaque cellule de $\mathcal{E}$. Plus précisément, nous construisons généralement moins de segments que de cellules présentes dans $\mathcal{E}$. Par exemple, dans la figure 7.95, si $\delta = 1.0$, nous avons $|\mathcal{E}| = 24$,

[2] En géométrie discrète, on parlerait de résolution de la grille

$|\mathcal{L}| = 5$, alors qu'une technique classique d'approximation comme celle de J. M. Snyder aboutirait à un ensemble de 14 segments. Dans la section suivante, nous proposons des expérimentations plus poussées pour comparer la qualité et la rapidité de reconstruction de notre algorithme avec deux méthodes classiques issues de la littérature. De plus, nous présentons des exemples d'applications des algorithmes de mises à jour (raffinement et groupement de cellules) que nous avons exposés dans le chapitre 3.

7.4 Expérimentations et analyse des algorithmes d'approximation de courbes implicites proposés

Les algorithmes que nous avons présentés dans le chapitre 3 permettent de définir un système global d'approximation de courbes implicites, qui raffine ou regroupe les intervalles en considérant un ou plusieurs processus :

- ▷ Comme les autres algorithmes, nous pouvons contrôler la topologie, et mettre à jour les intervalles seulement s'ils respectent une contrainte topologique donnée.
- ▷ Contrairement à ce qui est réalisé habituellement dans ces méthodes, on peut estimer la qualité de l'approximation polygonale obtenue. Par exemple, nous proposons d'étudier la courbure locale, qui peut être une caractéristique pertinente pour améliorer la reconstruction.
- ▷ Les processus de raffinement et de groupement peuvent être gérés de manière supervisée. En fait, un utilisateur peut choisir "à la main" les cellules qu'il désire mettre à jour par ces schémas.

Dans cette section, nous illustrons deux applications de notre système pour la représentation géométrique et topologique de courbes implicites. Nous comparons en premier lieu la qualité et la rapidité de notre méthode avec des méthodologies classiques et nous montrons ensuite des exemples de processus manuels de raffinement et de groupement. Enfin, nous présentons un programme de raffinement automatique basé sur la courbure locale de la reconstruction polygonale de la courbe.

7.4.1 Reconstruction polygonale statique de courbes implicites

Nous comparons la rapidité et la qualité de notre méthode avec deux algorithmes classiques de tracé de courbes implicites [Lopes *et al.*, 2001, Snyder, 1992b]. Nous avons choisi plusieurs fonctions f que nous approximons dans un domaine $\Omega = [-\omega;\omega] \times [-\omega;\omega] \subset \mathbb{R}^2$. Nous considérons dans un premier temps deux versions pour l'étape de décomposition en intervalles. Dans la décomposition spatialement adaptative [Snyder, 1992b], nous testons si la taille des cellules subdivisées R est suffisamment petite pour une bonne approximation : $w(\square f(R)) < \delta;\ \delta \in \mathbb{R}$ est notre contrainte d'acceptation. La décomposition géométriquement adaptative [Lopes *et al.*, 2001] ajoute la première dérivée pour construire moins de cellules dans $\mathcal{E}$. Dans ce cas, nous devons vérifier si une cellule R respecte $w(\square f(R)) < \delta \vee w(\square f'(R)) < \beta,\ \delta,\beta \in \mathbb{R}$. Dans nos

expérimentations, nous avons choisi $\delta = \frac{R}{2^{10}}$ et β variable. Nous rappelons que $w(\square f'(R))$ est construit avec les deux composantes du gradient normalisé de f. Une fois l'ensemble des cellules $\mathcal{E}$ construit, nous proposons de construire une approximation polygonale avec l'algorithme de J. M. Snyder [Snyder, 1992b], où les intersections entre la courbe $\mathcal{C}$ et une cellule R sont calculées grâce à l'arithmétique d'intervalles. Nous présentons donc une comparaison (en qualité et temps d'exécution) entre quatre méthodes.

- ▷ $\texttt{IA}_\texttt{s}$ [Snyder, 1992b] est l'algorithme original de J. M. Snyder, utilisant uniquement l'analyse en intervalles pour approximer $\mathcal{C}$.
- ▷ $\texttt{IA}_\texttt{g}$ [Lopes *et al.*, 2001] considère aussi la première dérivée de f pendant la décomposition en intervalles.
- ▷ Nous notons également l'utilisation de l'algorithme 3.7 sur les cellules construites par ces méthodes $\texttt{DG}_\texttt{s}$ and $\texttt{DG}_\texttt{g}$ respectivement.

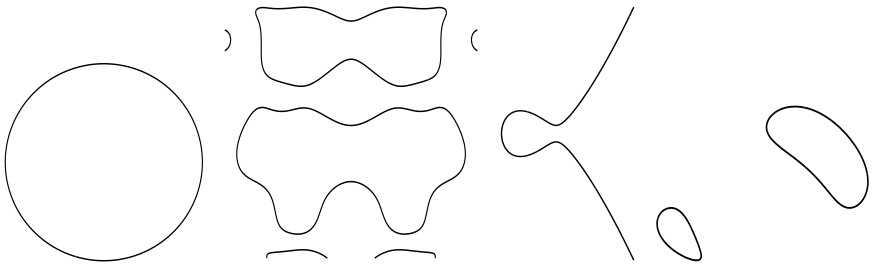
Dans la liste suivante sont présentées les équations $f(x, y) = 0$ associées aux courbes que nous avons choisies pour ces expériences, avec la largeur du domaine 2-D ω (voir figure 7.4.1 pour des tracés simples de ces courbes) :

Cercle $x^2 + y^2 - 1 = 0$, $\omega = 1.30$

Trigo $x^2 + y^2 + \cos(2\pi x) + \sin(2\pi y) + \sin(2\pi x^2)\cos(2\pi y^2) - 1 = 0$, $\omega = 1.1$ (issu de [Snyder, 1992b])

Cubique $y^2 - x^3 + x - 0.5 = 0$, $\omega = 5.2$ (issu de [Lopes *et al.*, 2001])

Deux blobs $\frac{601}{9} - \frac{872}{3}x + 544x^2 - 512x^3 + 256x^4 - \frac{2728}{9}y + \frac{2384}{3}xy - 768x^2y + \frac{5104}{9}y^2 - \frac{2432}{3}xy^2 + 768x^2y^2 - 512y^3 + 256y^4 = 0$, $\omega = 0.5$ (issus de [Martin *et al.*, 2002])



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

Fig. 7.96: Un aperçu des quatre courbes testées. De la gauche vers la droite : *cercle*, *trigo*, *cubique*, et *deux blobs*.

Nous présentons dans la table 7.11 les résultats de nos comparaisons entre les quatre méthodes. Le niveau k est considéré dans le calcul de δ : $\delta = \frac{R}{2^k}$. Pour chaque test, nous calculons un ensemble de segments de droites $\mathcal{L}$. Nous donnons le nombre de segments $\#\mathcal{L}$ pour mettre en avant la complexité de l'approximation polygonale, et par conséquent le temps requis pour la tracer. Le temps d'exécution $\mathcal{T}$ ne prend pas en compte l'étape d'analyse en intervalles, et est exposé en milli-secondes. Pour évaluer la qualité de chaque

méthode, nous avons choisi d'estimer la distance entre un point $p_i = (x_i, y_i) \in \mathcal{L}$ et $\mathcal{C}$ avec l'arithmétique d'intervalles. Par construction, nous savons que la distance entre p_i et $\mathcal{C}$ est dans le pire cas $\delta\sqrt{2}$. Donc, nous calculons une décomposition en intervalles très fine dans le domaine $[x_i + \delta\sqrt{2}, x_i - \delta\sqrt{2}] \times [y_i - \delta\sqrt{2}, y_i + \delta\sqrt{2}]$. Puis, nous cherchons l'intervalle 2-D le plus proche $\overline{\epsilon}_i$ de p_i. Nous considérons le centre de $\overline{\epsilon}_i$, que nous notons $\overline{\epsilon}_i.mid$. Une fois que nous avons calculé la distance Euclidienne $d_e(p_i, \overline{\epsilon}_i.mid)$ pour chaque point p_i, nous déterminons l'erreur moyenne de l'approximation polygonale sur l'ensemble des n_p points de $\mathcal{L}$

$$D = \frac{1}{n_p} \sum_{i=1}^{i=n_p} d(p_i, \overline{\epsilon}_i.mid). \tag{7.61}$$

Comme nous pouvons le constater dans la figure 7.97, notre méthode est très compétitive pour des descriptions fines de la courbe (*e.g.* $k = 10$ avec $\mathtt{DG_s}$) car elle approxime efficacement $\mathcal{C}$ avec moins de segments (et donc un temps d'exécution réduit) que dans la méthode classique $\mathtt{IA_s}$. Ce comportement peut être observé même pour une fonction très complexe comme *deux blobs* : nous traçons la courbe en 0.1 secondes avec une erreur d'environ 4.0×10^{-4}. Puisque nous ne considérons que les cellules pour calculer l'approximation polygonale, un pavage plus grossier du plan avec des cellules de tailles trop variables altère significativement la reconstruction. Par exemple, les niveaux les plus bas de décomposition k et les hautes valeurs de β impliquent que les segments calculés avec $\mathtt{DG_s}$ sont moins corrects que ceux déduits de l'approche $\mathtt{IA_g}$. L'avantage de construire des cellules plus grandes où une dérivée de f est petite engendre que les algorithmes classiques sont plus adaptés que le notre, comme on peut le voir dans la figure 7.97. En fait, Lopes *et al.* ont choisi ce critère pour calculer moins de segments dans $\mathcal{L}$, tout en gardant une qualité d'approximation très semblable à celle proposée par J. M. Snyder. D'une manière plus générale, comme D représente une erreur moyenne, notre méthode peut être jusqu'à environ 1 000 fois moins précise qu'une des deux méthodes classiques (par exemple en comparant $\mathtt{IA_g}$ et $\mathtt{DG_g}$, avec *de blobs* et $\beta = 0.20$). Néanmoins, la méthode que nous avons proposé aboutit à une reconstruction polygonale efficace, puisque nous obtenons rapidement (un temps toujours très inférieur à 1 seconde) peu de segments avec une approximation correcte de la courbe. Nous pouvons choisir un niveau de décomposition très élevé (*e.g.* $k = 10$) par l'approche de J. M. Snyder, et nous obtenons une approximation avec une erreur moyenne très faible, de l'ordre de 5.0×10^{-4}.

7.4.2 Quelques exemples d'opérations de mise à jour de la reconstruction d'une courbe implicite

Dans cette section, nous avons choisi d'étudier *trigo*, la fonction f dans l'intervalle $\Omega = [-1.1; 1.1] \times [-1.1; 1.1]$, telle que $f(x, y) = x^2 + y^2 + \cos(2\pi x) + \sin(2\pi y) + \sin(2\pi x^2)\cos(2\pi y^2)$ issu de [Snyder, 1992b]. Comme nous l'avons présenté dans ce chapitre, nous discrétisons en premier lieu cette courbe en un ensemble $\mathcal{E}$ grâce à l'approche de J. M. Snyder. La largeur δ des intervalles finalement construits est fixée : $\delta = 0.1$

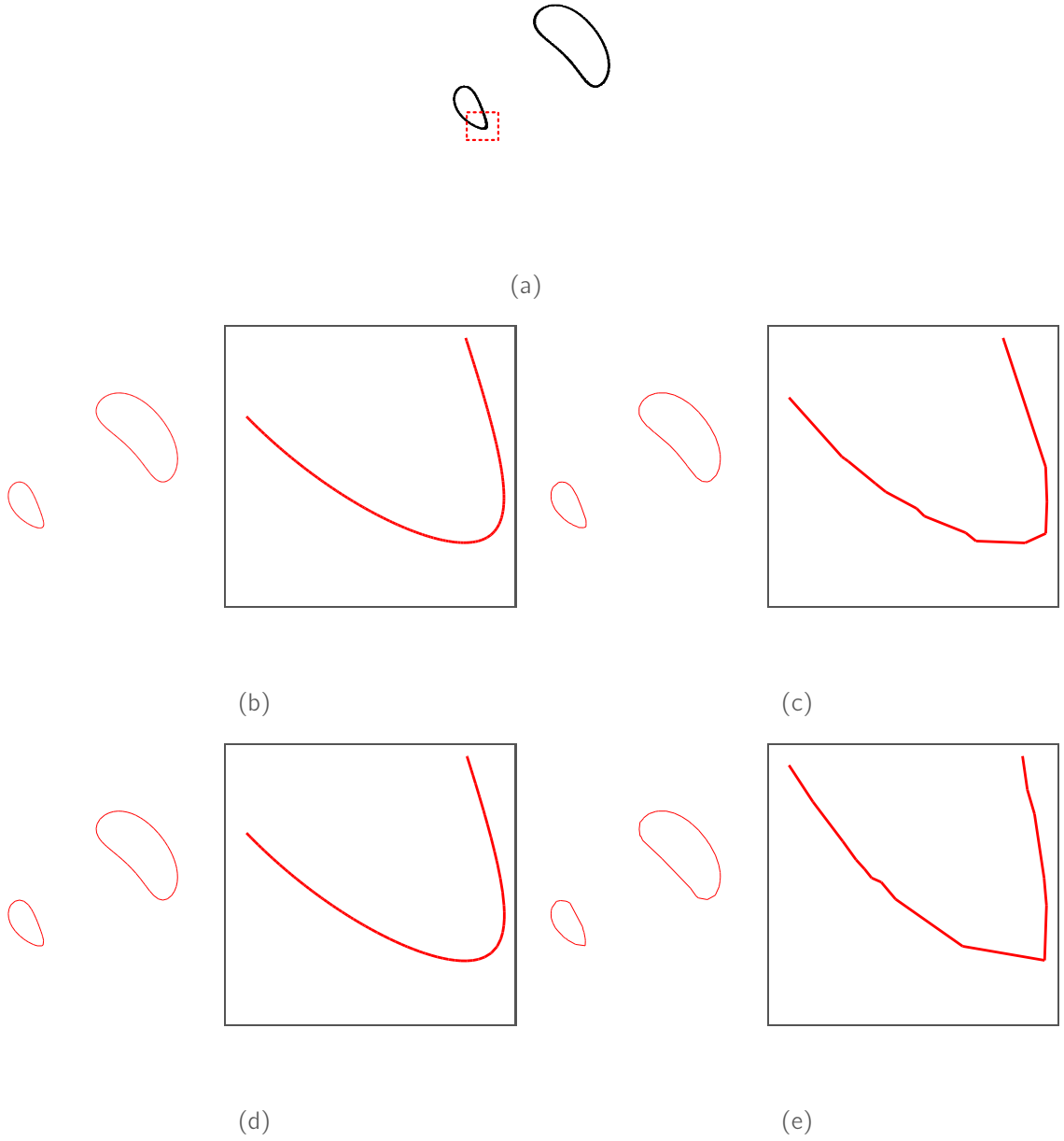


Fig. 7.97: L'approximation polygonale de la courbe *deux blobs* (de [Martin *et al.*, 2002]) obtenue pour les quatre algorithmes testés. Nous avons présenté une région zoomée avec un rectangle en pointillés en (a) pour chaque cas. (b) et (c) correspondent à $\mathtt{IA}_\mathtt{s}$ et $\mathtt{DG}_\mathtt{s}$ au niveau $k = 10$, tandis que (d) et (e) sont tracés grâce à $\mathtt{IA}_\mathtt{g}$ et $\mathtt{DG}_\mathtt{g}$ définis par $\beta = 0.20$. Dans cette région, nous avons observé que le nombre de segments est 204 pour $\mathtt{IA}_\mathtt{s}$, 115 pour $\mathtt{IA}_\mathtt{g}$ et 14 pour $\mathtt{DG}_\mathtt{s}$ et $\mathtt{DG}_\mathtt{g}$.

$\mathtt{IA}_s$ [Snyder, 1992b]

f	k	$\#\mathcal{L}$	T	D
Cercle	4	52	21	$6.38\ 10^{-4}$
	6	164	100	$1.69\ 10^{-4}$
	8	620	252	$4.23\ 10^{-5}$
	10	2384	951	$1.02\ 10^{-5}$
Trigo	4	128	133	$5.25\ 10^{-4}$
	6	354	367	$1.62\ 10^{-4}$
	8	1482	1627	$3.02\ 10^{-5}$
	10	4778	5249	$1.02\ 10^{-5}$
Cubique	4	30	41	$2.62\ 10^{-3}$
	6	120	90	$6.67\ 10^{-4}$
	8	422	185	$1.63\ 10^{-4}$
	10	1642	722	$4.20\ 10^{-5}$
2 blobs	4	52	282	$1.72\ 10^{-4}$
	6	148	675	$4.31\ 10^{-5}$
	8	518	1580	$1.08\ 10^{-5}$
	10	1866	4978	$2.69\ 10^{-6}$

$\mathtt{DG}_s$

f	k	$\#\mathcal{L}$	T	D
Cercle	4	10	1	$1.02\ 10^{-1}$
	6	20	2	$1.11\ 10^{-2}$
	8	56	13	$2.85\ 10^{-3}$
	10	182	199	$7.77\ 10^{-4}$
Trigo	4	44	4	$4.08\ 10^{-2}$
	6	58	6	$1.28\ 10^{-2}$
	8	143	84	$2.55\ 10^{-3}$
	10	366	928	$7.31\ 10^{-4}$
Cubique	4	11	1	$1.45\ 10^{-1}$
	6	13	1	$5.37\ 10^{-2}$
	8	29	6	$1.16\ 10^{-2}$
	10	80	81	$2.69\ 10^{-3}$
2 blobs	4	4	1	$1.40\ 10^{-2}$
	6	6	5	$5.88\ 10^{-3}$
	8	30	19	$1.19\ 10^{-3}$
	10	72	157	$4.29\ 10^{-4}$

$\mathtt{IA}_g$ [Lopes *et al.*, 2001]

f	β	$\#\mathcal{L}$	T	D
Cercle	0.05	196	76	$3.67\ 10^{-5}$
	0.10	100	41	$6.79\ 10^{-5}$
	0.20	52	19	$2.56\ 10^{-4}$
	0.40	28	12	$2.07\ 10^{-4}$
Trigo	0.20	2800	3032	$1.20\ 10^{-5}$
	0.25	2604	2808	$1.21\ 10^{-5}$
	0.30	2374	2573	$1.23\ 10^{-5}$
	0.35	2110	2284	$1.27\ 10^{-5}$
Cubique	0.20	1298	604	$1.12\ 10^{-5}$
	0.25	1248	581	$1.24\ 10^{-5}$
	0.30	1214	563	$1.40\ 10^{-5}$
	0.35	1150	533	$1.45\ 10^{-5}$
2 blobs	0.20	1262	3460	$8.31\ 10^{-6}$
	0.25	1093	3073	$8.32\ 10^{-6}$
	0.30	975	2547	$8.43\ 10^{-6}$
	0.35	739	2009	$8.48\ 10^{-6}$

$\mathtt{DG}_g$

f	β	$\#\mathcal{L}$	T	D
Cercle	0.05	26	4	$0.98\ 10^{-3}$
	0.10	20	1	$1.57\ 10^{-2}$
	0.20	12	< 1	$4.53\ 10^{-2}$
	0.40	6	< 1	$1.11\ 10^{-1}$
Trigo	0.20	304	521	$1.17\ 10^{-3}$
	0.25	278	385	$1.31\ 10^{-3}$
	0.30	224	265	$1.70\ 10^{-3}$
	0.35	220	183	$1.99\ 10^{-3}$
Cubique	0.20	101	48	$2.11\ 10^{-3}$
	0.25	98	45	$2.60\ 10^{-3}$
	0.30	99	40	$3.83\ 10^{-3}$
	0.35	81	35	$7.42\ 10^{-3}$
2 blobs	0.20	70	85	$4.13\ 10^{-3}$
	0.25	75	71	$4.80\ 10^{-3}$
	0.30	68	56	$5.49\ 10^{-3}$
	0.35	61	40	$6.20\ 10^{-3}$

Tab. 7.11: La comparaison de notre contribution avec les méthodologies classiques. La distance moyenne D entre $\mathcal{L}$ et $\mathcal{C}$ est estimée grâce à l'arithmétique d'intervalles. Le temps T est présenté en milli-secondes.

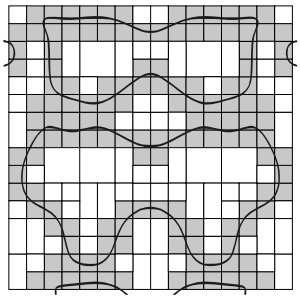
(voir la figure 7.98). Nous exposons maintenant quelques expérimentations du processus de raffinement (algorithme 3.8). En fait, nous mettons à jour successivement les éléments calculés par l'algorithme 3.7 avec des raffinements locaux par inclusion. Nous sélectionnons plusieurs cellules à chaque temps, et nous utilisons l'algorithme 3.8. Les nouvelles cellules contenues dans le raffinement sont déduites par le même algorithme d'analyse en intervalles, avec une largeur plus petite ($\delta = 0.05$). Dans la figure 7.98, nous montrons quatre temps de mise à jour. La première procédure de mise à jour, avec les données indiquées à $t = 0$, ne modifient pas le graphe de Reeb ($\mathcal{G}_0 = \mathcal{G}_1$) tandis que la reconstruction géométrique ($\mathcal{A}_1$ et $\mathcal{L}_1$) est plus précise. Au temps $t = 1$, nous choisissons deux cellules qui modifient le graphe de Reeb $\mathcal{G}_1$ en divisant les arêtes $m - m$ et $s - s$. Enfin, sélectionner les quatre cellules permet de créer deux nouvelles arêtes dans $\mathcal{G}_3$. Pour les opérations de groupement (algorithme 3.9), nous montrons dans la figure 7.98 les résultats de trois mises à jour de la reconstruction de la courbe. La première ne modifie pas le graphe de Reeb, la suivante permet de construire un graphe de Reeb plus simple en supprimant une arête $b - m$, et en connectant les deux noeuds voisins s. La dernière modification créée un noeud *split* en connectant deux arêtes.

7.4.3 Un schéma de raffinement automatique basé sur la courbure locale

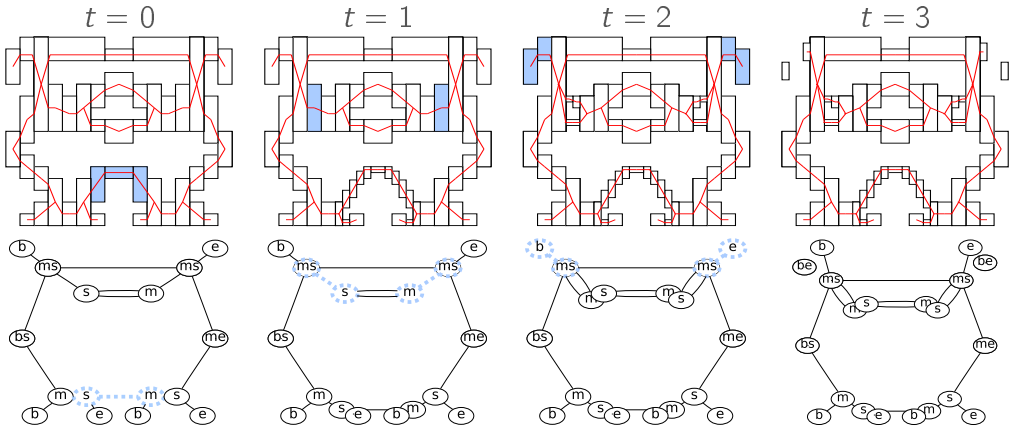
Ici, nous proposons d'étudier la polygonalisation initiale $\mathcal{L}$ d'un objet irrégulier $\mathcal{E}$ pour la raffiner lorsqu'un point dans $\mathcal{L}$ possède un rayon de courbure élevé. Le rayon de courbure σ_q d'un point $q \in \mathcal{L}$ est calculé en considérant les trois points consécutifs $p, q, r \in \mathcal{L}$ et le cercle $C_{(p,q,r)}$ passant par p, q et r

$$\sigma_q = \frac{1}{rayon(C_{(p,q,r)})}. \tag{7.62}$$

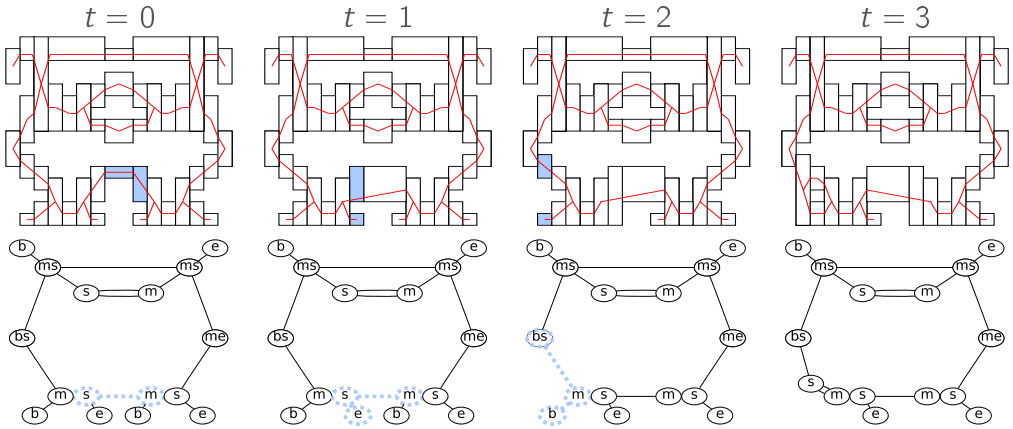
Avec ce processus, nous illustrons l'utilisation d'un critère de raffinement automatique. Plus précisément, nous fixons en premier lieu un rayon de courbure maximal $\overline{\sigma}$, et nous cherchons dans chaque k-arc de $\mathcal{A}$ une cellule R où est situé un point q avec un rayon $\sigma_q > \overline{\sigma}$. En fait, nous supposons que la courbe réelle $\mathcal{C}$ est lisse et de courbure bornée. Finalement, nous raffinons la cellule R et nous mettons à jour la reconstruction avec l'algorithme 3.8. Nous pouvons raffiner $\mathcal{L}$ itérativement jusqu'à ce que la polygonalisation respecte cette inégalité en chaque point $q \in \mathcal{L}$. Dans la figure 7.100, nous exposons quelques itérations de ce processus automatique de raffinement pour la fonction *cercle* d'équation $x^2 + y^2 - 1.0 = 0$. Nous avons fixé dans cet exemple $\overline{\sigma} = 0.1$ et la largeur des cellules issues des analyses en intervalles suivantes $\delta_1 = 0.1$, et $\delta_2 = 0.05$. Pour chaque itération sont présentés le recodage en k-arcs $\mathcal{A}$ et l'approximation polygonale $\mathcal{L}$. Le graphe de courbure est également illustré, où nous traçons la courbure σ_q pour chaque point q dans le sens anti-trigonométrique, et la courbure maximale $\overline{\sigma}$ est la ligne en pointillés. Enfin, les cellules raffinées et les points de haute courbure sont mis en valeur pour chaque étape. Dans la figure 7.99 est illustré le résultat de ce processus.



(a)

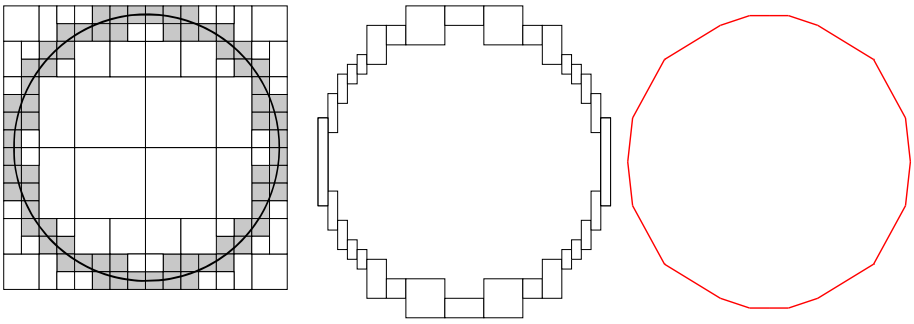


(b)



(c)

Fig. 7.98: Résultats de différents schémas de raffinement et de groupement. La courbe associée à $x^2 + y^2 + \cos(2\pi x) + \sin(2\pi y) + \sin(2\pi x^2)\cos(2\pi y^2) = 1$ sur $[-1.1; 1.1] \times [-1.1; 1.1]$ est discrétisée par intervalles grâce à l'algorithme `solve()` (a). Pour chaque procédure de mise à jour, nous présentons l'ensemble des cellules recodant l'objet traité $\mathcal{E}$, et la polygonalisation obtenue. Les cellules sélectionnées $R_1, R_2, \ldots R_n$ pour l'itération suivante sont présentées en bleu, et les arêtes associées dans le graphe de Reeb sont en pointillés.



(a) (b) (c)

Fig. 7.99: Résultat du processus de raffinement automatique décrit dans la figure 7.100. En (a) est donné l'ensemble des cellules $\mathcal{E}$ en entrée, en (b) l'ensemble des k-arcs recodant finalement $\mathcal{E}$, est en (c) la polygonalisation obtenue par le schéma automatique.

7.5 Bilan et perspectives

Dans ce chapitre, nous avons proposé un système global d'approximation de courbes implicites par des segments de droites. Nous avons décrit l'arithmétique d'intervalles, outil classique qui permet de développer des techniques de tracé robustes et rapides. Nous avons montré que la propriété d'inclusion de cette arithmétique est naturellement liée à la propriété d'inclusion du modèle de supercouverture sur $\mathbb{I}$-grilles. Clairement, les algorithmes de reconstruction géométrique et topologique d'objets irréguliers isothétiques sont adaptés pour représenter de manière efficace un objet $\mathcal{E}$ issu d'une analyse en intervalles 2-D pour encadrer une courbe planaire. Nous avons montré que notre approche est très compétitive (en considérant la qualité d'approximation et le temps d'exécution), comparée aux méthodologies classiques.

L'analyse en intervalles par un algorithme comme `solve()` est déterminante pour une bonne approximation de la courbe. Nous aimerions étudier d'autres théories très répandues pour optimiser cette étape comme les coefficients de Bernstein ou les fonctions spline. De plus, nous avons montré que la contrainte d'acceptation (*e.g.* la paramétrisabilité de J.M. Snyder) peut prendre en compte efficacement les variations de la courbe au sein d'un intervalle testé. La technique de cône de visibilité que nous proposons pour vectoriser un k-arc élémentaire de $\mathcal{E}$ pourrait être guidée par l'arithmétique d'intervalles pour améliorer la mise à jour du cône. Ainsi, cela réduirait la préimage calculée en considérant la courbe réelle représentée. Enfin, nous aimerions travailler sur une extension 3-D de notre système, où l'objectif serait d'approximer une surface implicite par des segments de plan. La plupart des algorithmes de la littérature que nous avons présentées sont aisément adaptables en 3-D, et nous pourrions là encore comparer notre contribution avec ces techniques.

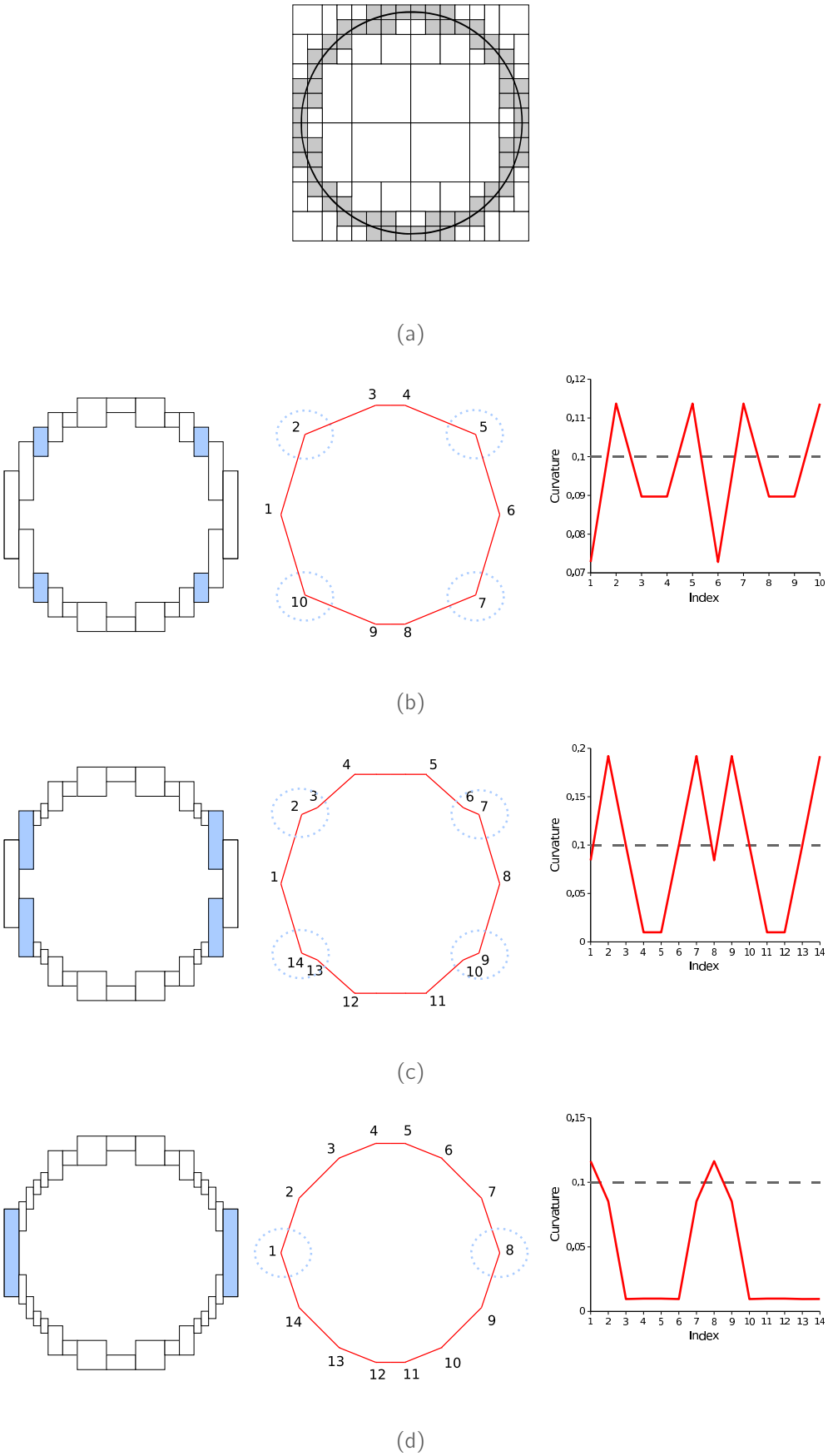


Fig. 7.100: Quelques itérations du schéma de raffinement automatique proposé. Nous montrons en haut les cellules en entrée de notre système. Le résultat final de ces trois étapes de raffinement est illustré dans la figure 7.99.

Conclusion et perspectives générales

Dans cette partie de bilan, nous reprenons nos différentes contributions exposées dans les trois parties de cette thèse. Pour chacune d'entre elles, nous indiquons également quelques pistes de recherches.

Un modèle générique de grille irrégulière isothétique

Les représentations par grilles irrégulières isothétiques sont très répandues dans les domaines liés à l'image 2-D et 3-D. Nous avons étudié notamment le codage par RLE, les décompositions quadtree/octree et le kd-tree, ainsi que certaines de leurs applications, à savoir la modélisation géométrique, les simulations de phénomènes naturels et l'accélération des algorithmes de la réalité augmentée et du jeu vidéo. Grâce à cette dernière étude, nous avons aussi montré les différentes optimisations possibles (en espace mémoire, vitesse d'accès, facilité de mise à jour, *etc.*) de ces structures classiques pour l'informatique graphique.

Nous avons proposé un modèle de $\mathbb{I}$-grille générique qui permet de représenter n'importe quelle grille irrégulière isothétique, et de l'organiser grâce à un ordre lexicographique. Nous utilisons dans cette thèse cet ordre pour parcourir une $\mathbb{I}$-grille de manière linéaire (selon le nombre total de cellules), grâce à une approche de *scan line*. Les différentes notions que nous avons introduites dans cette partie permettent de caractériser les objets contenus dans une $\mathbb{I}$-grille (k-arcs, k-objets, *etc.*). Nous nous sommes aussi intéressés au modèle de discrétisation par supercouverture étendu et aux distances discrètes sur ces grilles. La fin de cette partie concerne les structures de données possibles pour représenter une $\mathbb{I}$-grille, et nous avons proposé un algorithme simple de construction du graphe d'adjacence.

Pour aller plus loin

Dans notre étude sur les algorithmes liés au jeu vidéo, nous avons envisagé de mener une étude ultérieure sur la définition d'une structure de données générique qui permettrait d'accélérer ces algorithmes. La masse de données importante (maillages triangulaires par exemple) serait un support utile pour tester les structures de données accélératrices, et construire par exemple des modèles hybrides.

La théorie du signal non-uniforme que nous avons présentée dans cette thèse contient un ensemble d'outils mathématiques généralisés pour des signaux irréguliers ou incomplets. Il serait intéressant de trouver d'autres applications que celles proposées jusqu'à présent (reconstruction et compression principalement).

En ce qui concerne le sujet précis de notre thèse, on pourrait en premier lieu s'intéresser à l'adaptation d'autres définitions géométriques discrètes sur $\mathbb{I}$-grille comme la convexité, la circularité, *etc.* Cela permettrait de donner d'autres caractéristiques de formes dans notre application de reconnaissance de caractères ambigus par exemple. Une caractéristique intéressante pourrait être enfin la (non) régularité d'une forme et sa discrétisation sur $\mathbb{I}$-grille, comme ce qui est étudié dans [Stelldinger et Terzic, 2008] (pour un exemple

de r-régularité, voir la figure 1). Ces notions seraient un support idéal pour étudier les outils de la morphologie mathématique sur $\mathbb{I}$-grille (dilatation, érosion, *etc.*).

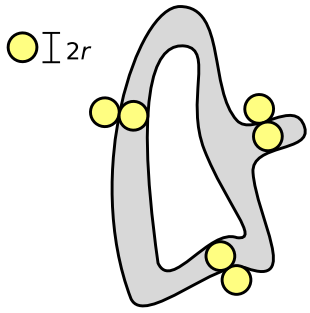


Fig 1: Une forme 2-D r-régulière [Stelldinger et Terzic, 2008]. Pour chaque bord d'une forme r-régulière, il existe une boule ouverte tangente de rayon r, à l'intérieur et à l'extérieur.

Reconstruction géométrique et topologique d'objets complexes sur $\mathbb{I}$-grille

Dans ce chapitre, nous avons développé un système complet qui représente la topologie d'un objet complexe 2-D sur $\mathbb{I}$-grille grâce à un graphe de Reeb, puis de calculer une polygonalisation exacte qui respecte le modèle de supercouverture. Grâce à l'ordre lexicographique que nous avons défini sur une $\mathbb{I}$-grille, nous avons établi naturellement un lien entre l'approche *scan line* pour parcourir les cellules, et la définition de graphe de Reeb. De plus, nous avons obtenu un algorithme incrémental rapide pour calculer cette représentation. La polygonalisation est réalisée par une approche de cône de visibilité, qui permet de calculer un représentant euclidien d'un k-arc en temps linéaire (en le nombre de cellules du k-arc). Nous avons proposé une nouvelle version de cet algorithme, où l'on cherche à optimiser la taille des cônes construits pour obtenir des segments de droite plus fidèles à la forme de l'objet original.

Ces représentations topologique et géométrique peuvent être mises à jour rapidement, lorsqu'une cellule est raffinée (*i.e.* subdivisée), ou quand plusieurs cellules sont regroupées ensemble. La définition du graphe de Reeb nous permet d'effectuer des opérations locales, que ce soit pour la mise à jour des éléments topologiques, ou le calcul d'une nouvelle reconstruction polygonale. Nous avons présenté diverses applications de ces outils, notamment dans le cadre de l'analyse d'images et de l'approximation de courbes implicites (qui est détaillée davantage dans la partie III de cette thèse).

Pour aller plus loin

Pour développer notre système sur des $\mathbb{I}$-grilles 3-D, nous devrions tout d'abord étudier la définition de la fonction de hauteur (ou fonction de Morse) nécessaire à la construction

du graphe de Reeb. Plusieurs approches existent, et ont été longuement étudiées pour des maillages triangulaires. Pour notre part, nous préférons éviter de changer de représentation, c'est-à-dire trianguler l'objet irrégulier 3-D (par un marching cubes par exemple), puis construire un graphe de Reeb associé à ce maillage. En effet, cette conversion implique un coût significatif en terme de temps de calcul. De plus, nous souhaitons conserver la structure de $\mathbb{I}$-grille dans nos traitements. La construction du graphe de Reeb implique alors la définition de points critiques (selles, maximas, et minimas) et la notion de lacet sur des $\mathbb{I}$-grilles 3-D.

La reconstruction polyhédrale d'une surface 3-D irrégulière pourrait être une piste de recherche intéressante. Elle pourrait être guidée par le graphe de Reeb, comme nous l'avons proposé en 2-D. Nous souhaiterions adapter en premier lieu notre technique de cône de visibilité, puis étudier l'approche par préimage généralisée de M. Dexet [Dexet et Andres, 2006]. Le problème se ramène dans notre système à reconstruire chaque portion de surface 3-D avec des morceaux de plans appartenant à la préimage (voir figure 2). Dans le cadre du rendu de surfaces implicites par l'arithmétique d'intervalles, ces méthodes permettraient de construire une décomposition en morceaux de plans plus compacte que les maillages triangulaires obtenus par des techniques classiques.

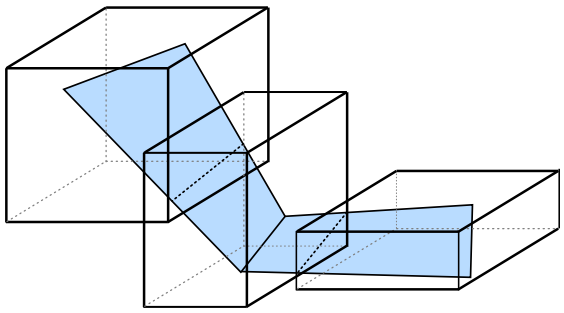


Fig 2: La reconstruction d'une portion de surface 3-D irrégulière par des morceaux de plans.

Algorithmes de transformée en distance sur $\mathbb{I}$-grille

La transformée en distance est une technique largement étudiée depuis une vingtaine d'années. Les algorithmes permettant de la calculer sont nombreux, et ils sont généralement basés sur la construction d'un diagramme de Voronoï discret. Pour obtenir cette transformation sur $\mathbb{I}$-grille, nous avons tout d'abord étudié le lien entre diagramme de Voronoï et distance entre cellules. Sur $\mathbb{I}$-grille, nous sommes ainsi parvenu à définir deux distances entre une cellule de premier plan et sa plus proche cellule de fond : l'une basée sur la distance entre le centre des cellules, l'autre en considérant la frontière des cellules (qui ont aboutit respectivement à la $\mathbb{I}$-CDT et la $\mathbb{I}$-BDT). Dans les deux cas, nous avons mis en évidence le lien avec deux types de diagrammes de Voronoï ; le premier est basé sur un ensemble de points, et le second sur un ensemble de segments.

Nous avons ensuite adapté trois techniques de transformée en distance sur $\mathbb{I}$-grille. Nous les avons développées car elles répondaient successivement à des problématiques différentes. L'extension de H. Breu [Breu *et al.*, 1995] permet de calculer la $\mathbb{I}$-CDT en temps linéaire (en le nombre de cellules de la $\mathbb{I}$-grille), mais elle n'est pas extensible à d-D, et est difficilement adaptable au calcul de la $\mathbb{I}$-BDT. Ensuite, nous avons proposé de réécrire l'algorithme décrit dans [Saito et Toriwaki, 1994] (rendu linéaire grâce aux travaux de [Hirata, 1996, Meijster *et al.*, 2000]), qui permet d'obtenir la $\mathbb{I}$-CDT et la $\mathbb{I}$-BDT en d-D. Néanmoins, elle possède une complexité en temps non-optimale, en considérant la structure de données que nous avons mise en place (matrice irrégulière). Enfin, notre approche basée sur [Maurer *et al.*, 2003] résout ce dernier problème avec une complexité linéaire en la taille de la matrice irrégulière.

Pour aller plus loin

Dans cette thèse, nous avons élaboré un ensemble d'expérimentations pour comparer, en 2-D, les algorithmes qui calculent la $\mathbb{I}$-CDT et la $\mathbb{I}$-BDT que nous avons développés. Nous pourrions faire de même en étudiant le comportement des techniques d-D sur les grilles les plus répandues en 3-D (anisotropes, FCC, BCC, octree, *etc.*).

Nous avons traité dans cette thèse de la transformée en distance basée sur la distance euclidienne. Nous pourrions généraliser nos travaux à d'autres métriques (chanfrein, chessboard, *etc.*). De même que les auteurs de [Maurer *et al.*, 2003] indiquent que leur algorithme est extensible à plusieurs distances, on pourrait améliorer notre adaptation sur $\mathbb{I}$-grille, et ainsi prendre en compte différentes métriques.

Nous souhaitons également étudier la construction du diagramme de Voronoï en d-D. Nous avons en effet montré qu'elle peut être envisagée par plusieurs approches classiques (*e.g.* l'enveloppe convexe), mais on pourrait également s'inspirer d'une technique de subdivision, comme le montrent les auteurs de [Park *et al.*, 2005] en 2-D et 3-D. Cette approche permettrait de calculer la $\mathbb{I}$-CDT en chaque cellule de premier plan en d-D en construisant et en manipulant le diagramme des cellules de fond (voir figure 3).

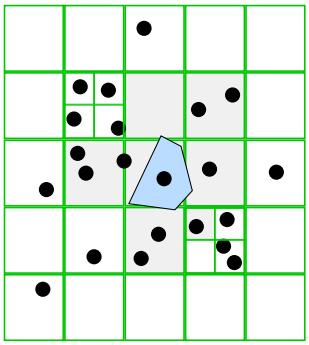


Fig 3: Diagramme de Voronoï par grilles de subdivision en 2-D [Park *et al.*, 2005]. Pour une cellule de Voronoï donnée (ici au centre), la principale problématique est de déterminer quelles parties des grilles parcourir pour construire les cellules adjacentes.

Enfin, nous souhaitons développer les outils nécessaires à la construction d'un axe médian réversible des objets que nous traitons sur $\mathbb{I}$-grille étiquetée. La notion de boule ouverte irrégulière (partie I de cette thèse) sera un support intéressant pour définir un algorithme de transformée en distance réversible, et permettra de reconstruire complètement l'objet irrégulier initial à partir de son axe médian.

Applications des outils de géométrie discrète sur $\mathbb{I}$-grille

Lors de notre stage de Master 2 recherche, nous avons découvert la modélisation par $\mathbb{I}$-grille, et son utilité dans les méthodes numériques. Après avoir traité des images 3-D issues de l'imagerie médicale, nous avons étudié la compression de ces images pour accélérer les simulations de radiothérapie. Grâce à un codage RLE dans le sens de l'irradiation, nous avons montré que la géométrie du patient devait être soigneusement optimisée pour permettre d'avoir des simulations avec des temps d'exécution acceptables pour l'utilisateur (moins de 2 heures de simulation avec un RLE contre 6 heures avec une grille classique de voxels).

Nous avons ensuite utilisé les outils de reconstruction d'objets 2-D dans une application de distinction de caractères ambigus. Grâce aux deux représentations que nous calculons pour une image binaire, nous proposons des caractéristiques pertinentes pour discriminer des images de '8' et de 'B'. Le classifieur `AdaBoost` et l'optimisation de sa base d'apprentissage par une recherche locale sont des premières pistes qui aboutissent à des résultats très satisfaisants (taux de reconnaissance de plus de 93% contre 85% avec les techniques précédentes développées au sein de l'entreprise Foxstream).

Enfin, nous nous sommes intéressés au tracé de courbes implicites grâce à l'arithmétique d'intervalles. Cette théorie est naturellement à mettre en relation avec notre modèle de $\mathbb{I}$-grille, comme l'arithmétique sur $\mathbb{Z}^2$ avec la grille régulière classique. L'algorithme de reconstruction du pavage issu d'une analyse en intervalles permet d'approximer la courbe avec une bonne qualité, comparée à celle des algorithmes de tracé classiques. De plus, nous calculons une structure polygonale avec peu de segments, ce qui accélère le tracé final de la courbe.

Pour aller plus loin

Bien que la modélisation géométrique 3-D pour la simulation soit devenue une thématique éloignée de notre thèse, nous portons toujours intérêt à cette discipline. La partie I de cette thèse montre que les méthodes numériques nécessitent des représentations adaptatives pour accélérer les traitements. Comme pour notre étude sur les structures accélératrices pour la réalité augmentée, on pourrait s'intéresser à la comparaison de différentes modélisations géométriques pour déterminer quelle serait la meilleure, ou envisager de combiner plusieurs structures.

Dans le chapitre traitant de notre application de reconnaissance de caractères, nous avons évoqué la possibilité d'utiliser d'autres techniques d'optimisation combinatoire. Elle

permettraient de calculer plus rapidement et de manière plus sûre une base optimale pour l'apprentissage du classifieur. Nous collaborons actuellement sur la manière de représenter des images issues de *story boards* grâce à une polygonalisation. Cette démarche a pour but de caractériser les éléments (personnages, décors, *etc.*) dessinés dans les croquis, afin de les indexer. Le principal problème que nous devons résoudre est de définir des caractéristiques pertinentes pour modéliser correctement la structure des dessins.

Enfin, nous aimerions nous intéresser dans le futur au rendu de surfaces implicites 3-D. Comme nous l'avons évoqué précédemment, il faudrait comparer les résultats obtenus avec ceux issus des techniques qui proposent des représentations triangulaires. Il serait également intéressant d'implémenter d'autres critères d'arrêt pour l'algorithme d'analyse d'intervalles, afin d'inclure des contraintes topologiques (comme la paramétrisabilité globale [Snyder, 1992b]) pour déterminer si une cellule doit être subdivisée ou non (quelques exemples sont illustrés dans la figure 4).

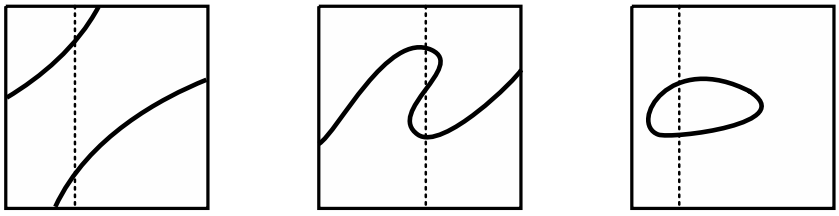


Fig 4: Quelques exemples de non paramétrisabilité globale en X d'une cellule [Snyder, 1992b]. Dans chacun des trois cas, l'équation $f(x, y) = 0$ possède plusieurs solutions pour une abscisse X donnée (comme selon la droite en pointillés). Dans ces cas, l'algorithme `solve()` doit subdiviser ces cellules.

Comme le montrent les paragraphes précédents, cette thèse ouvre de nombreuses perspectives de recherches, liées à la définition d'objets et d'algorithmes sur grilles irrégulières isothétiques, ainsi qu'à l'utilisation de ces dernières dans de nombreuses applications.

Bibliographie

[Agostinelli et al., 2003] Agostinelli, S. et al. (2003). **Geant4 - A Simulation Toolkit**. *Nuclear Instruments and Methods*, A506(3):250–303.

[Al-Abudi *et al.*, 2005] Al-Abudi, B. Q., George, L. A. et Hussain, A. A.-K. (2005). **Hierarchical Multilevel Block Truncation Coding Based on Horizontal-Vertical Partitioning For Color Images**. *International Journal on Graphics, Vision and Image Processing*, 6:31–36.

[Alberti et Mourrain, 2007] Alberti, L. et Mourrain, B. (2007). **Visualisation of Implicit Algebraic Curves**. *In 15th Pacific Conference on Computer Graphics and Applications (PG'07)*, pages 303–312.

[Alhalabi et Tougne, 2005] Alhalabi, F. et Tougne, L. (2005). **Toward Polygonalisation of Thick Discrete Arcs**. *In 11th International Conference on Computer Analysis of Images and Patterns (CAIP 2005)*, pages 197–204.

[Anagnostopoulos *et al.*, 2005] Anagnostopoulos, C., Alexandropoulos, T., Boutas, S., Loumos, V. et Kayafas, E. (2005). **A Template-Guided Approach to Vehicle Surveillance and Access Control**. *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS 2005)*, pages 534–539.

[Andres, 2000] Andres, E. (2000). **Modélisation Analytique Discrète d'Objets Géométriques.** Habilitation à diriger des recherches, UFR Sciences Fondamentale et Appliquées - Université de Poitiers.

[ANIMAL, 2008] ANIMAL (2008). **AN IMAge Library**. http ://animal.sourceforge.net.

[Arth *et al.*, 2007] Arth, C., Limberger, F. et Bischof, H. (2007). **Real-Time License Plate Recognition on an Embedded DSP-Platform**. *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR '07)*, pages 1–8.

[Arya *et al.*, 2002] Arya, S., Malamatos, T. et Mount, D. M. (2002). **Space-Efficient Approximate Voronoi Diagrams**. *In The 34th annual ACM symposium on Theory of computing (STOC'02)*, pages 721–730.

[Attali *et al.*, 2007] Attali, D., Boissonnat, J.-D. et Edelsbrunner, H. (2007). **Stability and Computation of the Medial Axis — a State-of-the-Art Report**. *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*.

[Bastian *et al.*, 2008a] Bastian, P., Blatt, M., Dedner, A., Engwer, C., Klöfkorn, R., Kornhuber, R., Ohlberger, M. et Sander, O. (2008a). **A Generic Grid Interface for Parallel and Adaptive Scientific Computing. Part II : Implementation and Tests in DUNE**. *Computing*, 82(2):121–138.

[Bastian *et al.*, 2008b] Bastian, P., Blatt, M., Dedner, A., Engwer, C., Klöfkorn, R., Ohlberger, M. et Sander, O. (2008b). **A Generic Grid Interface for Parallel and Adaptive Scientific Computing. Part I : Abstract Framework**. *Computing*, 82(2):103–119.

[Bentley, 1975] Bentley, J. L. (1975). **Multidimensional Binary Search Trees Used for Associative Searching**. *Communication on the ACM*, 18(9):509–517.

[Bernard et le Pennec, 2002] Bernard, C. et le Pennec, E. (2002). **Adaptation of Regular Grid Filterings to Irregular Grids**. Rapport technique, Ecole Polytechnique, Centre de Mathématiques Appliquées.

[Bertalmío *et al.*, 2000] Bertalmío, M., Sapiro, G., Caselles, V. et Ballester, C. (2000). **Image Inpainting**. *In 27th annual conference on Computer graphics and interactive techniques (SIGGRAPH '00)*.

[Biasotti *et al.*, 2008] Biasotti, S., Giorgi, D., Spagnuolo, M. et Falcidieno, B. (2008). **Reeb Graphs for Shape Analysis and Applications**. *Theorical Computer Science*, 392(1-3):5–22.

[Boissonnat *et al.*, 2006] Boissonnat, J.-D., Cohen-Steiner, D., Mourrain, B., Rote, G. et Vegter, G. (2006). **Meshing of Surfaces**. *In* Boissonnat, J.-D. et Teillaud, M., éditeurs : *Effective Computational Geometry for Curves and Surfaces*. Springer.

[Boissonnat *et al.*, 2003] Boissonnat, J.-D., Cohen-Steiner, D. et Vegter, G. (2003). **Meshing Implicit Surfaces with Certified Topology**. Rapport technique 4930, INRIA.

[Boissonnat *et al.*, 2002] Boissonnat, J.-D., Devillers, O., Pion, S., Teillaud, M. et Yvinec, M. (2002). **Triangulations in CGAL**. *Computational Geometry : Theory and Applications*, 22:5–19.

[Boissonnat et Teillaud, 2006] Boissonnat, J.-D. et Teillaud, M., éditeurs (2006). **Effective Computational Geometry for Curves and Surfaces**. Springer.

[Bremananth *et al.*, 2005] Bremananth, R., Chitra, A., Seetharaman, V. et Nathan, V. S. L. (2005). **A Robust Video Based License Plate Recognition System**. *International Conference on Intelligent Sensing and Information Processing*, pages 175–180.

[Bresenham, 1965] Bresenham, J. (1965). **Algorithm for Computer Control of Digital Plotter**. *IBM System Journal*, 4:25–30.

[Bresenham, 1977] Bresenham, J. (1977). **A Linear Algorithm for Incremental Display of Circular Arcs**. *Communications of the ACM*, 20(2):100–106.

[Breton, 2003] Breton, R. (2003). **Reconstruction Inversible d'Objets Discrets 2D**. Thèse de doctorat, Université de Poitiers, Poitiers, France.

[Breu *et al.*, 1995] Breu, H., Gil, J., Kirkpatrick, D. et Werman, M. (1995). **Linear Time Euclidean Distance Algorithms**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):529–533.

[Bronstein *et al.*, 2007] Bronstein, A. M., Bronstein, M. M. et Kimmel, R. (2007). **Weighted Distance Maps Computation on Parametric Three-Dimensional Manifolds**. *Journal of Computational Physics*, 225(1):771–784.

[Broumandnia et Shanbehzadeh, 2007] Broumandnia, A. et Shanbehzadeh, J. (2007). **Fast Zernike Wavelet Moments for Farsi Character Recognition**. *Image and Vision Computing*, 25(5):717–726.

[Bühler, 2001] Bühler, K. (2001). **Linear Interval Estimations for Parametric Objects - Theory and Application**. *Computer Graphics Forum*, 20(3).

[Bühler, 2002] Bühler, K. (2002). **Implicit Linear Interval Estimations**. *In 18th spring conference on Computer graphics (SCCG '02)*, pages 123–132.

[Burge et Kropatsch, 1999] Burge, M. et Kropatsch, W. G. (1999). **A Minimal Line Property Preserving Representation of Line Images**. *Computing*, 62(4):355–368.

[Buzer, 2002] Buzer, L. (2002). **An Incremental Linear Time Algorithm for Digital Line and Plane Recognition Using a Linear Incremental Feasibility Problem**. *In 10th International Conference on Discrete Geometry for Computer Imagery (DGCI 2002)*, pages 372–381.

[Cai et Sakas, 2000] Cai, W. et Sakas, G. (2000). **DRR Volume Rendering Using Splatting in Shear-Warp Context**. *IEEE Nuclear Science Symposium Conference Record*, 3(19):12–17.

[Cazals *et al.*, 1995] Cazals, F., Drettakis, G. et Puech, C. (1995). **Filtering, Clustering and Hierarchy Construction : a New Solution for Ray-Tracing Complex Scenes**. *Computer Graphics Forum*, 14(3):371–382.

[CGAL, 2008] CGAL (2008). **Computational Geometry Algorithms Library**. http ://www.cgal.org.

[Chang *et al.*, 2004] Chang, S. L., Chen, L. S., Chung, Y. C. et Chen, S. W. (2004). **Automatic License Plate Recognition**. *IEEE Transactions on Intelligent Transportation Systems*, 5(1):42–53.

[Charlap, 1995] Charlap, D. (1995). **The BMP File Format : Part I & II**. *Dr. Dobb's Journal*, 20(228 & 229).

[Chassery et Montanvert, 1991] Chassery, J. et Montanvert, A., éditeurs (1991). **Géometrie Discrète en Imagerie**. Hermès.

[Chehadeh *et al.*, 1996] Chehadeh, Y., Coquin, D. et Bolon, P. (1996). **A Skeletonization Algorithm Using Chamfer Distance Transformation Adapted to Rectangular Grids**. *13th International Conference on Pattern Recognition (ICPR'96)*, 2:131–135.

[Coeurjolly, 2002a] Coeurjolly, D. (2002a). **Algorithmique pour l'Analyse et la Modélisation en Géométrie Discrète**. Thèse de doctorat, Université Lumière Lyon 2.

[Coeurjolly, 2002b] Coeurjolly, D. (2002b). **Visibility in Discrete Geometry : an Application to Discrete Geodesic Paths**. *In 10th International Conference on Discrete Geometry for Computer Imagery*, pages 326–327.

[Coeurjolly, 2005] Coeurjolly, D. (2005). **Supercover Model and Digital Straight Line Recognition on Irregular Isothetic Grids**. *In 12th International Conference on Discrete Geometry for Computer Imagery*, pages 311–322. LNCS 3429.

[Coeurjolly, 2007] Coeurjolly, D. (2007). **Algorithmique pour l'Analyse et la Modélisation en Géométrie Discrète**. Université Claude Bernard Lyon 1.

[Coeurjolly et Montanvert, 2007] Coeurjolly, D. et Montanvert, A. (2007). **Optimal Separable Algorithms to Compute the Reverse Euclidean Distance Transformation and Discrete Medial Axis in Arbitrary Dimension**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):437–448.

[Coeurjolly *et al.*, 2007] Coeurjolly, D., Montanvert, A. et Chassery, J.-M., éditeurs (2007). **Géométrie Discrète et Images Numériques**. Hermès Paris.

[Coeurjolly et Zerarga, 2006] Coeurjolly, D. et Zerarga, L. (2006). **Supercover Model, Digital Straight Line Recognition and Curve Reconstruction on the Irregular Isothetic Grids**. *Computers & Graphics*, 30(1):46–53.

[Cohen-Or et Kaufman, 1995] Cohen-Or, D. et Kaufman, A. (1995). **Fundamentals of Surface Voxelization**. *Graphical models and image processing (GMIP)*, 57(6):453–461.

[Comba et Stolfi, 1993] Comba, J. L. D. et Stolfi, J. (1993). **Affine Arithmetic and its Applications to Computer Graphics**. *In VI Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens (SIBGRAPI'93)*, pages 9–18.

[Cordella et Vento, 2000] Cordella, L. et Vento, M. (2000). **Symbol Recognition in Documents : a Collection of Techniques ?** *International Journal on Document Analysis and Recognition*, 3(2):73–88.

[Cousty *et al.*, 2008] Cousty, J., Bertrand, G., Couprie, M. et Najman, L. (2008). **Fusion Graphs : Merging Properties and Watersheds**. *Journal of Mathematical Imaging and Vision*, 30(1):87–104.

[Cuisenaire, 1999] Cuisenaire, O. (1999). **Distance Transformations : Fast Algorithms and Applications to Medical Image Processing**. Thèse de doctorat, Université Catholique de Louvain, Louvain-La-Neuve, Belgique.

[Damiand *et al.*, 2004] Damiand, G., Bertrand, Y. et Fiorio, C. (2004). **Topological Model for Two-Dimensional Image Representation : Definition and Optimal Extraction Algorithm.** *Computer Vision and Image Understanding*, 93(2):111–154.

[de A. Lotufo et Zampirolli, 2001] de A. Lotufo, R. et Zampirolli, F. A. (2001). **Fast Multidimensional Parallel Euclidean Distance Transform Based on Mathematical Morphology**. *In 14th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2001)*, pages 100–105.

[de Berg *et al.*, 2000] de Berg, M., van Kreveld, M., Overmars, M. et Schwarzkopf, O. (2000). **Computational Geometry : Algorithms and Applications**. Springer-Verlag.

[de Figueiredo et Stolfi, 1997] de Figueiredo, L. H. et Stolfi, J. (1997). **Self-Validated Numerical Methods and Applications**. Brazilian Mathematics Colloquium monographs. IMPA/CNPq, Rio de Janeiro, Brazil.

[de Figueiredo et Stolfi, 2004] de Figueiredo, L. H. et Stolfi, J. (2004). **Affine Arithmetic : Concepts and Applications**. *Numerical Algorithms*, 37(12):147–158.

[Debled-Rennesson *et al.*, 2005] Debled-Rennesson, I., Feschet, F. et Rouyer-Degli, J. (2005). **Optimal Blurred Segments Decomposition in Linear Time**. *In 12th International Conference on Discrete Geometry for Computer Imagery (DGCI 2005)*, pages 371–382.

[Debled Rennesson et Reveilles, 1995] Debled Rennesson, I. et Reveilles, J.-P. (1995). **A Linear Algorithm for Segmentation of Digital Curves**. *International Journal of Pattern Recognition and Artificial Intelligence*, 9(6):635–662.

[Debled-Rennesson *et al.*, 2004] Debled-Rennesson, I., Tabbone, S. et Wendling, L. (2004). **Fast Polygonal Approximation of Digital Curves**. *17th International Conference on Pattern Recognition (ICPR 2004)*, 1:465–468.

[Devillers, 1998] Devillers, O. (1998). **Improved Incremental Randomized Delaunay Triangulation**. *In 14th Annual ACM Symposium on Computational Geometry*, pages 106–115.

[Dexet, 2006] Dexet, M. (2006). **Architecture d'un Modeleur Géométrique à Base Topologique d'Objets Discrets et Méthodes de Reconstruction en Dimensions 2 et 3**. Thèse de doctorat, Université de Poitiers.

[Dexet et Andres, 2006] Dexet, M. et Andres, E. (2006). **A Generalized Preimage for the Standard and Supercover Digital Hyperplane Recognition**. *In 13th International Conference on Discrete Geometry for Computer Imagery (DGCI 2006)*, pages 639–650.

[Dexet et Andres, 2008] Dexet, M. et Andres, E. (2008). **A Generalized Preimage for the Digital Analytical Hyperplane Recognition**. *Discrete Applied Mathematics*, In Press, Corrected Proof.

[D'Haene *et al.*, 2004] D'Haene, M., Eeckhaut, H., Christiaens, M. et Stroobandt, D. (2004). **An Exploration of the Hardware Implementation of Quadtree Compression**. *In 15th ProRISC Workshop*, pages 364–368.

[Dmitriev, 2000] Dmitriev, K. (2000). **Efficiency Issues on Ray Tracing Machine**. *In The 10th International Conference on Computer Graphics and Vision*, pages 99–103.

[Dorigo *et al.*, 1996] Dorigo, M., Maniezzo, V. et Colorni, A. (1996). **Ant System : Optimization by a Colony of Cooperating Agents**. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, 26(1):29–41.

[Droske *et al.*, 2001] Droske, M., Meyer, B., Rumpf, M. et Schaller, C. (2001). **An Adaptive Level Set Method for Medical Image Segmentation**. *In Annual Symposium on Information Processing in Medical Imaging*.

[Duan *et al.*, 2005] Duan, T. D., Du, T. L. H., Phuoc, T. V. et Hoang, N. V. (2005). **Building an Automatic Vehicle License-Plate Recognition System**. *In IEEE International Conference on Research, Innovation and Vision for the Future (RIVF'05)*, pages 59–63.

[Early et Long, 2001] Early, D. S. et Long, D. G. (2001). **Image Reconstruction and Enhanced Resolution Imaging from Irregular Samples**. *IEEE Transations on Geoscience and Remote Sensing*, 39(2).

[Edelsbrunner *et al.*, 2008] Edelsbrunner, H., H., J., Mascarenhas, A., Pascucci, V. et Snoeyink, J. (2008). **Time-Varying Reeb Graphs for Continuous Space–Time Data**. *Computational Geometry : Theory and Applications*, 41(3):149–166.

[Elberand et Kim, 2001] Elberand, G. et Kim, M. (2001). **Geometric Constraint Solver using Multivariate Rational Spline Functions**. *In 6th ACM Symposium on Solid Modeling and Applications (SMA'01)*, pages 1–10.

[Elgammal et Ismail, 2001] Elgammal, A. et Ismail, M. (2001). **Graph-Based Segmentation and Feature-Extraction Framework for Arabic Text Recognition**. *In 6th International Conference on Document Analysis and Recognition (ICDAR 01)*.

[Erickson et Har-Peled, 2002] Erickson, J. et Har-Peled, S. (2002). **Optimally Cutting a Surface into a Disk**. *In 18th Annual Symposium on Computational Geometry (SCG'02)*, pages 244–253.

[Fabbri *et al.*, 2008] Fabbri, R., Costa, L. D. F., Torelli, J. C. et Bruno, O. M. (2008). **2D Euclidean Distance Transform Algorithms : A Comparative Survey**. *ACM Computing surveys*, 40(1):1–44.

[Fabri, 2007] Fabri, A. (2007). **Voronoi Diagrams in CGAL, the Computational Geometry Algorithms Library**. *4th International Symposium on Voronoi Diagrams in Science and Engineering 2007 (ISVD'07)*, pages 8–14.

[Fadili *et al.*, 2007] Fadili, M., Starck, J.-L. et Murtagh, F. (2007). **Inpainting and Zooming Using Sparse Representations**. *The Computer Journal*. (À paraître).

[Farin, 1993] Farin, G. (1993). **Curves and Surfaces for Computer Aided Geometric Design, a Practical Guide**. Academic Press Professional, Inc., 3rd édition.

[Faudot et Michelucci, 2007] Faudot, D. et Michelucci, D. (2007). **A New Robust Algorithm to Trace Curves**. *Reliable Computing*, 13(4):309–324.

[Finkel et Bentley, 1974] Finkel, R. A. et Bentley, J. L. (1974). **Quad Trees : A Data Structure for Retrieval on Composite Keys**. *Acta Informatica*, 4:1–9.

[Foley *et al.*, 1997] Foley, J. D., van Dam, A., Feiner, S. K. et Hughes, J. F. (1997). **Computer Graphics - Principles and Practice**. Addison-Wesley.

[Foley et Sugerman, 2005] Foley, T. et Sugerman, J. (2005). **KD-tree Acceleration Structures for a GPU Raytracer**. *In ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware (HWWS '05)*, pages 15–22.

[Forman, 1998] Forman, R. (1998). **Morse Theory for Cell Complexes**. *Advances in Mathematics*, 134(1):90–145.

[Fouard et Malandain, 2005] Fouard, C. et Malandain, G. (2005). **3-D Chamfer Distances and Norms in Anisotropic Grids**. *Image Vision Computing*, 23(2):143–158.

[Fouard *et al.*, 2007] Fouard, C., Strand, R. et Borgefors, G. (2007). **Weighted Distance Transforms Generalized to Modules and their Computation on Point Lattices**. *Pattern Recognition*, 40(9):2453–2474.

[Freeman, 1979] Freeman, H. (1979). **Algorithm for Generating a Digital Straight Line on a Triangular Grid**. *IEEE Transactions on Computer*, 28(2):150–152.

[Freund et Schapire, 1995] Freund, Y. et Schapire, R. E. (1995). **A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting**. *In 2nd European Conference on Computational Learning Theory (EuroCOLT'95)*, pages 23–37.

[Frisken et Perry, 2002] Frisken, S. F. et Perry, R. N. (2002). **Simple and Efficient Traversal Methods for Quadtrees and Octrees**. *Journal of Graphic Tools*, 7(3):1–11.

[Fu *et al.*, 2004] Fu, C. W., Wong, T. T., Tong, W. S., Tang, C. K. et Hanson, A. J. (2004). **Binary-Space-Partitioned Images for Resolving Image-Based Visibility**. *IEEE Transactions on Visualization and Computer Graphics*, 10(1):58–71.

[Fuchs *et al.*, 1980] Fuchs, H., Kedem, Z. M. et Naylor, B. F. (1980). **On Visible Surface Generation by a Priori Tree Structures**. *In 7th annual conference on Computer graphics and interactive techniques (SIGGRAPH'80)*, pages 124–133.

[Gandoin et Devillers, 2002] Gandoin, P.-M. et Devillers, O. (2002). **Progressive Lossless Compression of Arbitrary Simplicial Complexes**. *In 29th annual conference on Computer graphics and interactive techniques (SIGGRAPH '02)*.

[Gargantini, 1982] Gargantini, I. (1982). **An effective way to represent quadtrees**. *Communications on the ACM*, 25(12):905–910.

[Glover, 1989a] Glover, F. (1989a). **Tabu Search - Part I**. *ORSA Journal on Computing*, 1(3):190–206.

[Glover, 1989b] Glover, F. (1989b). **Tabu Search - Part II**. *ORSA Journal on Computing*, 2(1):4–32.

[Goldsmith et Salmon, 1987] Goldsmith, J. et Salmon, J. (1987). **Automatic Creation of Object Hierarchies for Ray Tracing**. *IEEE Computer Graphics Applications*, 7(5): 14–20.

[Golomb, 1966] Golomb, S. W. (1966). **Run Length Encodings**. *In IEEE Transactions on Information Theory IT-12*, pages 399–401.

[Gramain, 1971] Gramain, A. (1971). **Topologie des surfaces**. Presses Universitaires Françaises.

[Grochenig et Razafinjatovo, 1996] Grochenig, K. et Razafinjatovo, H. (1996). **On Landau's Necessary Density Conditions for Sampling and Interpolation of Band-Limited Functions**. *Journal of the London Mathematical Society*, 54(3):557–565.

[Guan et Ma, 1998] Guan, W. et Ma, S. (1998). **A List-Processing Approach to Compute Voronoi Diagrams and the Euclidean Distance Transform**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(7):757–761.

[Ham et Young, 2003] Ham, F. et Young, Y. N. (2003). **A Cartesian Adaptive Level Set Method for Two-Phase Flows**. Rapport technique, Center of Turbulence Research.

[Hart, 1999] Hart, J. (1999). **Computational Topology for Shape Modeling**. *In Shape Modeling International (SMI'99)*, pages 36–43. IEEE Computer Society.

[Hassouna et Farag, 2007] Hassouna, M. et Farag, A. (2007). **MultiStencils Fast Marching Methods : a Highly Accurate Solution to the Eikonal Equation on Cartesian Domains**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(9):1563–1574.

[Havran, 2000] Havran, V. (2000). **Heuristic Ray Shooting Algorithms**. Thèse de doctorat, Faculty of Electrical Engineering.

[Herman, 1998] Herman, G. T. (1998). **Geometry of Digital Spaces**. Birkhauser Boston.

[Hesselink *et al.*, 2005] Hesselink, W. H., Visser, M. et Roerdink, J. (2005). **Euclidean Skeletons of 3D Data Sets in Linear Time by the Integer Medial Axis Transform**. *In* Ronse, C., Najman, L. et Decencière, E., éditeurs : *Computational Imaging and Vision*, volume 30, pages 259–268. Springer-Verlag.

[Hétroy, 2003] Hétroy, F. (2003). **Méthodes de Partitionnement de Surfaces**. Thèse de doctorat, Institut National Polytechnique de Grenoble, Grenoble, France.

[Hilaire et Tombre, 2005] Hilaire, X. et Tombre, K. (2005). **Robust and Accurate Vectorization of Line Drawings**. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Hirata, 1996] Hirata, T. (1996). **A Unified Linear-Time Algorithm for Computing Distance Maps**. *Information Processing Letters*, 58(3):129–133.

[Horn *et al.*, 2007] Horn, D. R., Sugerman, J., Houston, M. et Hanrahan, P. (2007). **Interactive K-d tree GPU Raytracing**. *In Symposium on Interactive 3D graphics and games (I3D '07)*, pages 167–174.

[Horowitz et Pavlidis, 1977] Horowitz, S. et Pavlidis, T. (1977). **Picture Segmentation by a Directed Split and Merge Procedure**. *In Computer Methods in Images Analysis*, pages 101–11.

[Hu *et al.*, 2002] Hu, C., Xu, S. et Yang, X. (2002). **A Review on Interval Computation - Software and Applications**. *International Journal of Computational and Numerical Analysis and Applications*, 1(2):149–162.

[Huang *et al.*, 2007] Huang, J., Ertekin, S., Song, Y., Zha, H. et Giles, C. L. (2007). **Efficient Multiclass Boosting Classification with Active Learning**. *In 7th SIAM International Conference on Data Mining (SDM 2007)*, pages 297–308.

[Hubert-Tremblay *et al.*, 2006] Hubert-Tremblay, V., Archambault, L., Tubic, D., Roy, R. et Beaulieu, L. (2006). **Octree Indexing of DICOM Images for Voxel Number Reduction and Improvement of Monte Carlo Simulation Computing Efficiency**. *Medical Physics*, 33(8):2819–2831.

[Irving *et al.*, 2006] Irving, G., Guendelman, E., Losasso, F. et Fedkiw, R. (2006). **Efficient Simulation of Large Bodies of Water by Coupling Two and Three Dimensional Techniques**. *In ACM SIGGRAPH 2006 Papers*, pages 805–811.

[Jackson *et al.*, 2007] Jackson, D. J., Ren, H., Wu, X. W. et Ricks, K. G. (2007). **A Hardware Architecture for Real-time Image Compression using a Searchless Fractal Image Coding Method**. *Journal of Real-Time Image Processing*, 1(3):225–237.

[Jevans et Wyvill, 1989] Jevans, D. et Wyvill, B. (1989). **Adaptive Voxel Subdivision for Ray Tracing**. *In Graphics Interface'89*, pages 164–172.

[Jiang et Paganetti, 2004] Jiang, H. et Paganetti, H. (2004). **Adaptation of Geant4 to Monte Carlo Dose Calculations Based on CT Data**. *Medical Physics*, 31(10):2811–2818.

[Jung et Gupta, 1996] Jung, D. et Gupta, K. (1996). **Octree-Based Hierarchical Distance Maps for Collision Detection**. *IEEE International Conference on Robotics and Automation*, 1:454–459.

[Karavelas, 2004] Karavelas, M. (2004). **A Robust and Efficient Implementation for the Segment Voronoi Diagram**. *In International Symposium on Voronoi Diagrams in Science and Engineering (VD2004)*, pages 51–62.

[Kirkpatrick *et al.*, 1983] Kirkpatrick, S., Gelatt, C. D. et Vecchi, M. P. (1983). **Optimization by Simulated Annealing**. *Science*, 220(4598):671–680.

[Klette et Rosenfeld, 2004] Klette, R. et Rosenfeld, A. (2004). **Digital Geometry : Geometric Methods for Digital Picture Analysis**. Morgan Kaufmann Publishers Inc.

[Klimaszewski et Sederberg, 1997] Klimaszewski, K. et Sederberg, T. (1997). **Faster Ray Tracing using Adaptive Grids**. *IEEE Computer Graphics and Applications*, 17(1):42–51.

[Knoll, 2006] Knoll, A. (2006). **A Survey of Octree Volume Rendering Techniques**. *In 1st IRTG Workshop*.

[Kotsiantis *et al.*, 2006] Kotsiantis, S., Zaharakis, I. et Pintelas, P. (2006). **Machine Learning : a Review of Classification and Combining Techniques**. *Artificial Intelligence Review*, 26(3):159–190.

[Kovalcík et Tobola, 2005] Kovalcík, V. et Tobola, P. (2005). **Dynamic Bounding Volume Hierarchies for Occlusion Culling**. *In Virtual Environments*, pages 91–96.

[Kubica *et al.*, 2005] Kubica, J., Moore, A., Connolly, A. et Jedicke, R. (2005). **A Multiple Tree Algorithm for the Efficient Association of Asteroid Observations**. *In 11th ACM SIGKDD international conference on Knowledge discovery in data mining (KDD '05)*, pages 138–146.

[Lam *et al.*, 1992] Lam, L., Lee, S. W. et Suen, C. Y. (1992). **Thinning Methodologies - a Comprehensive Survey**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9):869–885.

[Landau, 1967] Landau, H. (1967). **Necessary Density Conditions for Sampling and Interpolation of Certain Entire Functions**. *Acta Mathematica*, 117(1):37–52.

[Larsson et Akenine-Möller, 2006] Larsson, T. et Akenine-Möller, T. (2006). **A Dynamic Bounding Volume Hierarchy for Generalized Collision Detection**. *Computers & Graphics*, 30(3):451–460.

[le Pennec et Mallat, 2005] le Pennec, E. et Mallat, S. (2005). **Sparse Geometric Image Representations with Bandelets**. *IEEE Transactions on Image Processing*, 14(4):423–438.

[Lienhardt, 1991] Lienhardt, P. (1991). **Topological Models for Boundary Representation : a Comparison with n-Dimensional Generalized Maps**. *Computer-Aided Design*, 23(1):59–82.

[Liu et Dori, 1998] Liu, W. et Dori, D. (1998). **A Survey of Non-thinning Based Vectorization Methods**. *In Joint IAPR International Workshops on Advances in Pattern Recognition (SSPR '98/SPR '98)*, pages 230–241.

[Lopes *et al.*, 2001] Lopes, H., Oliveira, J. B. et de Figueiredo, L. H. (2001). **Robust Adaptive Approximation of Implicit Curves**. *In XIV Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'01)*, page 10.

[Lorensen et Cline, 1987] Lorensen, W. E. et Cline, H. E. (1987). **Marching Cubes : A High Resolution 3D Surface Construction Algorithm**. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169.

[Losasso *et al.*, 2006] Losasso, F., Fedkiw, R. et Osher, S. (2006). **Spatially Adaptive Techniques for Level Set Methods and Incompressible Flow**. *Computers & Fluids*, 35(10):995–1010.

[Losasso *et al.*, 2004] Losasso, F., Gibou, F. et Fedkiw, R. (2004). **Simulating Water and Smoke with an Octree Data Structure**. *ACM Transactions on Graphics*, 23(3): 457–462.

[Luczak et Rosenfeld, 1976] Luczak, E. et Rosenfeld, A. (1976). **Distance on a Hexagonal Grid**. *IEEE Transactions on Computer*, 25:532–533.

[Ma et Ellis, 2004] Ma, B. et Ellis, R. (2004). **Surface-Based Registration with a Particle Filter.** *In MICCAI (1)*, volume 3216, pages 566–573.

[MacDonald et Booth, 1990] MacDonald, J. D. et Booth, K. S. (1990). **Heuristics for Ray Tracing Using Space Subdivision**. *The Visual Computer*, 6(6):153–166.

[Malgouyres, 2002] Malgouyres, R. (2002). **Algorithmes pour la Synthèse d'Images et l'Animation 3D**. Dunod.

[Manzanera et Jolion, 1995] Manzanera, A. et Jolion, J.-M. (1995). **Pyramide Irrégulière : une Représentation pour la Vision Exploratoire**. *Traitement du Signal*, 12(2):169–176.

[Marfil *et al.*, 2006] Marfil, R., Molina-Tanco, L., Bandera, A., Rodríguez, J. A. et Sandoval, F. (2006). **Pyramid Segmentation Algorithms Revisited**. *Pattern Recognition*, 39(8):1430–1451.

[Martin *et al.*, 2002] Martin, R., Shou, H., Voiculescu, I., Bowyer, A. et Wang, G. (2002). **Comparison of Interval Methods for Plotting Algebraic Curves**. *Computer Aided Geometric Design*, 19(7):553–587.

[Marvasti, 2001] Marvasti, F. A. (2001). **Nonuniform sampling : Theory and Practice**. Plenum Publishers Co.

[Matsumoto, 2002] Matsumoto, Y. (2002). **An Introduction to Morse Theory**. American Mathematical Society.

[Maurer *et al.*, 2003] Maurer, C. R., Qi, R. et Raghavan, V. (2003). **A Linear Time Algorithm for Computing Exact Euclidean Distance Transforms of Binary Images in Arbitrary Dimensions**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):265–270.

[Megiddo, 1984] Megiddo, N. (1984). **Linear Programming in Linear Time when the Dimension is Fixed**. *Journal of the ACM*, 31(1):114–127.

[Meijster *et al.*, 2000] Meijster, A., Roerdink, J. et Hesselink, W. H. (2000). **A General Algorithm for Computing Distance Transforms in Linear Time**. *In Mathematical Morphology and its Applications to Image and Signal Processing*, pages 331–340. Kluwer.

[Meir et Rätsch, 2003] Meir, R. et Rätsch, G. (2003). **Advanced Lectures on Machine Learning**, chapitre An Introduction to Boosting and Leveraging, pages 118–183. Springer.

[Mertzios et Karras, 1999] Mertzios, B. et Karras, D. (1999). **On Applying Fast and Efficient Methods in Pattern Recognition**. *Signal Processing for Multimedia*.

[Milnor, 1963] Milnor, J. (1963). **Morse Theory**. Princeton University Press.

[Montanvert *et al.*, 1991] Montanvert, A., Meer, P. et Rosenfeld, A. (1991). **Hierarchical Image Analysis Using Irregular Tessellations**. *Pattern Analysis for Machine Intelligence*, 13(4):307–316.

[Moore, 1966] Moore, R. E. (1966). **Interval Analysis**. Prentice-Hall.

[Moore et Yang, 1959] Moore, R. E. et Yang, C. T. (1959). **Interval Analysis I**. Rapport technique LMSD-285875, Lockheed Missiles and Space Division, Sunnyvale, CA, USA.

[Mourrain, 2002] Mourrain, B. (2002). **Topology of Implicit Curves and Surfaces**. http ://www-sop.inria.fr/galaad/mourrain/Cours/20021022topology.pdf.

[Nagy, 2005] Nagy, B. (2005). **A Comparison Among Distances Based on Neighborhood Sequences in Regular Grids**. *In 14th Scandinavian Conference on Image Analysis (SCIA05)*, pages 1027–1036.

[Naito *et al.*, 2000] Naito, T., Tsukada, T., Yamada, K., Kozuka, K. et Yamamoto, S. (2000). **Robust License-Plate Recognition Method for Passing Vehicles under Outside Environment**. *IEEE Transactions on Vehicular Technology*, 49(6):2309–2319.

[Nyquist, 1928] Nyquist, H. (1928). **Certain Topics in Telegraph Transmission Theory**. *In American Institute of Electrical Engineers Transactions*, volume 47, page 617.

[OpenCV, 2008] OpenCV (2008). **Intel Open Source Computer Vision library**. http ://www.sourceforge.net/projects/opencvlibrary.

[Paganetti, 2004] Paganetti, H. (2004). **Four-Dimensional Monte Carlo Simulation of Time-Dependant Geometries**. *Physics in Medicine and Biology*, 49(6):75–81.

[Paglieroni, 1992] Paglieroni, D. W. (1992). **Distance Transforms : Properties and Machine Vision Applications**. *Graphical Models for Image Processing*, 54(1):56–74.

[Park *et al.*, 2005] Park, S. H., Lee, S. S. et Kim, J. H. (2005). **The Delaunay Triangulation by Grid Subdivision**. *In Computational Science and Its Applications (ICCSA 2005)*, pages 1033–1042. 10.1007/11424857_111.

[Pascucci *et al.*, 2007] Pascucci, V., Scorzelli, G., Bremer, P.-T. et Mascarenhas, A. (2007). **Robust on-line Computation of Reeb Graphs : Simplicity and Speed**. *ACM Transactions on Graphics*, 26(3):58.

[Pavlidis, 2003] Pavlidis, T. (2003). **36 Years on the Pattern Recognition Front**. *Pattern Recognition Letters*, 24(1-3):1–7.

[Peeters *et al.*, 1979] Peeters, F., Verbeeten, B. et Venema, H. W. (1979). **Prix Nobel de médecine et de physiologie en 1979 pour A. M. Cormack et G. N. Hounsfield]**. *Nederlands tijdschrift voor geneeskunde*, 123(51):2192–3.

[Persoon et Fu, 1986] Persoon, E. et Fu, K. S. (1986). **Shape Discrimination using Fourier Descriptors**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(3):388–397.

[Pham *et al.*, 2005] Pham, D. T., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S. et Zaidi, M. (2005). **The Bees Algorithm**. Rapport technique, Manufacturing Engineering Centre, Cardiff University, Royaumes-Unis.

[Pharr et Humphreys, 2004] Pharr, M. et Humphreys, G. (2004). **Physically Based Rendering**. Elsevier.

[Plantinga et Vegter, 2004] Plantinga, S. et Vegter, G. (2004). **Isotopic Approximation of Implicit Curves and Surfaces**. *In Eurographics/ACM SIGGRAPH symposium on Geometry processing (SGP'04)*, pages 245–254.

[Plantinga et Vegter, 2007] Plantinga, S. et Vegter, G. (2007). **Isotopic Meshing of Implicit Surfaces**. *The Visual Computer*, 23(1):45–58.

[Plewa *et al.*, 2005] Plewa, T., Linde, T. et Weirs, V., éditeurs (2005). **Adaptive Mesh Refinement - Theory and Applications. Proceedings of the Chicago Workshop on Adaptive Mesh Refinement in Computational Science and Engineering)**, volume 41. Springer.

[Popinet, 2003] Popinet, S. (2003). **Gerris : a Tree-Based Adaptive Solver for the Incompressible Euler Equations in Complex Geometries**. *Journal of Computeational Physics*, 190(2):572–600.

[Preparata et Shamos, 1985] Preparata, F. P. et Shamos, M. I. (1985). **Computational Geometry : an Introduction**. Springer-Verlag.

[Preusser et Rumpf, 1999] Preusser, T. et Rumpf, M. (1999). **An Adaptive Finite Element Method for Large Scale Image Processing**. *In 2nd International Conference on Scale-Space Theories in Computer Vision (SCALE-SPACE'99)*, pages 223–234.

[Pruess et Garcia, 2000] Pruess, K. et Garcia, J. (2000). **A Systematic Approach to Local Grid Refinement in Geothermal Reservoir Simulation**. *In World Geothermal Congress*.

[Ramponi et Carrato, 2001] Ramponi, G. et Carrato, S. (2001). **An Adaptive Irregular Sampling Algorithm and its Application to Image Coding**. *Image Vision Computing*, 19(7):451–460.

[Reeb, 1946] Reeb, G. (1946). **Sur les Points Singuliers d'une Forme de Pfaff Complétement Intégrable ou d'une Fonction Numérique**. *Comptes Rendus de L'Académie ses Sciences*, pages 847–849.

[Remy et Thiel, 2000] Remy, E. et Thiel, E. (2000). **Optimizing 3D Chamfer Masks with Norm Constraints**. *In 7th International Workshop on Combinatorial Image Analysis (IWCIA'00)*, pages 39–56.

[Rodrigues et al., 2004] Rodrigues, P. et al. (2004). **Application of Geant4 Radiation Transport Toolkit to Dose Calculations in Anthromorphic Phantoms**. *Applied Radiation and Isotopes*, 61:1451–1461.

[Rosenfeld et Pfaltz, 1966] Rosenfeld, A. et Pfaltz, J. L. (1966). **Sequential Operations in Digital Picture Processing**. *Journal of the ACM*, 13(4):471–494.

[Rosenfeld et Pfaltz, 1968] Rosenfeld, A. et Pfaltz, J. L. (1968). **Distance Functions on Digital Pictures**. *Pattern Recognition*, 1(1):33–61.

[Rubin et Whitted, 1980] Rubin, S. M. et Whitted, T. (1980). **A 3-dimensional Representation for Fast Rendering of Complex Scenes**. *In 7th annual conference on Computer graphics and interactive techniques (SIGGRAPH '80)*, pages 110–116.

[Réveillès, 1991] Réveillès, J.-P. (1991). **Géométrie discrète, calcul en nombres entiers et algorithmique**. Thèse de doctorat, Université Louis Pasteur.

[Saito et Toriwaki, 1994] Saito, T. et Toriwaki, J. I. (1994). **New Algorithms for Euclidean Distance Transformation of an n-Dimensional Digitized Picture with Applications**. *Pattern Recognition*, 27(11):1551–1565.

[Samet, 1983] Samet, H. (1983). **A Quadtree Medial Axis Transform**. *Communications of the ACM*, 26(9):680–693.

[Samet, 1990a] Samet, H. (1990a). **Applications of Spatial Data Structures : Computer Graphics, Image Processing, and GIS**. Addison-Wesley Longman Publishing Co., Inc.

[Samet, 1990b] Samet, H. (1990b). **The Design and Analysis of Spatial Data Structures**. Addison-Wesley Longman Publishing Co., Inc.

[Sarrut et Guigues, 2008] Sarrut, D. et Guigues, L. (2008). **Region-Oriented CT Image Representation for Reducing Computing Time of Monte Carlo Simulations**. *Medical Physics*, 35(4):1452–1463.

[Schapire, 2003] Schapire, R. E. (2003). **The Boosting Approach to Machine Learning : An Overview**. *In* Denison, D. D., Hansen, M. H., Holmes, C., Mallick, B. et Yu, B., éditeurs : *Nonlinear Estimation and Classification*. Springer.

[Schneider *et al.*, 2000] Schneider, W., Bortfeld, T. et Schlegel, W. (2000). **Correlation between CT Numbers and Tissue Parameters Needed for Monte Carlo Simulations of Clinical Dose Distributions**. *Physics in Medicine and Biology*, 45:459–478.

[Schouten et van den Broek, 2004] Schouten, T. et van den Broek, E. (2004). **Fast Exact Euclidean Distance (FEED) Transformation**. *In International Conference on Pattern Recognition (ICPR'04)*, volume 3, pages 594–597.

[Selman *et al.*, 1992] Selman, B., Levesque, H. et Mitchell, D. (1992). **A New Method for Solving Hard Satisfiability Problems**. *In 10th National Conference on Artificial Intelligence (AAAI'92)*, pages 440–446.

[Sethian, 1999] Sethian, J. A. (1999). **Level Set Methods and Fast Marching Methods**. Cambridge University Press.

[Sethian, 2001] Sethian, J. A. (2001). **Evolution, Implementation, and Application of Level Set and Fast Marching Methods for Advancing Fronts**. *Journal of Computational Physics*, 169(2):503–555.

[Shridhar et Badreldin, 1984] Shridhar, M. et Badreldin, A. (1984). **High Accuracy Character Recognition Algorithm Using Fourier and Topological Descriptors**. *Pattern Recognition*, 17(5):515–524.

[Sintorn et Borgefors, 2004] Sintorn, I.-M. et Borgefors, G. (2004). **Weighted Distance Transforms for Volume Images Digitized in Elongated Voxel Grids**. *Pattern Recognition Letters*, 25(5):571–580.

[Sivignon *et al.*, 2004] Sivignon, I., Breton, R., Dupont, F. et Andres, E. (2004). **Discrete Analytical Curve Reconstruction without Patches**. *Image and Vision Computing*, 23(2):191–202.

[Sloboda et Zat'ko, 1995] Sloboda, F. et Zat'ko, B. (1995). **On Boundary Approximation**. *In 6th International Conference on Computer Analysis of Images and Patterns (CAIP'95)*, pages 488–495.

[Sloboda et Zat'ko, 2001] Sloboda, F. et Zat'ko, B. (2001). **On Approximation of Jordan Surfaces in 3D**. *In Digital and Image Geometry*, pages 365–388.

[Snyder, 1992a] Snyder, J. M. (1992a). **Generative Modeling for Computer Graphics and CAD : Symbolic Shape Design using Interval Analysis**. Academic Press Professional, Inc.

[Snyder, 1992b] Snyder, J. M. (1992b). **Interval Analysis for Computer Graphics**. *Computer Graphics*, 26(2):121–130.

[Soille, 2003] Soille, P. (2003). **Morphological Image Analysis**. Springer-Verlag, 2nd édition.

[Song *et al.*, 2002] Song, J., Cai, M., Lyu, M. R. et Cai, S. (2002). **Graphics Recognition from Binary Images : one Step or two Steps**. *16th International Conference on Pattern Recognition*, 3:135–138.

[Stelldinger et Terzic, 2008] Stelldinger, P. et Terzic, K. (2008). **Digitization of Non-Regular Shapes in Arbitrary Dimensions**. *Image and Vision Computing*, 26:1338–1346.

[Stolte, 2005] Stolte, N. (2005). **Arbitrary 3D Resolution Discrete Ray Tracing of Implicit Surfaces**. *In 12th International Conference on Discrete Geometry for Computer Imagery (DGCI 2005)*, volume 3429 de *Lecture Notes in Computer Science*, pages 414–426. Springer.

[Strand et Borgefors, 2005] Strand, R. et Borgefors, G. (2005). **Distance transforms for three-dimensional grids with non-cubic voxels**. *Compututer Vision and Image Understanding*, 100(3):294–311.

[Strohmer, 1993] Strohmer, T. (1993). **Efficient Methods for Digital Signal and Image Reconstruction from Nonuniform Samples**. Thèse de doctorat, Institut für Mathematik der Universität Wien, Wien, Germany.

[Szécsi et Benedek, 2002] Szécsi, L. et Benedek, B. (2002). **Improvements on the kd-tree**. *In First Hungarian Conference on Computer Graphics and Geometry*.

[Taubin, 1994] Taubin, G. (1994). **An Accurate Algorithm for Rasterizing Algebraic Curves and Surfaces**. *In IEEE Computer Graphics and Applications*.

[Teschner *et al.*, 2003] Teschner, M., Heidelberger, B., Müller, M., Pomeranerts, D. et Gross, M. (2003). **Optimized Spatial Hashing for Collision Detection of Deformable Objects**. *In Vision, Modeling, Visualization (VMV 2003)*, pages 47–54.

[Thatcher, 2000] Thatcher, U. (2000). **Spatial Partitioning Schemes**, pages 444–453. Charles River Media.

[Thome, 2007] Thome, N. (2007). **Représentations Hiérarchiques et Discriminantes pour la Reconnaissance de Formes, l'Identification des Personnes et l'Analyse des Mouvements dans les Séquences d'Images**. Thèse de doctorat, Université Lumière Lyon 2.

[Tierny *et al.*, 2006] Tierny, J., Vandeborre, J.-P. et Daoudi, M. (2006). **Graphes de Reeb de Haut Niveau de Maillages Polygonaux 3D**. *In COmpression et REprésentation des Signaux Audiovisuels (CORESA'06)*, pages 172–177.

[Tung, 2005] Tung, T. (2005). **Indexation 3D de Bases de Données d'Objets 3D par Graphes de Reeb Améliorés**. Thèse de doctorat, Telecom Paris, ENST/TIC, Paris, France.

[Vacavant, 2005] Vacavant, A. (2005). **Représentation d'Images 3D pour les Simulations d'Interactions Rayonnement Matière en Radiothérapie**. Rapport technique, Université Claude Bernard Lyon 1.

[Vázquez, 1999] Vázquez, C. (1999). **Reconstruction d'Images Irrégulièrement Echantillonnées : Application à la Reconstruction de Vues Intermédiaires et au Codage Vidéo**. Thèse de doctorat, Université du Québec, Québec, Canada.

[Veltkamp et Latecki, 2006] Veltkamp, R. et Latecki, L. (2006). **Data Science and Classification**, chapitre Properties and Performance of Shape Similarity Measures, pages 47–56. Springer.

[Verhaegen et Devic, 2005] Verhaegen, F. et Devic, S. (2005). **Sensitivity Study for CT Image Use in Monte Carlo Treatment Planning**. *Physysics in Medicine and Biology*, 50:937–946.

[Voronoi, 1908] Voronoi, G. (1908). **Nouvelles Applications des Paramètres Continus à la Théorie des Formes Quadratiques. Deuxième Mémoire : Recherches sur les Parallélloèdres Primitifs**. *Journal für die reine und angewandte Mathematik*, 134:198–287.

[Vörös, 2001] Vörös, J. (2001). **Low-Cost Implementation of Distance Maps for Path Planning using Matrix Quadtrees and Octrees**. *Robotics and Computer-Integrated Manufacturing*, 17:447–459(13).

[Wald *et al.*, 2007] Wald, I., Boulos, S. et Shirley, P. (2007). **Ray Tracing Deformable Scenes Using Dynamic Bounding Volume Hierarchies**. *ACM Transactions on Graphics*, 26(1):6.

[Wallace, 1991] Wallace, G. K. (1991). **The JPEG Still Picture Compression Standard**. *Communications on the ACM*, 34(4):30–44.

[Wang et Bertrand, 1992] Wang, X. et Bertrand, G. (1992). **Some Sequential Algorithms for a Generalized Distance Transformation Based on Minkowski Operations**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(11):1114–1121.

[Wenyin et Dori, 1997] Wenyin, L. et Dori, D. (1997). **A protocol for Performance Evaluation of Line Detection Algorithms**. *Machine Vision and Applications*, 9(5/6): 240–250.

[Wenyin et Dori, 1999] Wenyin, L. et Dori, D. (1999). **From Raster to Vectors : Extracting Visual Information from Line Drawings**. *Pattern Analysis and Application*, 2(1):10–21.

[Whang *et al.*, 1995] Whang, K. Y., Song, J. W., Chang, J. W., Kim, J. Y., Cho, W. S., Park, C. M. et Song, I. Y. (1995). **Octree-R : An Adaptive Octree for Efficient Ray Tracing**. *IEEE Transactions on Visualization and Computer Graphics*, 1(4):343–349.

[Xiao *et al.*, 2003] Xiao, Y., Siebert, P. et Werghi, N. (2003). **A Discrete Reeb Graph Approach for the Segmentation of Human Body Scans**. *In 4th International Conference on 3D Digital Imaging and Modeling*, pages 378–385.

[Yu *et al.*, 2006] Yu, Z. S., Cai, Y. Z., Oh, M. J., Kim, T. W. et Peng, Q. S. (2006). **An Efficient Method for Tracing Planar Implicit Curves**. *Journal of Zhejiang University - Science A*, 7(7):1115–1123.

[Zhang et Lu, 2004] Zhang, D. et Lu, G. (2004). **Review of Shape Representation and Description Techniques**. *Pattern Recognition*, 37:1–19.

Publications de l'auteur

Revue internationale

[Vacavant *et al.*, 2008] Vacavant, A., Coeurjolly, D. et Tougne, L. (2008). **A framework for dynamic implicit curve approximation by an irregular discrete approach**. *Graphical Models*. (**En révision**).

Conférences internationales

[Chateau *et al.*, 2005] Chateau, T., Vacavant, A. et Lavest, J.-M. (2005). **Skin Detection and Tracking by Monocular Vision**. *In International Symposium on Signals, Circuits & Systems (ISSCS 2005)*, Iasi, Roumanie.

[Guigues *et al.*, 2006] Guigues, L., Sarrut, D., Vacavant, A., Dufour, N., Ricol, M.-C., Testa, E., Boutemeur, M., Freud, N. et Létang, J.-M. (2006). **A Platform for Monte Carlo Simulation of Cancer Therapy with Photon and Light Ions Beams based on the Geant 4 Toolkit**. *In Nuclear Science Symposium*, San Diego, California, USA.

[Thome et Vacavant, 2007] Thome, N. et Vacavant, A. (2007). **A Combined Statistical-Structural Strategy for Alphanumeric Recognition**. *In 3rd International Symposium on Visual Computing (ISVC 2007)(2)*, volume 4842 de *Lecture Notes in Computer Science*, pages 529–538, Lake Tahoe, Nevada, USA. Springer.

[Vacavant et Chateau, 2005] Vacavant, A. et Chateau, T. (2005). **Realtime head and hands tracking by monocular vision**. *In IEEE International Conference on Image Processing '05 (ICIP 2005)*, Gênes, Italie.

[Vacavant *et al.*, 2006a] Vacavant, A., Coeurjolly, D. et Tougne, L. (2006a). **Dynamic Reconstruction of Complex Planar Objects on Irregular Isothetic Grids**. *In 2nd International Symposium on Visual Computing (ISVC 2006) (2)*, volume 4292 de *Lecture Notes in Computer Science*, pages 205–214, Lake Tahoe, Nevada, USA. Springer.

[Vacavant *et al.*, 2006b] Vacavant, A., Coeurjolly, D. et Tougne, L. (2006b). **Topological and Geometrical Reconstruction of Complex Objects on Irregular Isothetic Grids**. *In 13th International Conference on Discrete Geometry for Computer Imagery (DGCI 2006)*, volume 4245 de *Lecture Notes in Computer Science*, pages 470–481, Szeged, Hongrie. Springer.

[Vacavant *et al.*, 2008] Vacavant, A., Coeurjolly, D. et Tougne, L. (2008). **Distance Transformation on Two-Dimensional Irregular Isothetic Grids**. *In 14th International Conference on Discrete Geometry for Computer Imagery (DGCI 2008)*, volume 4992 de *Lecture Notes in Computer Science*, pages 238–249, Lyon, France. Springer.

Revue nationale

[Chateau et Vacavant, 2005] Chateau, T. et Vacavant, A. (2005). **Suivi de gestes temps réel par traitements d'images couleur**. *Traitement du Signal, numéro spécial Imagerie Couleur*, 21(1).

Conférences nationales

[Vacavant, 2007] Vacavant, A. (2007). **Outils de la Géométrie Discrète pour le Tracé de Courbes Implicites par Arithmétique d'Intervalles**. *In Rencontres Arithmétiques de l'Informatique Mathématique (RAIM 2007)*, Montpellier, France.

[Vacavant, 2008] Vacavant, A. (2008). **Approximation de Courbes Implicites grâce à des Outils Discrets Irréguliers**. *In Journées Informatique et Géométrie (JIG 2008)*, Dijon, France.

[Vacavant et Chateau, 2005] Vacavant, A. et Chateau, T. (2005). **Suivi de la Tête et des Mains en Vision Monoculaire Temps Réel**. *In ORASIS2005, neuvième congrès des jeunes chercheurs en vision par ordinateur*, Clermont-Ferrand, France.

[Vacavant *et al.*, 2006] Vacavant, A., Coeurjolly, D. et Tougne, L. (2006). **Reconstruction Topologique et Géométrique d'Objets Complexes sur Grilles Isothétiques Irrégulières**. *In COmpression et REprésentation des Signaux Audiovisuels*, Caën, France.

[Vacavant et Sarrut, 2005] Vacavant, A. et Sarrut, D. (2005). **Structuration sur Grilles Isothétiques Irrégulières dans les Simulations de Radiothérapie**. *In Journées Informatique et Géométrie (JIG 2005)*, Paris, France.

Index des auteurs