Courbes remplissant l'espace et leur application en traitement d'images

Giap Nguyen

Université de La Rochelle

École doctorale S2IM

Thèse présentée par :

Giáp NGUYỄN

Pour obtenir le grade de : **Docteur de l'université de La Rochelle**
Discipline : **Informatique**

Courbes remplissant l'espace et leur application en traitement d'images

Directeur de thèse : Rémy MULLOT
Co-encadrants scientifiques : Patrick FRANCO
Jean-Marc OGIER

Soutenue publiquement le 14 novembre 2013, devant le jury composé de :

Rolf INGOLD	Professeur, Université de Fribourg, Président du jury
Nicole VINCENT	Professeur, Université Paris Descartes, Rapporteur
Jean-Philippe DOMENGER	Professeur, Université de Bordeaux, Rapporteur
Rémy MULLOT	Professeur, Université de La Rochelle
Patrick FRANCO	Maître de conférences, Université de La Rochelle
Jean-Marc OGIER	Professeur, Université de La Rochelle

Dernière modification : 8 octobre 2013

Thèse réalisée au laboratoire L3i
l3i.univ-larochelle.fr

financiée par la région Poitou-Charentes

Courbes remplissant l'espace et leur application en traitement d'images

Résumé : Les courbes remplissant l'espace sont connues pour la capacité d'ordonner les points multidimensionnels sur une ligne en tout conservant la localité, *i.e.* les points proches sont toujours proches sur la ligne. La conservation de la localité est beaucoup recherchée dans plusieurs applications. La courbe de Hilbert est la courbe remplissant l'espace qui conserve le mieux la localité. Cette courbe est originalement proposée en 2D, *i.e.* n'est qu'applicable aux points dans un espace 2D.

Pour une perspective d'application dans le cas multidimensionnel, nous proposons dans cette thèse une généralisation de la courbe de Hilbert. La courbe généralisée est définie en s'appuyant sur la propriété essentielle de la courbe de Hilbert qui crée son niveau de conservation de la localité : l'adjacence. Ainsi, elle évite la dépendance du motif primitif RBG qui est le seul motif primitif de la courbe étendu par les recherches précédentes. Le résultat est donc une famille de courbe conservant bien la localité.

L'optimisation de la conservation de la localité est aussi abordée pour permettre de retrouver la courbe qui conserve le mieux la localité. Pour cet objectif, nous proposons une mesure de la conservation de la localité. En s'appuyant sur les paramètres, cette mesure peut adapter aux différentes situations applicatives comme le changement de métrique ou de taille de localité.

La construction est une partie importante de la thèse, elle est la base du calcul de l'index utilisé dans l'application. Pour un calcul de l'index rapide, la courbe de Hilbert autosimilaire est utilisée. C'est la courbe de Hilbert satisfait les conditions de la courbe remplissant l'espace définie dans Chapitre 4.

La courbe généralisée est enfin appliquée dans la recherche d'image. Il s'agit d'une recherche par le contenu où chaque image est caractérisée par un vecteur multidimensionnel. Les images sont ordonnées par la courbe sur une ligne, ainsi, la recherche est simplifiée en une recherche sur une liste ordonnée. En donnant une image d'entrée, les images similaires sont celles correspondantes aux index voisins de l'index de l'image d'entrée. La conservation de la localité garantit que ces index correspondent aux images similaires.

Mots clés : courbe remplissant l'espace, courbe de Hilbert, conservation de la localité, CBIR

Spacer-filling curves and their application in image processing

Summary : The space-filling curves are known for the ability to order the multidimensional points on a line while preserving the locality, *i.e.* the close points are closely ordered on the line. The locality preserving is wished in many applications. Hilbert curve is the best locality preserving space-filling curve. This curve is originally proposed in 2D, *i.e.* it is only applied to points in a 2D space.

For application in the multidimensional case, we propose in this thesis a generalization of Hilbert curve. Generalized curve is based on the essential property of Hilbert curve that creates its level of locality preserving : the adjacency. Thus, it avoids the dependence on the pattern RBG, which is the only pattern of the curve extended by previous researches. The result is a family of curves preserving well the locality.

The optimization of the locality preserving is also addressed to find out the best locality preserving curve. For this purpose, we propose a measure of the locality preserving. Based on the parameters, this measure can adapt to different application situations such as the change of metric or locality size.

The curve construction is an important part of the thesis. It is the basis of the index calculation used in application. For a rapid index calculation, the self-similar Hilbert curves is used. They are Hilbert curves satisfying the self-similar conditions specified in Chapitre 4.

The generalized curve is finally applied in image search. It is the question of the content-based image search (CBIR) where each image is characterized by a multidimensional vector. Images are ordered by the curve of a line, and the search is simplified to the search on an ordered list. By giving an input image, similar images are those corresponding to neighbors of the index of the input. The locality preserving ensures that these indexes correspond to similar images.

Keywords : space-filling curve, Hilbert curve, locality preserving, CBIR

Remerciements

A l'issue de la rédaction de cette recherche, je suis convaincue que la thèse est loin d'être un travail solitaire. En effet, je n'aurais jamais pu réaliser ce travail doctoral sans le soutien de plusieurs personnes dont la générosité, la bonne humeur et l'intérêt manifestés à l'égard de ma recherche m'ont permis de progresser dans cette phase délicate de ``l'apprenti-chercheur''.

En premier lieu, je tiens à remercier mon directeur de thèse et mes encadrants scientifiques, monsieur Rémy MULLOT, monsieur Patrick FRANCO et monsieur Jean-Marc Ogier pour la confiance qu'ils m'ont accordé en acceptant d'encadrer ce travail doctoral, pour leurs multiples conseils scientifiques et stratégiques.

Je souhaiterais exprimer ma gratitude à madame Nicole VINCENT, à monsieur Jean-Philippe DOMENGER et à monsieur Rolf INGOLD pour avoir accepté de juger mes travaux et pour avoir dégagé du temps pour y consacrer.

Je remercie tout l'équipe du laboratoire L3i pour son accueil chaleureux et pour les qualités humaines de ses membres.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Résumé:
Nous introduisons dans ce chapitre le thème de la thèse qui comprend la notion de la courbe remplissant l'espace, son objectif, ses applications possibles, les problèmes ouverts et notre proposition permettant résoudre ces problèmes. La structure de la thèse est aussi présentée à la fin du chapitre.

1.1 La recherche dans les grandes bases d'images

1.1.1 La recherche d'images : par mots clés vs. par contenu

Avec le développement des outils de création (appareils photo, smartphones, caméras embarquées, …) et des systèmes de stockage et partage (disques durs, serveurs, réseaux sociaux), les images sont plus en plus nombreuses dans les systèmes informatiques. Dans ce contexte, la recherche est immanquable pour exploiter les images.

D'une façon générale, une recherche retourne les documents qui satisfont le mieux à une requête. Nous constatons que la formulation de la requête influence l'efficacité de cette recherche. Par exemple, dans la recherche de documents textuels, le type de recherche le plus connu, on formule normalement les requêtes avec les mots clés. Nous témoignons l'efficacité de cette formulation de requête via l'usage et l'efficacité de différents moteurs de recherche comme Google, Bing, Yahoo. C'est une formulation naturelle pour les documents textuels parce qu'ils sont composés à partir des mots. Cependant, une formulation naturelle de requête n'est pas toujours facile pour d'autres types de documents comme l'image, le son, le vidéo. Pour les images, une solution similaire peut aussi être utilisée : la recherche d'images par mots clés. Dans ce type de recherche, chaque image est liée aux quelques termes concernant son contenu par l'étiquetage. Effectivement, la recherche d'images par mots clés est implémentée dans les services de recherche d'images connues comme Google Images, Bing Images.

Cependant, cette solution expose clairement deux points faibles :

— Les mots clés ne garantit pas la description du contenu de l'image : le contenu de l'image est visuel, il ne peut pas être intégralement représenté par les mots clés.

— Cette approche nécessite un coût humain important pour l'étiquetage des images.

Dans ce contexte, on propose la recherche d'image par contenu (*CBIR - Content-based image retrieval*). Le CBIR rassemble les techniques de la vision par ordinateur pour décrire le contenu de l'image. Cette approche permet d'éviter les manipulations manuelles par une description automatique du contenu de l'image.

1.1.2 CBIR : la caractérisation multidimensionnelle

Dans le CBIR, le contenu d'une image est souvent décrit par un descripteur ayant la forme d'un vecteur multidimensionnel, qui est souvent appelé signature de l'image. Par exemple, le descripteur SIFT (*Scale-invariant feature transform*) est de dimension 128 [104], l'histogramme de dimension 166 est utilisé dans [144], la description de forme par les moments de Zernike d'ordre 9 est de dimension 30 [118].

Ces exemples montrent aussi que la dimension des descripteurs peut être grande. En effet, un descripteur de grand dimension permet de décrire mieux le contenu de l'image. Par contre, il reste à noter que cette description est de bas niveau.

1.1.3 Indexation des images : défi multidimensionnel

Pour répondre à la requête, le système lance la recherche qui est en principe une comparaison de la signature du document avec les documents dans la base pour retrouver ceux qui sont les plus similaires. Cependant, une comparaison avec tous les documents dans la base est coûteux, surtout dans le cas des grands bases. C'est pourquoi on fait l'indexation, l'étape qui permet d'organiser les informations concernant les documents afin d'accélérer la comparaison.

À titre exemple, pour mieux chercher dans une séquence des nombres, on ordonne cette séquence. Dans une séquence ordonnée, nous pouvons appliquer la recherche par dichotomie qui examine au maximum $\log_2(n)$ parmi n éléments de la séquence. Dans les applications réelles, la table de hachage [36] et l'arbre-B [43] sont les structures souvent utilisées pour l'indexation.

Cependant, ces méthodes ne fonctionnent qu'avec les données 1-D. Dans le cas de donnée multidimensionnelle, quelques méthodes sont proposées comme arbre-KD [166], arbre-R [67]. Cependant, ces méthodes montrent de faibles performances [19, 101, 116] pour le cas des très grandes dimensions des descripteurs utilisées dans le CBIR.

1.2 Courbes remplissant l'espace

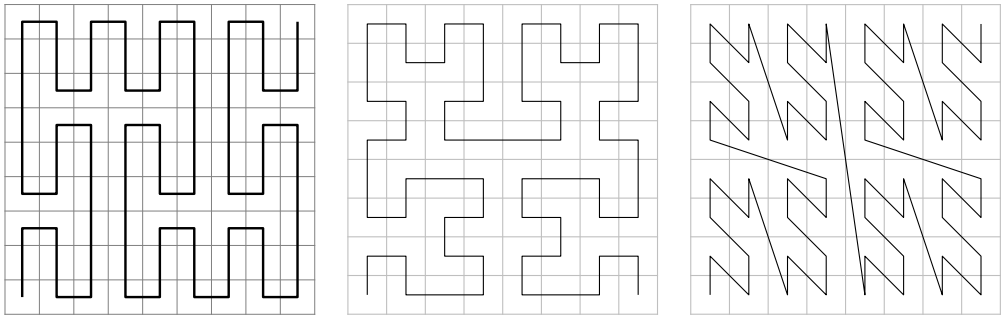
Dans cette thèse, pour résoudre le dysfonctionnement des méthodes existantes, nous proposons d'appliquer les courbes remplissant l'espace pour permettre de surpasser l'obstacle des grandes dimensions.

1.2.1 Remplissage de l'espace

Une courbe remplissant l'espace est un chemin qui passe par tous les points de l'espace. Ainsi, nous pouvons la considérer comme un ordonnancement de ces points. La Figure 1.1 montre 3 courbes remplissant l'espace connues proposées par Peano, Hilbert, Lebesgue.

1.2.2 Indexation multidimensionnelle : réduction de dimension

La réduction de dimension est une piste pour l'indexation multidimensionelle [76]. La correspondance d'un point à son index par la courbe remplissant l'espace



(Peano) *(Hilbert)* *(Lebesgue)*

FIGURE 1.1 – Les courbes remplissant l'espace connues proposées par Peano, Hilbert et Lebesgue. Une courbe remplissant l'espace est un chemin passe par tous les points de l'espace.

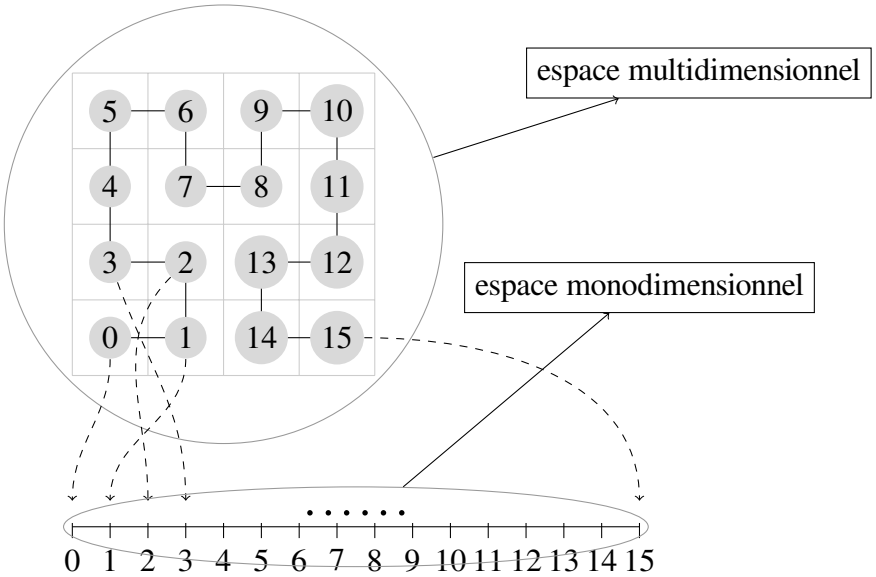
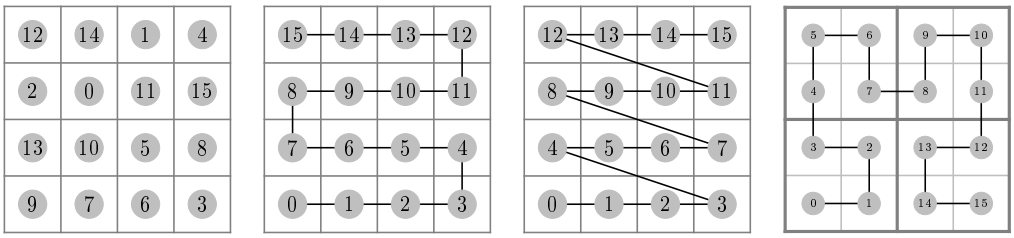


FIGURE 1.2 – L'indexation avec la courbe remplissant l'espace. Chaque point multidimensionnel correspond à un index 1-D. L'espace multidimensionnel est donc projeté sur une ligne (l'espace 1-D).

est une réduction de dimension à 1. Cette réduction de dimension nous permet de transformer l'indexation multidimensionnelle vers l'indexation 1-D [*cf.* Figure 1.2]. L'indexation 1-D est minutieusement étudiée avec plusieurs méthodes efficaces. La table de hachage [36] et l'arbre-B [43], que nous avons ci-dessus abordés, sont des méthodes typiques de l'indexation 1-D.

1.2.3 Conservation de la localité

La correspondance $\mathbf{d}$-D à 1-D facilite l'indexation dans le cas de grandes dimensions mais elle ne garantit pas l'efficacité de l'indexation. En effet, il existe de multiples façons pour réaliser la correspondance $\mathbf{d}$-D à 1-D. Par exemple, les courbes nommées Scan et Sweep [*cf.* Figure 1.3 *(Scan,Sweep)*] sont des méthodes simples permettant de passer par tous les points de l'espace. Un ordonnancement aléatoire peut aussi faire correspondre des points à leurs index 1-D [*cf.* Figure 1.3 *(Aléatoire)*].



(Aléatoire) *(Scan)* *(Sweep)* *(Hilbert)*

FIGURE 1.3 – Exemples d'ordonnance de l'espace de dimension 2.

L'avantage des courbes remplissant l'espace par rapport d'autres ordonnancements est la conservation de la localité : capacité d'ordonner les points auprès de leurs voisins. À titre d'exemple, nous considérons la courbe Scan, la courbe de Hilbert dans la Figure 1.3, et les points correspondants aux index 0, 1, 2, 3 dans chaque courbe. La courbe de Hilbert ordonne ces points de façon à ce que des points proches dans l'espace se retrouvent quasi séquentiellement lors du parcours dans l'espace et donc sur la courbe 1-D décrivant ce parcours. Cela est moins vrai pour Scan. Ainsi, la courbe de Hilbert conserve mieux la localité par rapport la courbe Scan.

1.3 Recherche d'images dans les grandes base : indexation multidimensionnelle par les courbes remplissant l'espace

Les courbes remplissant l'espace sont appliquées dans plusieurs domaines de l'informatiques, des mathématiques et l'électroniques. Dans l'informatique, les courbes remplissant l'espace sont connues avec les applications dans la base de données [56, 92] et la compression d'images [88, 112].

Notre objectif dans cette thèse est d'exploiter la capacité d'appliquer des courbes remplissant l'espace dans la recherche des images dans les grandes bases. Il s'agit

d'une recherche CBIR avec les images caractérisées par les descripteurs de grande dimension.

Nous décrivons ici une simplification de notre application qui va être détaillée dans le Chapitre 6. L'objectif de cet exemple est d'illustrer le système de recherche [*cf.* Chapitre 6] et l'intérêt des courbes remplissant l'espace.

1.3.1 Système de recherche proposé : idée générale

Le système de recherche proposée comprend deux parties : l'indexation des images et la recherche. Dans l'indexation, chaque image est mise en relation avec son index (l'ordre sur la courbe remplissant l'espace choisie) et stockée dans la base dans l'ordre des index. Comme la courbe remplissant l'espace conserve la localité, les images ordonnées proches dans la base sont similaires.

Ainsi, la recherche revient alors simplement à la recherche sur l'espace des index (1-D) : à la suite du positionnement de l'index de l'image d'entrée, les images correspondantes aux index voisins sont résultats.

1.3.2 Exemple : Indexation d'images couleur

À titre d'illustration, nous considérons les images simples de couleur homogène. Chaque image contient une seule couleur [*cf.* Figure 1.4]. Il y a 64 images, les couleurs étant décrites dans le système RVB (rouge, vert, bleu) avec les valeurs qui varient entre 0 et 255. Les 64 couleurs examinées (r, v, b) satisfont $r, v, b \in \{0, 85, 170, 255\}$.

Nous considérons aussi deux différentes méthodes d'ordonnancement : Sweep et la courbe de Hilbert. La méthode d'ordonnancement Sweep est simple, elle ordonne les couleurs comme suit :

$$
\begin{array}{cccc}
 & (0,0,0) & \to (85,0,0) & \to (170,0,0) & \to (255,0,0) \\
\to & (0,85,0) & \to (85,85,0) & \to (170,85,0) & \to (255,85,0) \\
\to & (0,170,0) & \to (85,170,0) & \to (170,170,0) & \to (255,170,0) \\
\to & (0,255,0) & \to (85,255,0) & \to (170,255,0) & \to (255,255,0) \\
\to & (0,0,85) & \to (85,0,85) & \to (170,0,85) & \to (255,0,85) \\
\vdots & & & & \\
\to & (0,255,255) & \to (85,255,255) & \to (170,255,255) & \to (255,255,255)
\end{array}
$$

Le résultat de cet ordonnancement est figuré dans la Figure 1.4.

Le défaut de cet ordonnancement est les grandes différences de couleur entre plusieurs couples d'images consécutives. Par exemple, les deux couleurs ordonnées consécutives (255,0,0) et (0,85,0) sont très différentes. Dans la Figure 1.4 les ``*'' marques quelques changements brusques de couleurs des images consécutives.

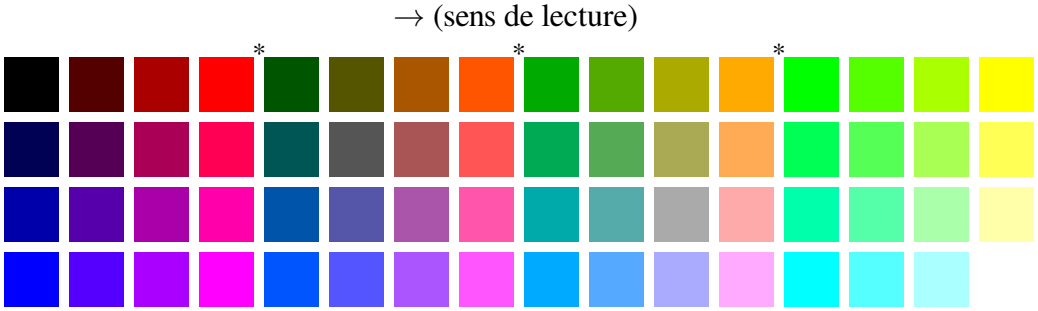


FIGURE 1.4 – Les images ordonnées selon un parcours Sweep.

La courbe de Hilbert permet d'éviter ce problème avec sa propriété de conservation de la localité. Dans cet ordonnancement, deux images ordonnées consécutives ont toujours des couleurs similaires. La Figure 1.5 montre l'ordre des images selon la courbe de Hilbert.

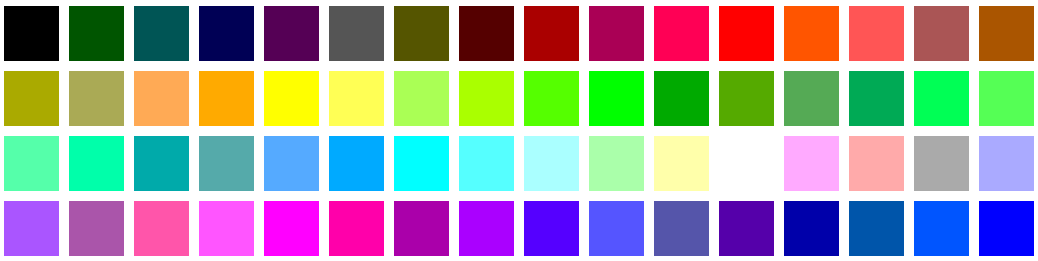


FIGURE 1.5 – Les images ordonnées par la courbe de Hilbert.

Nous considérons que cet ordonnancement est une indexation des images et l'index d'une image est son ordre. Avec cette indexation, la recherche d'image se transforme en une recherche de l'index dans une séquence ordonnée des index. La recherche dans une séquence est investiguée dans la littérature avec plusieurs méthodes efficaces proposées. Par exemple, la recherche par dichotomie est une méthode efficace et peut être appliquée pour retrouver l'index de requête et ses voisins.

Dans le Chapitre 6, nous développons une application sur ce même principe mais avec une grande dimension et d'autres courbes remplissant l'espace. Cette application nous permet de comparer l'efficacité de nouvelles courbes que nous allons développer dans le Chapitre 4 et le Chapitre 5.

1.4 Objective et organisation de la thèse

1.4.1 Objectif

Notre objectif général est de faciliter l'utilisation et d'optimiser l'efficacité des courbes remplissant l'espace dans la recherche d'images. Concrètement, les deux objectifs principaux sont :

— *optimisation la conservation de la localité : chercher la courbe qui conserve mieux la localité.* Comme la conservation de la localité influe fortement sur l'efficacité de l'indexation pour permettre de retrouver les sorties les plus pertinentes, elle doit être mesurée et utilisée comme le critère pour le choix de la courbe remplissant l'espace.

— *optimisation la correspondance n-D - 1-D : développer un algorithme rapide pour calculer l'index à partir d'un point.* Ce calcul est appliqué sur chaque image pour retrouver son index, que ce soit l'indexation ou dans la recherche. Ainsi, sa complexité influence fortement sur la performance du système de recherche.

En fixant la priorité sur conservation de la localité, nous exploitons ainsi plusieurs possibilités d'extension multidimensionnelle de la courbe de Hilbert. Une méthode de construction générale applicable à toutes ces courbes est également proposée.

1.4.2 Problèmes ouverts

La courbe de Hilbert est proposé depuis plus de 100 ans et trouve son utilité dans nombreuses applications. Cependant, il n'existe pas de définition de la courbe de Hilbert multidimensionnelle permettant sa mise en œuvre [1].

Sans l'éclairage de définition, les propositions pour la construction des courbes sont incomplètes. Concrètement, en dehors des constructions de la courbe 2-D, qui sont déjà suggérées par Hilbert, il n'existe que deux méthodes principales [23, 29] permettant la construction de courbes multidimensionnelles [2] [*cf.* Section 2.4]. Cependant, ces deux méthodes ne construisent qu'une seule extension (une seule courbe à chaque dimension donnée). Cela ne respecte pas la tendance générale de l'extension multidimensionnelle, qui produit plusieurs possibilités d'extension. Cette limite supprime la possibilité de choisir des courbes, surtout dans le but d'optimisation de la conservation de la localité.

1. À notre connaissance, la seule définition est trouvée dans [133]. Cependant, celle-ci est abstraite et peu utile aux applications

2. Les autres méthodes de construction de la courbe de Hilbert sont concrétisations, restrictions ou répétitions de ces deux méthodes

1.4.3 Démarche

Pour parvenir à ces objectifs, nous proposons de nous centrer sur les points suivants :

Définition La définition existante de la courbe de Hilbert multidimensionnelle est abstraite. Elle n'est pas suffisamment précise pour la transposer directement dans des applications informatiques. Une définition concrète est nécessaire pour fixer l'objet des opérations et analyses sur les courbes remplissant l'espace comme la construction et l'optimisation de la conservation de la localité. Les courbes de Hilbert multidimensionnelle doivent conserver les propriétés de la courbe de Hilbert originelle 2-D. Cela permet de maintenir la bonne conservation de la localité de la courbe de Hilbert. En effet, nous imposons que la définition généralisée doit sortir la courbe de Hilbert originelle dans le cas de 2 dimensions.

Construction de courbe La construction est l'étape indispensable pour toute application. Une construction complexe (en temps ou en mémoire) induit la même complexité sur l'application. Nous proposons dans cette thèse une méthode de construction de courbe avec deux propriétés :

- Universalité : cette méthode peut construire toutes les courbes remplissant l'espace définies dans la Chapitre 5.
- Efficacité : elle peut construire rapidement des courbes avec un espace mémoire réduit. Nous nous sommes penchés sur l'optimisation de nos algorithmes, ce qui a donné naissance à une version ``en ligne''.

Conservation de la localité La conservation de la localité est une propriété recherchée dans les applications des courbes remplissant l'espace. Nous essaierons d'optimiser cette propriété. En sachant qu'il y a plusieurs courbes de Hilbert quand la dimension est supérieure à 2, l'optimisation choisit, parmi ces courbes, la courbe qui maximise la conservation de la localité.

1.4.4 Organisation de la thèse

Dans le chapitre suivant (Chapitre 2), nous décrivons brièvement des recherches dans la littérature qui concernent les courbes remplissant l'espace : la proposition des courbes, la construction, la conservation de localité, l'application, etc.

Le Chapitre 3 détaille la conservation de la localité, la propriété importante des courbes remplissant l'espace. Afin d'optimiser la conservation de la localité des courbes remplissant l'espace, nous devons d'abord déterminer la mesure de la conservation de la localité. En évitant les défauts des mesures existantes, nous proposons

une nouvelle mesure de la conservation de la localité. C'est une mesure paramétrable, qui peut être modifiée pour adapter aux différentes situations de l'application. Les comparaisons montrent que, avec les paramètres pertinents, cette mesure peut s'approcher des mesures existantes.

Les contenus concernant la courbe remplissant l'espace sont décrits dans le Chapitre 4. Elle concerne les notions générales de la courbe remplissant l'espace comme la définition de l'espace, de la courbe remplissant l'espace et aussi sa conservation de la localité et ses applications.

La proposition de l'extension de la courbe de Hilbert est décrite dans le Chapitre 5. La définition, la construction et l'optimisation de la courbe de Hilbert multidimensionnelle sont présentées dans ce chapitre.

En appliquant les éléments détaillés dans les chapitres précédents, le Chapitre 6 développe les algorithmes de correspondance des points avec leur index ainsi que les outils permettant la mise en œuvre de l'application. En s'appuyant sur ces algorithmes, nous expérimentons la performance des courbes remplissant l'espace dans la recherche d'image. Une nouvelle courbe qui est née par la définition proposée est comparée avec la courbe de Hilbert connue construite par l'algorithme de Butz.

Chapitre 2

État de l'art

Résumé:

Ce chapitre décrit brièvement la notion ``courbe remplissant l'espace'', ses propriétés utiles dans les applications, sa construction, etc. et la positionement de notre proposition par rapport aux travaux existants.

2.1 Les courbes remplissant l'espace

2.1.1 La naissance des courbes remplissant l'espace

L'ordonnancement total d'un ensemble multi-dimensionnel peut être réalisé par la correspondance de chaque point multidimensionnel avec un index dans un ensemble mono-dimensionnel. Autrement dit, les points multidimensionnels sont associés avec leur nombre ordinal (index). Cela est toujours faisable même dans le cas des espaces continus selon la découverte de Cantor publiée en 1878 [34] : il existe une bijection entre les variétés différentiables de dimension finie, même dans le cas où leurs dimensions sont différentes. Par conséquence, il y a une bijection ϕ entre $[0,1]$ et $[0,1]^{\mathbf{d}}$:

$$\phi : [0,1] \leftrightarrow [0,1]^{\mathbf{d}}$$

Cependant, une bijection quelconque ne garantit pas l'intérêt dans l'application. La bijection mentionnée ci-dessus a plus de sens si elle est continue. En d'autres termes, les points (dans $[0,1]^{\mathbf{d}}$) correspondant aux valeurs proches (dans $[0,1]$) sont aussi proches. Il n'y a donc pas de saut de la bijection ϕ. Toutefois, un an après la découverte de Cantor, Netto a montré qu'il est obligatoire que ϕ soit discontinue [117].

Néanmoins, la question de l'existence d'une surjection continue de $[0,1]$ dans $[0,1]^{\mathbf{d}}$ est toujours ouverte. Une surjection d'un espace 1-D dans un espace $\mathbf{d}$-D permet de faire correspondre chaque point multidimensionnel avec au moins un index 1-D distingué. Effectivement, une surjection d'un espace 1-D sur un espace multidimensionnel est réalisable et cela est matérialisé par la proposition de la première courbe remplissant l'espace : la courbe de Peano [127].

2.1.2 La courbe de Péano

La courbe proposée par Péano en 1890 remplit une aire plane [127], c'est-à-dire l'espace de 2-D. Dans son article, une courbe qui remplit une aire plane est définie comme une courbe qui passe par tous les points d'un carré :

Définition 1 (Courbe remplissant l'aire plane)**.** *Deux fonctions x et y, uniformes et continues d'une variable réelle t, qui, lorsque t varie dans l'intervalle* $[0,1]$*, prennent toutes les couples de valeurs telles que* $0 \leq x \leq 1$, $0 \leq y \leq 1$.

Pour éclairer cette définition, la notion de courbe est aussi abordée :

Définition 2. *Une courbe continue est le lieu des points dont les coordonnées sont des fonctions continues d'une variable.*

Le coeur de l'article [127] est la correspondance analytique d'une valeur de t dans l'intervalle [0,1] avec les coordonnées d'un point (x, y) dans le carré $[0, 1]^2$, Peano écrit d'abord ces nombres sous la base 3. C'est-à-dire :

$$\begin{aligned} t &= 0, a_1 a_2 a_3 \ldots \\ x &= 0, b_1 b_2 b_3 \ldots \\ y &= 0, c_1 c_2 c_3 \ldots \end{aligned}$$

avec $a_i, b_i, c_i \in \{0, 1, 2\} \ \ \forall i \in \{1, 2, 3, \cdots\}$. Notons que l'écriture d'un nombre $u \in [0, 1]$ sous la base 3 est $0, u_1 u_2 u_3 \ldots$ si $u = u_1 3^{-1} + u_2 3^{-2} + u_3 3^{-3} + \ldots$.

La relation entre ces éléments est définie avec l'aide de la fonction $k(v) = 2 - v$ $\forall v \in \{0, 1, 2\}$. L'application p fois de k est notée par k^p, $k^0(v) = v, k^p(v) = k(k^{p-1}(v)) \ \forall v \in \{0, 1, 2\}$. Notons que si $w = k^p(v)$, alors, $v = k^p(w)$ pour $p \in \{1, 2, 3, \cdots\}$ et $v, w \in \{0, 1, 2\}$.

La relation entre x, y et t est formulée comme suit :

$$\begin{aligned} a_1 &= b_1 \\ a_2 &= k^{b_1}(c_1) \\ a_3 &= k^{c_1}(b_2) \\ a_4 &= k^{b_1+b_2}(c_2) \\ &\vdots \\ a_{2n-1} &= k^{c_1+c_2+\ldots+c_{n-1}}(b_n) \\ a_{2n} &= k^{b_1+b_2+\ldots+b_n}(c_n) \end{aligned}$$

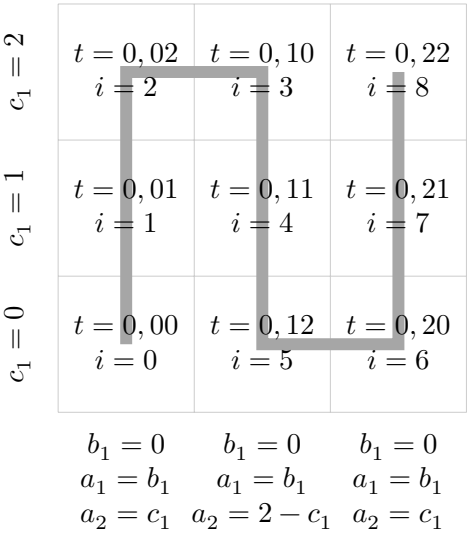
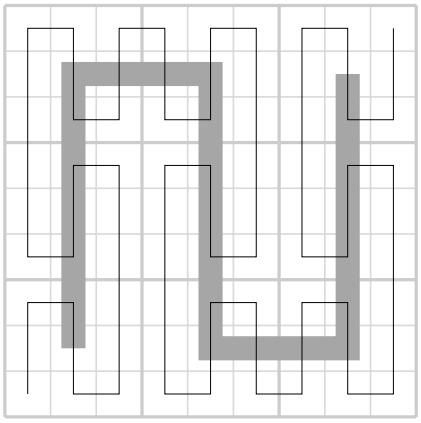


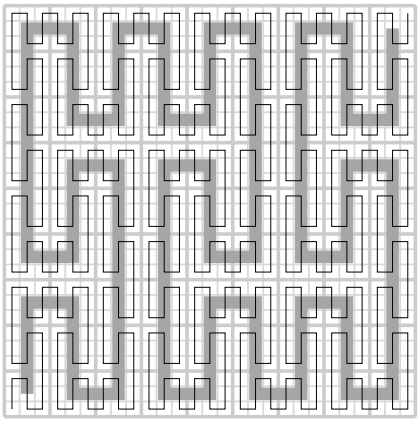
FIGURE 2.1 – La courbe de Peano d'ordre 1 : Il n'y a que le premier chiffre (*i.e.* b_1, c_1) de chaque coordonnée qui est considéré. L'index t est déduit à partir l'extrait concernant de la relation ci-dessus : $a_1 = b_1$; $a_2 = k^{b_1}(c_1)$; $t = 0, a_1 a_2$. Dans une application informatique, nous pouvons obtenir l'index en entier par la formule $i = 3a_1 + a_2$. La courbe est le parcours traversant tous les points avec l'ordre croissant de l'index i.

En appliquant cette relation avec la précision d'un chiffre (sans le 0 et la virgule, *i.e.* $x = 0, b_1$ et $y = 0, c_1$) nous avons la courbe de Peano d'ordre 1 [*cf.* Figure 2.1.2].

Si la précision de 2 chiffres est considérée (*i.e.* $x = 0, b_1 b_2$ et $y = c_1 c_2$), la relation ci-dessus permet de construit la courbe de Peano d'ordre 2. La courbe d'ordre 3 est construite de façon analogue [*cf.* Figure 2.2].



(ordre 2) : 2 chiffres de chaque coordonnées sont considérés *i.e.* $x = 0, b_1 b_2$ et $y = 0, c_1 c_2$. En appliquant la relation entre les coordonnées et l'index, nous avons
$a_1 = b_1; a_2 = k^{b_1}(c_1);$
$a_3 = k^{c_1}(b_2); a_4 = k^{b_1+b_2}(c_2);$
$t = 0, a_1 a_2 a_3 a_4;$
$i = 3^3 a_1 + 3^2 a_2 + 3 a_3 + a_4$



(ordre 3) : 3 chiffres de chaque coordonnées sont considérés *i.e.* $x = 0, b_1 b_2 b_3$ et $y = 0, c_1 c_2 c_3$. En appliquant la relation entre les coordonnées et l'index, nous avons
$a_1 = b_1; a_2 = k^{b_1}(c_1);$
$a_3 = k^{c_1}(b_2); a_4 = k^{b_1+b_2}(c_2);$
$a_5 = k^{c_1+c_2}(b_3); a_6 = k^{b_1+b_2+b_3 J(c_3)};$
$t = 0, a_1 a_2 a_3 a_4 a_5 a_6;$
$i = 3^5 a_1 + 3^4 a_2 + 3^3 a_3 + 3^2 a_4 + 3 a_5 + a_6$

FIGURE 2.2 – Les courbes de Peano d'ordre 2 et 3.

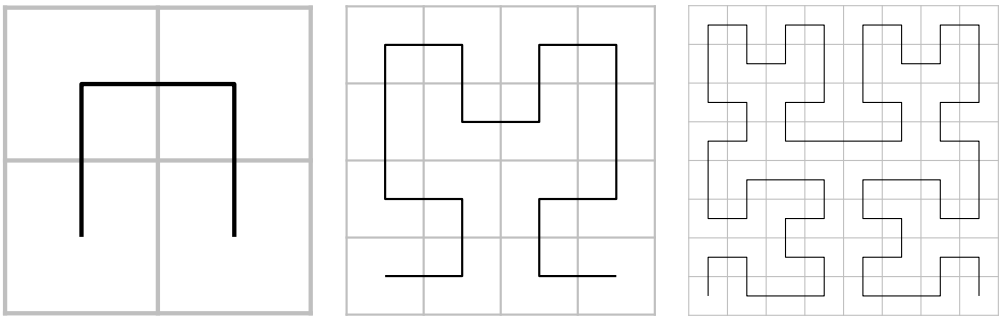
Nous apercevons que la courbe d'ordre 2 peut être construite par un remplacement de chaque point de la courbe d'ordre 1 par une courbe d'ordre 1 elle-même ou une transformée (réflexion par rapport l'axe x ou/et l'axe y) d'un motif qui est également la courbe d'ordre 1. Le remplacement chaque point de la courbe d'ordre 2 par une transformée du motif donne la courbe d'ordre 3.

Un tel processus construit la courbe de Peano d'ordre quelconque. Nous disons dans ce cas que la courbe est auto-similaire. Cette propriété est définie et analysée dans la Chapitre 5. L'auto-similitude est un facteur important de la conservation de la localité [*cf.* Chapitre 3] comme elle permet d'ordonner consécutivement les points dans une localité (qui remplace un point de la courbe d'ordre inférieur) : les points proches correspondent aux index proches. La conservation de la localité est brièvement abordée dans ce chapitre [*cf.* Sous-section 2.2.4] et plus détaillée dans

le Chapitre 3.

2.1.3 La courbe de Hilbert

En réponse à la proposition de Peano, Hilbert en 1891 propose une manière alternative de faire correspondre les points d'un carré avec ceux d'un segment [72]. La courbe de Hilbert est souvent employée dans les applications parce qu'elle conserve mieux la localité selon différents critères.



(ordre 1) *(ordre 2)* *(ordre 3)*

FIGURE 2.3 – La courbe de Hilbert jusqu'à l'ordre 3.

La construction de la courbe de Hilbert est décrite comme suit :

Le carré (l'espace 2-D) est d'abord divisé en 4 parties carrés égales par deux lignes orthogonales. Ces parties sont ensuite ordonnées selon la condition suivante :

Condition 1 (Hilbert-Peano [1])**.** *Deux parties ayant un bord commun ont deux index consécutifs*

Les mêmes opérations (division et ordonnancement) sont ensuite appliquées sur chacune des parties (résultant de la première division). Il en résulte 16 sous-parties dont leurs index sont choisis en respectant la Condition 1. Ce processus est répété sur chaque sous-partie créée.

La Figure 2.3 montre les 3 premières étapes de la construction de la courbe de Hilbert.

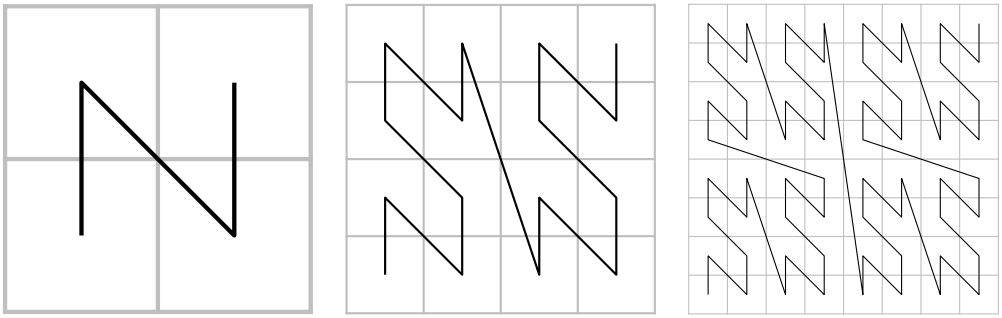
En tant que courbe qui conserve mieux la localité [56, 113] parmi les courbes connues, la courbe de Hilbert est largement utilisée dans les applications. Pour cette raison, la courbe est aussi l'objet principal de la thèse. Cependant, la définition originelle de Hilbert ne concerne que des courbes en 2-D. Une extension multidimensionnelle de la courbe est nécessaire pour les applications multidimensionnelles. La

1. Nous appelons cette condition *Hilbert-Peano* parce qu'elle est ``propriété commune'' aux courbes proposées par Hilbert et Peano

Chapitre 5 présente une généralisation de la courbe de Hilbert au cas multidimensionnel en évitant les limites des travaux existants relatés [*cf.* Section 2.3].

2.1.4 La courbe de Lebesgue

La courbe de Lebesgue [94] est aussi nommée ordre-Z pour sa forme d'une lettre Z ou Morton [114] pour la proposition de la courbe.



(ordre 1) *(ordre 2)* *(ordre 3)*

FIGURE 2.4 – La courbe de Lebesgue jusqu'à l'ordre 3.

La courbe ou plus précisément, sa version discrète, est connue par sa construction aisée : l'interfoliage des coordonnées binaires. Cependant, la version discrète, décrite par Schoenberg en 1938 [138], n'est pas une vraie courbe puisqu'elle comprend des sauts, ce qui réduit la conservation de la localité. La courbe fait correspondre un index écrit en binaire $t = (t^{(0)}t^{(1)} \cdots t^{(n)} \cdots)_2$ avec un point $\mathbf{d}$-dimensionnel $a = (a_0, a_1, \ldots, a_{\mathbf{d}-1})$ par l'*interfoliage des coordonnées* représentées en binaire comme suit :

$$\begin{array}{lcccccc} a_0 & = & (t^{(0)} & t^{(\mathbf{d})} & \ldots & t^{(k\mathbf{d})} & \ldots)_2 \\ a_1 & = & (t^{(1)} & t^{(\mathbf{d}+1)} & \ldots & t^{(k\mathbf{d}+1)} & \ldots)_2 \\ \vdots & & & & & & \\ a_{\mathbf{d}}-1 & = & (t^{(\mathbf{d}-1)} & t^{(\mathbf{d}+\mathbf{d}-1)} & \ldots & t^{(k\mathbf{d}+\mathbf{d}-1)} & \ldots)_2 \end{array}$$

La Figure 2.4 montre 3 premiers ordres de la courbe de Lebesgue.

2.1.5 D'autres courbes remplissant une aire plane

En dehors de ces 3 courbes remplissant l'espace abordées ci-dessus, il existe plusieurs autres courbes remplissant un espace. Par la relation avec ces 3 courbes connues, on peut diviser en 2 catégories de courbes :

— Les variantes de ces 3 courbes qui sont proposées en respectant les contraintes de la courbe originale. Par exemple, la courbe de Moore [133] est une variante connue de la courbe de Hilbert. On peut trouver 4 autres variantes de la courbe de Hilbert dans [103]. La courbe de Wunderlich [133] est une variante de la courbe de Peano.
— Les autres courbes remplissant l'espace. Les courbes de Polya, Schoenberg et Jordan sont dans cette catégorie. Nous les trouvons dans [133]. Ce livre contient un bon catalogue des courbes remplissant l'espace. Ces courbes remplissant l'espace ne sont pas obligées de remplir un carré. Dans ce livre, nous trouvons une définition de la courbe remplissant l'espace : courbe qui a un volume positif.

Comme les 3 courbes proposées par Peano, Hilbert et Lebesgue remplissent un carré, elles sont plus faciles à contrôler, par rapport à des courbes qui remplissent des formes moins régulières. C'est une des raisons pour lesquelles elles sont utilisées en informatique. La popularité d'une courbe est aussi liée à la conservation de la localité et la facilité de sa construction.

2.2 Les propriétés

En considérant qu'une courbe remplissant une aire plane est un cas particulier de la courbe remplissant un espace et que par ailleurs, une courbe remplissant un espace dispose des propriétés de la courbe remplissant une aire plan, nous pourrons utiliser ces propriétés généralisées aux courbes remplissant l'espace.

2.2.1 Fractal, remplissage de l'espace

Un objet est dit fractal si :

Condition 2 (Fractal)**.** *Sa dimension de Hausdorff est strictement supérieure à sa dimension topologique.*

En effet, Benoît Mandelbrot a utilisé le terme *courbe fractale* pour désigner les courbes satisfaisant la Condition 2 comme des courbes de Koch ainsi que toutes les courbes remplissant l'espace. Comme il s'agit de courbes, leur dimension topologique est égale à 1, tandis que la dimension de la courbe de Koch est d'environ 1,26, celles de Peano et de Hilbert 2-D sont égales à 2, qui est la dimension topologique d'un carré. Cela explique également pourquoi ces courbes remplissent une aire plane.

2.2.2 Auto-similitude

Les courbes remplissant l'espace sont auto-similaires, c'est-à-dire, leurs sous-parties sont les copies d'elles-mêmes à des échelles différentes.

En étant auto-similaire, une courbe remplissant l'espace peut être construite par un processus récursif.

Visuellement, les sous-parties d'une courbe auto-similaire conservent la forme de la courbe. Par conséquent, une transformation comme la rotation ou la réflexion peut être appliquée. Par exemple, dans les cas de la courbe de Peano ou de la courbe de Hilbert, leurs sous-parties peuvent être les copies à une échelle inférieure ou éventuellement avec les rotations d'angle $\pi/2$ ou $-\pi/2$.

Les transformations créent la possibilité de varier la structure des courbes remplissant l'espace et d'optimiser quelques propriétés de la courbe. Typiquement, ce type de transformations de la courbe de Hilbert permet d'optimiser la conservation de la localité ce qui est très importante dans les applications [*cf.* Chapitre 3]. Évidement, l'adaptation à ces transformations pose alors des difficultés pour la construction des courbes.

2.2.3 Non auto-croisement

L'auto-croisement des courbes remplissant l'espace est interdit dans les applications où l'index correspondant à chaque point doit être unique.

Par exemple, dans une base de données relationnelle, chaque clé primaire qui permet d'identifier uniquement une ligne dans une table doit être unique. Deux lignes distinctes d'une table ne peuvent pas avoir une même clé primaire. En supposant que des index des points (représentant des lignes de données) sur une courbe remplissant l'espace sont employés comme la clé primaire, ces index doivent être uniques.

Si une courbe remplissant l'espace se croise, les points d'intersection correspondent à 2 index ou plus. En conséquent, elle ne peut pas être appliquée dans les applications nécessitant un index unique.

La version discrétisée des courbes remplissant l'espace classiques comme les courbes de Peano, Hilbert ou Lebesgue, qui sont bijectives, garantit qu'il n'y a pas d'auto-croisement dans ces courbes (discrètes).

2.2.4 La conservation de la localité

La conservation de la localité est un point essentiel d'une courbe remplissant l'espace qui permet de faire correspondre des points proches avec les index proches. La conservation de la localité est parfois appelée *la conservation de voisinages*puisqu'elle mesure la capacité à faire correspondre les voisins dans l'espace multi-dimensionnel avec leur index (dans l'espace 1-D) qui sont aussi voisins.

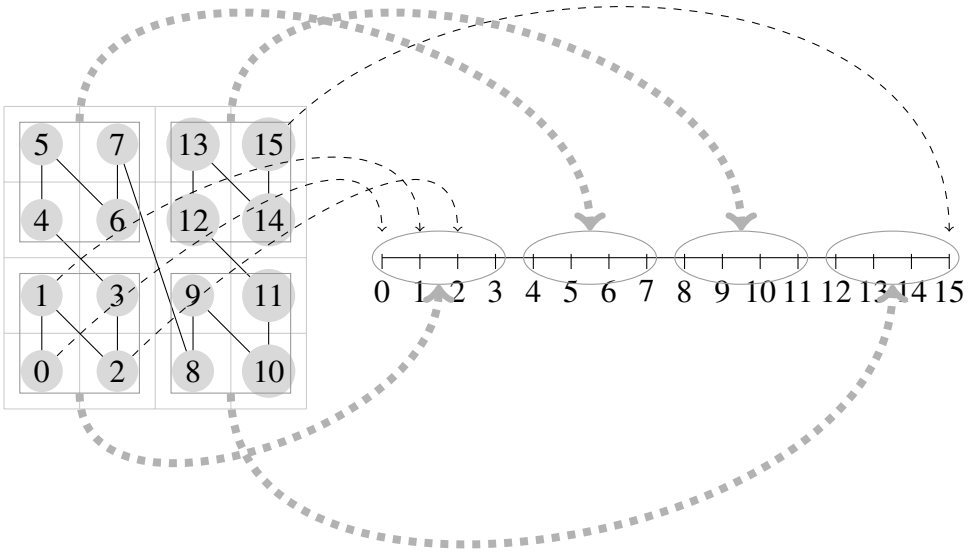


FIGURE 2.5 – La conservation de la localité de la courbe de Lebesgue. Chaque localité correspond à un segment des index. Ainsi, les points d'une localité sont ordonnés consécutivement. Cependant, si nous considérons l'ordonnancement des points dans chaque localité, il existe des points qui ne sont pas voisins, et qui sont consécutivement ordonnés. La courbe de Hilbert enlève ce saut de voisinage et permet une meilleure conservation de la localité [*cf.* Figure 2.6].

La conservation de la localité est un point important pour les applications informatiques et est souvent un critère essentiel pour le choix de la courbe remplissant l'espace. À titre d'exemple, dans l'application de recherche d'images dans une grande base [*cf.* Chapitre 6], des images similaires peuvent être rapidement retrouvées avec un système réalisant l'indexation et la recherche comme suit :

— L'indexation : chaque image est faite correspondre à un index. Les index sont ensuite ordonnés dans une liste. Avec cet ordonnancement, le positionnement d'un index est rapide, par exemple, par la recherche binaire.
— La recherche : les images issues de la recherche correspondent aux index voisins de l'index de l'image d'entrée.

Ce système n'est utile que si la correspondance entre l'image et l'index conserve bien la localité. La conservation de la localité garantit que les index consécutifs correspondent aux images similaires. Les sorties du système sont alors similaires

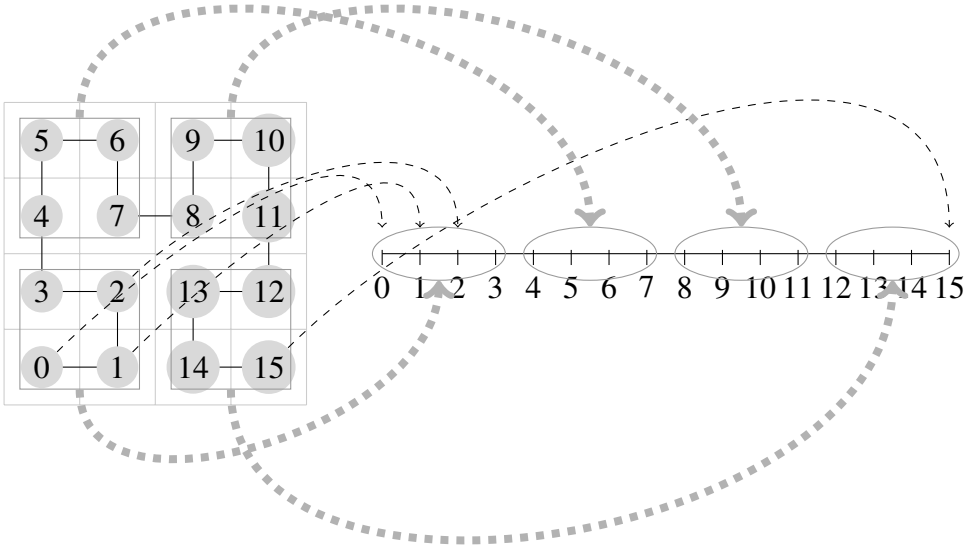


FIGURE 2.6 – La conservation de la localité de la courbe de Hilbert. Chaque localité correspond à un segment des index. Ainsi, les points d'une localité sont ordonnés consécutivement. De plus, dans chaque localité, chaque point est ordonné à côté 2 de ses voisins.

à l'image d'entrée.

L'auto-similitude [*cf.* Sous-section 2.2.2] est un acteur de la conservation de la localité. Avec l'auto-similitude, les points de chaque sous-parties sont consécutivement ordonnés [*cf.* Figure 2.5].

Malgré que toutes les courbes remplissant l'espace sont pareillement autosimilaires, elles conservent différemment la localité. La courbe de Hilbert qui optimise l'ordonnancement local conserve le mieux localité [*cf.* Figure 2.6]. La conservation dominant de la localité de la courbe de Hilbert est confirmée par différentes mesures [60, 113, 128].

Pour notre application, plus haut le niveau de la conservation de la localité, meilleure sera la similarité à l'image d'entrée.

Dans cette thèse, un des buts principaux est d'optimiser la conservation de la localité des courbes remplissant l'espace. La Chapitre 3 détaille la notion de conservation de la localité et présente une proposition d'une mesure de conservation de la localité qui généralise les mesures de la littérature [60, 113, 128].

2.3 L'extension multidimensionnelle

Dans leur version originale, les courbes de Hilbert, Peano et Lebesgue ne remplissent qu'un espace 2-D. Cependant, les applications informatiques actuelles demandent souvent des dimensions plus importantes que 2. Par exemple, dans le traitement d'images, la signature SIFT connue [104] est de 128 dimensions. Dans notre application de recherche d'images (par le contenu) [*cf.* Chapitre 6], les images peuvent être caractérisées par les moments de Zernike, qui sont les outils très populaires dans le domaine du traitement d'image. Cela contribue à manipuler un vecteur de caractéristique de dimension $\mathbf{d} = 30$.

Pour être appliquée dans ces cas, une extension multidimensionnelle des courbes remplissant l'espace est nécessaire. Quelques extensions existent.

L'extension multidimensionnelle peut être simple comme le cas de la courbe de Lebesgue. La courbe de Lebesgue 2-D est construite par l'interfoliage de 2 coordonnées. Pour le cas $\mathbf{d}$-dimensionnel, la courbe est construite de la même façon : l'interfoliage de $\mathbf{d}$ coordonnées. Dans la description de la courbe de Lebesgue, nous avons donné la méthode de construction générale pour le cas multidimensionnel.

Concernant la courbe de Peano, l'extension est un peu plus difficile à mettre en œuvre, mais elle reste déterministe parce que la courbe est définie par des relations formelles. Une extension multidimensionnelle de cette courbe est proposée dans [133].

Parmi les 3 courbes, la courbe de Hilbert est plus difficile à généraliser. En effet, la courbe est proposée par une règle de construction respectant la Condition 1 : l'adjacence des parties ordonnées consécutives. Pour satisfaire cette condition, la copie avec changement d'échelle de la courbe n'est pas suffisant pour créer des sous-parties de la courbe à chaque itération de la construction récursive. Les transformations sont nécessaires. Dans le cas multidimensionnel, le nombre de possibilités de transformation augmente. Cela est source de difficultés pour une extension au cas multidimensionnel.

Il existe des algorithmes qui permettent de générer une version multidimensionnelle de la courbe de Hilbert. C'est le cas de l'algorithme de Butz [29] et l'application du schéma de Bially [23] au cas de la courbe de Hilbert, par Faloutsos [56] ou par Lawder [93].

Cependant, ces algorithmes ne traduisent qu'une seule version multidimensionnelle de la courbe de Hilbert (formulée originalement en 2-D), alors que nous pensons qu'il existe plusieurs solutions pour satisfaire la Condition 1. Cette diversité des solutions (que nous baptisons famille) joue un rôle clés dans la définition de nouvelles courbes multidimensionelles présentant des niveaux de localité différents.

Nous montrerons [*cf.* Chapitre 5] que ces niveaux de localité sont en moyenne comparable à la version multidimensionnelle de la courbe de Hilbert (vu par les algorithmes de Butz et Lawder) existante jusqu'ici.

La variété des solutions est analysée dans le Chapitre 5.

2.4 La construction de courbes : analyse et positionnement parmi les travaux existants

La construction de courbes fait correspondre les points multi-dimensionnels avec leurs index. Il s'agit d'une étape incontournable pour l'application des courbes remplissant l'espace et influe directement sur la performance des courbes remplissant l'espace dans les applications.

La construction d'une courbe remplissant l'espace peut être réalisée de deux façons :

1. analytiquement : l'index de chaque point est déduit depuis ses coordonnées. Par exemple, Peano décrit la relation entre les coordonnées d'un point et son index via des fonctions.
2. énumérativement : on établit une liste de points ordonnés selon les index. Par exemple, la courbe de Hilbert d'ordre 1 est une liste des points : $(0,0), (0,1), (1,1), (1,0)$.

La courbe de Peano et la courbe de Lebesgue sont faciles à construire comme cela a été décrite dans la définition même de la courbe. Pour la courbe de Peano, le calcul de la formule définit simplement la courbe. La courbe de Lebesgue est générée par l'interfoliage des coordonnées.

La construction de la courbe de Hilbert est la plus difficile parmi les 3 courbes parce qu'elle doit réaliser des transformations, qui doivent satisfaire la Condition 1.

La construction de la courbe de Hilbert 2-D n'est pas très complexe. Cependant, la construction de cette courbe devient de plus en plus complexe avec l'augmentation de la dimension de l'espace. En généralisant à travers les ordres, à la dimension d'évolution $\mathbf{d}$ fixée, une courbe de Hilbert d'ordre $\mathbf{n}$ est constituée de $2^{\mathbf{d}}$ sous-courbes. Chaque sous-courbe est une version transformée de la courbe de Hilbert d'ordre antérieur $\mathbf{n}-1$ [*cf.* Figure 2.3 *(ordre 2), (ordre 3)*]. De plus, on remarque que pour $\mathbf{n}$ donné, les transformations géométriques (rotation, réflexion) diffèrent suivant la localisation même des sous-courbes (exemple pour $\mathbf{n}=2$, [*cf.* Figure 2.3 *(ordre 2)*](coin inférieur gauche et droit). L'enchaînement de ces transformations est un facteur clé pour construire une courbe qui préserve la localité. Or, lorsque les dimensions augmentent, la détermination de ces transformations et de leurs enchaînements n'est plus trivial y compris en faisant référence aux règles de Hilbert (établies en dimension 2) ou au modèle abstrait proposé dans [133]. La dimensionnalité est source de difficulté de construction. Cela se confirme dans la littérature où finalement peu d'algorithmes (à notre connaissance) hormis les célèbres schémas de Bially [22], Butz [29] opèrent au-delà de la dimension 3.

Bially [22] fait référence à des machines états-transition [*cf.* Figure 2.7] pour le calcul de l'index d'un point (à partir de ses coordonnées binaires) sur une courbe de dimension arbitraire ($\mathbf{d} >= 2$) [2] . Les matrices de transformations - qui participent à la construction même de ces machines - sont étroitement liées à des discussions portant essentiellement sur un seul type de courbe multidimensionnelle, la courbe RBG (reflected binary Gray code) [25]. C'est une approche claire, interprétable, servant de référence à divers articles [56, 93]. Néanmoins, ces machines doivent être re-générées lors de tout changement de dimension d'entrée d (modification du nombre d'états, de transitions, etc.). La génération de ces machines, dans sa version automatique [56], se heurte à la difficulté d'arbitrer parmi un ensemble de solutions qui apparaissent à diverses étapes de la conception. Ce qui en pratique rend son implémentation (dans sa version automatique) délicate au-delà de la dimension 5.

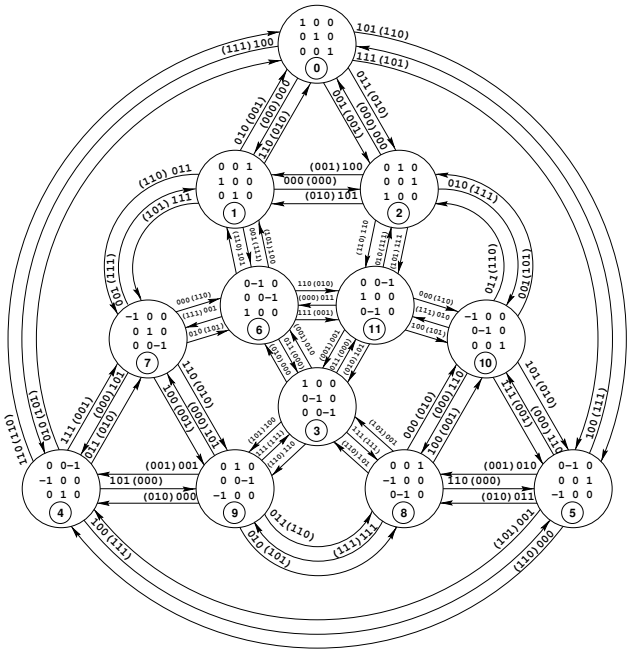


FIGURE 2.7 – Exemple de machine états-transitions [93] pour le calcul de l'index d'un point sur la courbe de Hilbert en $d = 3$ quelque soit n. Cette machine fait référence à un modèle (interne) de construction de courbes multidimensionnelles s'appuyant sur un motif primitif de type RBG. On reconnaît les matrices de transformation (cercles) traduisant l'enchaînement des sous-courbes à travers les divers ordres n.

Le schéma de Butz dans [29] s'appuie sur un modèle de transition valide quelle que soit la dimension, codé par une série d'opérations binaires. Il permet de retrouver

2. communément appelée courbe de Hilbert

un point de la courbe à partir de son index et non l'inverse. Or, de nombreuses applications nécessitent également le calcul de l'index (réversibilité).

Notons que dans ces propositions, les réflexions menées sur les transformations, leurs enchaînements reposent essentiellement sur la courbe multidimensionnelle RBG, *i.e.* finalement sur un nombre limité de manière de remplir l'espace. Est-il possible de construire d'autres courbes pour remplir un espace multidimensionnel qui vérifieraient un niveau de localité comparable à la courbe de Hilbert ?

2.5 Applications

Les courbes remplissant l'espace sont employées dans des champs disciplines aussi variés telles que l'informatique, les mathématiques [28, 30] et l'électroniques [68, 108, 109].

Nous dressons une liste (non exhaustive) d'applications qui concernent divers problèmes dans le domaine informatique au sens large.

— Base de données [56, 92]
— Traitement d'image [31, 90, 129]
— Compression d'image [88, 112]
— Vidéo [106, 107]
— Halftoning [9, 150, 165]
— Graphique [44, 50]
— Indexation [37, 79, 98, 142, 160]
— Données spatiales [87, 89, 145]
— Structure des données [8, 65, 81]
— Algorithmique [12, 23, 49, 123]

Ces applications s'appuient quasi systématiquement sur 2 propriétés des courbes remplissant l'espace :

1. la correspondance les points multidimensionnels avec les index 1-D,
2. et la conservation de la localité.

Avec la conservation de la localité, la courbe remplissant l'espace fait correspondre une grande partie des voisins d'un point avec les index autour de l'index de ce point. Cela permet de mettre en relation des opérations dans l'espace multidimensionnel avec les opérations correspondantes dans l'espace 1-D. Cette idée est illustrée, dans le Chapitre 1, à travers un exemple d'application qui concerne l'ordonnancement d'images couleurs.

	Courbe		
	Peano	Lebesgue	Hilbert
Base	3	2	2
Conservation de la localité	+	+	++
Extension multidimensionnelle	-	+	- -
Construction	+	+	- -

TABLEAU 2.1 – Vue globale des courbes remplissant l'espace : approche qualitative. Si la courbe de Hilbert est celle qui conserve le mieux la localité [56, 113], elle reste la plus difficile à construire au-delà de quelques dimensions ($\mathbf{d} \geq 4$).

2.6 Conclusion

Les courbes remplissant l'espace sont des objects utilisées en informatique avec deux usages principaux :

— Correspondance des points multidimensionnels avec les index 1-D. Cet usage peut être considéré comme une réduction de dimension, qui permet de simplifier les données et les opérations agissant sur ces données. Cette réduction de dimension à 1-D permet de facilité la recherche des voisins.

— Conservation de la localité. La localité des points est conservée dans le nouvel espace 1-D. Les voisins d'un point correspond aux index autour de l'index de ce point. Avec l'hypothèse que les points proches représentent les entités similaires ou à relation forte, la courbe remplissant l'espace permet d'ordonner de façon proche ces entités dans espace 1-D. En s'appuyant sur cette caractéristique, la courbe remplissant l'espace est une façon d'ordonner des données, qui permet d'accélérer leurs exploitations.

Pour être utilisée, nous avons besoin de construire la courbe elle-même. Pour le cas de dimension supérieure à 2, une extension multidimensionnelle est aussi nécessaire.

Le Tableau 2.1 compare qualitativement les points importants des trois courbes les plus connues. Dans ce tableau, nous revenons également sur la base de la courbe, qui est la taille sur chaque dimension de la courbe du premier ordre. Elle influence l'implémentation, une courbe binaire (de base 2) était mieux adaptée à la culture binaire de l'informatique. Pour cette raison, la fréquence des applications de la courbe de Peano est moins importante que les deux autres.

Malgré la difficulté de l'extension multidimensionnelle et de sa construction, la courbe de Hilbert est largement utilisée dans les applications pour sa conservation de la localité. Notons enfin que dans les propositions d'extension multidimensionnelle et de construction de la courbe de Hilbert, les réflexions menées sur les transformations et leurs enchaînements reposent essentiellement sur la courbe multidimensionnelle RBG, *i.e.* finalement sur un nombre limité de manière de remplir l'espace. Est-il

possible de construire d'autres courbes pour remplir un espace multidimensionnel qui vérifieraient un niveau de localité comparable à la courbe de Hilbert ?

Cette thèse tente de répondre à cette question en proposant une nouvelle définition d'une courbe remplissant l'espace adaptée au cas multidimensionnel. Cette approche généralise au cas multidimensionnel la démarche de Hilbert établie originalement en dimension 2. Elle est présentée dans le Chapitre 5. Il en résulte une famille de courbes dont le niveau de conservation de la localité (mesuré dans le Chapitre 3) est comparable à l'extension de la courbe de Hilbert réalisée par l'algorithme de Butz.

Chapitre 3

Conservation de localité

Résumé:

Nous présentons dans ce chapitre une proposition de définition de la conservation de la localité ainsi que sa mesure. Comme la localité est une notion qui se varie selon l'application, nous proposons une mesure adaptable de la conservation de la localité qui possède des paramètres modifiables pour des applications différentes.

Des applications de cette mesure dans le cas de l'espace métrique et la recherche des images seront décrites dans le Chapitre 6.

La *conservation de la localité*, est abordée populairement dans les recherches des courbes remplissant l'espace [31, 60, 113]. En effet, elle est normalement mentionnée comme la raison principale de l'utilisation de la courbe remplissant l'espace. Cependant, quand cette notion ne se limite pas dans le cadre des courbes remplissant l'espace, nous la trouvons également dans plusieurs autres thématiques de recherches [*cf.* Sous-section 3.1.1].

La conservation de la localité représente la capacité d'une fonction à faire correspondre des points proches dans l'espace source aux points proches dans l'espace cible. Une fonction peut conserver totalement la localité d'un point, par exemple pour l'application identité $f(x) = x$. Cependant, la conservation de la localité des fonctions est souvent partielle.

À notre connaissance, les recherches concernant la conservation de la localité utilisent cette notion de façon implicite ou dans un sens strict. Quelques articles proposent des mesures de la conservation de la localité sans aucune définition [1].

Dans cette section, nous proposons à la fois la définition de la conservation de la localité et sa mesure. C'est une mesure universelle qui s'adapte à la variation d'objectifs liés à des applications. En s'appuyant sur la mesure définie, nous pouvons choisir la fonction à appliquer selon le critère de conservation de la localité. Concrètement, dans nos recherches, l'optimisation de la conservation de la localité des courbes remplissant l'espace va être détaillée [*cf.* Chapitre 4].

3.1 La conservation de la localité : les applications, les travaux relatés et les mesures

3.1.1 Localité dans les applications

Initialement, la localité est la proximité spatiale ou temporelle, qui peut conduire à

- un traitement commun. Par exemple, dans la classification, les objets similaires sont mis dans une même classe,
- la corrélation de quelques caractéristiques. Par exemple, les pixels dans un même segment d'une image ont souvent des couleurs similaires.

Le terme "localité" est trouvé dans plusieurs domaines de l'informatique. Nous listons ci-dessous les plus connues et l'application typique de la localité dans ces domaines :

1. Sur les 100 premiers résultats de recherche pour les termes *localité* et son équivalent en anglais *locality* des moteurs de recherche Google, Google Scholar, ScienceDirect, SpringerLink, Microsoft Academic Search

Indexation de données [32, 46, 63, 70, 99, 105, 121]. C'est avant tout notre cas d'étude - application [*cf.* Chapitre 6]. Il s'agit d'une indexation où les points multi-dimensionnels proches correspondent aux index proches. L'idée est de conserver au maximum la segmentation de la base de données dans l'espace des index. Dans le cas idéal, chaque classe de données similaires correspond à un segment des index. Le positionnement des index est plus facile à traiter par rapport à la recherche des points dans l'espace d'origine. La recherche est ainsi accélérée en évitant la navigation directe sur l'espace des données. Les voisins d'une entrée sont donc trouvés depuis les index proches.

Compression d'images [24, 74, 80, 96]. Dans une image, les pixels proches ont souvent les couleurs similaires. Cette propriété est largement utilisée pour la compression des images, par exemple, la compression par des techniques des ondelettes ou par fractal.

Réduction de dimension [39, 62, 64, 97, 122, 126, 156--158]. La technique typique dans cette catégorie est LPP (locality preserving projections - projections conservant localité). En discriminant avec PCA (*principal component analysis* - analyse en composantes principales), qui est une technique de réduction de dimension au niveau global, LPP réduit la dimensionnalité via des analyses locales pour présenter un bon taux de réduction.

Traitement parallèle, système de multiprocesseur [1,4,10,11,61,71,82]. La localité présente un fort intérêt dans le traitement parallèle. Typiquement, un système possédant plusieurs processeurs est amené à partager une tâche avec des données sauvegardées sur une mémoire commune. Chaque processeur possède en plus une mémoire rapide et la tâche peut être divisée en petites tâches qui vont être distribuées aux processeurs. En traitant ces tâches, la communication et la synchronisation entre des processus occupent une quantité importante de traitements. La réduction de la communication est un objet essentiel dans l'optimisation de ces systèmes. L'idée est ici de charger chaque mémoire privée par les données qui sont fortement dépendants et réduire maximum la dépendance d'autres données. Cela permet une fréquence réduite de la synchronisation.

Gestion de mémoires [2, 48, 53, 95, 102, 124, 141, 155]. L'idée est similaire à l'optimisation de localité pour le traitement parallèle. Il existe en effet plusieurs types de mémoires avec des vitesses différentes. Par exemple, la mémoire cache est plus rapide par rapport la mémoire vive (RAM) et naturellement que les disques durs qui sont encore plus lents. L'idée est de charger une mémoire plus rapide par les données

plus fréquentes. Cette fréquence est liée à la localité temporelle : une bonne localité garantit une durée courte d'accès aux données.

Réduction des communications dans les réseaux [26, 66, 69, 77, 78, 125, 146, 162]. Dans le modèle de réseau pair à pair, la charge du réseau pour partager des fichiers peut être réduit si les téléchargements entre des ordinateurs de courtes distances sont prioritaires.

Ces exemples illustrent clairement que la localité est un principe générique dans différents domaines. La localité peut être spatiale, temporelle ou logique. Les éléments voisins sont souvent corrélatifs dans le traitement. Par conséquent, une structure conservant la localité facilite les traitements.

3.1.2 La conservation de la localité

Il est souvent difficile voire impossible de résoudre une problématique à partir des données originelles. Il est parfois nécessaire de transformer les données afin de les rendre plus facile à exploiter dans l'objectif que l'on s'est donné. Par exemple, la reconnaissance des lignes droites dans une image est facile à mettre en place avec la transformation de Hough [54]. L'utilisation des transformations sont très fréquentes dans le traitement de l'image, par exemple pour la transformation de Fourier [100], les ondelettes [110], transformation de Radon [21], etc.

La relation spatiale des points est fréquemment une contrainte du problème. Il est souvent nécessaire de la conserver dans le cas où une transformation est utilisée. Par exemple, la recherche d'images (par le contenu) que nous présentons dans le Chapitre 6 réalise une transformation qui fait correspondre chaque vecteur multidimensionnel (représentant d'une images) à un index 1-D. Cette transformation facilite l'indexation car cette indexation dans l'espace 1-D est clairement plus facile par rapport celle dans l'espace multidimensionnel. D'autre part, le but étant de retrouver les images similaires, la relation de similarité doit être présentée dans la nouvelle représentation (l'espace 1-D).

En effet, la conservation de la localité accompagne souvent des travaux de recherches concernant les courbes remplissant l'espace. Toutefois, elle n'est pas utilisée de manière exclusive dans le cadre des courbes remplissant l'espace. Elle paraît également dans d'autres types de transformations [33, 70, 122]. Par exemple, le travail de recherche dans [33] concerne en principe une optimisation de la conservation de la localité d'une représentation de données dans un espace de plus faible dimension.

3.1.3 Les mesures

Dans le cadre de la mise en œuvre des applications, il est utile de quantifier la localité des transformations et leur conservation. Une mesure nous permet de comparer la conservation de la localité des transformations et déterminer la meilleure selon ce critère.

Dans la littérature, nous ne trouvons que 3 mesures qui sont proposées pour évaluer les courbes remplissant l'espace. Dans la plupart des travaux de recherche référant la conservation de la localité, cette qualité n'est pas explicitement définie et quantifiée [33, 70, 122].

Ces 3 mesures sont proposées par les auteurs suivants :

Perez, Kamata et Kawaguchi [128]

$$L(C) = \sum_{i,j \in [N^m], i<j} \frac{|i-j|}{\mathfrak{d}(C(i), C(j))}$$

avec C la fonction qui fait correspondre chaque index à son point sur la courbe remplissant l'espace choisie. L'espace est de dimension m et de taille N sur chaque dimension. Ainsi, il y a en total N^m points qui correspondent à N^m index. Dans une perspective de réduction de la dimension, nous pouvons convenir que l'espace multidimensionnel est la source et les index sont dans la cible. Pour faire correspondre les points de la source à la cible, nous utilisons donc C^{-1}.

Mitchison et Durbin [111]

$$L_q(C) = \sum_{\{i,j \in [N^2] : i<j, d(C(i),C(j))=1\}} |i-j|^q$$

avec C la fonction qui fait correspondre chaque index à son point sur une courbe remplissant un espace 2-dimensionnel de taille N^2 et q un paramètre.

Bongki Moon [113] Cette mesure de la conservation de la localité des courbes remplissant l'espace utilise la notion *grappe* (*cluster*) qui est un segment des index consécutifs correspondants aux points dans une localité source définie. Pour cette mesure, les localités source sont des rectangles. Une localité source peut correspondre à plusieurs grappes. L'ensemble des grappes de la localité L_1 est noté $\mathsf{C}(L_1)$. La localité L_1 est bien conservée si $\#\mathsf{C}(L_1)$ est petit. En généralisant, une courbe conserve bien la localité si la somme des nombres de grappes correspondant aux toutes les localités est petite.

En analysant ces mesures, nous constatons que :

— ces mesures ne permettent que l'évaluation la conservation de la localité des courbes remplissant l'espace ou une transformation qui peut faire correspondre un point multidimensionel à une valeur.
— les deux premières mesures évaluent la conservation de la localité en utilisant la distance entre des points. Elles ne peuvent pas être employées dans un espace sans une métrique définie.

Enfin, chacune des mesures ci-dessus est définie suivant un objectif et se place dans une application particulière. Elle ne s'adapte pas lors de la variation de la définition de la localité.

3.2 Définition et mesure

3.2.1 Motivation

La définition de la localité dépend de l'application et peut varier beaucoup d'un cas à l'autre. De plus, une application peut examiner plusieurs localités avec des pondérations différentes. Par exemple, dans la segmentation d'image, la taille des régions est variable. Chaque région peut être considérée comme une localité. Par conséquent, il est nécessaire d'examiner des localités de tailles différentes.

Pour ces raisons, nous allons proposer une mesure adaptable en s'appuyant sur les caractéristiques suivantes :

— indépendance de la métrique : la localité peut être définie avec ou sans une métrique,
— paramétrique : en utilisant les localités comme paramètres, la mesure varie selon les localités en s'adaptant au besoin de l'application,
— pondérable : par la pondération, il est possible de combiner les différentes localités.

3.2.2 Localité

Dans un sens général, la localité d'un point est la zone dans laquelle ce point peut être trouvé. La localité est donc différente du voisinage, qui est défini comme l'entourage d'un point. La localité d'un point n'est pas obligé d'être centré sur ce point. Ainsi, nous définissons :

Définition 3 (Localité)**.** *Une localité est l'ensemble des points dans un segment de l'espace quelconque.*

Chaque point dans une localité est aussi dit un membre de cette localité.

Comme le segment de l'espace peut être librement défini, cette définition est adaptable. Chaque application peut définir la localité propre avec son objectif, la localité étant défini selon l'objectif de l'application.

La notion de la localité a naturellement une certaine relation avec le voisinage. Un voisinage est la localité comprenant son centre et ainsi que tous ses voisins. Cependant, la relation inverse n'est pas juste : la localité d'un point ne centre pas obligatoirement sur ce point.

Une localité peut être définie via le segment de l'espace. Par exemple dans l'espace 2-D, l'ensemble des points dans un rectangle (défini via ses sommets, par exemple) est une localité. Un point dans un espace métrique et tous ses voisins sont dans une localité. Comme le voisinage dans l'espace métrique est l'ensemble des points qui se placent dans un rayon défini depuis le centre [*cf.* Définition 4], le segment de cette localité est un cercle.

Définition 4 (Voisinage)**.** *Dans l'espace métrique (X,d), la localité de rayon* r>*0 d'un point* x *est la boule centrée en x et de rayon r :*

$$V^r_{(X,d)}(x) = \{t \in X | d(t,x) \leq r\}$$

La localité peut être directement définie par ses membres. La zone est le segment minimum qui contient tous ses membres. Par exemple, une localité peut être définie comme k-ppv (*k* plus proches voisins) d'un point. Dans le cas d'un graphe, une localité peut être définie comme l'ensemble des noeuds adjacents d'un noeuds (les noeuds ayant le lien direct avec lui).

Dans cette thèse, les localités sont aussi notées par lettre L. S'il y a différentes localités, elles sont numérotés $L_1, L_2, L_3, \cdots$. De la même manière, le voisinage est noté par lettre V. De plus, des exposants et des index peuvent apporter des informations complémentaires. Par exemple, le voisinage de x dans rayon r de l'espace métrique (X, d) est noté : $V^r_{(X,d)}(x)$. L'ensemble de toutes les localités et celui de tous les voisinages définis sur l'ensemble X sont notés respectivement par $\mathcal{L}(X)$ et $\mathcal{V}(X)$

3.2.3 Conservation de localité

Avec les localités source L_1 et cible L_2, la conservation de la localité d'une fonction f est définie comme suit :

Définition 5 (Conservation de localité)**.** *Posons deux localités L_1, L_2. La conservation de la localité L_1 dans la localité L_2 de l'application $f : X \to Y$ est mesurée par*

$$\zeta_f(L_1, L_2) = \frac{\#f(L_1) \cap L_2}{\#L_1} \tag{3.1}$$

Où $f(L_1)$ est l'ensemble des images produites par f des points dans L_1.

Explication: Suite à une application de f, la localité L_1 est transformée en $f(L_1)$. À noter que la localité $f(L_1)$ et L_2 sont rarement identiques, même avec des localités bien choisies. Les points conservés sont dans l'ensemble $f(L_1) \cap L_2$. Ainsi, la conservation de la localité est le ratio entre le nombre de points conservés et la totalité des points de la localité source.

La valeur de la conservation de la localité est limitée entre 0 et 1. La valeur 0 est une perte totale de la localité. Au contraire, la valeur 1 est assignée à la conservation totale de localité. Plus cette valeur est grande, mieux la fonction conserve la localité.

Exemple 1. Examinons la conservation de la localité dans la figure 3.1. La localité source est le voisinage $V_1(x)$ de x et celle cible est le voisinage $V_2(f(x))$ de son image $f(x)$. Parmi 5 points de $V_1(x)$, il y en a seulement 3 qui sont conservés dans $V_2(f(x))$ (les *s et x). Leurs images sont les $\circ$s et $f(x)$. En dehors de ceux-ci, la localité $V_2(f(x))$ contient en plus un autre voisin ($\diamond$) qui n'est pas l'image d'aucun voisin de x. Ainsi, la conservation de la localité de la fonction f au point x est $3/5$.

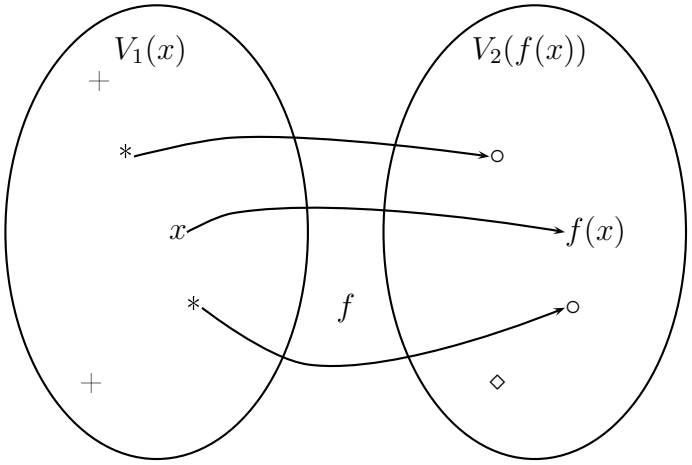


FIGURE 3.1 – Conservation de voisinage.

En s'appuyant sur cette définition, la conservation de la localité de la fonction f avec le paramètre de localité (V_1, V_2) sur l'espace total est formulée par :

$$\zeta_{(V_1,V_2)}(X) = \frac{\sum_{x \in X} \zeta_{(V_1,V_2)}(x)}{\#X} \tag{3.2}$$

3.3 Conservation de la localité combinée

Des localités différentes peuvent être intéressantes parce qu'elles jouent certains rôles différents dans les applications réelles. Ainsi, une mesure de conservation de la

localité qui combine plusieurs conservations de la localité individuelles peut être souhaitée. Cependant, les rôles des localités sont souvent dissemblables. Par exemple, un voisinage avec un rayon petit peut être plus intéressant car l'impact de celui-ci est plus important au centre.

Pour cette raison, nous proposons une mesure de conservation de la localité qui satisfait deux aspects :

— Composée depuis différentes localités,

— Soulignant les rôles différents de ces localités.

La différence entre les rôles des localités est soulignée par la pondération.

Définition 6 (Conservation de localité pondérée sur tout l'espace)**.** *La conservation de la localité de la fonction $f : X \to Y$ avec la pondération de localités p est mesurée par :*

$$\zeta(f, p, \psi) = c(X, Y) \sum_{L_1 \in \mathcal{L}(X)} \sum_{L_2 \in \mathcal{L}(Y)} p(L_1, L_2)(\zeta_f(L_1, L_2))^{\psi} \tag{3.3}$$

$c(X, Y)$ est le facteur général. Par exemple, $c(X, Y) = \frac{1}{\#X}$ est la normalisation si les valeurs moyennes sont souhaitées. Par défaut, cette normalisation est désactivée par la valeur 1.

La puissance de conservation de la localité ψ est le paramètre supplémentaire flexible permettant de simplifier la formule. Par exemple, pour neutraliser la conservation de la localité quelconque : nous pouvons choisir $\psi = 0$; pour réduire l'importance d'une conservation de la localité : nous pouvons choisir $\psi < 0$. Pour les cas simples, les valeurs $0, 1, -1$ sont utilisées. Cependant, une valeur quelconque peut être choisie pour un but particulier.

Le choix de la pondération dans la mesure de la conservation de la localité des fonctions permet d'accentuer les différents aspects de la localité. Ce choix dépend du but de chaque application.

Exemple 2. Examinons les localités qui sont les voisinages V_1, V_2 dans deux espaces métriques quelconques. Ils représentent les ensembles des points qui se placent dans les boules de rayons r_1, r_2.

$$p(V_1, V2) = p(r_1, r_2) = \begin{cases} \frac{1}{r_1} & \text{si } r_2 = r_1^2 \\ 0 & \text{si } r_2 \neq r_1^2 \end{cases}$$

Dans cet exemple, on n'examine que les voisins conservés dans le rayon $r_2 = r_1^2$ et souligne les localités de rayon petit au niveau de la source.

Comme la pondération dispose de deux localités comme variables, elle peut souligner le rôle de chaque localités et la relation entre celles-ci. Pour éclairer la formalisation d'une pondération, nous pouvons la décomposer en trois parties :

$$p(L_1, L_2) = s_1(L_1)s_2(L_2)l(L_1, L_2)$$

où :

— s_1 est l'accentuation du rôle de L_1

— s_2 est l'accentuation du rôle de L_2

— l est l'accentuation de la relation entre L_1 et L_2

En utilisant cette décomposition, les membres de la pondération dans l'exemple 2 sont :

— $s_1 r_1) = \frac{1}{r_1}$

— $s_2(r_2) = 1$

— $l(r_1, r_2) = \begin{cases} 1 & \text{si } r_2 = r_1^2 \\ 0 & \text{si } r_2 \neq r_1^2 \end{cases}$

Ainsi, nous simplifions la formulation de la pondération par les formulations de ses composants.

3.4 Comparer avec d'autres mesures existantes

Dans cette section, nous essayons de reformuler les mesures existantes décrites dans Sous-section 3.1.3 par le paramétrage de notre mesure. Les éléments à définir sont les localités, leurs pondérations et leurs puissances (ψ).

La mesure proposée par Perez, Kamata et Kawaguchi

$$L(C) = \sum_{i,j \in [N^m], i<j} \frac{|i-j|}{\mathfrak{d}(C(i), C(j))}$$

avec C la fonction qui fait correspondre l'index avec son point sur la courbe remplissant l'espace choisie. L'espace est de dimension m et de taille N sur chaque dimension. Ainsi, il y a en total N^m points qui correspondent à N^m index. Dans la direction de réduction de dimension, nous pouvons convenir que l'espace multidimensionnel est la source et les index sont dans la cible. Pour faire correspondre les points de la source à la cible, nous utilisons C^{-1}.

Ces mesures peuvent être analysées par :

— $\mathcal{L}(X) = \{V_1^{r_1}(x) | V_1^{r_1}(x) = \{t : d(t, x) = r_1\}, x \in X, r_1 > 0\}$,

— $\mathcal{L}(Y) = \{V_2^{r_2}(i) | V_2^{r_2}(i) = \{j : j - i = r_2\}, i \in Y, r_2 > 0\}$,

— $p(V_1^{r_1}, V_2^{r_2}) = \frac{r_2}{r_1}$ ou $s_1(V_1^{r_1}) = \frac{1}{r_1}, s_2(V_2^{r_2}) = r_2$,

$l(V_1^{r_1}(x), V_2^{r_2}(i)) = \begin{cases} 1 & \text{si } x = C(i) \\ 0 & \text{si non} \end{cases}$,

— $\psi = 0$

En effet, $\mathcal{L}(X)$ contient les localités définies comme les sphères de rayon r_1 centrées aux xs. Chaque localité dans $\mathcal{L}(Y)$ ne contient qu'un seul point, qui est effectivement un numéro satisfaisant $j-i=r_2$. La conservation de la localité n'a de sens que dans le cas où son poids n'est pas nul. Pour cette raison, nous ne mesurons que la conservation de la localité dans le cas que $l(V_1^{r_1}, V_2^{r_2}) \neq 0$, c'est-à-dire, $x = C(i)$. Pour chaque i et $x = C(i)$, la conservation de la localité entre deux localités $V_1^{r_1}(x)$ et $V_2^{r_2}(i)$ est mesure par $\frac{\#C^{-1}(V_1)\cap V_2}{\#V_1} = \frac{\#C^{-1}(V_1)\cap\{j\}}{\#V_1}$, avec j est le seul index de V_2. Cette mesure n'a qu'une valeur significative ($\neq 0$) quand $C^{-1}(V_1)\cap\{j\}\neq\emptyset$, cela équivaut $C(j)\in V_1$. Cette contrainte est satisfaite quand $r_1 = \mathfrak{d}(x, C(j)) = \mathfrak{d}(C(i), C(j))$.

En remplaçant ces éléments dans la formule de mesure combinée, nous avons la mesure proposée par Perez, Kamata et Kawaguchi :

$$\begin{aligned}\zeta(f,p,\psi) &= \sum_{L_1\in\mathcal{L}(X)}\sum_{L_2\in\mathcal{L}(Y)} p(L_1,L_2)(\zeta_f(L_1,L_2))^{\psi} \\ &= \sum_{j>i}\frac{r_2}{r_1} = \sum_{j>i}\frac{j-i}{\mathfrak{d}(C(i),C(j))} = L(C)\end{aligned}$$

La mesure proposée par Mitchison et Durbin

$$L_q(C) = \sum_{\{i,j\in[N^2]:i<j,d(C(i),C(j))=1\}} |i-j|^q$$

De la même manière, cette mesure peut être décomposée en :
— $V_1^{r_1}(x) = \{t : d(t,x) = r_1\}, V_2^{r_2}(i) = \{j : |i-j| = r_2\}$
— $p(V_1,V_2) = r_2^q$ où $s_1(V_1) = \begin{cases}1 & \text{si } r_1 = 1\\ 0 & \text{si } r_1 \neq 1\end{cases}, s_2(V_2) = r_2^q$,
$l(V_1^{r_1}(x), V_2^{r_2}(i)) = \begin{cases}1 & \text{si } x = C(i)\\ 0 & \text{si non}\end{cases}$,
— $\psi = 0$

La mesure proposée par Bongki Moon Cette mesure peut être formulée par :
— $\mathcal{L}(X)$ est l'ensemble tous les rectangles dans X
— $\mathcal{L}(Y)$ est l'ensemble des clusters
— $p(L_1,L_2) = \begin{cases}1 & \text{si } L_2\in\mathsf{C}(L_1)\\ 0 & \text{si } L_2\notin\mathsf{C}(L_1)\end{cases}$,
ou $s_1(L_1)=1, s_2(L_2)=1$
$l(L_1,L_2) = \begin{cases}1 & \text{si } L_2\in\mathsf{C}(L_1)\\ 0 & \text{si } L_2\notin\mathsf{C}(L_1)\end{cases}$
— $\psi = 0$

Comme $\psi = 0$, ces mesures neutralisent la conservation de la localité mais retiennent seulement leurs rôles et la relation entre elles.

Conclusion

Dans cette section, nous avons proposé une mesure générale de conservation de la localité. C'est une mesure paramétrique, qui permet de combiner différentes mesures de conservation de la localité.

La perspective de cette proposition de mesure est de mieux adapter au besoin des différentes applications, qui agissent sur plusieurs localités avec des niveaux d'impact différents.

Dans ce sens, la mesure proposée couvre les autres mesures de conservation de la localité en choisissant les paramètres bien adaptés. En effet, nous avons décrit les paramètres pour le cas de 3 mesures connues. Ainsi, notre proposition de mesure est aussi une génération de ces 3 mesures. Le Tableau 3.1 résume les paramètres pour ces mesures.

Mesure		**Perez, Kamata et Kawaguchi**	**Mitchison et Durbin**	**Bongki Moon**
Localité source		$V_1^{r_1}(x) = \{t : \mathfrak{d}(t,x) = r_1\}$	$V_1^{r_1}(x) = \{t : d(t,x) = r_1\}$	rectangle
Localité cible		$V_2^{r_2}(i) = \{j : j - i = r_2\}$	$V_2^{r_2}(i) = \{j : \lvert i - j\rvert = r_2\}$	grappe
Poid	s_1	$\frac{1}{r_1}$	$\begin{cases} 1 & \text{si } r_1 = 1 \\ 0 & \text{si } r_1 \neq 1 \end{cases}$	1
	s_2	r_2	r_2^q	1
	l	$\begin{cases} 1 & \text{si } x = C(i) \\ 0 & \text{si non} \end{cases}$	$\begin{cases} 1 & \text{si } x = C(i) \\ 0 & \text{si non} \end{cases}$	$\begin{cases} 1 & \text{si } L_2 \in \mathsf{C}(L_1) \\ 0 & \text{si } L_2 \notin \mathsf{C}(L_1) \end{cases}$

TABLEAU 3.1 – Paramètres décrivant les mesures existantes par la nouvelle mesure générale proposée.

3.5 La localité dans la recherche des plus proches voisins

3.5.1 Recherche des plus proches voisins

La recherche des plus proches voisins est un problème très courant en l'informatique [7, 18, 20, 85, 139, 147, 159]. C'est une recherche des points les plus proches par rapport à une référence dans un espace métrique. Elle est présente dans plusieurs domaines, dont :

— La reconnaissance de formes [15, 86, 104],
— La classification automatique [5, 6, 152] avec sa méthode des k plus proches voisins qui est largement utilisée,
— Partitionnement de données [51, 52, 148]
— La vision par ordinateur [55, 149, 163]
— La recherche d'images par le contenu (CBIR) [35, 59, 143]
— Théorie des codes [91, 153], typiquement, décodage par maximum de vraisemblance [38, 47, 154]
— La compression des données [131, 135, 136]
— Les autres domaines : marketing [27, 40, 115, 134], correcteur orthographique [137], détection du plagiat [161, 164].

La solution la plus simple est la recherche linéaire qui consiste à examiner tous les éléments de la base de données. Cependant, comme il s'agit d'une recherche exhaustive, elle est très coûteuse, en particulier dans le cas de grandes dimensions et de grandes bases de données.

Pour une recherche efficace dans le contexte multidimensionel,certaines méthodes tentent de partitionner l'espace comme l'arbre k-d (*k-d tree*) [17, 58, 166], l'arbre-R (*R-tree*) et ses extensions [14, 67, 140]. Le principe de ces méthodes est d'utiliser des structures de données permettant d'indexer l'espace en conservant la localité. Cela revient à partitionner l'espace en segments et que les données dans chaque segment correspondent aux index proches.

Le partitionnement de données, qui est le coeur de ces méthodes, nous permet aussi de les distinguer. Par exemple, l'arbre k-d propose un partitionnement binaire fixé. L'arbre-R divise l'espace en s'adaptant aux données. Chaque segment est défini par le rectangle contenant ses points. Les différentes façons de partitionnement cause un écart de performance dans l'indexation, qui peut être expliqué par l'écart de la conservation de la localité des méthodes de partitionnement.

3.5.2 Approche pour la recherche des plus proches voisins

Nous définissons la recherche des plus proches voisins comme suit :

Définition 7. *Examinons un espace $\mathcal{P}$ avec la métrique $d_{\mathcal{P}}$, ${\mathcal{S}_{\mathcal{P}}}^k(p_0)$ est l'ensemble de k plus proches voisins dans $\mathcal{P}$ du point p_0 si $\forall p \in \mathcal{P}$:*

$$\begin{cases} d_{\mathcal{P}}(p, p_0) < \max_{p_i \in {\mathcal{S}_{\mathcal{P}}}^k(p_0)} d_{\mathcal{P}}(p_i, p_0) \Rightarrow p \in\in {\mathcal{S}_{\mathcal{P}}}^k(p_0) \\ d_{\mathcal{P}}(p, p_0) > \max_{p_i \in {\mathcal{S}_{\mathcal{P}}}^k(p_0)} d_{\mathcal{P}}(p_i, p_0) \Rightarrow p \notin\in {\mathcal{S}_{\mathcal{P}}}^k(p_0) \end{cases}$$

Nous appelons le rayon de ${\mathcal{S}_{\mathcal{P}}}^k(p_0)$ la valeur

$$\max_{p_i \in {\mathcal{S}_{\mathcal{P}}}^k(p_0)} d_{\mathcal{P}}(p_i, p_0)$$

La définition examine deux cas :

1. Si
$$d_{\mathcal{P}}(p, p_0) < \max_{p_i \in {\mathcal{S}_{\mathcal{P}}}^k(p_0)} d_{\mathcal{P}}(p_i, p_0)$$
on est sûr que p est dans ${\mathcal{S}_{\mathcal{P}}}^k(p_0)$,
2. Si
$$d_{\mathcal{P}}(p, p_0) > \max_{p_i \in {\mathcal{S}_{\mathcal{P}}}^k(p_0)} d_{\mathcal{P}}(p_i, p_0)$$
on est sûr que p n'est pas dans ${\mathcal{S}_{\mathcal{P}}}^k(p_0)$.

Pour les points que $d_{\mathcal{P}}(p, p_0) = \max_{p_i \in {\mathcal{S}_{\mathcal{P}}}^k(p_0)} d_{\mathcal{P}}(p_i, p_0)$, on a au moins un point appartient ${\mathcal{S}_{\mathcal{P}}}^k(p_0)$, les autres points sont incertains.

Recherche des proches voisins La définition ci-dessus est une approche traditionnelle et connue de la recherche des plus proches voisins. C'est une recherche stricte des plus proches voisins. Cependant, dans certain contexte, une approche moins stricte peut être utilisée.

Par exemple, dans le contexte de recherche d'images par le contenu dans les bases d'images très grandes, la question de temps d'exécution est toujours posée. La condition *plus proche* peut être ainsi moins exigeante, une sortie des images similaires à celle d'entrée étant acceptable. De plus, comme ces bases d'images sont très grandes, il y a plusieurs images similaires à celle d'entrée et l'écart de *niveau similaire* n'est pas très grand. Ainsi, une sortie des images *similaires*, qui ne sont pas obligées *les plus similaires*, à celle d'entrée peut être acceptée.

Définition 8. *Dans un espace $\mathcal{P}$ avec métrique $d_{\mathcal{P}}$, une recherche des k' proches voisins parmi les k plus proches voisins dans $\mathcal{P}$ du point p_0 permet d'identifier certains k' points dans ${\mathcal{S}_{\mathcal{P}}}^k(p_0)$.*

Indexation Une recherche linéaire, qui examine successivement des éléments dans la base, est inefficace et parfois impossible, surtout dans les grandes bases de données. Dans un système de recherche d'information en général et dans celui des plus proches voisins en particulier, on fait une étape préparatoire supplémentaire : l'indexation. L'indexation est une étape permettant de structurer l'espace de recherche pour faciliter et accélérer la recherche. En accompagnant l'indexation, une méthode de recherche propre doit être proposée.

Mathématiquement, une *indexation* est simplement une bijection f entre un espace des données $\mathcal{P}$ et un espace des index $\mathfrak{I}$. Après cette indexation, $i = f(p) \in \mathfrak{I}$ est appelé l'index de p.

Un exemple simple et connu de l'indexation peut être retrouvé dans la recherche des nombres dans une liste. L'indexation est le tri de la liste, l'index d'un nombre

est son ordre dans la liste classée. Une recherche par dichotomie peut accompagner cette indexation.

Une autre indexation connue est l'arbre-B. L'arbre-B a plusieurs variations (arbre-B+, arbre-B*) et adaptations dans des contextes particuliers, comme arbre-R pour les bases de données spatiales. Il s'agit d'une structure de données en arbre équilibré. Les données sont stockées dans les noeuds, qui sont représentés par leur clé. Une identification de donnée est un parcours à travers ces noeuds. Ainsi, l'index d'un point est l'ensemble des noeuds parcourus. À noter que dans les noeuds, les clés sont aussi classées et recherchées par dichotomie. Une indexation complexe peut comprendre alors d'autres indexations.

Nous évaluons une indexation via le temps et la complexité de mise en oeuvre des éléments suivants :

— f : l'identification de l'index depuis les données
— f^{-1} : l'identification des données depuis l'index
— la recherche sur $\mathfrak{I}$

Une indexation est bonne si

— elle nous permet de convertir facilement dans les deux sens : des données aux index et l'inverse,
— la recherche dans l'espace des index est rapide.

Pour la recherche des plus proches voisins, en dehors d'identification des données, l'indexation représente en plus le voisinage. Elle permet de localiser un groupe assez petit de voisins où parmi eux, les plus proches voisins sont rapidement retrouvées.

Dans ce cas-ci, l'indexation est en plus évaluée par un autre critère : la capacité de représenter le voisinage. Cette qualité peut être formulée sous forme de la conservation de la localité, qui est présentée au début de ce chapitre.

3.6 Conservation de localité de l'indexation

3.6.1 Recherche des plus proches voisins

Dans l'espace $\mathcal{P}$, chaque indexation $f : \mathcal{P} \to \mathfrak{I}$ correspond à une valeur

$$k_f = \max_{p \in \mathcal{P}} \{\min\{h \in \mathbf{N} : f(\mathcal{S}_{\mathcal{P}}{}^k(p)) \subset \mathcal{S}_{\mathfrak{I}}{}^h(f(p))\}\} \tag{3.4}$$

k_f est une valeur minimale satisfaisant que les k_f proches voisins de l'index $f(p)$ d'un point p quelconque dans l'espace d'index contient tous les index de k plus proches voisins de p.

Pour déterminer $\mathcal{S}_{\mathcal{P}}{}^k(p_0)$, nous pouvons déterminer d'abord $\mathcal{S}_{\mathfrak{I}}{}^{k_f}(f(p_0))$. Ainsi, $\mathcal{S}_{\mathcal{P}}{}^k(p_0) = \mathcal{S}^k_{\mathcal{P}^*}(p_0)$ où $\mathcal{P}^* = \{p \in \mathcal{P} : f(p) \in \mathcal{S}_{\mathfrak{I}}{}^{k_f}(f(p_0))\}$. Visuellement, pour

chercher k plus proches voisins de p_0, nous faisons la recherche sur l'espace des index. Le résultat est un voisinage de l'index de p_0 qui contient tous les index des k plus proches voisins de p_0. Un raffinement est enfin appliqué sur les points correspondants à ces index pour identifier les vrais voisins.

En conséquence, une indexation est valide si le ratio k/k_f est grand parce qu'elle garantit que le nombre des proches voisins (k_f) à examiner pour retrouver les k plus proches voisins n'est pas grand.

Si on considère que la localité des points et des index est leur voisinage, la qualité d'une indexation peut être traduite par la conservation de la localité. Autrement dit, pour chaque point p_0, sa localité L_1 est son voisinage de rayon

$$r_1 = \max_{p_i \in \mathcal{S}_{\mathcal{P}}{}^k(p_0)} d_{\mathcal{P}}(p_i, p_0)$$

et la localité L_2 de l'index i est son voisinage de rayon

$$r_2 = \max_{j \in \mathcal{S}_{\mathfrak{I}}{}^{k_f}(i)} d_{\mathfrak{I}}(i, j)$$

En appliquant la formule 3.1, nous avons la conservation de la localité de la bijection réciproque f^{-1} de f (comme f est bijective) au point p_0 est :

$$\zeta_{f^{-1}}(L_2, L_1) = \frac{\#f^{-1}(L_2) \cap L_1}{\#L_2}$$

Comme $f(L_1) \subset L_2 \Rightarrow L_1 \subset f^{-1}(L_2) \Rightarrow f^{-1}(L_2) \cap L_1 = L_1$, nous avons

$$\zeta_{f^{-1}}(L_2, L_1) = \frac{\#L_1}{\#L_2} = \frac{k}{k_f}$$

Ainsi, mieux f^{-1} conserve la localité, meilleure sera l'indexation.

3.6.2 Recherche des proches voisins

Avec l'indexation f de $\mathcal{P}$, une approche pour la recherche des k proches voisins de p est de sortir tous les points correspondant aux k plus proches voisins dans l'espace d'index $\mathfrak{I}$ de l'index $f(p) \in \mathfrak{I}$. Comme le raffinement pour les plus proches voisins est supprimé, le temps d'exécution est réduit. Cependant, nous pouvons poser les deux questions suivantes :

1. est-ce que ces voisins sont assez proches du point d'entrée ?
2. parmi ces voisins, combien de voisins sont assez proches du point d'entrée ?

Ces questions peuvent être formulée comme suit. Elle relève également du principe de la conservation de la localité.

Question 1 Appelons k' le nombre que les k' plus proches voisins de p contiennent tous ses k proches voisins. Plus k' est petit, plus les voisins sont proches du point d'entrée. Ainsi, une indexation est bonne si le ratio k/k' est grand. Notons que nous avons toujours $k < k'$ et $0 < k/k' \leq 1$.

Si on considère toujours que la localité est le voisinage, la localité L_1 de p est son voisinage avec le rayon étant celui de ${\mathcal{S}_{\mathcal{P}}}^{k'}(p)$ et localité L_2 de l'index $f(p)$ est sont voisinage avec le rayon étant celui de ${\mathcal{S}_{\mathcal{I}}}^{k}(f(p))$. Nous avons $\#L_1 = k', \#L_2 = k$.

En appliquant aussi la formule 3.1, nous avons la conservation de la localité de f au point p est :

$$\zeta_f(L_1, L_2) = \frac{\#f(L_1) \cap L_2}{\#L_1}$$

Comme $L_1 \supset f^{-1}(L_2) \Rightarrow f(L_1) \supset L_2 \Rightarrow f(L_1) \cap L_2 = L_2$, nous avons

$$\zeta_f(L_1, L_2) = \frac{\#L_2}{\#L_1} = \frac{k}{k'}$$

Ainsi, mieux f conserve la localité, à savoir k/k' est grand, meilleure sera l'indexation. Dans le cas idéal, f conserve totalement la localité, c'est-à-dire $k/k' = 1$, les proches voisins sont aussi les *plus* proches voisins. Cependant, à notre connaissance, il n'existe pas d'indexation idéale qui simplifie ces opérations. Un exemple classique de l'indexation idéale est l'identité I, $I(p) = p$.

Question 2 Dans le cas d'une recherche des k plus proches voisins ${\mathcal{S}_{\mathcal{P}}}^{k}(p_0)$ de p, meilleure sera une indexation, plus grand sera le nombre de voisins dans ${\mathcal{S}_{\mathcal{P}}}^{k}(p_0)$ trouvés dans $f^{-1}({\mathcal{S}_{\mathcal{I}}}^{k}(f(p)))$. Notons ce nombre k''. Alors, $k'' = \#{\mathcal{S}_{\mathcal{P}}}^{k}(p_0) \cap f^{-1}({\mathcal{S}_{\mathcal{I}}}^{k}(f(p)))$.

Posons la localité L_1 de p est le voisinage avec le rayon étant celui de ${\mathcal{S}_{\mathcal{P}}}^{k}(p_0) \cap f^{-1}({\mathcal{S}_{\mathcal{I}}}^{k}(f(p)))$ et la localité L_2 de $f(p)$ est le voisinage avec le rayon étant celui de ${\mathcal{S}_{\mathcal{I}}}^{k}(f(p))$. Nous avons $\#L_1 = k'', \#L_2 = k$.

En appliquant la formule 3.1, nous avons la conservation de la localité de la bijection réciproque f^{-1} de f (comme f est bijective) au point p est :

$$\zeta_{f^{-1}}(L_2, L_1) = \frac{\#f^{-1}(L_2) \cap L_1}{\#L_2}$$

Comme $f(L_1) \subset L_2 \Rightarrow L_1 \subset f^{-1}(L_2) \Rightarrow f^{-1}(L_2) \cap L_1 = L_1$, nous avons

$$\zeta_{f^{-1}}(L_2, L_1) = \frac{\#L_1}{\#L_2} = \frac{k''}{k}$$

En conclusion, mieux f^{-1} conserve la localité, meilleure sera l'indexation.

3.6.3 Évaluation d'indexations

Nous avons constaté le lien existant entre la conservation de la localité de l'indexation et l'efficacité de la recherche, quel que soit le type de la recherche : recherche des plus proches voisins (stricte) ou recherche des proches voisins (relâchée). Le tableau ci-dessous résume cette situation.

Conservation de la localité	Recherche des plus proches voisins	Recherche des proches voisins	
		Q1	Q2
f		✓	
f^{-1}	✓		✓

TABLEAU 3.2 – Influence de la conservation de la localité de l'indexation sur les types de recherches de plus proches voisins.

Les mesures de la conservation de la localité de l'indexation et celle de la bijection réciproque peuvent donc être utilisées pour évaluer les indexations. Elles peuvent être séparément ou mutuellement utilisées.

Concernant les applications, les paramètres de la conservation de la localité peuvent nous intéresser. La formule 3.4 est un exemple : dans la recherche de k plus proches voisins, une question directe pour l'indexation est de définir quelle limite de la recherche dans l'espace des index permet de sortir tous les index de k plus proches voisins.

3.7 Conclusion

Dans ce chapitre, nous avons abordé les éléments concernant la conservation de la localité. Dans la plupart des applications, les entités sont examinées dans la relation avec leur entourage, qui sont représentés par la notion *localité*. Il parait assez naturel que le traitement de chaque entité concerne sa localité. Ainsi, une transformation qui est utilisée pour faciliter le traitement a une meilleure performance si elle conserve la localité.

Dans une application donnée, il existe souvent plusieurs transformations permettant de simplifier ses opérations. Ces transformations ont des différences conservation de la localité. Une optimisation peut être utilisée pour retrouver la meilleure transformation. Pour parvenir à cet objectif, nous avons défini une mesure de la conservation de la localité. Il s'agit d'une mesure générique qui peut être appliquée dans des situations différentes par paramétrage. En s'appuyant sur cette mesure, nous pouvons sélectionner la bonne transformation pour une application donnée. La mesure est justifiée dans ce chapitre mais également par les résultats de l'application dans la Chapitre 6.

La conservation de la localité est ensuite analysée dans le contexte de l'indexation, surtout pour la recherche des proches voisins, une recherche populaire pour les grandes bases d'images. Ces travaux illustrent la relation existant entre la performance de l'indexation et sa conservation de la localité.

Chapitre 4

Courbe remplissant l'espace et l'autosimilitude

Résumé:

Dans ce chapitre, nous proposons une nouvelle définition d'une courbe remplissant l'espace, qui permet une implémentation informatique au-delà de quelques dimensions ($d>4$*). S'appuyant sur l'autosimilitude, cette définition apporte 2 avantages principaux :*

— *une définition explicite, concise et ouvrant à une implémentation informatique,*

— *un bon niveau de conservation de la localité.*

Nous montrons qu'une courbe remplissant l'espace peut être définie à partir d'un motif primitif et d'une liste de transformations (similitudes). De plus, nous démontrons qu'il est possible de décomposer une similitude en deux opérations simples : réflexion et permutation. Cette décomposition permet de mettre en œuvre des courbes remplissant l'espace par des algorithmes légères, sans demande des ordinateurs puissants.

Une vue générale de la conservation de la localité des courbes est présentée via des tests comparatifs utilisant la mesure proposée dans le Chapitre 3. Nous observons qu'il existe une partition en termes de localité entre les courbes vérifiant l'autosimilitude au sens de Définition 12 et les autres. La courbe de Hilbert est celle qui maximise la conservation de la localité ce qui conforme aux résultats déjà observés dans la littérature.

Sur la base de ces résultats, nous proposons dans la Chapitre 5 une généralisation multidimensionnelle de la courbe de Hilbert en souhaitant construire d'autres courbes multidimensionnelles conservant aussi bien la localité.

4.1 Contexte et objectifs

Malgré le fait que les courbes remplissant l'espace sont largement utilisées dans les applications informatiques, elles s'appuient essentiellement sur des propositions particulières. Il n'existe aucune définition précise de la courbe remplissant l'espace générale. La manque de définition empêche la généralisation multidimensionnelle et l'exploitation des courbes remplissant l'espace pour retrouver celles qui conservent le mieux possible la localité, propriété souvent cherchée dans les applications.

Face à ce défit, nous proposons une définition de la courbe remplissant l'espace en nous appuyant sur la notion l'autosimilitude. L'autosimilitude est déterminée comme la propriété commune des courbes remplissant l'espace.

En s'appuyant sur la définition proposée, nous déterminons ensuite des éléments permettant de préciser ce qu'est une courbe remplissant l'espace : le motif et les transformations des sous-courbes. Quoique la transformation de chaque sous-courbe est une similitude, elle reste encore vague et sa représentation n'est pas encore homogène. Nous proposons de simplifier chaque transformation en deux opérations simples : une réflexion et une permutation des coordonnées. Ces 2 opérations sont faciles à implémenter d'un point de vue informatique.

Enfin, la conservation des courbes remplissant l'espace est confirmée via des comparaisons utilisant notre mesure de la conservation de localité décrite dans le Chapitre 3. Plus précisément, ces mesures sont adaptées au cas de la recherche des images que nous détaillerons dans le Chapitre 6. La comparaison montre également que la courbe de Hilbert est celle qui conserve mieux la localité. La courbe de Hilbert va être développée dans le chapitre suivant [*cf.* Chapitre 5].

4.2 Une courbe remplissant l'espace : un ordonnancement autosimilaire

4.2.1 Ordonnancement de l'espace

Une courbe remplissant un espace $\mathbf{d}$-dimensionnel est un chemin qui passe par tous les points de cet espace. L'ordre de passage des points le long de la courbe considérée réalise un ordonnancement de l'espace.

Formellement, un ordonnancement de l'espace [1] est une application $\mathfrak{F}$ qui associe à chaque point de l'espace $\{1, 2, \cdots, n\}^{\mathbf{d}}$ un index (valeur mono-dimensionnelle) dans $\{1, 2, \cdots, n^{\mathbf{d}}\}$ comme suit :

Définition 9 (Ordonnancement de l'espace)**.** *Un ordonnancement de l'espace est une*

1. Dans la thèse, seul les espaces discrétisés seront considérés (discrétisation d'un hypercube).

bijection :

$$\mathfrak{F} : \{1, 2, \cdots, n\}^{\mathbf{d}} \rightarrow \{1, 2, \cdots, n^{\mathbf{d}}\} \tag{4.1}$$

La Figure 4.1 illustre des exemples d'espaces discrétisés $\{1, 2, \cdots, n\}^{\mathbf{d}}$ de dimension $\mathbf{d}$=2 et 3 à la résolution $n = 4$.

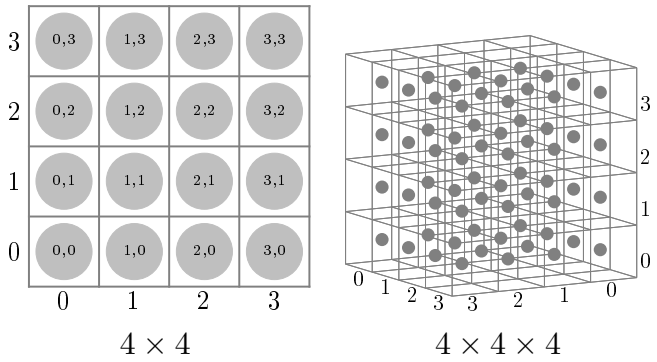


FIGURE 4.1 – Cas d'espaces discrétisés en dimension $\mathbf{d}$=2 et 3 : un espace correspond à grille de points représentés par des disques contenant leurs coordonnées (2-D) ou des boules (3-D).

L'application $\mathfrak{F}$ n'est pas unique. Il existe en effet plusieurs manières d'ordonner un espace. D'ailleurs, le nombre de façon d'ordonner augmente suivant la résolution et la dimension de l'espace suivant $n^{\mathbf{d}}!$.

La Figure 4.2 illustre 3 façons d'ordonner les points d'un espace.

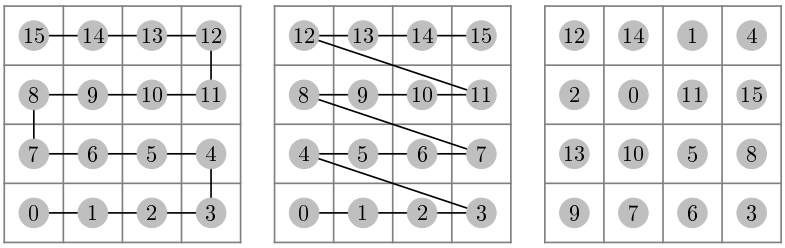


FIGURE 4.2 – Plusieurs façons d'ordonner un espace $\mathfrak{F}$. Cas de deux ordonnancements déterministes (Scan et Sweep) et un ordonnancement aléatoire. Les courbes Scan et Sweep très connues ordonnent tous les points de la même ligne de manière consécutive ; les points consécutifs ont des index consécutifs. Seul diffère le point correspondant au début de chaque ligne. Dans l'ordonnancement *(Aléatoire)* la distribution des index aux points de l'espace suit une loi uniforme.

4.2.2 $\mathfrak{F}$: modèle et implémentation informatique

Ordonner un espace consiste à déterminer l'application $\mathfrak{F}$ [*cf.* Sous-section 4.2.1]. Dans la littérature, plusieurs modèles existent (liste, algorithme, formule etc.) mais tous ne permettent pas une implémentation informatique efficace. Nous en donnons un bref aperçu.

- $\mathfrak{F}$ peut être définie par une liste de points ordonnés selon leurs index. C'est la façon la plus simple de description. Néanmoins, ce modèle requiert des ressources machine importantes : consommation mémoire pour la sauvegarde de la liste de points (le nombre de points d'un espace de dimension $\mathbf{d}$ et de résolution n est $n^{\mathbf{d}}$), et temps de calcul pour déterminer l'index d'un point donné.
- $\mathfrak{F}$ peut être définie par un algorithme, par exemple, dans le cas de la courbe de Lebesgue (illustrée dans Figure 2.4). Le calcul de l'index d'un point donné s'effectue par l'interfoliage des coordonnées de ce point [*cf.* Sous-section 2.1.4].
- $\mathfrak{F}$ peut être définie par une formule. Par exemple, dans le cas de la courbe Sweep (illustrée dans la Figure 4.2), le calcul de l'index $\mathbf{i}$ d'un point de coordonnées (x, y) s'effectue selon : $\mathbf{i} = \mathfrak{F}(x, y) = ny + x$

Nous constatons que ce dernier modèle est compact et que le calcul de l'index d'un point est rapide. Cependant, il n'est pas toujours possible d'extraire un modèle efficace pour $\mathfrak{F}$ au sens où :

— ce modèle permet une implémentation informatique dans des perspectives multidimensionnelles (limite des machines d'états au-delà de $\mathbf{d}$>4 [*cf.* Section 2.4]),

— ce modèle doit permettre un calcul de l'index dans un temps raisonnable (adapté à une application comme celle proposée dans le Chapitre 6).

N'oublions pas qu'outre la représentation pour $\mathfrak{F}$, le parcours de l'espace qu'elle traduit doit aussi présenter un niveau acceptable de conservation de la localité. Or, nous observons dans la littérature que la localité d'un parcours (traduit par $\mathfrak{F}$) et la difficulté d'établir un modèle de représentation efficace sont liés. Par exemple, la courbe de Hilbert présente un niveau de localité supérieur (y compris avec la mesure proposée dans le Chapitre 3) à celle de Lebesgue [*cf.* Tableau 4.1], mais son implémentation reste plus complexe.

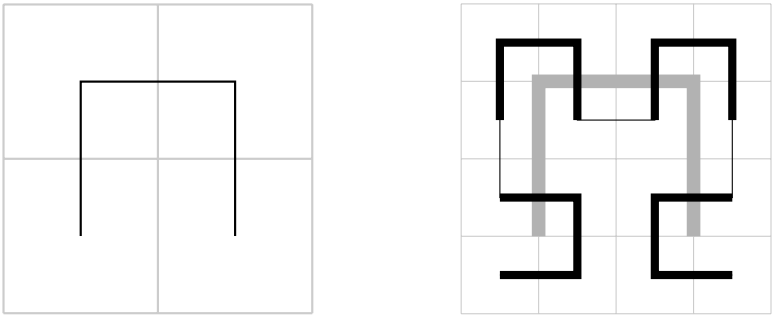
Nous pensons que l'autosimilitude est un facteur clé pour construire des parcours avec un bon niveau de localité tout en permettant une représentation de la courbe ($\mathfrak{F}$) autorisant un calcul de l'index (d'un point donné) en un temps raisonnable, c'est-à-dire compatible avec une application telle que celle présentée dans le Chapitre 6.

Dans la suite, nous commençons par décrire le concept général de l'autosimilitude. Ensuite, nous revisitons l'autosimilitude à la lumière de l'héritage. L'héritage [*cf.* Définition 10] est un moyen de construire une courbe dont le niveau de localité est conservé à travers les ordres $\mathbf{n}$.

4.2.3 La notion générale de l'autosimilitude

Intuitivement, une courbe remplissant l'espace est dite autosimilaire dès lors que ses parties ont également la forme de la courbe elle-même. La Figure 4.3 illustre l'autosimilitude dans la courbe de Hilbert. En observant la courbe d'ordre $\mathbf{n}$=2, nous nous apercevons qu'elle est constituée de copies de la courbe d'ordre antérieur ($\mathbf{n}$=1). Nous appelons *sous-courbe* chacune de ces copies (exacte ou transformée).

De plus, la disposition de ces sous-courbes est imposée par la courbe d'ordre $\mathbf{n}$=1. Ainsi, la courbe d'ordre $\mathbf{n}$=1, que nous appelons le *motif primitif*, est une description grossière de la forme d'une courbe remplissant l'espace. Cette forme est aussi répétée dans les sous-courbes.



(ordre $\mathbf{n}$*=1, motif primitif)* *(ordre* $\mathbf{n}$*=2, les sous-courbes sont épaises)*

FIGURE 4.3 – Illustration de l'autosimilitude dans la courbe de Hilbert. Dans la courbe d'ordre $\mathbf{n}$=2, les deux sous-courbes en haut sont les copies de la courbe d'ordre $\mathbf{n}$=1, les sous-courbes en bas sont les rotations de $90°$ (à gauche) et $-90°$ (à droite) de celle-ci. L'ordre des sous-courbes de la courbe d'ordre $\mathbf{n}$=2 suit l'ordre des points de la courbe d'ordre $\mathbf{n}$=1.

4.2.4 Héritage

L'héritage guide la manière avec laquelle les points d'une courbe d'ordre $\mathbf{n}$ sont remplacés pour créer une courbe d'ordre supérieur $\mathbf{n}$+1. L'héritage doit garantir que :

— les points de la courbe d'ordre supérieur (qui remplacent un même point de la courbe d'ordre inférieur) soient *consécutivement* ordonnés,

— *l'ordre de parcours* de la courbe d'ordre inférieur soit conservé par la courbe d'ordre supérieur.

L'héritage est donc défini par :

Définition 10 (Héritage)**.** *Considérons 2 ordonnancements de l'espace :*

$$\begin{aligned} \mathcal{C} &: \{0,1,\dots,n-1\}^{\mathbf{d}} \rightarrow \{0,1,\dots,n^{\mathbf{d}}-1\} \\ \mathcal{C}' &: \{0,1,\dots,\mathbf{b}n-1\}^{\mathbf{d}} \rightarrow \{0,1,\dots,(\mathbf{b}n)^{\mathbf{d}}-1\} \end{aligned}$$

Nous disons "$\mathcal{C}'$ hérite de $\mathcal{C}$ de rapport $\mathbf{b}$[2]*" si*

$$\left\lfloor \frac{\mathcal{C}'(x)}{\mathbf{b}^{\mathbf{d}}} \right\rfloor = \mathcal{C}\left(\left\lfloor \frac{x}{\mathbf{b}} \right\rfloor\right)$$

Notons que $\lfloor a \rfloor$ est la partie entière de a et, pour le cas d'un vecteur, $\lfloor x \rfloor = (\lfloor x_0 \rfloor, \lfloor x_1 \rfloor, \dots, \lfloor x_{\mathbf{d}-1} \rfloor)$. Ainsi, $\lfloor x/\mathbf{b} \rfloor = (\lfloor x_0/\mathbf{b} \rfloor, \lfloor x_1/\mathbf{b} \rfloor, \dots, \lfloor x_{\mathbf{d}-1}/\mathbf{b} \rfloor)$.

Nous faisons correspondre chaque point $k = (k_0, k_1, \dots, k_{\mathbf{d}-1})$ sur l'ordonnancement $\mathcal{C}$ avec un ensemble $\mathcal{S}_k$ des points sur l'ordonnancement $\mathcal{C}'$ suivant :

$$\mathcal{S}_k = \{x = (x_0, x_1, \dots, x_{\mathbf{d}-1}) : k_i\mathbf{b} \leq x_i < (k_i+1)\mathbf{b}\}$$

Nous appelons $\mathcal{S}_k$ le *sous-espace* correspondant à k.

Ainsi, $k_i \leq x_i/\mathbf{b} < k_i + 1 \;\; \forall x \in \mathcal{S}_k, i \in \{1, 2, \dots, \mathbf{d}\}$, alors

$$\left\lfloor \frac{x}{\mathbf{b}} \right\rfloor = k \;\; \forall x \in \mathcal{S}_k$$

Par conséquent, la valeur de $\mathcal{C}(\lfloor x/\mathbf{b} \rfloor)$ est fixée à $\mathbf{i} = \mathcal{C}(k)$ (l'index de k sur $\mathcal{C}$) pour tous les points x dans $\mathcal{S}_k$. Selon la Définition 10, nous avons $\lfloor \mathcal{C}'(x)/\mathbf{b}^{\mathbf{d}} \rfloor = \mathcal{C}(\lfloor x/\mathbf{b} \rfloor) = \mathbf{i}$. Alors, $\mathbf{i}\mathbf{b}^{\mathbf{d}} \leq \mathcal{C}'(x) < (\mathbf{i}+1)\mathbf{b}^{\mathbf{d}} \;\; \forall x \in \mathcal{S}_k$.

Autrement dit, les points dans le sous-espace $\mathcal{S}_k$ correspondent aux index dans le segment $[\mathbf{i}\mathbf{b}^{\mathbf{d}}, (\mathbf{i}+1)\mathbf{b}^{\mathbf{d}}]$. Notons que le nombre des points et celui des index sont égaux. Ils sont $\mathbf{b}^{\mathbf{d}}$. De plus, $\mathcal{C}'$ est bijective. Chaque point correspond à un index.

En résumé, *les points dans chaque $\mathcal{S}_k$ correspondent aux index consécutifs. Autrement dit, ils sont consécutivement ordonnés.*

À noter que, *les sous-espaces $\mathcal{S}_k$ ($k \in \{0,1,\cdots,\mathbf{b}-1\}^{\mathbf{d}}$) sont ordonnés selon l'index $\mathcal{C}(k)$ de leurs correspondances k.* Concrètement, si $x \in \mathcal{S}_k, x' \in \mathcal{S}_h$ et $\mathcal{C}(k) < \mathcal{C}(h)$, nous avons toujours : $\mathcal{C}'(x) < \mathcal{C}'(x')$.

En conclusion, nous dirons qu'un ordonnancement $\mathcal{C}$' *hérite* au sens de la Définition 10 de l'ordonnancement $\mathcal{C}$ si $\mathcal{C}$' satisfait les deux 2 conditions $H1$ et $H2$ ci-dessous.

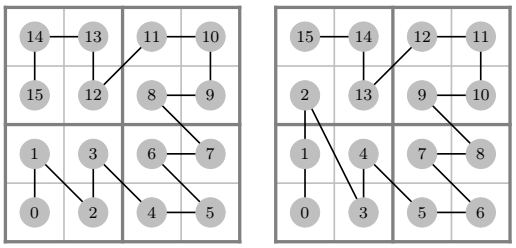
$H1$. *$\mathcal{C}'$ ordonne consécutivement les points appartenant à chaque sous-espace $\mathcal{S}_k$,*

$H2$. *$\mathcal{C}'$ ordonne les sous-espaces $\mathcal{S}_k$ dans le même ordre que leurs points correspondants k de $\mathcal{C}$.*

2. Dans ce chapitre, les illustrations sont de rapport $\mathbf{b}$=2.

Pour mieux cerner le rôle de chaque condition ($H1$, $H2$) dans la définition de l'*héritage* [*cf.* Définition 10], diverses configurations sont testées et analysées allant du respect partiel jusqu'au respect total des deux conditions.

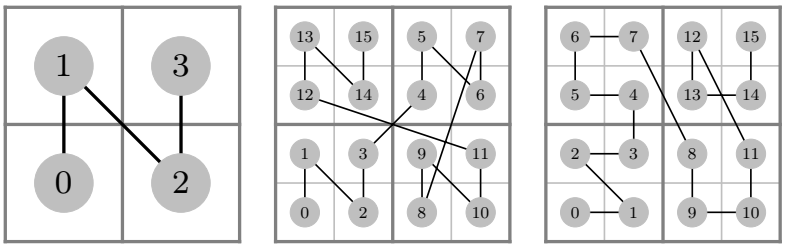
La Figure 4.4 illustre (en dimension $\mathbf{d}$=2 à l'ordre $\mathbf{n}$=2) le respect et la violation de $H1$.



($H1$) *($\cancel{H1}$)*

FIGURE 4.4 – Influence de l'*héritage* [*cf.* Définition 10] dans la construction d'un ordonnancement de l'espace. Cas de sous-courbes ($\mathbf{d}$=2) synthétisant le respect de la condition $H1$ ou non. Dans la figure de droite, le non-respect de $H1$ est symbolisé par les liaisons entre des points 1-2 et 2-3. Une sous-courbe doit remplir un sous-espace avant d'en visiter un autre.

La Figure 4.5 illustre le respect conjoint de $H1$ et $H2$.



(ordre $\mathbf{n}$*=1)* *($H1$ & $\cancel{H2}$)* *($H1$ & $H2$)*

FIGURE 4.5 – Influence de l'*héritage* [*cf.* Définition 10] dans la construction d'un ordonnancement de l'espace : rôle de $H1$ et $H2$. Cas de courbes synthétisant le respect total (droite) ou partiel (milieu) de $H1$ et $H2$. Dans la courbe de droite, chaque sous-courbe remplit un sous-espace et l'ordre de parcours des sous-espaces est imposé par le motif primitif ($\mathbf{n}$=1).

La condition $H1$ contribue à définir un bon niveau de localité (à l'intérieur de chacun des sous-espaces $\mathcal{S}_k$). Pour que ce niveau de localité soit conservé lors de la connexion des sous-espaces la condition $H2$ est nécessaire mais pas suffisante. Il faut en plus que l'ordonnancement à l'ordre antérieur conserve également la localité. C'est la raison pour laquelle nous introduisons dans la partie suivante une nouvelle définition de l'autosimilitude.

4.2.5 La courbe remplissant l'espace : proposition de définition

Après avoir décrit le concept général de l'autosimilitude [*cf.* Sous-section 4.2.3], nous proposons une nouvelle formulation de l'autosimilitude qui, au contraire des définitions implicites consultées dans la littérature, permet d'envisager une implémentation informatique. Cette formulation s'appuiera sur l'héritage [*cf.* Définition 10].

Le motif primitif $\mathfrak{m}$ Comme nous avons introduit dans Sous-section 4.2.3, le motif primitif est une description de la forme d'un ordonnancement autosimilaire. Il est le modèle pour les sous-courbes à toutes les échelles de la courbe : chaque sous-courbe est une copie ou une similitude du motif.

Un motif peut être un ordonnancement de l'espace quelconque. Dans une courbe remplissant l'espace, la résolution d'un motif est définie comme la *base* $\mathfrak{b}$ de la courbe.

Une similitude Une similitude est une transformation qui multiplie toutes les distances par un rapport t défini. Ainsi, elle transforme un motif en une sous-courbe ayant la même forme. Les exemples typiques de la similitude sont la translation, la rotation, l'homothétie et la réflexion.

Définition 11 (Similitude)**.** *Examinons un espace X muni d'une métrique $\mathfrak{d}$. Une bijection $f : X \to X$ est dite une similitude de rapport r si*

$$\mathfrak{d}(f(a), f(b)) = t\mathfrak{d}(a, b) \quad \forall a, b \in X$$

Examinons un objet $A \subset X$, $f(A)$ est dit une similitude de A, A et $f(A)$ sont dits similaires.

Définition 12 (Autosimilitude)**.** *Un ordonnancement de l'espace $\mathcal{C}$ est dit autosimilaire de motif $\mathfrak{m}$ ayant la résolution $\mathfrak{b}$ si :*

- *S*1. *$\mathcal{C}$ est une similitude (au sens de Définition 11) de $\mathfrak{m}$,*
- *S*2. *ou :*
  - *A*1. *$\mathcal{C}$ hérite de rapport $\mathfrak{b}$ (au sens de Définition 10) d'un ordonnancement de l'espace qui est lui-même autosimilaire de $\mathfrak{m}$, et*
  - *A*2. *à l'intérieur de chaque sous-espaces, l'ordonnancement est une similitude de $\mathfrak{m}$.*

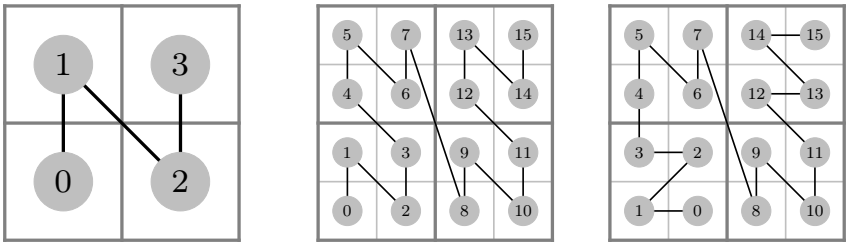
$\mathfrak{b}$ est dite la base de $\mathcal{C}$.

L'autosimilitude et la courbe remplissant l'espace

Définition 13 (Courbe remplissant l'espace)**.** *Nous disons qu'une courbe remplissant l'espace est un ordonnancement de l'espace autosimilaire au sens de Définition 12.*

Une courbe remplissant l'espace qui satisfait $S1$ est dite la courbe d'*ordre* 1. Normalement, la courbe d'ordre 1 est définie comme le motif $\mathfrak{m}$ de la courbe remplissant l'espace. Une courbe qui satisfait $S2$ est dite d'ordre $\mathbf{n}+1$ si elle hérite d'une courbe d'ordre $\mathbf{n}$.

Nous constatons que le respect de la Définition 12 donne naissance à une classe de courbes dont figurent certaines courbes remplissant l'espace connues comme celle de Lebesgue (Figure 4.6). À titre d'exemple, la Figure 4.6 illustre le cas de deux courbes différentes satisfaisant la Définition 12. Dans la courbe de Lebesgue (au centre), les sous-courbes sont des copies exactes du motif primitif (à gauche). Une application des transformations sur des sous-courbes crée des courbes alternatives comme la courbe de droite.



(Lebesgue, ordre $\mathbf{n}$*=1)* *(Lebesgue, ordre* $\mathbf{n}$*=2)* *(Autre, ordre* $\mathbf{n}$*=2)*

FIGURE 4.6 – Application de la Définition 12, exemple de courbes autosimilaires. La courbe de Lebesgue d'ordre $\mathbf{n}$=1 vérifie $S1$. Les deux courbes à droite sont autosimilaires par la vérification de $S2$. Autrement dit, ces courbes à l'ordre $\mathbf{n}$=2, héritent de la courbe de Lebesgue d'ordre $\mathbf{n}$=1 ($S2A1$). Par contre, elles vérifient $S2A2$ différemment.

4.3 Optimisation de la définition d'une courbe remplissant l'espace : simplification des similitudes

4.3.1 Nouvelle définition d'une courbe remplissant l'espace

L'autosimilitude au sens de la Définition 12 apporte l'avantage de la concision dans la définition d'une courbe remplissant l'espace [*cf.* Sous-section 4.2.2]. Ainsi, $\mathfrak{F}$ peut être définie sans déterminer une liste de points ordonnés selon leurs index (très coûteux en termes de ressources machine).

Puisque les sous-courbes sont des similitudes du motif primitif $\mathfrak{m}$, nous ne devons pas énumérer les points de chaque sous-courbes, mais simplement préciser leurs similitudes correspondantes.

Concrètement, une courbe remplissant l'espace peut être bien définie par :

1. un motif primitif $\mathfrak{m}$,
2. une liste de similitudes, chaque similitude transforme le motif en une sous-courbe correspondante.

La description du motif primitif est simplement une énumération des points qui le compose.

À chaque sous-courbe est associée une similitude du motif primitif $\mathfrak{m}$. Comme il existe plusieurs types de similitudes (rotation, réflexion, translation, etc.), dans la section suivante nous montrons qu'il est possible de décomposer ces similitudes en deux simples transformations : une réflexion et une permutation des coordonnées qui sont faciles à implémenter.

4.3.2 Isométrie

Comme la taille des sous-espaces et la taille de l'espace remplit par le motif de chaque courbe remplissant l'espace sont égales, le rapport des similitudes est constamment égale à 1. Dans ce cas, la similitude est une isométrie.

En appliquant au contexte de la thèse, nous définissons l'isométrie comme suit :

Définition 14 (Isométrie)**.** *Examinons un espace* $X = \{0, 1, \dots, k-1\}^{\mathbf{d}}$ *muni une métrique* $\mathfrak{d}$*. Une bijection* $f : X \to X$ *est dite une isométrie si*

$$\mathfrak{d}(f(x), f(y)) = \mathfrak{d}(x, y) \tag{4.2}$$

Des recherches en géométrie euclidienne montrent qu'une isométrie quelconque peut être décomposée en rotation, réflexion et déplacement [42]. Cette décomposition s'appelle la forme canonique de l'isométrie. Si nous considérons le centre de l'isométrie (point fixe) comme le centre de l'espace rempli par le motif, la translation est nulle. L'isométrie est donc décomposée en rotation et réflexion. Autrement dit, une isométrie $\mathfrak{Iso}$ peut s'écrire sous forme $\mathfrak{Iso} = \mathfrak{Ref} \circ \mathfrak{Rot}$.

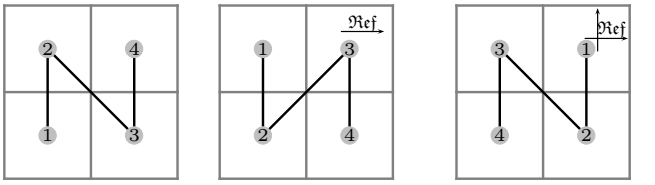
Dans le cas de la transformation des sous-courbes, nous montrons ci-dessous qu'il est possible de décomposer une telle isométrie $\mathfrak{Iso}$ en deux transformations simples, faciles à programmer, que sont la *réflexion* et la *permutation des coordonnées*.

Réflexion

La réflexion définie ici n'est pas identique à celle définie traditionnellement. Cependant, elles sont en principe similaires.

Une réflexion est simple par rapport axe u si elle transforme un point p_1 au point p_2 satisfaisant :

1. p_2 se situe sur la ligne orthogonale avec u et traversant par p_1.
2. $\mathfrak{d}(p_1, O) = \mathfrak{d}(p_2, O)$ où O est le centre du l'espace rempli par le motif.



(a) Origine $\mathfrak{Z}$ *(b)* $\mathfrak{Ref}^2_{\{x\}}(\mathfrak{Z})$ *(c)* $\mathfrak{Ref}^2_{\{x,y\}}(\mathfrak{Z})$

FIGURE 4.7 – Application de la Définition 15 : exemples de réflexions ($\mathbf{d}$=2, $\mathbf{b}$=2) simple ($A = \{0\}$) et complexe ($A = \{0, 1\}$).

Une réflexion complexe transforme successivement un point via plusieurs réflexions simples par rapport à des axes différents. L'ordre de ces réflexions étant indifférent, il en résulte la même transformée. La figure 4.7 (c) illustre le résultat obtenu lors de l'application d'une réflexion complexe du motif $\mathfrak{m}$ [*cf.* Figure 4.7 *(a)*] selon les axes x et y.

La réflexion (simple ou complexe) est entièrement définie comme suit :

Définition 15. *$p' = (p'_0, p'_1, \dots, p'_{\mathbf{d}-1})$ est le reflet du $p = (p_0, p_1, \dots, p_{\mathbf{d}-1})$ dans un motif de base $\mathbf{b}$ aux axes dans A, noté $p' = \mathfrak{Ref}^{\mathbf{b}}_A(p)$, si*

$$\begin{cases} p'_i = \mathbf{b} \text{ - } 1 \text{ - } p_i & si \quad i \in A \\ p'_i = \quad\quad p_i & si \quad i \notin A \end{cases}$$

En utilisant une métrique usuelle comme celle euclidienne ou celle de Manhattan, la réflexion est isométrique. Par exemple, considérons 2 points a et b, si la métrique euclidienne est utilisée, nous avons toujours $|a'_i - b'_i| = |a_i - b_i|$, alors :

$$\mathfrak{d}(a', b') = \sqrt{\sum_{i=0}^{\mathbf{d}-1} (a'_i - b'i)^2} = \sqrt{\sum_{i=0}^{\mathbf{d}-1} (a_i - bi)^2} = \mathfrak{d}(a, b)$$

De plus, nous retirons facilement le corollaire suivant :

Corollaire 1. *Examinons des réflexions et un point p quelconque dans un même espace de base $\mathbf{b}$, nous avons : $\mathfrak{Ref}^b_A(\mathfrak{Ref}^b_A(p)) = p$*

Permutation des coordonnées

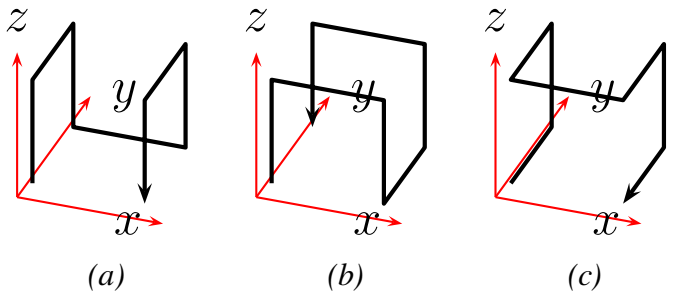
Une permutation des coordonnées est simplement un ré-ordonnancement des coordonnées des points transformés. Nous l'utilisons pour remplacer la rotation dans la décomposition canonique.

Définition 16. *Considérons une bijection* $f : \{0, 1, \dots, \mathbf{d}-1\} \to \{0, 1, \dots, \mathbf{d}-1\}$. *La fonction*

$$\begin{array}{rccc} \mathfrak{p}^f : & \{0,1,\dots,\mathfrak{r}-1\}^{\mathbf{d}} & \to & \{0,1,\dots,\mathfrak{r}-1\}^{\mathbf{d}} \\ & p = (p_0, p_1, \cdots, p_{\mathbf{d}-1}) & \mapsto & p' = (p_{f(0)}, p_{f(1)}, \cdots, p_{f(\mathbf{d}-1)}) \end{array}$$

est dite une permutation des coordonnées par rapport à f.

La Figure 4.8 illustre 2 permutations de coordonnées différentes d'un même motif de base $\mathbf{b}$=2 en $\mathbf{d}$=3. La figure *(b)* est le résultat d'un interchangement des coordonnées x et y de la figure *(a)*. La figure *(c)* est le résultat d'un interchangement des coordonnées y et z de la figure *(a)*.



(a) *(b)* *(c)*

FIGURE 4.8 – Application de la Définition 16 : exemples de permutations de coordonnées ($\mathbf{d}$=3, $\mathbf{b}$=2) du motif *(a)*. Pour *(b)* : $f(0) = 1, f(1) = 0, f(2) = 2$, pour *(c)* : $f(0) = 0, f(1) = 2, f(2) = 1$.

Comme la permutation de coordonnées est simplement un changement de l'ordre des coordonnées et que ce changement est identique pour tous les points, la distance entre les deux points quelconques reste inchangée après la transformation (avec une métrique usuelle (euclidienne, Manhattan)).

La réflexion et la permutation des coordonnées sont isométriques pour les métriques usuelles (euclidienne, Manhattan). Cependant, d'autres métriques peuvent être utilisées dans les applications, comme le montre Section 4.3.2.

Métrique

Dans cette partie, nous proposons une classe de métriques avec lesquelles les deux transformations ci-dessus sont isométriques. La métrique proposée est défini dans Équation 4.3.

$$\mathfrak{d}(x,y) = f\left(\sum_{i=0}^{\mathbf{d}-1} g(x_i - y_i)\right) \tag{4.3}$$

avec $f, g : \mathbb{R} \to \mathbb{R}$ sont des normes, c'est-à-dire, par exemple pour g, $\forall x, y \in R$:

1. $g(ax) = |a|g(x)$
2. $g(x+y) \leq g(x) + g(y)$
3. $g(x) = 0 \Leftrightarrow x = 0$

Depuis ces propriétés, nous pouvons retirer en plus deux autres :

4. $g(x) = g(-x)$ car $g(-x) = |-1|g(x)$. Nous avons aussi $g(x) = g(|x|)$.
5. $g(x) \geq 0$ parce que $2g(x) = g(x) + g(-x) \geq g(x-x) = 0$

En plus, si $x > y \geq 0$, nous avons

$$x = ay, a > 1 \Rightarrow g(x) = |a|g(y) > g(y)$$

Jusqu'à ici, $\mathfrak{d}(x,y)$ est simplement une fonction, pour être une métrique, elle doit satisfaire les conditions suivantes :

1. $\mathfrak{d}(x,y) = 0 \Leftrightarrow x = y$
2. $\mathfrak{d}(x,y) = \mathfrak{d}(y,x)$
3. $\mathfrak{d}(x,z) \leq \mathfrak{d}(x,y) + \mathfrak{d}(y,z)$

Théorème 1. *La fonction $\mathfrak{d}$ définie dans Équation 4.3 est une métrique*

Démonstration. Pour la condition 1, si $x = y$, nous avons, $x_i = y_i \ \ \forall i \in \{0, 1, \dots, \mathbf{d}-1\}$. Comme g est une norme,

$$g(x_i - y_i) = 0 \ \ \forall i \in \{0, 1, \dots, \mathbf{d}-1\}$$

f est aussi une norme, par conséquent, $d(x,y) = 0$.
Inversement,

$$d(x,y) = 0 \Rightarrow \sum_{i=0}^{\mathbf{d}-1} g(x_i - y_i) = 0$$

Comme $g(u) >= 0 \ \ \forall u \in R$, nous avons

$$g(x_i - y_i) = 0 \ \ \forall i \in \{0, 1, \dots, \mathbf{d}-1\}$$

En conséquence, $x_i - y_i = 0 \ \ \forall i \in \{0, 1, \dots, \mathbf{d}-1\}$. Ainsi, $x = y$.
En regardant la condition 2, comme g est une norme,

$$g(x_i - y_i) = g(y_i - x_i)$$

En conséquence, $\mathfrak{d}(x,y) = \mathfrak{d}(y,x)$.
Enfin, pour la condition 3,

$$\mathfrak{d}(x,z) = f\left(\sum_{i=0}^{\mathbf{d}-1} g(x_i - z_i)\right) = f\left(\sum_{i=0}^{\mathbf{d}-1} g(x_i - y_i + y_i - z_i)\right)$$

Comme g est une norme,

$$g(x_i - y_i + y_i - z_i) \leq g(x_i - y_i) + g(y_i - z_i)$$

Donc,

$$\sum_{i=0}^{\mathbf{d}-1} g(x_i - z_i) \leq \sum_{i=0}^{\mathbf{d}-1} g(x_i - y_i) + \sum_{i=0}^{\mathbf{d}-1} g(y_i - z_i)$$

f est aussi une norme,

$$\mathfrak{d}(x,z) \leq f\left(\sum_{i=0}^{\mathbf{d}-1} g(x_i - y_i) + \sum_{i=0}^{\mathbf{d}-1} g(y_i - z_i)\right) \leq \mathfrak{d}(x,y) + \mathfrak{d}(y,z)$$

□

En s'ajoutant aux 3 propriétés d'une métrique précisées ci-dessous, nous retirons en plus une autre propriété :

4. $\mathfrak{d}(x,y) >= 0 \ \ \forall x, y$

En effet, les métriques usuelles, comme celle euclidienne ou de Manhatan, sont des concrétisations de la métrique dérivée de la norme p (la norme du espace de Lebesgue) :

$$\|x\|_p = \left(\sum_{i=0}^{\mathbf{d}-1} |x_i|^p\right)^{1/p}$$

La métrique proposée est plus large que celle-ci avec $g(x) = |x|^p$, $f(x) = x^{1/p}$.

Corollaire 2. *La réflexion et la permutation des coordonnées sont isométriques avec la métrique 4.3.*

Démonstration. Nous examinons deux points quelconques $u = (u_0, u_1, \ldots, u_{\mathbf{d}-1})$ et $v = (v_0, v_1, \ldots, v_{\mathbf{d}-1})$, une réflexion $\mathfrak{Ref}_A$ et une permutation des coordonnées $\mathfrak{p}^f$ dans l'espace $\{0, 1, \ldots, b-1\}^d$.

Concernant la réflexion, nous avons,

$$\begin{aligned}
\mathfrak{Ref}_A(u)_i - \mathfrak{Ref}_A(v)_i &= \begin{cases} (b-1-u_i) - (b-1-v_i) & \text{si} \quad i \in A \\ -(u_i - v_i) & \text{si} \quad i \in A \end{cases} \\
&\forall i \in \{0, 1, \ldots, \mathbf{d}-1\} \\
\Rightarrow g(\mathfrak{Ref}_A(u)_i - \mathfrak{Ref}_A(v)_i) &= g(u_i - v_i) \quad \forall i \{0, 1, \ldots, \mathbf{d}-1\}
\end{aligned}$$

Ainsi, $\mathfrak{d}(\mathfrak{Ref}_A(u), \mathfrak{Ref}_A(v)) = \mathfrak{d}(u, v)$.
À propos de la permutation des coordonnées, comme f est une permutation

$$\sum_{i=0}^{\mathbf{d}-1} g(u_{f(i)} - v_{f(i)}) = \sum_{i=0}^{\mathbf{d}-1} g(|u_i - v_i|)$$

Donc, $\mathfrak{d}(\mathfrak{p}^f(u), \mathfrak{p}^f(v)) = \mathfrak{d}(u, v)$.

□

Théorème 2. *Toutes les isométries peuvent être décomposées en réflexion et permutation*

Démonstration. Pour démontrer le Théorème 2, nous considérons le Lemme 1.

Lemme 1. *Si le point* $p = (p_0, p_1, \dots, p_{\mathbf{d}-1}) \in \{0, 1, \dots, b-1\}^{\mathbf{d}}$ *satisfait*

$$\exists q : \mathfrak{d}(p, q) = f(\sum_{i=0}^{\mathbf{d}-1} g(b-1))$$

Nous avons $p_i \in \{0, b-1\} \ \ \forall i \in \{0, 1, \dots, \mathbf{d}-1\}$

Démonstration. Pour $s, t \in \{0, 1, \dots, b-1\}$, nous avons $|s-t| \leq b-1$ et

$$|s-t| = b-1 \Leftrightarrow s, t \in \{0, b-1\}$$

Si $\exists i \in \{0, 1, \dots, \mathbf{d}-1\} : p_i \notin \{0, b-1\}$, nous avons,

$$g(p_i - q_i) = g(|p_i - q_i|) < g(b-1)$$

Donc, $\mathfrak{d}(p, q) < f(\sum_{i=0}^{\mathbf{d}-1} g(b-1))$. Cela contredit avec la supposition du lemme.
On en déduit que $p_i \in \{0, b-1\} \ \ \forall \in \{0, 1, \dots, \mathbf{d}-1\}$.

□

Nous continuons de démontrer le Théorème 2.
Considérons une isométrie $\mathfrak{Iso}$ quelconque sur un espace d-dimensionnel de base b.
Posons que $a = (0, 0, \cdots, 0)$ et les points

$$e_i = (e_{i_0}, e_{i_1}, \dots, e_{i_{\mathbf{d}-1}}) \ \ (i \in \{0, 1, \dots, \mathbf{d}-1\})$$

satisfaisant

$$\begin{cases} e_{ii} = b-1 & \\ e_{ij} = 0 & \forall j \neq i \end{cases} \tag{4.4}$$

Pour un point u dans l'espace, écrivons

$$\bar{u} = (b-1-u_0, b-1-u_1, \cdots, b-1-u_{\mathbf{d}-1}$$

Rappelons que :

$$\mathfrak{d}(a, \bar{a})) = f(\sum_{i=0}^{\mathbf{d}-1} g(b-1))$$

et

$$\mathfrak{d}(e_i, \bar{e_i}) = f(\sum_{i=0}^{\mathbf{d}-1} g(b-1))$$

En conséquence,

$$\mathfrak{d}(\mathfrak{Iso}(a), \mathfrak{Iso}(\bar{a})) = f(\sum_{i=0}^{\mathbf{d}-1} g(b-1))$$

et

$$\mathfrak{d}(\mathfrak{Iso}(e_i), \mathfrak{Iso}(\bar{e_i})) = f(\sum_{i=0}^{\mathbf{d}-1} g(b-1))$$

En appliquant le Lemme 1, nous avons

$$\mathfrak{Iso}(a)_i, \mathfrak{Iso}(e_i)_j \in \{0, b-1\} \ \ \forall i, j \in \{0, 1, \ldots, \mathbf{d}-1\}$$

Comme

$$\mathfrak{d}(\mathfrak{Iso}(e_i), \mathfrak{Iso}(a)) = \mathfrak{d}(e_i, a) = b-1$$

les $\mathfrak{Iso}(e_i)$ ne diffèrent que $\mathfrak{Iso}(a)$ à une coordonnée et la différence est $b-1$.

Supposons que

$$A = \{i : \mathfrak{Iso}(a)_i = b-1, i \in \{0, 1, \ldots, \mathbf{d}-1\}\}$$

Nous voyons que

$$\mathfrak{Ref}_A^b(a) = (0, 0, \cdots, 0) = a$$

et comme les $\mathfrak{Iso}(e_i)$ ne diffèrent que $\mathfrak{Iso}(a)$ à une coordonnée et la différence est $b-1$, on a la même relation entre $\mathfrak{Ref}_A^b(\mathfrak{Iso}(e_i))$ et $\mathfrak{Ref}_A^b(\mathfrak{Iso}(a)) = (0, 0, \cdots, 0)$. On en déduit que

$$\forall i \in \{0, 1, \ldots, \mathbf{d}-1\} \ \ \exists j \in \{0, 1, \ldots, \mathbf{d}-1\} : \mathfrak{Ref}_A^b(\mathfrak{Iso}(e_i)) = e_j$$

Pour tous ces couples i, j, nous établissons la fonction f satisfaisant $f(i) = j$.

Appliquons la Corollaire 1, nous avons

$$\mathfrak{Iso}(e_i) = \mathfrak{Ref}_A^b(e_j) = \mathfrak{Ref}_A^b(\mathfrak{p}^f(e_i))$$

Comme $\mathfrak{p}^f(a) = a$, nous avons également,

$$\mathfrak{Iso}(a) = \mathfrak{Ref}_A^b(\mathfrak{p}^f(a))$$

Nous démontrons maintenant que $\mathfrak{Iso} = \mathfrak{Ref}_A^b \circ \mathfrak{p}^f$. Autrement dit,

$$\forall u \in \{0, 1, \dots, b-1\}^{\mathbf{d}}, \mathfrak{Iso}(u) = \mathfrak{Ref}_A^b(\mathfrak{p}^f(u))$$

Comme la réflexion et la permutation sont isométriques, nous avons que $\mathfrak{Ref}_A^b \circ \mathfrak{p}^f$ est aussi isométrique. Avec deux points u, v quelconques dans l'espace, nous avons $\mathfrak{d}(\mathfrak{Iso}(u), \mathfrak{Iso}(v)) = \mathfrak{d}(u, v)$ et $\mathfrak{d}(\mathfrak{Ref}_A^b(\mathfrak{p}^f(u)), \mathfrak{Ref}_A^b(\mathfrak{p}^f(v))) = \mathfrak{d}(u, v)$

Par conséquent,

$$\mathfrak{d}(\mathfrak{Iso}(u), \mathfrak{Iso}(v)) = \mathfrak{d}(\mathfrak{Ref}_A^b(\mathfrak{p}^f(u)), \mathfrak{Ref}_A^b(\mathfrak{p}^f(v)))$$

Appliquons cela, nous avons, avec un point u quelconque,

$$\begin{cases} \mathfrak{d}(\mathfrak{Iso}(u), \mathfrak{Iso}(a)) & = \mathfrak{d}(\mathfrak{Ref}_A^b(\mathfrak{p}^f(u)), \mathfrak{Ref}_A^b(\mathfrak{p}^f(a))) \\ \mathfrak{d}(\mathfrak{Iso}(u), \mathfrak{Iso}(e_i)) & = \mathfrak{d}(\mathfrak{Ref}_A^b(\mathfrak{p}^f(u)), \mathfrak{Ref}_A^b(\mathfrak{p}^f(e_i))) \ \ \forall i \in \{0, 1, \dots, \mathbf{d}-1\} \end{cases}$$

$$\Leftrightarrow \begin{cases} \sum_{j=0}^{\mathbf{d}-1} g(\mathfrak{Iso}(u)_j - \mathfrak{Iso}(a)_j) & = \sum_{j=0}^{\mathbf{d}-1} g(\mathfrak{Ref}_A^b(\mathfrak{p}^f(u))_j - \mathfrak{Ref}_A^b(\mathfrak{p}^f(a))_j) \\ \sum_{j=0}^{\mathbf{d}-1} g(\mathfrak{Iso}(u)_j - \mathfrak{Iso}(e_i)_j) & = \sum_{j=0}^{\mathbf{d}-1} g(\mathfrak{Ref}_A^b(\mathfrak{p}^f(u))_j - \mathfrak{Ref}_A^b(\mathfrak{p}^f(e_i))_j) \\ \forall i \in \{0, 1, \dots, \mathbf{d}-1\} \end{cases}$$

Appliquons la suppression à ces deux lignes, comme $\mathfrak{Iso}(e_i) = \mathfrak{Ref}_A^b \circ \mathfrak{p}^f(e_i)$ et $\mathfrak{Iso}(a) = \mathfrak{Ref}_A^b \circ \mathfrak{p}^f(a)$ ne diffèrent qu'à la coordonnée $f(i)$ et la différence est $b-1$, nous avons :

$$\begin{array}{lll} & g(\mathfrak{Iso}(u)_{f(i)} - \mathfrak{Iso}(e_i)_{f(i)}) & = g(\mathfrak{Ref}_A^b(\mathfrak{p}^f(u))_{f(i)} - \mathfrak{Ref}_A^b(\mathfrak{p}^f(e_i))_{f(i)}) \\ & \forall i \in \{0, 1, \dots, \mathbf{d}-1\} & \\ \Leftrightarrow & \mathfrak{Iso}(u)_{f(i)} - \mathfrak{Iso}(e_i)_{f(i)}) & = \mathfrak{Ref}_A^b(\mathfrak{p}^f(u))_{f(i)} - \mathfrak{Ref}_A^b(\mathfrak{p}^f(e_i))_{f(i)} \\ & \forall i \in \{0, 1, \dots, \mathbf{d}-1\} & \\ \Leftrightarrow & \mathfrak{Iso}(u)_{f(i)} & = \mathfrak{Ref}_A^b(\mathfrak{p}^f(u))_{f(i)} \ \ \forall i \in \{0, 1, \dots, \mathbf{d}-1\} \\ \Leftrightarrow & \mathfrak{Iso}(u)_i & = \mathfrak{Ref}_A^b(\mathfrak{p}^f(u))_i \ \ \forall i \in \{0, 1, \dots, \mathbf{d}-1\} \\ \Leftrightarrow & \mathfrak{Iso}(u) & = \mathfrak{Ref}_A^b(\mathfrak{p}^f(u)) \end{array}$$

□

Une isométrie $\mathfrak{Iso}$ dans l'espace $\{0, 1, \dots, \mathbf{b}-1\}^{\mathbf{d}}$ peut être décomposée comme suit :

*De*1. Déterminer l'ensemble A des coordonnées égales à $\mathbf{b}-1$ de l'image de $(0, 0, \cdots 0)$.

$$A = \{i \in \{0, 1, \dots, \mathbf{d}-1\} : \mathfrak{Iso}(0, 0, \cdots, 0)_i = \mathbf{b}-1\}$$

*De*2. Déterminer la bijection de permutation f. Examinons l'application $\mathfrak{Ref}_A^{\mathbf{b}} \circ \mathfrak{Iso}$ et les e_is sont définis dans la formule 4.4. Nous constatons que pour chaque e_i, il existe un seul index j satisfaisant $\mathfrak{Ref}_A^{\mathbf{b}}(\mathfrak{Iso}(e_i))_j \neq 0$ qui permet de définir f comme suit : $f(i) = j$.

*De*3. $\mathfrak{Iso} = \mathfrak{Ref}_A^b \circ \mathfrak{p}^f$

PROCESSUS 1: Décomposition d'une isométrie quelconque.

En s'appuyant sur le Théorème 2, nous pouvons décomposer une isométrie en une réflexion et une permutation des coordonnées selon le Processus 1.

Exemple 3. Décomposition d'une isométrie quelconque en réflexion et permutation : application du Processus 1.

Considérons l'isométrie $\mathfrak{Iso}$ qui transforme $\mathfrak{m}_1$ en $\mathfrak{m}_3$ illustrée dans la Figure 4.9, nous appliquons le processus comme suit :

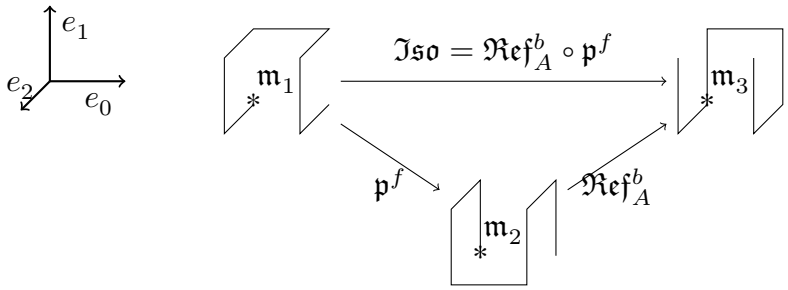


FIGURE 4.9 – Illustration d'une décomposition d'une isométrie. Les motifs commencent par le symbole ``*''. L'isométrie qui transforme $\mathfrak{m}_1$ en $\mathfrak{m}_3$ peut être décomposée en $\mathfrak{Ref}_A^b$ et $\mathfrak{p}^f$.

— Déterminons $A = \{i \in \{0, 1, \dots, \mathbf{d}-1\} : \mathfrak{Iso}(0, 0, \cdots, 0)_i = 1\}$. Comme le point $(0, 0, 0)$ du $\mathfrak{m}_1$ est correspondant au point $(0, 1, 1)$ du $\mathfrak{m}_3$, $\mathfrak{Iso}(0, 0, 0) = (0, 1, 1)$ ou $\mathfrak{Iso}(0, 0, 0)_0 = 0, \mathfrak{Iso}(0, 0, 0)_1 = 1, \mathfrak{Iso}(0, 0, 0)_2 = 1$. Ainsi, $A = \{1, 2\}$

— Selon 4.4, $e_0 = (1,0,0); e_1 = (0,1,0); e_2 = (0,0,1)$. Les points $(1,1,1),(0,1,0),(0,0,1)$ du $\mathfrak{m}_3$ sont consécutivement les correspondances de e_0, e_1, e_2 du $\mathfrak{m}_1$. Autrement dit,

$$\begin{aligned}\mathfrak{Iso}(e_0) &= (1,1,1)\\ \mathfrak{Iso}(e_1) &= (0,1,0)\\ \mathfrak{Iso}(e_2) &= (0,0,1)\end{aligned}$$

Appliquons $\mathfrak{Ref}_A^{\mathbf{b}}$ sur ces points, nous avons

$$\begin{aligned}\mathfrak{Ref}_A^{\mathbf{b}}(\mathfrak{Iso}(e_0)) &= (1,0,0)\\ \mathfrak{Ref}_A^{\mathbf{b}}(\mathfrak{Iso}(e_1)) &= (0,0,1)\\ \mathfrak{Ref}_A^{\mathbf{b}}(\mathfrak{Iso}(e_0)) &= (0,1,0)\end{aligned}$$

Selon *De2* : $f(i) = j$ si $\mathfrak{Ref}_A^{\mathbf{b}}(\mathfrak{Iso}(e_0)) \neq 0$, nous avons $f(0) = 0, f(1) = 2, f(2) = 1$.

En résumé, les descriptions de la réflexion et la permutation des coordonnées sont $A = 1,2$ et $f(0) = 0; f(1) = 2; f(2) = 1$.

4.4 Conservation de localité des courbes remplissant l'espace et mesures

La conservation de la localité des courbes remplissant l'espace est souvent citée comme une justification de leur utilisation. Cette propriété est l'une des conséquences de l'*autosimilitude* : Intuitivement, les points dans chaque partie de l'espace sont consécutivement ordonnés. Ainsi, cette localité (une partie de l'espace) s'est associée à une localité dans l'espace des index (le segment des index consécutifs). Ainsi, la localité est conservée par les courbes remplissant l'espace.

Dans cette section, nous mesurons le niveau de conservation de la localité de diverses courbes remplissant l'espace classiques, parmi lesquelles figurent des courbes vérifiant l'*autosimilitude* au sens de Définition 12 (Lebesgue, Hilbert) et d'autres qui ne la vérifient pas (Scan, Sweep, Aléatoire). Nous constatons qu'il existe une partition en termes de niveau de localité entre les courbes vérifiant ou pas la condition d'*autosimilitude*.

La conservation de la localité est estimée avec la mesure proposée dans le Chapitre 3.

4.4.1 Expérimentation

Ces évaluations sont réalisées sur des courbes de différentes dimensions $\mathbf{d} = 2,3,4$ à l'ordre $\mathbf{n}$= 6. La courbe multidimensionnelle dite de Hilbert est générée

avec l'algorithme de Butz [29] ; la courbe de Lebesgue est générée par interfoliage des coordonnées [*cf.* Sous-section 2.1.4] ; la génération des autres courbes est décrite en Annexe A.

Ces courbes sont à l'ordre 6, c'est-à-dire qu'une courbe $\mathbf{d}$-dimensionnelle contient $2^{6\mathbf{d}} = 64^{\mathbf{d}}$ points. Cet ordre est choisi selon 2 critères :

1. Les mesures sont réalisées sur une machine standard (décrite en Annexe B), les index sont sauvegardés sous forme de nombres entiers de 32 bits. Par conséquent, avec ce codage, le nombre des points de l'espace possible est limité à 2^{32} ou l'ordre $\mathbf{n}$ est limitée à $32/\mathbf{d}$.
2. L'ordre doit être assez grand pour limiter l'effet des marges dans l'estimation de la conservation de la localité. À la marge, les points n'ont qu'une partie de leurs voisins.

Les estimations menées sur la courbe aléatoire sont des moyennes calculées sur un échantillon de 1000 tirages d'ordonnancement aléatoires selon une loi uniforme.

4.4.2 Mesure

Pour appliquer la mesure proposée dans le Chapitre 3, nous utilisons la métrique euclidienne pour évaluer les distances. Les localités sont déterminées selon le rayon :

— pour un rayon r_1 de l'espace d'origine donné, la localité d'un point p est $\mathcal{V}_{r_1}(P) = \{v : \mathfrak{d}(v,p) \leq r_1\}$

— le rayon r_2 est choisi pour que la localité $\{j : \mathfrak{d}(j,i) \leq r_2\}$ de l'index i de p soit de la même taille que la localité de p.

4.4.3 Résultats

Les tableaux 4.1, 4.2 et 4.3 synthétisent les résultats des mesures correspondant aux dimensions 2, 3 et 4 avec différents rayons r_1. À titre indicatif, les courbes sont illustrées dans les figures 4.2, 2.4 et 2.3.

Rayon r_1	Hilbert	Lebesgue	Sweep	Scan	Aléatoire
1	**0.61**	**0.61**	0.60	**0.61**	0.20
2	**0.66**	0.55	0.39	0.41	0.08
3	**0.61**	0.58	0.28	0.32	0.05
4	**0.64**	0.55	0.22	0.28	0.04

TABLEAU 4.1 – Résultat de la conservation de la localité de plusieurs courbes remplissant l'espace en dimension $\mathbf{d}$=2 à l'ordre $\mathbf{n}$=6. Les chiffres en gras correspondent au plus haut niveau de localité.

Rayon r_1	Hilbert	Lebesgue	Sweep	Scan	Aléatoire
1	**0.44**	**0.44**	0.43	**0.44**	0.15
2	**0.37**	0.31	0.20	0.21	0.04
3	**0.32**	0.26	0.11	0.13	0.02
4	**0.26**	0.23	0.07	0.09	0.01

TABLEAU 4.2 – Résultat de la conservation de la localité de plusieurs courbes remplissant l'espace en dimension $\mathbf{d}$=3 à l'ordre $\mathbf{n}$=6. Les chiffres en gras correspondent au plus haut niveau de localité.

.

Rayon r_1	Hilbert	Lebesgue	Sweep	Scan	Aléatoire
1	**0.34**	**0.34**	0.33	**0.34**	0.12
2	**0.25**	0.23	0.12	0.13	0.03
3	**0.16**	0.13	0.06	0.06	0.01
4	**0.13**	0.11	0.03	0.04	0.00

TABLEAU 4.3 – Résultat de la conservation de la localité de plusieurs courbes remplissant l'espace en dimension $\mathbf{d}$=4 à l'ordre $\mathbf{n}$=6. Les chiffres en gras correspondent au plus haut niveau de localité.

.

À la lecture des résultats (tableaux 4.1, 4.2 et 4.3) nous observons que :

— la courbe qui présente le plus haut niveau de conservation de la localité est la courbe de Hilbert (surtout lors que le rayon de localité r_1 augmente). L'application de notre mesure [*cf.* Chapitre 3] conduit donc à des résultats conformes à ceux déjà observés dans la littérature [60, 113, 128].

— il existe une partition en termes de résultats entre les courbes vérifiant la condition d'*autosimitude* au sens de Définition 12 et les autres. Notamment, on voit apparaître l'émergence des courbes de Hilbert et Lebesgue face aux courbes Sweep, Scan et Aléatoire, quels que soient la dimension $\mathbf{d}$ et le rayon r_1. Notons que ces observations sont également confirmées par d'autres tests à des dimensions supérieures ($\mathbf{d}$=5, 6, 7). La vérification de la Définition 12 contribue à construire des courbes ayant un haut niveau de conservation de la localité.

4.5 Conclusion

Dans une application, une courbe remplissant l'espace est évaluée selon deux critères :

— le calcul de l'index d'un point à partir ses coordonnées. Ce critère dépend de

la définition de la courbe. Une définition concise permet un calcul rapide de l'index,

— la conservation de la localité.

En s'appuyant sur ces deux critères, nous avons introduit et analysé différents aspects de la courbe remplissant l'espace. Tout d'abord, les notions concernant et la définition de la courbe remplissant l'espace sont exposées. Pour définir la courbe remplissant l'espace, nous proposons une définition de l'autosimilitude, la propriété clé de la courbe remplissant l'espace. Cette propriété apporte aux courbes remplissant l'espace l'avantage sur les deux critères ci-dessus :

— l'autosimilitude permet de définir concisément une courbe remplissant l'espace. En étant autosimilaire, une courbe est composée des sous-courbes composées de similitudes du motif primitif. Ainsi, au lieu d'être définie par l'énumération de points, une sous-courbe est représentée par la similitude correspondante. Avec ce modèle compact, le calcul de l'index est plus rapide parce qu'il n'est plus un positionnement dans une liste des points.

— la courbe conserve bien la localité sur les deux champs : local et global. Comme les points de chaque sous-espace sont ordonnés consécutivement, la courbe conserve la localité au niveau local. De plus, une courbe remplissant l'espace hérite de la courbe à l'ordre inférieur, qui conserve bien la localité. Ainsi, elle hérite globalement la conservation de la localité de cette dernière.

En s'intéressant à la concision de la définition des courbes remplissant l'espace, nous proposons une simplification des similitudes par la décomposition d'une similitude en deux transformations simples : réflexion et permutation des coordonnées.

À la fin du chapitre, la conservation de la localité des ordonnancements de l'espace différents est estimée avec notre mesure proposée dans le Chapitre 3. Les résultats montrent qu'en vérifiant l'*autosimilitude*, les courbes remplissant l'espace sont les courbes qui conservent le mieux la localité par rapport aux autres ordonnancements de l'espace. Parmi les courbes remplissant l'espace, la courbe de Hilbert est celle qui maximise la conservation de la localité. En effet, plusieurs travaux de recherche dans la littérature soulignent déjà la conservation de la localité supérieure de la courbe de Hilbert. Nous choisissons la courbe de Hilbert pour analyser et extraire les bonnes propriétés en constituant une courbe qui conserve bien la localité [*cf.* Chapitre 5].

Chapitre 5

Proposition d'une famille de courbe conservant bien la localité : généralisation multidimensionnelle de la courbe de Hilbert

Résumé:

Suite à l'étude de l'optimisation de la conservation de la localité des courbes remplissant l'espace, nous présentons une proposition de généralisation multidimensionnelle de la courbe de Hilbert, la courbe qui conserve mieux la localité selon plusieurs critères, en retenant la propriété essentielle : l'adjacence. En effet, pour optimiser la conservation de la localité, dans un premier temp nous nous affranchissons de l'autosimilitude en maintenant seulement l'héritage pour créer la souplesse de la construction de courbe, le résultat est une classe des courbes.

Les mesures de conservation de la localité montrent que ces courbes conservent bien la localité. De plus, avec la diversification des courbes, l'optimisation de conservation de la localité est possible.

Par contre, parfois une grande dimension et la vitesse du calcul de l'index sont demandées. Dans ce cas, nous revenons sur l'autosimilitude. Nous proposons alors une optimisation de la conservation de la localité des courbes de Hibert autosimilaires en soulignant sur le choix du motif primitif et la jonction des sous-courbes.

5.1 Motivation

5.1.1 Positionnement par rapport aux travaux existants

Dans les travaux de recherche relatifs à l'extension multidimensionnelle de la courbe de Hilbert [*cf.* Section 2.3], les réflexions menées sur les transformations, leurs enchaînements reposent essentiellement sur un seul motif primitif : le RBG $\mathbf{d}$-dimensionnel avec $\mathbf{d}$ donné, *i.e.* finalement sur un nombre limité de manière de remplir l'espace.

En effet, pour les paramètres de dimension et d'ordre fixés, les auteurs ci-dessus n'essaient que de proposer une seule courbe de Hilbert multidimensionnelle avec l'objectif de comparer la conservation de la localité de la courbe de Hilbert avec les autres courbes remplissant l'espace. Pour cet objectif, les transformations des sous-courbes sont aussi fixées dans les méthodes mentionnées ci-dessus sans regarder la possibilité de modifier les transformations.

Concrètement, selon ces recherches, une courbe $\mathbf{d}$-dimensionnelle dite de Hilbert possède les propriétés communes suivantes :

1. une courbe remplissant l'espace satisfaisant l'*autosimilitude* (au sens de Définition 12),
2. sa base est égale à 2,
3. son motif primitif est celui RBG [*cf.* Section 5.3.1],
4. les transformations des sous-courbes doivent satisfaire la condition suivante : si a et b sont des sous-courbes consécutives (a précède b), le dernier point de a et le premier point de b sont adjacents [*cf.* Définition 17].

En effet, l'application de ces méthodes ne donne naissance qu'à une unique courbe dite de Hilbert. *Est-il possible de construire d'autres courbes pour remplir un espace multidimensionnel qui vérifieraient un niveau de conservation de la localité comparable à la courbe de Hilbert ?*

5.1.2 Motivations et orientations

Nous proposons une nouvelle méthode ([*cf.* Sous-section 5.2.2]) permettant de construire une famille de courbes remplissant un espace multidimensionnel de niveau de conservation de la localité comparable à la courbe de Hilbert ([*cf.* Section 5.4]). Nous avons souligné [*cf.* Sous-section 5.1.1] que l'identification des transformations géométriques (à l'ordre $\mathbf{n}$ donné) joue un rôle dans l'articulation des $2^{\mathbf{d}}$ sous-courbes, contribuant ainsi à construire globalement une courbe conservant la localité. Nous pensons que le contrôle des points de départs et d'arrivées (extrémités) au niveau de chaque sous-courbe est un facteur déterminant dans le cadre d'une extension multidimensionnelle. En imposant via une contrainte ([*cf.* Définition 18] / $C3$) que la

distance entre les points d'arrivées et de départs de deux sous-courbes consécutives soit unitaire (adjacence), on garantit la continuité inter-sous-courbes ce qui permet :

— de contourner la difficulté [1] d'identifier explicitement un ensemble de matrices de transformations ([*cf.* Sous-section 5.2.2]) ;

— d'éviter toute rupture de topologie (saut dans l'espace original) nuisible pour la conservation de la localité.

De plus, en formulant des règles guidant la construction des sous-courbes même ([*cf.* Sous-section 5.2.2] / conditions $C1$ et $C3$), nous montrons qu'aux extrémités fixées il existe plusieurs courbes remplissant un espace. Ceci est également vrai à l'ordre $\mathbf{n} = 1$ ce qui nous permet de sortir du seul recours à la courbe RBG comme motif primitif ([*cf.* Section 5.4]) et de pouvoir enfin définir une famille de courbes qui conservent bien la localité.

Ces hypothèses sont confirmées par les expérimentations menées dans la Section 5.4.

5.2 Courbe de Hilbert généralisée

Dans cette section, nous proposons une définition de la courbe de Hilbert généralisée en précisant la propriété essentielle de la courbe de Hilbert : l'adjacence. Différentes illustrations permettent de distinguer les courbes qui satisfont cette définition et ainsi que les autres.

5.2.1 Adjacence

L'adjacence est la propriété qui crée le niveau remarquable de conservation de la localité de la courbe de Hilbert. L'adjacence est formulée comme suit :

Définition 17 (Adjacence)**.** *Deux points dans un espace* $\mathbf{d}$*-dimensionnel*

$$\begin{aligned} x &= (x_0, x_1, \dots, x_{\mathbf{d}-1}) \\ y &= (y_0, y_1, \cdots, y_{\mathbf{d}-1}) \end{aligned}$$

sont dits adjacents si $\exists i \in \{0, 1, \dots, \mathbf{d}-1\}$ *de façon que*

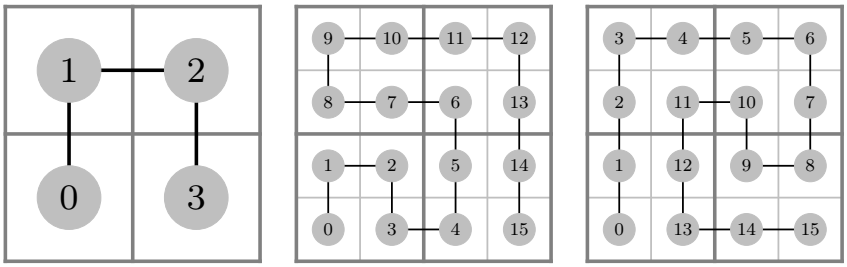
$$\begin{cases} |x_i - y_i| = 1 \\ x_j = y_j \quad \forall j \in \{0, 1, \dots, \mathbf{d}-1\} \quad j \neq i \end{cases}$$

1. croissante avec les dimensions.

En d'autres termes, deux points sont dits adjacents si la distance entre eux est l'unité. *Un ordonnancement de l'espace est dit adjacent si chaque couple de deux points consécutifs est adjacent.*

Malgré que l'adjacence soit une propriété importante pour la conservation de localité, elle est étroitement associée à l'*héritage*.

La courbe de Scan est un exemple. Cette courbe est adjacente mais elle ne conserve pas bien la localité. Le point crucial est qu'elle ne satisfait pas l'héritage. La Figure 5.1 montre une courbe adjacente remplissant un espace de taille 2×2 qui est aussi le motif primitif de la courbe de Hilbert 2-dimensionnelle et deux courbes adjacentes de taille 4×4 autre que la courbe de Hilbert [*cf.* Figure 2.3] et la courbe Scan [*cf.* Figure 4.2].



(a) *(b)* *(c)*

FIGURE 5.1 – Exemple de courbes vérifient l'*adjacence* [*cf.* Définition 17]. La propriété adjacente seule ne suffit pas pour bien conserver la localité. Aux ordres supérieurs ($\mathbf{n}$>1), il faut y rajouter la propriété d'*héritage* [*cf.* Définition 10]. Les courbes *(b)* et *(c)* ne respectent pas totalement les conditions de la courbe de Hilbert.

5.2.2 Définition et exemples d'application

Nous utilisons le postulat qu'il existe plusieurs courbes au-delà de la dimension 2, autres que celles générées par l'algorithme de Butz [29] ou Lawder [93]. Notre hypothèse est qu'en proposant une extension au cas multidimensionnel [*cf.* Définition 18] de la définition de Hilbert établie en dimension 2 [*cf.* Sous-section 2.1.3], nous accédons à l'ensemble de ces courbes.

Nous proposons une méthode qui consiste à discrétiser un espace de dimension $\mathbf{d}$, à l'ordre $\mathbf{n}$, en $2^{\mathbf{nd}}$ sous-espaces dont l'ordre de parcours satisfait un niveau de conservation de la localité (au sens de [56, 60, 113]) comparable à celui de la courbe de Hilbert. Nous retenons dans ces courbes l'adjacence.

Notre définition de la courbe de Hilbert multidimensionnelle ne contient que les conditions minimales d'une courbe de Hilbert pour permettre une variation large en cherchant à retrouver des bonnes courbes remplissant l'espace.

Définition 18 (Courbe de Hilbert généralisée)**.**

*G*1. *À l'ordre* $\mathbf{n}$*=1 : une courbe de Hilbert à l'ordre* $\mathbf{n}$*=1 est un ordonnancement d'un espace* $\mathcal{C}$ *de résolution* $\mathbf{b}$*=2 vérifiant l'adjacence [cf. Définition 17],*

*G*2. *À l'ordre* $\mathbf{n}$*>1 : un ordonnancement de l'espace* $\mathcal{C}$ *est dit une courbe de Hilbert d'ordre* $\mathbf{n}$ *(>1) si :*

*D*1. $\mathcal{C}$ *hérite de rapport* $\mathbf{b}$*=2 (au sens de Définition 10) d'une courbe de Hilbert* $\mathcal{C}'$ *d'ordre* $\mathbf{n}$*-1,*

et

*D*2. *à l'intérieur de chaque sous-espace* $\mathcal{S}_k$*,* $\mathcal{C}$ *vérifie l'adjacence.*

Par conséquent, une courbe de Hilbert $\mathcal{C}$ comme ci-dessus est un ordonnancement de l'espace vérifiant 3 conditions :

*C*1. $\mathcal{C}$ *ordonne consécutivement les points appartenant à chaque sous-espace* $\mathcal{S}_k$*,*

*C*2. $\mathcal{C}$ *ordonne les sous-espaces* $\mathcal{S}_k$ *dans le même ordre que leurs points correspondants* k *de* $\mathcal{C}'$*.*

*C*3. $\mathcal{C}$ *est adjacente.*

L'ajout de la condition *C*3 (l'adjacence) au respect conjoint des conditions *C*1 et *C*2 permet de construire une courbe dont le niveau de conservation de la localité est comparable à la courbe de Hilbert originelle ([*cf.* Section 5.4]).

La condition *C*3 opère localement (dès l'ordre 1, selon *G*1) dans l'ordonnancement des points, venant compléter la condition *C*1 ([*cf.* Figure 5.2] : illustrations à $\mathbf{n} = 1$) ; mais également (dès l'ordre 2, selon *G*2*D*2) dans l'enchaînement des ordonnancements locaux complétant ainsi la condition *C*2 ([*cf.* Figure 5.3] : illustrations à $\mathbf{n} = 2$). Ainsi tous les points appartenant à un sous-espace donné doivent certes être connectés consécutivement avant d'être enchaînés aux sous-espaces voisins. De plus, ces connections tout comme ces enchaînements doivent vérifier l'adjacence ([*cf.* Figure 5.3], milieu et droit).

En imposant une contrainte d'adjacence aux points consécutifs, la condition *C*3 crée une partition, en termes de localité, entre les courbes obéissant à la Définition 18 et d'autres types de courbes comme celle de Lebesgue par exemple.

Finalement, la définition proposée [*cf.* Définition 18] est suffisamment large pour que plusieurs courbes (une famille de courbes) puissent en satisfaire toutes les conditions ([*cf.* Figure 5.3] - les 2 courbes à droite de la figure).

La Section 5.3 est consacrée à l'apport d'éléments de preuve concernant sa validité.

Notamment, nous répondrons à trois questions clés :

— Existe-t-il, en dimension $\mathbf{d}$ donnée, des courbes vérifiant toutes les conditions de la Définition 18 ?

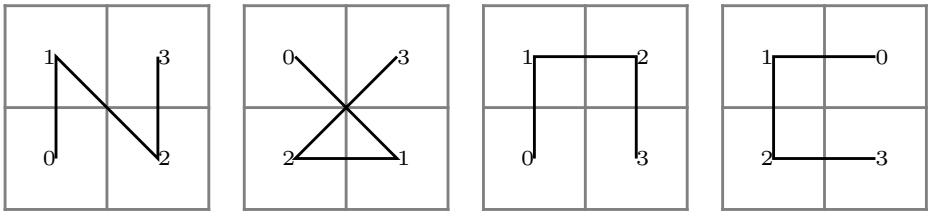


FIGURE 5.2 – Influence de la condition $C3$ à l'ordre 1 dans le processus de construction des courbes. Cas de sous-courbes synthétisant le respect de la condition $C3$ (2 courbes à droite) ou non (2 courbes à gauche) sachant le respect de la condition $C1$.

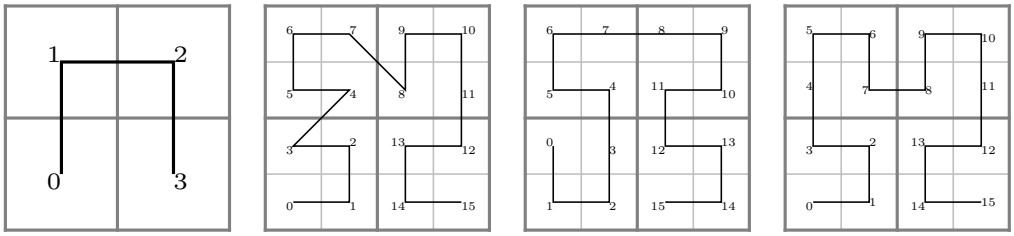


FIGURE 5.3 – Vers le respect de l'ensemble des conditions : influence de la condition $C3$ à l'ordre 2 dans la construction des courbes. Sachant le respect de $C1$ et $C2$, cas de courbes synthétisant le respect ou non de $C3$. Les 2 courbes à droite satisfont toutes les conditions. On reconnaît la courbe de Hilbert à l'extrême droite.

— Si oui, combien de courbes ?
— Ont-t-elles le même niveau de conservation de la localité ?

Ces éléments présentent l'intérêt de mieux identifier les degrés de liberté apparaissant dans le processus de construction ce qui s'avère utile dans la perspective d'une implémentation informatique.

5.3 Processus de construction et preuves

Nous rappelons que ce processus consiste à discrétiser un espace de dimension $\mathbf{d}$, à l'ordre $\mathbf{n}$, en $2^{\mathbf{nd}}$ points dont l'ordre de parcours satisfait un niveau de conservation de la localité comparable à celui issu de la courbe de Hilbert. Lors du passage de l'ordre $\mathbf{n}$ à $\mathbf{n}+1$, chacun des $2^{\mathbf{nd}}$ points est discrétisé localement en $2^{\mathbf{d}}$ points, ce qui donne naissance à $2^{\mathbf{nd}}$ sous-espaces de $2^{\mathbf{d}}$ points. Deux types de contraintes doivent être vérifiées :

— une contrainte locale portant sur l'ordre de parcours des $2^{\mathbf{d}}$ points dans chaque sous-espace (conditions $C1$ et $C3$) ;

— une contrainte globale s'exerçant sur l'articulation (l'enchaînement) des divers parcours établis localement sur chacune des $2^{\mathbf{nd}}$ sous-courbes qui remplissent les sous-espaces (conditions $C2$ et $C3$).

La contrainte globale est plus restrictive que la contrainte locale. Autrement dit, nous devons d'abord identifier les points de départ et d'arrivée (dans chaque sous-espace) garantissant les conditions $C2$ et $C3$ avant de remplir l'espace ainsi délimité par des sous-courbes vérifiant la contrainte locale.

Ceci est synthétisé par le Processus 2.

— **Entrée :** un espace de dimension $\mathbf{d}$
— **Répéter jusqu'à l'ordre $\mathbf{n}$ désiré :**

$e1$. Découper les points, chaque point deviendra un sous-espace de $2^{\mathbf{d}}$ points

$e2$. Choisir un couple d'extrémités dans chaque sous-espace vérifiant les conditions $C2$ et $C3$

$e3$. Choisir un parcours entre les couples d'extrémités vérifiant les conditions $C1$ et $C3$

Processus 2: Générer une courbe multidimensionnelle (de niveau de conservation de la localité comparable à la courbe de Hilbert).

Par souci de clarté, nous aborderons dans un premier temps la construction de la courbe vérifiant les contraintes locales (le motif primitif) [*cf.* Sous-section 5.3.1]. Nous traiterons ensuite la construction de la courbe satisfaisant la contrainte globale [*cf.* Sous-section 5.3.2].

5.3.1 Construction d'une courbe à l'ordre 1 : le motif primitif

En comparant avec les propriétés de la courbe de Hilbert décrite par Butz, Faloutsos et Lawder, les courbes proposées [*cf.* Définition 18] peuvent être décomposées en sous-courbes de différentes formes. En d'autres termes, elles sont construites depuis différents motifs primitifs, qui ne sont pas obligatoirement RBG. Un motif primitif peut être résultat de l'application de la Définition 18 à l'ordre $\mathbf{n} = 1$ ($G1$) ou un parcours à l'intérieur d'un sous-espace ($G2D2$).

Concrètement, comme la courbe de Hilbert satisfait la condition $C1$, chacun de ses motifs primitifs remplit un sous-espace de résolution $\mathbf{b}$=2. Un motif primitif $\mathfrak{m}$ est donc une séquence de $2^{\mathbf{d}}$ points $\mathfrak{m}_i \in \{0,1\}^{\mathbf{d}} \ \ \forall i \in \{0,1,\cdots,2^{\mathbf{d}}\}$, chaque point $\mathfrak{m}_i \ \ \forall i \in \{0,1,\cdots,2^{\mathbf{d}}\}$ est représenté par $\mathbf{d}$ coordonnées :

$$\mathfrak{m}_i = (\mathfrak{m}_{i_0}, \mathfrak{m}_{i_1}, \ldots, \mathfrak{m}_{i_{\mathbf{d}-1}})$$

où $\mathfrak{m}_{ij} \in \{0,1\} \ \ \forall i \in \{0,1,\ldots,2^{\mathbf{d}}-1\}, \ \ j \in \{0,1,\ldots,\mathbf{d}-1\}$.

De plus, comme la courbe satisfait l'adjacence (condition $C3$), chaque motif primitif est une partie (ou une similitude d'une partie) de la courbe, il satisfait aussi l'adjacence. C'est-à-dire,

$\forall i \in \{0, 1, \ldots, 2^{\mathbf{d}} - 1\}$ $\exists k \in \{0, 1, \ldots, \mathbf{d} - 1\}$ tel que :

$$\begin{cases} \mathfrak{m}_{i+1j} = \mathfrak{m}_{ij} & \forall j \neq k \\ \mathfrak{m}_{i+1k} = 1 - \mathfrak{m}_{ik} \end{cases}$$

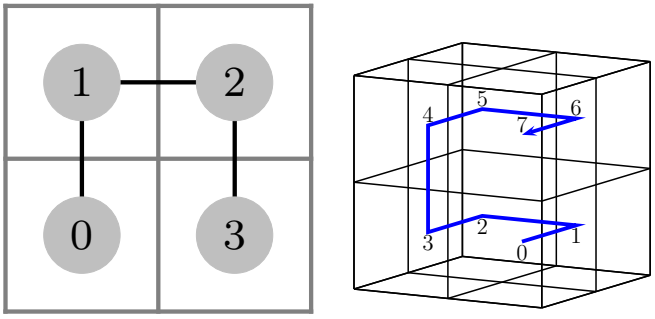
Motif RBG

Notons que le recours classique au motif primitif RBG comme courbe de Hilbert $\mathbf{d}$-dimensionnelle d'ordre 1, satisfait les conditions $C1$ et $C3$ de notre proposition quel que soit $\mathbf{d}$. Il est donc un motif primitif de la courbe de Hilbert ([*cf.* Figure 5.4]).

Le motif RBG s'appuie sur le code binaire réfléchi de Gray. Un code de Gray contient une séquence des mots de taille fixée satisfaisant la distance de Hamming entre chacun des deux codes consécutifs est de 1. Autrement dit, si on considère qu'un mot est un point dont les coordonnées sont des lettres, une séquence de points satisfaisant l'adjacence est un code de Gray. Dans ce cas-là, la dimension est la taille des mots. Par conséquence, les motifs primitifs de la courbe de Hilbert et de la courbe de Peano sont des exemples du code de Gray.

Le code binaire réfléchi de Gray (RBG) est une implémentation du code de Gray binaire qui permet de générer des mots binaires de longueur finie tel que deux mots consécutifs ne diffèrent que d'un seul bit. Parmi les codes de Gray, le code de Gray binaire est l'objet de plusieurs recherches du fait que beaucoup d'applications dans le monde de l'électronique (analogue et numérique) [25], s'appuient sur ce principe. Le code RBG est construit par la réflexion du code de taille inférieure.

Un motif primitif RBG $\mathbf{d}$-dimensionnel est un code RBG avec les mots de longueur $\mathbf{d}$. La Figure 5.4 montre deux motifs primitifs RBG de dimension 2 et 3.



*(**d**=2)* *(**d**=3)*

FIGURE 5.4 – Remplir un espace de dimension 2 et 3 : recours classique au motif primitif RBG.

En effet, le RBG est le motif primitif unique de la courbe de Hilbert multidimensionnelle dans les extensions mentionnées.

Comme les codes consécutifs dans une séquence de code RBG satisfont naturellement les conditions de la courbe de Hilbert (l'adjacence et la binarité), l'application de ce code à la courbe est évidente. Un autre avantage pour cette application est la détermination rapide de l'ordre d'un code dans une séquence RBG. Cela est réalisé via le décalage de bit et l'opérateur XOR (OU exclusif) [*cf.* Algorithme 1]. Cette approche permet d'éviter de générer le code complet avec tous les mots. Une génération du code complet nécessite un grand espace de mémoire et beaucoup temps d'exécution. De plus, la recherche des mots dans un code complet est longue.

Algorithme 1 Déterminer ordre d'un point dans une séquence RBG.

procédure IndexerRBG($a = (a_1 a_2 \dots a_d)_2$)
    /* $a = (a_1 a_2 \dots a_d)_2$ est la représentation binaire de a */

    /* $b = a \oplus$ DecalageDroite$(a, 1)$ */
    /* DecalageDroite(a, n) décale a à droite n bit */
    $b_1 = a_1, b_2 = a_2 \oplus a_1, b_3 = a_3 \oplus a_2, \dots, b_d = a_d \oplus a_{d-1}$
    retourner $b = (b_1 b_2 \dots b_d)_2$
fin procédure

Donnant une longueur de mot, nous pouvons toujours générer un motif primitif RBG. Quel que soit la dimension $\mathbf{d}$, il existe au moins un motif primitif.

Autres motifs primitifs : sortir du cas RBG

En dehors de la limite du motif primitif RBG, un motif primitif de la courbe de Hilbert ne doit que satisfaire la condition $C3$. Nous sommes donc a priori libre de choisir des points de départ et d'arrivée autres que ceux imposés par la courbe RBG. Ces éléments seront développés par la suite.

Nous pouvons facilement vérifier que dans le cas 2-dimensionnelle, il y a 8 motifs primitifs qui satisfont l'adjacence. Mais ces motifs primitifs sont similaires (au sens de Définition 11) au motif primitf RBG de dimension $\mathbf{d}$=2. Cependant, quand la dimension augmente, nous avons des motifs primitifs qui ne sont pas similaires à RBG. Par exemple, dans le cas 3-dimensionnelle, nous avons 54 motifs primitifs avec 3 formes différentes montrées dans la Figure 5.6.

Nous fournissons l'Algorithme 2 permettant d'énumérer tous les motifs primitifs à la dimension $\mathbf{d}$ donnée. La figure 5.5 illustre la diversité des motifs primitifs solution en dimension 2.

La condition $C3$, qui porte sur l'adjacence des points consécutifs, est une contrainte souple au gré de l'augmentation des dimensions. En effet, elle peut être satisfaite dans

Algorithme 2 Chercher tous les motifs primitifs possibles en dimension $\mathbf{d}$.

procédure ÉNUMÉRATIONDEMOTIFS($\mathbf{d}$)
    `Séquence` $\leftarrow \emptyset$ /* Initialiser la `Séquence` */
    pour $p \in \{0,1\}^{\mathbf{d}}$ **faire**
        /* Ajouter le premier point dans la `Séquence`, il n'y a pas de condition sur le premier point, un point quelconque peut être candidat */
        PUSH(`Séquence`, p) /* ajouter le point p à la fin de la `Séquence` */
        /* Établir des motifs via l'ajout consécutif de points dans la `Séquence` */
        PROLONGATIONDESÉQUENCE(`Séquence`) /* [*cf.* Algorithme 3] */
    fin pour
fin procédure

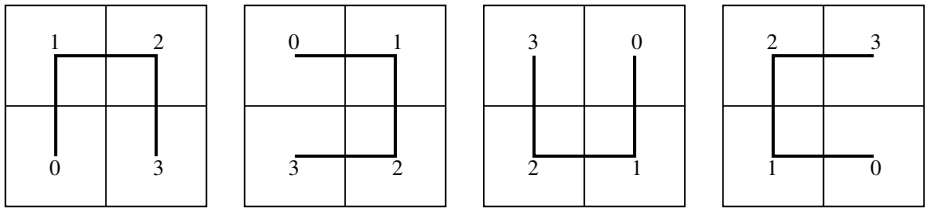


FIGURE 5.5 – Plusieurs façons de remplir un espace de dimension 2 : exemple de motifs primitifs résultant de l'application de la Définition 18 (on reconnaît la courbe RBG (a) utilisée dans la littérature).

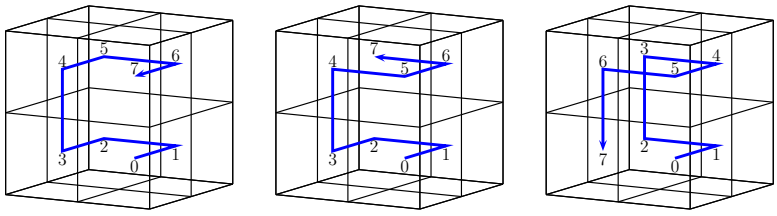


FIGURE 5.6 – Plusieurs façons de remplir un espace de dimension 3 : exemple de motifs primitifs résultant de l'application de la Définition 18 (Seule la courbe RBG (a) est utilisée dans la littérature).

de multiples directions. Cette souplesse autorise l'existence de plusieurs motifs primitifs solution, ouvrant le choix à d'autres motifs primitifs que RBG.

Notons que les motifs primitifs satisfont par construction la contrainte locale. Ils sont donc des candidats aux sous-courbes quel que soit l'ordre $\mathbf{n}$ *choisi.*

Algorithme 3 Établir des motifs primitifs via la prolongation de séquence.

```
1: /* Ajouter consécutivement dans la Séquence des points n'est pas passés.
      Un point ajouté doit être adjacent avec le dernier point de la Séquence ac-
      tuelle, la modification chaque coordonnée du dernier point de la Sé-
      quence produit des candidats. */
2: procédure ProlongationDeSéquence(Séquence, d)

3:     pour i ← 1 à d faire
4:         PointSuivant←dernier(Séquence) /* DERNIER(Séquence) le dernier
               point de la Séquence */
5:         /* Le point suivant : depuis le dernier point de la Séquence, modifier une
               coordonnée (la distance entre ce deux points est égale à 1) */
6:         PointSuivant[i] ← 1 − PointSuivant[i]

7:         /* le motif passe par chaque point une seule fois, si le point est dans la
               Séquence, il ne peut pas être rajouté dans la Séquence */
8:         si PointSuivant ∉ Séquence alors
9:             push(Séquence, PointSuivant)

10:            /* Une Séquence est un motif si elle contient tous les points de l'es-
                   pace {0, 1}^d */
11:            si Séquence est un motif alors /* Séquencecontient 2^d points */
12:                push(LesMotifs, Séquence)
13:            sinon

14:                /* Si la Séquence n'est pas encore un motif, essayer d'ajouter
                       un autre point */
15:                ProlongationDeSéquence(Séquence)
16:            fin si
17:        fin si
18:    fin pour
19: fin procédure
```

5.3.2 Le passage aux ordres supérieurs

Les développements suivants (Théorème 4 et Théorème 5) ont pour but de cerner (localement dans chaque sous-espace) les couples d'extrémités susceptibles de garantir la contrainte globale pour lesquels on est sûr qu'il existe une sous-courbe (motif primitif) réalisant leur connexion.

Autrement dit, est-il possible de construire une sous-courbe avec deux extrémités fixées ?

Théorème 3 (Distance impaire)**.** *Entre deux extrémités séparées par une distance impaire, il existe toujours au moins un motif primitif permettant de les connecter.*

Démonstration. Nous démontrons le théorème 3 par induction.

Quel que soit $\mathbf{d}$, il existe toujours au moins un motif primitif entre deux extrémités dont la distance est 1. Par exemple, la courbe RBG ([*cf.* Sous-section 5.3.1]).

Supposons que nous pouvons toujours construire un motif primitif entre deux extrémités dont la distance est impaire et vaut $2m-1$ ($\forall m \in \mathbb{N}, 2m-1 \geq 1, 2m+1 \leq d$).

Soient deux points $a = a_1a_2...a_d$ et $b = b_1b_2...b_d$ respectivement le point de départ et d'arrivée satisfaisant $\|a, b\| = 2m + 1$. Car $2m - 1 \geq 1 \Rightarrow \|a, b\| = 2m+1 \geq 3$, nous pouvons toujours trouver deux différentes positions i, j telles que $a_i \neq b_i$, $a_j \neq b_j$, $i < j$.

En examinant deux points :

$$\begin{aligned} a' &= a_0a_1...a_{i-1}\mathbf{a_i}a_{i+1}...\mathbf{a_{j-1}}\mathbf{a_{j+1}}...a_{\mathbf{d}-1} \\ b' &= b_0b_1...b_{i-1}\mathbf{a_i}b_{i+1}...\mathbf{b_{j-1}}\mathbf{b_{j+1}}...b_{\mathbf{d}-1} \end{aligned}$$

dans l'espace de dimension $(d-1)$, nous voyons que $\|a', b'\| = 2m - 1$, donc, nous pouvons construire un motif primitif $M = (a', ..., b')$ entre a' et b'. En ajoutant a_j à la position j de chaque point appartenant à M, M devient un autre parcours M' (en dimension $\mathbf{d}$) reliant a et $b'' = b_1b_2...b_{i-1}\mathbf{a_i}b_{i+1}...b_{j-1}\mathbf{a_j}b_{j+1}...b_d$; notons que la distance entre deux points consécutifs de M' est toujours égale à 1.

Examinons ensuite deux autres points en dimension $\mathbf{d}-1$:

$$\begin{aligned} b^{(3)} &= b_0b_1...b_{i-1}\mathbf{a_i}b_{i+1}...\mathbf{b_{j-1}}\mathbf{b_{j+1}}...b_{\mathbf{d}-1} \\ b^{(4)} &= b_0b_1...b_{i-1}\mathbf{b_i}b_{i+1}...\mathbf{b_{j-1}}\mathbf{b_{j+1}}...b_{\mathbf{d}-1} \end{aligned}$$

nous avons $d(b^{(3)}, b^{(4)}) = 1$.

Par conséquent, nous pouvons construire un motif primitif $N = (b^{(3)}, ..., b^{(4)})$ entre eux. En ajoutant b_j à la position j de chaque point appartenant à N, N devient un parcours N' en dimension $\mathbf{d}$ reliant $b^{(5)} = b_1b_2...b_{i-1}\mathbf{a_i}b_{i+1}...b_{j-1}\mathbf{b_j}b_{j+1}...b_d$ à b, la distance entre deux points consécutifs de N' est toujours égale à 1.

Tous les points appartenant à un parcours donné (M' ou bien N') sont deux à deux différents. De plus, parce que $a_j \neq b_j$, tous les points dans M' sont deux à deux différents avec ceux de N'. Par conséquent, tous les points dans les deux parcours sont deux à deux différents. On est donc sûr de ne pas passer deux fois au même endroit.

Puisque $d(b'', b^{(5)}) = 1$, la distance entre deux certains points consécutifs du parcours joint $M'N' = (a, ..., b'', b^{(5)}, ..., b)$ est toujours égale à 1.

M et N sont les motifs primitifs dans l'espace de dimension $(d-1)$. Ainsi, le nombre de points dans chaque motif primitif est 2^{d-1} et la totalité des points dans $M'N'$ est $2 * 2^{d-1} = 2^d$. Ainsi on est sûr que l'espace est totalement rempli.

En conclusion, $M'N'$ est un motif primitif dans l'espace de dimension $\mathbf{d}$.

Nous venons donc de démontrer qu'il est possible de construire un motif primitif entre deux points séparés par une distance impaire (et inférieure à $\mathbf{d}$). □

Théorème 4 (Distance paire)**.** *Entre deux extrémités séparées par une distance paire, il n'existe pas de motif primitif permettant de les connecter.*

Démonstration. Si la distance entre deux extrémités est paire, alors cela signifie que le nombre de modifications à appliquer sur les coordonnées du point de départ pour atteindre celles du point d'arrivée est pair. Autrement dit, relier deux points séparés par une distance paire passe nécessairement par la connexion d'un nombre impair de points intermédiaires. Comme le nombre total de points d'un motif primitif de dimension $\mathbf{d}$ est de $2^{\mathbf{d}}$ donc toujours pair cela implique un nombre de points intermédiaires pair $(2^{\mathbf{d}} - 2)$. Par conséquent, il ne peut pas exister de chemin connectant deux extrémités séparées par une distance paire et donc il n'existe pas de motif primitif entre eux. □

En conclusion, d'après les théorèmes 3 et 4, seules les extrémités séparées par une distance impaire (et inférieure à $\mathbf{d}$*) peuvent être connectées par un motif primitif.*

Théorème 5. *Avec le point de départ fixé, le nombre minimum de possibilités des points d'arrivées d'une sous-courbes est* $2^{\mathbf{d}-2}$

Démonstration. Pour le dernier sous-espace, parmi $2^{\mathbf{d}}$ points, nous avons $2^{\mathbf{d}-1}$ points ayant la distance impaire au point de départ. Tous ces $2^{\mathbf{d}-1}$ points sont candidats pour le point terminal de la courbe.

Pour les autres sous-espaces, le point d'arrivée d'une sous-courbe doit être adjacent au sous-espace suivant. L'hyperplan orthogonal à l'axe liant ces deux sous-courbes coupe le sous-espace actuel en deux, une moitié approchant le sous-espace suivant. Si les coordonnées des points sont codées en binaire, seule une coordonnée des points approchant le sous-espace suivant est identique, $d-1$ autres coordonnées sont différentes. Par conséquent, il existe 2^{d-1} points approchant le sous-espace suivant et donc candidats comme point d'arrivée.

Parmi les $d-1$ autres coordonnées, une fois les valeurs des $d-2$ coordonnées choisies, la valeur de la coordonnée restante fixe la distance au point de départ et donc sa parité. Par conséquent, il existe 2^{d-2} points approchant le sous-espace suivant séparés du point de départ par une distance impaire.

Parce que la distance d'un point à lui-même (égale à 0) n'est pas impaire, le nombre de possibilité pour définir un point d'arrivée (toujours séparé par une distance impaire du point de départ) est 2^{d-2}. □

5.4 Diversité des courbes et conservation de la localité

5.4.1 Diversité des courbes générées

Le Processus 2 - éclairé par les analyses menées dans les sections 5.3.1 et 5.3.2 - permet de cerner les diverses possibilités pour la construction d'une courbe multi-dimensionnelle :

— choix du motif primitif pour la courbe à l'ordre $\mathbf{n}$=1,
— choix des extrémités (au niveau de chaque sous-courbe),
— choix du parcours entre ces extrémités (définition des sous-courbes).

Un choix de ces trois paramètres est appelé une configuration.

Son exécution délivre une famille de courbes, dont 5 sont données à titre d'exemples dans la Figure 5.7. Chaque courbe correspond à une configuration de construction (notée T, BB, BS, CB, CS) *i.e.* à un choix d'un triplet parmi les degrés de libertés décrits ci-dessus. Ces configurations couvrent plusieurs cas :

— deux motifs primitifs sont utilisés : par exemple, la configuration T obéit au motif primitif (b) illustré dans la Figure 5.6, alors que les autres (BB,BS,CB,CS) obéissent au même motif primitif RBG (a).
— BB et CB partagent les mêmes points de départs et d'arrivées : les mêmes motifs primitifs mais diffèrent par au moins un choix d'extrémités.
— BB et BS partagent les mêmes points de départs et d'arrivées : les mêmes motifs primitifs, les mêmes extrémités mais diffèrent toujours par le choix du parcours entre ces extrémités.

Finalement, on observe [*cf.* Figure 5.7] des courbes ayant (ou pas) les mêmes points de départ et d'arrivée et décrivant un parcours de l'espace différent.

Dans la section suivante, le niveau de conservation de la localité de chacune de ces courbes est estimé puis comparé à celui de la courbe de Hilbert. Cette dernière est prise comme référence car il a été montré que parmi un ensemble de courbes remplissant l'espace (Peano, Lebesgue), elle possède le plus haut niveau de conservation de la localité [56, 60, 113].

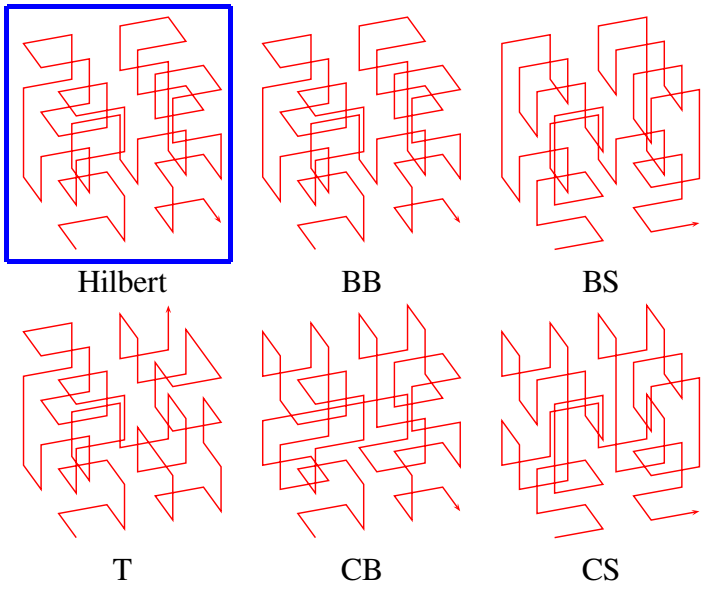


FIGURE 5.7 – Exemple de 5 courbes ($\mathbf{d} = 3, \mathbf{n} = 2$) résultant du Processus 2 et positionnement par rapport à la courbe de Hilbert (encadré). Chaque courbe correspond à une configuration de construction (triplet d'entrée). On remarque que la courbe construite selon la configuration BB, correspond à la courbe de Hilbert. La courbe de Hilbert est donc une sortie du Processus 2.

5.4.2 Mesure de la conservation de la localité des courbes générées

Plusieurs mesures ont été proposés dans la littérature [111, 128, 151] pour estimer la conservation de la localité d'une courbe remplissant l'espace. L'idée centrale est d'estimer si l'indexation (le calcul de l'index d'un point $\mathbf{d}$-dimensionnel sur une courbe donnée) est une opération conforme au sens de la conservation des distances inter-points dans l'espace original et dans l'espace des index).

Dans cette section, la conservation de la localité de chacune des courbes - décrites dans la Sous-section 5.4.1 - résultant du Processus 2, est estimée. La conservation de la localité moyenne de rayon 2, notée L, est estimée selon notre mesure proposée dans le Chapitre 3. Une ``bonne'' courbe maximise L. Le rayon 2 est choisi parce que la taille des espaces examinés est relativement petite. En plus de la mesure principale L, V est l'écart type de la conservation de la localité des points de la courbe. Une faible valeur de V montre l'uniformité de la conservation de la localité de la courbe.

Nous estimons aussi la conservation de la localité des courbes par deux autres mesures, notées M_1 et M_2. M_1 obéit à la définition de Perez et al. dans [128] et M_2 correspond à une version de M_1 établie dans le pire des cas (il s'agit de mesurer un

rapport maximal et non plus moyen).

La mesure M_1 quantifie la conservation moyenne de la localité des courbes :

$$M_1(C) = \frac{1}{N} \sum_{i,j \in R_C, i<j} \frac{|i-j|}{\mathfrak{d}(C(i), C(j))}$$

Avec : C une courbe remplissant l'espace multidimensionnel, R_C l'espace unidimensionnel (l'espace des index). i, j les index affectés aux points $C(i), C(j)$ sur la courbe C. N est le nombre de couples i, j et $\mathfrak{d}(C(i), C(j))$ la distance euclidienne entre $C(i)$ et $C(j)$.

La mesure M_2 quantifie la conservation de la localité des courbes dans le pire des cas :

$$M_2(C) = \max_{i,j \in R_C, i<j} \frac{|i-j|}{\mathfrak{d}(C(i), C(j))}$$

Une ``bonne'' courbe est une courbe qui minimise ces deux mesures.

Plusieurs courbes (30 au total) sont construites en dimension 3 et 4 balayant les ordres $\mathbf{n} = 2, 3, 4$ selon les 5 configurations (T, BB, BS, CB, CS) décrites dans la Sous-section 5.4.1. La courbe dite de Hilbert résulte de l'application de l'algorithme de Butz [29].

Discussion : À la lecture de ces résultats [*cf.* Tableau 5.1], dans le cadre de cette expérience, on observe que :

— Quel que soit $\mathbf{d}, \mathbf{n}$, la configuration BB correspond à la courbe de Hilbert. Il est donc logique de retrouver les mêmes estimations.

— Quel que soit $\mathbf{d}, \mathbf{n}$, le niveau de conservation de la localité moyen selon L et M_1 correspondant aux 5 configurations est assez proche de celui de la courbe de Hilbert. L'écart maximum relatif reste toujours inférieur à 0,14963% selon M_1 et 5,26% selon L, ce qui tend à confirmer nos hypothèses sur la conservation de la localité ([*cf.* Sous-section 5.1.2]).

— Il est possible de construire des courbes ayant un meilleur niveau de conservation de la localité que la courbe de Hilbert et cela quel que soit le critère (L, M_1 ou M_2) considéré (configuration CB en dimension 3 selon M_1, T en dimension 3 et 4 selon M_2 [*cf.* Tableau 5.1]. Bien que numériquement le gain moyen selon M_1 (engendré par CB) semble peu significatif (0,035%), ce n'est pas le cas selon M_2, où la configuration T, en dimension 3 et 4, procure un gain de conservation de la localité moyen (respectivement maximum) par rapport à la courbe de Hilbert de 24,358% (respectivement 27,118%).

— Notre mesure L caractérise le niveau de conservation de la localité de la courbe dépendamment au rayon choisi. À l'ordre 2 [*cf.* Tableau 5.1], le rayon 2 couvre bien l'espace. C'est pourquoi la conservation de la localité de toutes

Courbe		L	V	M_1	M_2
Hilbert		1.0	0.0	8.434138	59.0
Notre proposition	BB	1.0	0.0	8.434138	59.0
	BS	1.0	0.0	8.436178	61.0
	CB	1.0	0.0	**8.431279**	59.0
	CS	1.0	0.0	8.436178	61.0
	T	1.0	0.0	8.435158	**43.0**

$\mathbf{d} = 3, \mathbf{n} = 2$

Courbe		L	V	M_1	M_2
Hilbert		1.0	0.0	**28.667378**	247.0
Notre proposition	BB	1.0	0.0	**28.667378**	247.0
	BS	1.0	0.0	28.667798	253.0
	CB	1.0	0.0	28.679817	247.0
	CS	1.0	0.0	28.681465	253.0
	T	1.0	0.0	28.710273	**189.0**

$\mathbf{d} = 4, \mathbf{n} = 2$

Courbe		L	V	M_1	M_2
Hilbert		0.76	0.20	33.261915	471.0
Notre proposition	BB	0.76	0.20	33.261915	471.0
	BS	**0.76**	**0.19**	33.269824	493.0
	CB	0.72	0.20	**33.249778**	475.0
	CS	0.72	0.20	33.270267	493.0
	T	0.74	0.20	33.275373	**347.0**

$\mathbf{d} = 3, \mathbf{n} = 3$

Courbe		L	V	M_1	M_2
Hilbert		**0.68**	**0.21**	**226.085288**	3951.0
Notre proposition	BB	**0.68**	**0.21**	**226.085288**	3951.0
	BS	0.67	0.21	226.088161	4061.0
	CB	0.67	0.20	226.179165	3959.0
	CS	0.67	0.20	226.192087	4061.0
	T	0.67	0.21	226.421427	**3035.0**

$\mathbf{d} = 4, \mathbf{n} = 3$

Courbe		L	V	M_1	M_2
Hilbert		0.59	0.19	132.809301	3767.0
Notre proposition	BB	0.59	0.19	132.809301	3767.0
	BS	0.60	0.19	132.840889	3949.0
	CB	0.56	0.18	**132.760701**	3803.0
	CS	0.57	0.18	132.842745	3949.0
	T	**0.60**	**0.18**	133.149893	**2905.0**

$\mathbf{d} = 3, \mathbf{n} = 4$

Courbe		L	V	M_1	M_2
Hilbert		0.48	0.18	**1803.318995**	63215.0
Notre proposition	BB	0.48	0.18	**1803.318995**	63215.0
	BS	0.48	0.18	1803.341506	64989.0
	CB	**0.48**	**0.17**	1804.056001	63351.0
	CS	**0.48**	**0.17**	1804.158155	64989.0
	T	0.47	0.17	1805.982096	**48573.0**

$\mathbf{d} = 4, \mathbf{n} = 4$

TABLEAU 5.1 – Estimation de la conservation de la localité, selon 2 mesures plusieurs critères : M_1, M_2 issues de la littérature [128] et L, V proposées dans le Chapitre 3, des courbes générées par notre Processus 2 selon 5 configurations [*cf.* Sous-section 5.4.1] aux dimensions $\mathbf{d}$ = 3, 4 et aux ordres $\mathbf{n}$ = 2, 3, 4. Comparaison avec la courbe de Hilbert (Algorithme de Butz). Les chiffres en gras correspondent aux meilleurs niveaux de conservation de la localité.

les courbes est totale. Avec des espaces de taille plus grande, la conservation de la localité n'est plus totale. À l'ordre 3 [*cf.* Tableau 5.1], la courbe BS 3-dimensionnelle montre son meilleur niveau de conservation de la localité. Notons que les courbes générées par l'algorithme de Butz et la configuration BB possèdent la même conservation de la localité moyenne mais leur écart type est plus grand. De façon identique, la courbe qui conserve mieux la localité est aussi déterminée par l'index auxiliaire à l'ordre 4 [*cf.* Tableau 5.1]. Par exemple, au cas 3-dimensionnel, la courbe générée par la configuration T est celle qui conserve mieux la localité malgré qu'elle a le même niveau de conservation de la localité que la courbe générée par la configuration BB.

5.5 Courbes et applications

En confrontant avec la définition de la courbe remplissant l'espace, cette généralisation de la courbe de Hilbert ne vérifie pas l'autosimilitude. Cette contrainte est remplacée par celui plus souple : l'héritage. En effet, pour optimiser la conservation de la localité, ce relâchement des contraintes est nécessaire car il permet d'exploiter toutes les possibilités d'ordonner les points d'un sous-espaces en tout vérifiant l'adjacence ($C3$).

Ainsi, les sous-courbes ne sont pas obligatoirement similaires [*cf.* Définition 11]. Par conséquent, la courbe dépend éventuellement de plusieurs motifs primitifs et leur place mémoire est important dans le cas de grande dimension. Dans ce cas, nous revenons à l'autosimitude. Une courbe Hilbert autosimilaire est un ordonnancement vérifiant à la fois l'*adjacence* [*cf.* Définition 17] et l'*autosimilitude* [*cf.* Définition 12].

Dans notre application [*cf.* Chapitre 6], nous choisissons la courbe Hilbert autosimilaire qui maximise la conservation de la localité selon notre mesure [*cf.* Chapitre 3]. Comme la courbe est autosimilaire, nous pouvons accélérer le calcul de l'index par l'application simplifiée de l'isométrie décrite dans Sous-section 4.3.2. Ainsi, la courbe solution possède deux avantages :

— un bon niveau de conservation de la localité,
— le calcul de l'index rapide.

5.6 Conclusion

Dans cette chapitre, nous avons proposé une généralisation multidimensionnelle de la courbe de Hilbert. Des éléments de preuve montrent que le résultat de cette généralisation est une classe des courbes qui conservent bien la localité : Cela revient à dire qu'il existe désormais plusieurs courbes qui généralisent la courbe de Hilbert, le nombre de courbes augmentant avec la dimension de l'espace.

La diversité des courbes crée la possibilité d'optimisation sur le niveau de la conservation de la localité. Plusieurs tests comparatifs ont été menés. Ces tests montrent que la conservation de la localité des courbes généralisées est comparable à celle de la courbe multidimensionnelle dite de Hilbert générée par l'algorithme de Butz ou de Bially.

Au-delà de la conservation de la localité, la facilité de construction et de calcul de l'index est aussi un facteur important dans la perspective d'une application réelle. Après d'avoir analysé la structure des courbes de Hilbert, nous proposons les processus et les algorithmes qui permettent de générer le motif, de déterminer les extrémités, qui sont les éléments principaux de la constructions de la courbe.

Pour les applications de grande dimension, nous proposons la courbe de Hilbert autosimilaire, qui optimise le calcul de l'index. Un seul motif primitif est alors

utilisé. Elle ne demande que peu d'espace pour sauvegarder les informations nécessaires pour la construction et le calcul de l'index. Le motif et les extrémités des sous-courbes sont choisis pour que la courbe maximise la conservation de la localité.

Chapitre 6

Application

6.1 Objectifs et idées centrales

Les Chapitres 4 et 5 s'inscrivent dans le cadre :

1. de la recherche de manières alternatives - aux courbes de Hilbert, Peano, Lebesgue - afin de remplir un espace, tout en maintenant un bon niveau de localité (mesurés Sous-section 5.4.2).
2. de développement de schémas d'implémentation restant opérationnels au-delà de quelques dimensions, avec pour toile de fond le calcul de l'index des points sur le long de la courbe considérée (Sous-section 4.3.2 et Section 5.3).

Ces directions offrent des perspectives en termes d'applications. Dans le domaine du traitement des images, des travaux récents s'appuient sur la construction de courbes remplissant l'espace à des fins de caractérisation [129] ou de compression [88]. Néanmoins ces modèles opèrent dans de faibles dimensions (2,3). Construire des courbes remplissant l'espace de dimensions supérieures pourrait permettre d'aborder d'autres problèmes, par exemple pour la recherche d'images dans de grandes bases. C'est la piste que nous explorons, dans ce chapitre. En effet, dans le cadre de la recherche des images par le contenu (CBIR), les images sont souvent caractérisées par des vecteurs de grandes dimensions (description numérique).

Pour accéder à de telles dimensions, ou tout du moins à des dimensions gravitant autour de 30, il est nécessaire de se pencher sur l'optimisation de nos algorithmes. C'est pourquoi, une version légère (en termes d'occupation mémoire) de l'algorithme dédié au calcul de l'index est proposée. Cette version, exploite le gain algorithmique issu du calcul d'une composition de transformations, et s'intitule *méthode en ligne*.

Idées centrales guidant l'application Dans le cadre de cette application, une image est représentée classiquement par diverses caractéristiques (de contours, régions) encapsulées dans un vecteur de $\mathbf{d}$ dimensions, dit vecteur de caractéristiques

feature vector. Une collection d'images est synthétisée par un ensemble de points **d**-dimensionnel évoluant dans l'espace des caractéristiques.

Indexation : Dans notre application, l'indexation fait correspondre simplement chaque image décrite par un descripteur multidimensionnel à son index sur la courbe remplissant l'espace choisie. Le calcul de l'index de ces points sur une courbe remplissant l'espace (des caractéristiques) est un moyen d'ordonner le long d'une ligne (espace des index) les points (les images) en fonction de leur localisation dans l'espace [*cf.* Figure 6.1]. L'ordre ainsi obtenu contribue à réduire la complexité algorithmique de la recherche d'une image requête, ce qui confère à notre technique la propriété de rapidité.

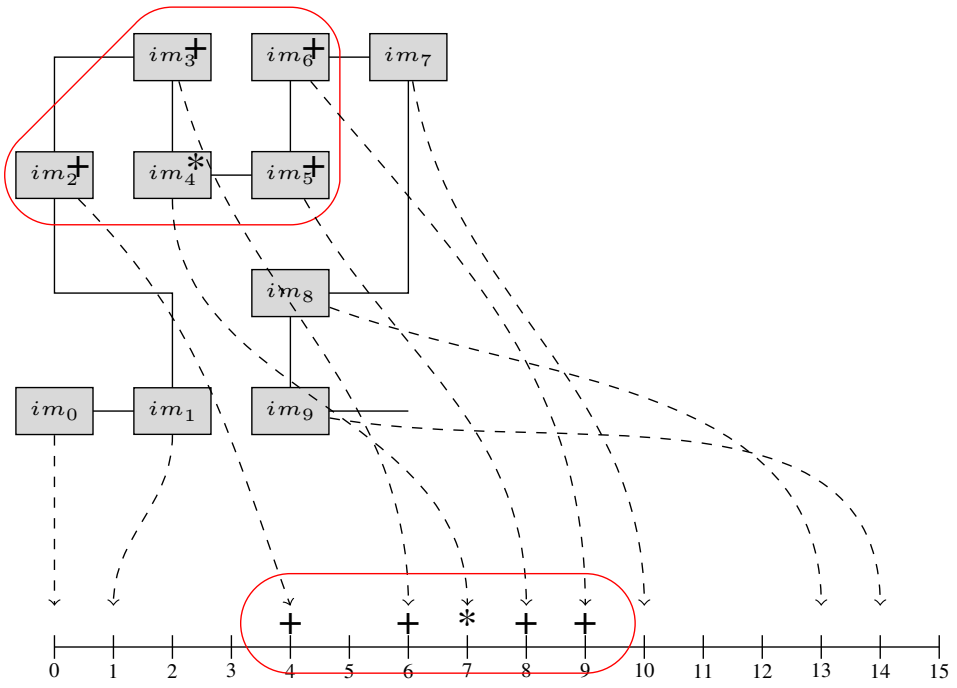


FIGURE 6.1 – Illustration de l'indexation des images et la recherche. Chaque image est associée à un index, l'indexation consiste à ordonner les images selon leur index. Les images similaires (+) sont celles qui ont les index proches de l'index de l'image de l'entrée (*).

La conservation de la localité permet de définir combien de points proches sur la ligne (espace des index) [*cf.* Figure 6.1] correspondent à des images aux caractéristiques effectivement proches (espace des caractéristiques- métrique donnée). La localité peut donc inférer sur les résultats de la recherche et donc sur les perfor-

mances notamment en termes de précision. C'est la raison pour laquelle, des tests comparatifs sont effectués lorsque l'espace des caractéristiques est rempli :

— par la courbe présentant le plus haut niveau de localité (issue de notre proposition, l'Algorithme 5),
— par la courbe dite de Hilbert (Algorithme de Butz), prise comme référence.

Recherche : Dans la phase de recherche, les images ayant les index les plus proches de l'index de l'image entrée vont être sélectionnées comme résultats. En effet, c'est une recherche dans laquelle les sorties ne sont pas obligatoirement les plus proches voisins de la requête mais une partie de cet ensemble. Réellement, pour les grandes bases d'image, la requête est souvent sous forme *``S'il y a des images similaires à cette image ?''*. Pour cela, on exige des réponses rapides et pertinentes plutôt que les meilleures réponses. Autrement dit, c'est une recherche rapide qui permet de trouver des images similaires à l'image entrée, ces résultats ne sont pas obligatoirement les plus proches de l'image entrée.

Il s'agit de tests à grande échelle (19270 images), menés sur une base résultant de l'union de bases de référence (GREC, MPEG, LEMS [*cf.* Section 6.3.2]). Notons que ces tests :

— mettent en jeu des images variées (graphiques, naturelles) , binaires, de taille réduite (300×300), ou ayant subis quelques dégradations (bruits),
— portent sur plusieurs phases, allant de l'indexation d'une image d'entrée à sa recherche, en passant par la mise à jour de la base.

L'utilité des courbes remplissant l'espace est démontrée via un système de recherche d'images avec 3 caractéristiques marquantes :

— indexation rapide,
— mise à jour flexible et rapide,
— réponse rapide.

Contenu du chapitre La section suivante [*cf.* Section 6.2] présente des méthodes de calcul de l'index : *hors ligne* et *en ligne*.

Ensuite, l'application de recherche d'image est présentée. L'indexation, la recherche détaillées ainsi que les évaluations, l'amélioration des résultats est présentées dans la Section 6.3.

6.2 Le calcul de l'index

6.2.1 Algorithme hors ligne

Comme nous avons précisé dans la Sous-section 4.2.2, une courbe remplissant l'espace peut être définie par une liste de points ordonnés selon leurs index. Cette

méthode est appelée l'énumération des points. La génération de la liste des points peut également s'appuyer sur la construction des courbes présentée dans Chapitre 4 et Chapitre 5.

En s'appuyant sur l'énumération des points, nous pouvons calculer facilement l'index d'un point en précisant son ordre dans la liste. Ainsi, la méthode *hors ligne* est simplement le positionnement du point de l'entrée dans une liste ordonnée des points de la courbe remplissant l'espace. L'Algorithme 4 décrit cette simple méthode de calcul de l'index.

Algorithme 4 Indexation hors ligne.

```
1: procédure IndexHorsLigne(Point, Liste)
2:     /* Chercher le point dans la liste d'énumération par d'examination consécutive des points, commencer au début de la liste */
3:     Index←1
4:     tant que Liste[Index] ≠ Point faire
5:         Index←Index+ 1
6:     fin tant que
7:     retourner Index
8: fin procédure
```

Malgré le fait que l'énumération des points est la façon la plus simple pour définir une courbe remplissant l'espace, elle peut se heurter à des limites pratiques, par exemple pour le cas de dimensions et d'ordres élevés. Effectivement, le nombre de points d'un espace augmente exponetiellement avec la dimension et sa résolution. Par exemple, la courbe de Hilbert $\mathbf{d}$-dimensionnelle d'ordre $\mathbf{n}$ remplit un espace contenant $2^{\mathbf{d}\times\mathbf{n}}$ points. Avec $\mathbf{d}$ et $\mathbf{n}$ grands, une énumération de tous ces points occupe un espace mémoire très important. Pour $\mathbf{d} = 10$ et $\mathbf{n} = 5$, cette l'énumération occupe $2^{50} \times$ *taille d'un point*. Chaque point occupe $10 \times 5 = 50$ bits équivalant 7 octets. Par conséquent, l'énumération occupe au moins 7×2^{50} octets, ce qui est supérieur à 7000 téraoctets.

Même dans le cas où nous arrivons à énumérer les points d'un grand espace, nous nous heurtons à un autre problème : la complexité de la recherche des points dans la liste. Dans notre test, la recherche du point $(11, 2, 8, 27, 19, 7)$ dans la liste des points de la courbe de Hilbert 6-dimensionnelle d'ordre 5 (générée par l'algorithme de Butz) prend 4,6 minutes. Ce temps de traitement ne permet pas d'appliquer la courbe dans les applications réelles.

En effet, dans les applications, la dimension de l'espace peut être de l'ordre de 100, voire 1000. L'ordre de la courbe doivent être assez grand pour bien distinguer des éléments de l'espace. Dans notre application, nous appliquons des courbes de

dimension 30 à l'ordre 8. L'énumération des points de ces courbes est impossible avec notre équipement informatique [*cf.* Annexe B].

En conclusion, l'algorithme hors ligne n'est envisageable que pour des courbes qui remplissent un espace de dimension et d'ordre faibles. Un algorithme en ligne, qui permet de retrouver l'index d'un point de la courbe remplissant l'espace sans énumérer tous les points de la courbes, est recommandé pour les autres cas. Dans la partie suivant, les éléments pour cet algorithme sont précisés.

6.2.2 L'algorithme en ligne : principe

L'algorithme *en ligne* calcule l'index d'un point sur une courbe remplissant l'espace définie par Définition 13 sans l'énumération des points de cette courbe.

La décomposition d'un index Selon Définition 12, une courbe remplissant l'espace $\mathcal{C}$ d'ordre $\mathbf{n}$ hérite d'une courbe remplissant l'espace $\mathcal{C}$' d'ordre $\mathbf{n}$-1 (une courbe d'ordre 1 hérite d'un point que nous considérons comme la courbe d'ordre 0). En conséquence, l'index d'un point p sur $\mathcal{C}$ peut être décomposé en :

$$\mathcal{C}(p) = \mathcal{C}'(p')\mathbf{b}^{\mathbf{d}} + t$$

où p' est le point correspondant au sous-espace $\mathcal{S}$ contient le point p et t est l'ordre de p à l'intérieur du $\mathcal{S}$, $\mathbf{b}$ et $\mathbf{d}$ sont consécutivement la base de $\mathcal{C}$ et la dimension de l'espace. Nous appelons p' le *père* de p parce que p résulte de la division de p'.

Ainsi, le calcul de l'index de p est fait si nous pouvons calculer l'index de p' sur $\mathcal{C}$' et t. En effet,

— le calcul de t est présenté ci-dessus,

— comme $\mathcal{C}$' est aussi une courbe remplissant l'espace [*cf.* Définition 12], l'index de p' peut être également calculé par cet algorithme (récursivité).

Le calcul de t : Supposons que f est l'isométrie qui transforme le motif $\mathfrak{m}$ en la sous-courbe remplissant $\mathcal{S}$.

Pour calculer t, nous devons d'abord déterminer la position relative p'' de p à l'intérieur $\mathcal{S}$, qui représente la position de p si l'origine de l'espace est fixée à l'origine de $\mathcal{S}$. Comme la résolution de chaque sous-espace est $\mathbf{b}$, la position relative peut être calculée comme suit :

$$p'' = p \bmod \mathbf{b} = (p_0 \bmod \mathbf{b}, p_1 \bmod \mathbf{b}, \dots, p_{\mathbf{d}-1} \bmod \mathbf{b})$$

t est désormais l'ordre de $f^{-1}(p'')$ sur le motif $\mathfrak{m}$: $t = \mathfrak{m}(f^{-1}(p''))$. Ainsi, la décomposition de l'index est réécrite comme suit :

$$\mathcal{C}(p) = \mathcal{C}'(p')\mathbf{b}^{\mathbf{d}} + \mathfrak{m}(f^{-1}(p'')) \tag{6.1}$$

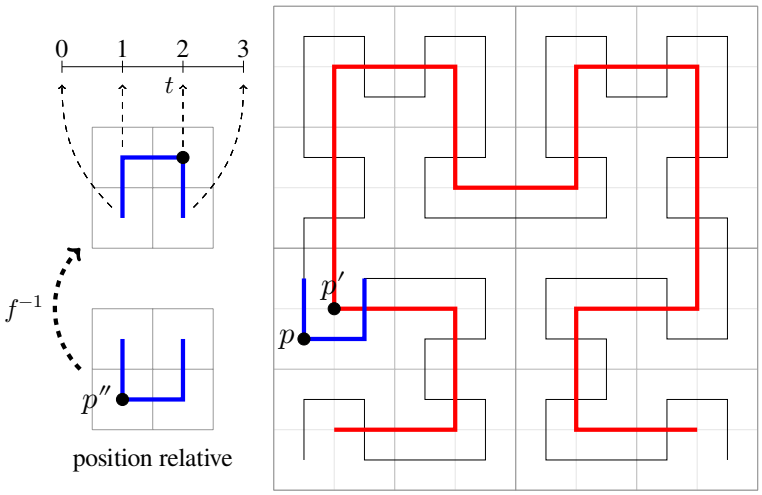


FIGURE 6.2 – Illustration de la décomposition de l'index $\mathcal{C}(p)$ du point p : $\mathcal{C}(p) = \mathcal{C}'(p')\mathbf{b}^{\mathbf{d}} + t$ (avec $\mathbf{b}$=2, $\mathbf{d}$=2 dans cette figure). $\mathcal{C}$ hérite de $\mathcal{C}$' (rouge) et p' est le point sur $\mathcal{C}$' correspondant au sous-espace $\mathcal{S}$ qui contient p. t est l'ordre du p sur la sous-courbe remplissant $\mathcal{S}$ (bleu), t peut être calculé par l'intermédiaire de la position relative p'' et la référence de l'ordre du point équivalent sur le motif via l'isométrie f.

6.2.3 L'algorithme en ligne : la répétition d'isométries

En appliquant la décomposition de l'index ci-dessus, nous devons mémoriser toutes les isométries appliquées aux sous-courbes. Notons que dans une courbe d'ordre $\mathbf{n}$, il y a $1 + \mathbf{b}^{\mathbf{d}} + \mathbf{b}^{\mathbf{2d}} + \cdots + \mathbf{b}^{(\mathbf{n}-1)\mathbf{d}}$ sous-courbes. Malgré que l'espace mémoire pour sauvegarder ces isométries est réduit par rapport l'espace de mémoire pour l'énumération de points, il reste encore important.

Pour cette raison, nous proposons de répéter une configuration simple des isométries qui maximise la conservation de la localité suivant le Processus 3.

La Figure 6.3 montre l'application d'une configuration dans la division des points.

— *Construction de configuration* : déterminer les isométries pour la division du motif qui optimisent la conservation de la localité, chaque index i sur le motif $\mathfrak{m}$ est mis en correspondance avec une isométrie $\mathfrak{Iso}(i)$,
— *Application de configuration* : l'isométrie appliquée pour la division d'un point p est déterminée comme suit :
  — déterminer l'ordre t de p dans sa sous-courbe,
  — l'isométrie appliquée pour la sous-courbe divisant p est $f(\mathfrak{Iso}(t))$, où f est l'isométrie qui transforme le motif $\mathfrak{m}$ en sous-courbe contenant p.

PROCESSUS 3: Construction et application d'une configuration

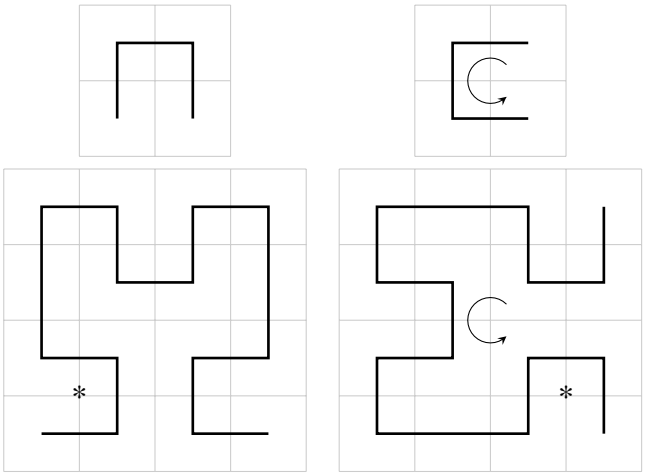


FIGURE 6.3 – Illustration de l'application d'une configuration *(a)* dans la division des points d'une courbe d'ordre 1 *(b, haute)* pour créer une courbe d'ordre 2 *(b, base)*. La configuration montre la division des points *(a, base)* du motif *(a, haute)*. Pour diviser les points d'une sous-courbe *(b)*, nous devons considérer l'isométrie f qui transforme *(a)* en *(b)*. La division du premier point *(b, *)* est le résultat de la composition de f et la première isométrie *(a, *)* de la configuration.

Déterminer des positions relatives : l'écriture des coordonnées en base $\mathbf{b}$

Selon Équation 6.1, pour calculer l'index d'un point p, nous devons déterminer sa position relative p'' et son père p'. À son tour, le calcul de l'index de p' a besoin de sa position relative. De la même façon, nous devons calculer la position relative de tous ses ascendants. Nous notons $p^{(k)}$ la position relative de l'ascendant de p sur la courbe d'ordre k.

En effet, toutes ces positions relatives peuvent être extraites depuis l'écriture en base $\mathbf{b}$ des coordonnées de p. Notons qu'un nombre k est écrit $(k^{(\mathbf{n})}k^{(\mathbf{n}-1)} \dots k^{(1)}k^{(0)})_{\mathbf{b}}$

en base $\mathbf{b}$ si

$$k = k^{(\mathbf{n})}\mathbf{b}^{\mathbf{n}} + k^{(\mathbf{n}-1)}\mathbf{b}^{\mathbf{n}-1} + \cdots + k^{(1)}\mathbf{b} + k^{(0)}$$

où $k^{(i)} \in \{0, 1, \ldots, \mathbf{b}\} \ \ \forall i \in \{0, 1, \ldots, \mathbf{n}\}$

Supposons que chaque coordonnée de p est écrite en base $\mathbf{b}$ sous forme $p_i = (p_i^{(\mathbf{n})} p_i^{(\mathbf{n}-1)} \ldots p_i^{(1)} p_i^{(0)})_{\mathbf{b}} \ \ \forall i \in \{0, 1, \ldots, \mathbf{d}-1\}$, chaque position relative est extraite comme suit :

$$p^{(k)} = (p_0^{(k)}, p_1^{(k)}, \ldots, p_{\mathbf{d}-1}^{(k)})$$

La décomposition des transformations

Nous réécrivons désormais Équation 6.1 comme suit :

$$\mathcal{C}(p) = \mathfrak{m}(f_1^{-1}(p^1))\mathbf{b}^{(\mathbf{n}-1)\mathbf{d}} + \mathfrak{m}(f_2^{-1}(p^{(\mathbf{n}-2)}))\mathbf{b}^{(\mathbf{n}-2)\mathbf{d}} + \ldots + \mathfrak{m}({f_{\mathbf{n}}}^{-1}(p^{(\mathbf{n})}))$$

Ainsi, il ne reste que les isométries $f_i \ \ \forall i \in 1, 2, \cdots, \mathbf{n}$ à déterminer. f_1 est déterminée par la configuration [*cf.* Processus 3]. Pour les autres isométries, selon Processus 3, $f_i = f_{i-1} \circ \mathfrak{Iso}(\mathfrak{m}(f_{i-1}^{-1}(p^{(i-1)}))) \ \ \forall i \in 2, 3, \cdots, \mathbf{n}$ [*cf.* Figure 6.4].

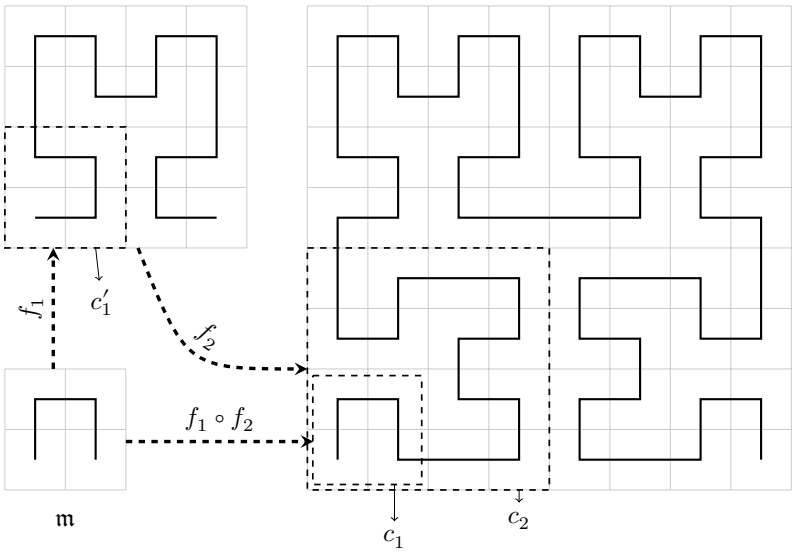


FIGURE 6.4 – Illustration la décomposition d'une isométrie : l'isométrie qui transforme $\mathfrak{m}$ en la sous-courbe c_1 est composée de f_1 qui transforme $\mathfrak{m}$ en c_1' (la mère de c_1) et f_2 qui transforme c_1' en c_1.

En conclusion, l'index d'un point peut être calculé sans aucune construction de courbe ou sous-courbe. Il résulte directement du motif via une composition des fonctions.

Algorithme en ligne

En s'appuyant sur ces analyses, nous proposons l'Algorithme 5 pour indexer des points.

Simplification d'une composition des transformations

Comme nous l'avons souligné lors de la discussion dans la Sous-section 4.3.2, une isométrie peut être simplifiée en une permutation des coordonnées et une réflexion, ceci afin de permettre de faciliter l'implémentation informatique de ces transformations. Dans l'Algorithme 5, il est nécessaire en plus de réaliser la composition de ces fonctions. Dans cette section, nous montrons la façon de déterminer la permutation des coordonnées et la réflexion d'une composition de deux isométries.

Considérons une composition des isométries $f = f_1 \circ f_2$. Supposons qu'elles sont décomposées en la permutation des coordonnées et la réflexion comme suit : $f = \mathfrak{Ref} \circ \mathfrak{p}$, $f_1 = \mathfrak{Ref}_1 \circ \mathfrak{p}_1$ et $f_2 = \mathfrak{Ref}_2 \circ \mathfrak{p}_2$ avec g_1, g_2 sont consécutivement les permutations de $\mathfrak{p}_1, \mathfrak{p}_2$ et $\mathfrak{Ref}_1, \mathfrak{Ref}_2$ sont les réflexions par rapport les axes dans A_1, A_2.

Examinons un point $x = (x_0, x_1, \dots, x_{\mathbf{d}-1})$ quelconque, nous avons $f(x) = f_1(f_2(x))$.

Supposons
$y = f_2(x) = \mathfrak{Ref}_2(\mathfrak{p}_2(x)) = \mathfrak{Ref}_2(x_{g_2(0)}, x_{g_2(1)}, \dots, x_{g_2(\mathbf{d}-1)})$
Avec $y = (y_0, y_1, \dots, y_{\mathbf{d}-1})$, nous avons

$$\begin{cases} y_i = \text{b - 1 - } x_{g_2(i)} & \text{si} \quad i \in A_2 \\ y_i = \quad\quad x_{g_2(i)} & \text{si} \quad i \notin A_2 \end{cases} \quad \forall i \in \{0, 1, \dots, \mathbf{d}-1\} \tag{6.2}$$

Supposons en suite,
$z = f_1(f_2(x)) = f_1(y) = \mathfrak{Ref}_1(\mathfrak{p}_1(y)) = \mathfrak{Ref}_1(y_{g_1(0)}, y_{g_1(1)}, \dots, y_{g_1(\mathbf{d}-1)})$
Avec $z = (z_0, z_1, \dots, z_{\mathbf{d}-1})$, nous avons

$$\begin{cases} z_j = \text{b - 1 - } y_{g_1(j)} & \text{si} \quad j \in A_1 \\ z_j = \quad\quad y_{g_1(j)} & \text{si} \quad j \notin A_1 \end{cases} \quad \forall j \in \{0, 1, \dots, \mathbf{d}-1\}$$

Remplaçons i par $g_1(j)$ dans Équation 6.2, nous avons

$$\begin{cases} y_{g_1(j)} = \text{b - 1 - } x_{g_2(g_1(j))} & \text{si} \quad g_1(j) \in A_2 \\ y_{g_1(j)} = \quad\quad x_{g_2(g_1(j))} & \text{si} \quad g_1(j) \notin A_2 \end{cases} \quad \forall i \in \{0, 1, \dots, \mathbf{d}-1\}$$

Enfin, nous avons,

Algorithme 5 Indexation en ligne.

procédure INDEXENLIGNE(`Point`, `Ordre`)

/* Initiation : f transforme le motif $\mathfrak{m}$ en la courbe d'ordre 1, nous utilisons l'identité mais une isométrie quelconque peut être utilisée */
$f \leftarrow$ `Identité`

/* Écriture du `Point` en base $\mathbf{b}$ */
/* `Point` $= \left[(p_0^{(n)}, p_1^{(n)}, \dots, p_{\mathbf{d}-1}^{(n)})^{(n)}(p_0^{(n-1)}, p_1^{(n-1)}, \dots, p_{\mathbf{d}-1}^{(n-1)})^{(n-1)} \dots (p_0^{(0)}, p_1^{(0)}, \dots, p_{\mathbf{d}-1}^{(0)})^{(0)}\right]_{\mathbf{b}} = \left(\texttt{Point}^{(n)}\texttt{Point}^{(n-1)} \dots \texttt{Point}^{(0)}\right)_{\mathbf{b}}$ */

pour $i \leftarrow n$ **à** 0 **faire**
  /* Déterminer le point sur le motif $\mathfrak{m}$ correspondant avec la position relative actuellement examinée. Comme f transforme les points sur le motif en les positions relatives sur la sous-courbe actuelle, f^{-1} fait l'inverse, elle transforme les positions relatives sur la sous-courbe actuelle en les points correspondants sur le motif */
  $m \leftarrow f^{-1}(\texttt{Point}^{(i)})$

  /* Déterminer l'index du correspondant dans le motif, il est aussi l'index de la position relative examinée dans la sous-courbe actuelle */
  $\texttt{Index}^{(i)} \leftarrow$ INDEXSURMOTIF(m)

  /* Mettre à jour l'isométrie de la sous-courbe qui remplace le point actuelle */
  $g \leftarrow \texttt{Isométrie}[\texttt{Index}^{(i)}]$
  $f \leftarrow f \circ g$
fin pour

/* Notons que le motif contient $\mathbf{b}^{\mathbf{d}}$ points, l'index est donc écrit comme suivant */
`Index` $\leftarrow \left(\texttt{Index}^{(n)}\texttt{Index}^{(n-1)} \dots \texttt{Index}^{(0)}\right)_{\mathbf{b}^{\mathbf{d}}}$
retourner `Index`
fin procédure

$$\begin{cases} z_j = b-1-(b-1-x_{g_2(g_1(j))}) & \text{si} \quad j \in A_1, g_1(j) \in A_2 \\ z_j = b-1-(\quad x_{g_2(g_1(j))}) & \text{si} \quad j \in A_1, g_1(j) \notin A_2 \\ z_j = \quad (b-1-x_{g_2(g_1(j))}) & \text{si} \quad j \notin A_1, g_1(j) \in A_2 \\ z_j = \quad (\quad x_{g_2(g_1(j))}) & \text{si} \quad j \notin A_1, g_1(j) \notin A_2 \end{cases}$$

$\forall i, j \in \{0, 1, ..., \mathbf{d}-1\}$
Simplifions cette formule, nous avons :

$$\begin{cases} z_j = b-1-x_{g_2(g_1(j))} & \text{si } j \in A_1, g_1(j) \notin A_2 \text{ ou } j \notin A_1, g_1(j) \in A_2 \\ z_j = \quad x_{g_2(g_1(j))} & \text{sinon} \end{cases}$$

$\forall j \in \{0, 1, ..., \mathbf{d}-1\}$
Par conséquent, nous pouvons décomposer $f = \mathfrak{Ref} \circ \mathfrak{p}$ avec
— $\mathfrak{Ref}$ est la réflexion par rapport les axes dans

$$A = \{j \in A_1 : g_1(j) \notin A_2\} \cup \{j \notin A_1 : g_1(j) \in A_2\}$$

— $\mathfrak{p}$ est la permutation des coordonnées selon la permutation $g = g_2 \circ g_1$

6.3 Application à la recherche d'images dans une grande base

6.3.1 La recherche d'images par leur contenu

La nouvelle tendance

La recherche d'image par le contenu est un principe de recherche d'images qui est utilisé par un grand nombre d'études [45,143] ou de services connus sur l'Internet comme Google Images ou Facebook pour la détection faciale.

En opposition à la recherche d'images par mots clés, la recherche d'images par le contenu est une approche visuelle. La requête est une image et la réponse est une liste d'images visuellement similaires à la requête.

L'avantage principal de la recherche d'images par le contenu est la logique naturelle de l'approche. Elle se rapproche de la fonction du système de vision humaine. De plus, elle contient potentiellement deux avantages importants :

— La description intégrale de l'image. L'image est elle-même une description visuelle d'une partie du monde. Une description autre que la description visuelle, comme les mots clés, ne la représente que partiellement. En décrivant les images par les signatures visuelles, la recherche d'images par le contenu permet d'une description plus complète de l'image.

— Automatisation de l'indexation. En évitant de représenter l'image par les mots clés, qui s'accompagne d'un coût humain important, la recherche d'images par le contenu diminue l'intervention de l'homme dans l'indexation des images.

Hautes-dimensions

Dans un système de recherche, les images similaires à une image requête sont essentiellement retrouvées par la comparaison sur le niveau de similarité : les sorties sont les images ayant les meilleurs niveaux de similarité par rapport l'image requête. Malgré que les images sont déjà sous forme numérique, la comparaison directe (pixel par pixel) n'a pas normalement de sens. Pour que les images soient comparables, on extrait une caractérisation de leur contenu. Le contenu de l'image peut être décrit par plusieurs aspects. Dans les recherches récentes, la couleur, la forme et la texture sont souvent utilisées pour décrire les images. Ils restent des descripteurs de bas niveau.

L'extraction du contenu est le travail principal de la recherche d'image par le contenu. Il existe plusieurs méthodes pour chaque type de contenu. Par exemple,

— la couleur peut être décrite dans les espaces de couleur différents comme RGB ou HSV [132],

— une forme peut être représentée par sa région ou son contour [132].

Le descripteur du contenu extrait est sous forme un vecteur numérique et, avec une normalisation parfois requise, il permet la comparaison pour mesurer la similarité des images. Pour représenter l'abondance du contenu de l'image, la dimension du vecteur est grande, même pour décrire chaque aspect du contenu. Par exemple, la couleur d'une image peut être représentée par un histogramme, qui est un vecteur multi-dimensionnel (166-d histogramme utilisé dans [144]). Le descripteur connu SIFT est un autre exemple. Ce descripteur est de 128 dimensions. Dans un système de recherche complet, plusieurs aspects du contenu sont associés.

L'augmentation de la dimension du descripteur permet de mieux décrire le contenu de l'image mais elle cause des problèmes pour l'indexation. Les méthodes d'indexation multi-dimensionnelle, par exemple, l'arbre k-d, l'arbre R et ses successeurs comme l'arbre R* et l'arbre R+ [76], ne répondent pas de façon pertinente avec les bases de données haute-dimensionnelles.

Dans une autre approche, nous pouvons utiliser des méthodes de réduction de la dimension comme ACP (Analyse en composantes principales) [57] mais ces techniques demandent une quantité importante de ressources (en temps et en espace mémoire).

Intérêt du recours à une courbe remplissant l'espace

Pour résoudre le problème de la dimension, nous faisons appel à une courbe remplissant l'espace. Une telle courbe, en faisant correspondre des points multi-

dimensionnels (espace réel) avec leurs index mono-dimensionnelles (espace des index), permet de convertir la recherche d'images en une recherche mono-dimensionnelle. Concrètement, le descripteur multi-dimensionnel de chaque image est en correspondance avec un index. La recherche est alors formulée par la recherche d'un index dans un ensemble d'index.

La correspondance entre un descripteur multi-dimensionnel et son index monodimensionnel peut être considérée comme une réduction de dimension. Comme toutes les réductions de dimension, l'index monodimensionnel ne conserve pas totalement les relations spatiales ou le voisinage des points multi-dimensionnels. Cependant, avec la conservation de la localité, les courbes remplissant l'espace optimisent la conservation du voisinage (versus les courbes Scan, Sweep et autres [*cf.* Sous-section 1.2.3]).

6.3.2 Expérimentations : Base d'images, descripteur, système de recherche

Les bases d'images considérées

Dans l'expérimentation de la performance des courbes remplissant l'espace, nous les testons dans le cadre de la recherche des images de formes. Les images sont collectées depuis 3 sources connues :

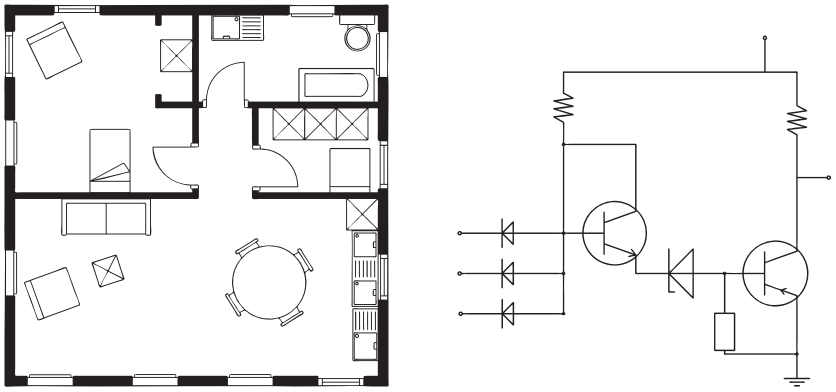
— *GREC* [1] Les images du concours [2] sur la reconnaissance de forme du GREC (IAPR [3] international workshop on graphics recognition).
— *MPEG* [4] Les images de forme d'expérimentation de la norme MPEG-7 [5]
— *LEMS* [6] La collection de forme du laboratoire LEMS [7]

GREC est une collection des symboles de dessins techniques de 2 domaines différents : l'architecture et l'électronique [*cf.* Figure 6.5]. Ces symboles ne contiennent que 2 types primitifs : les lignes droites et les arcs. Il n'y a pas de régions remplies, de

1. http://iapr-tc10.univ-lr.fr/index.php?option=com_content&view=article&id=162&Itemid=67
2. http://iapr-tc10.univ-lr.fr/index.php?option=com_content&view=article&id=154
3. International Association for Pattern Recognition www.iapr.org/
4. http://www.imageprocessingplace.com/downloads_V3/root_downloads/image_databases/MPEG7_CE-Shape-1_Part_B.zip
5. http://mpeg.chiariglione.org/standards/mpeg-7
6. http://vision.lems.brown.edu/sites/vision.lems.brown.edu/files/1070db.tar.gz
7. The Laboratory for Engineering Man/Machine Systems http://vision.lems.brown.edu/

textes ou d'autres éléments dans ces symboles. La Figure 6.6 montre des exemples de symboles utilisées.

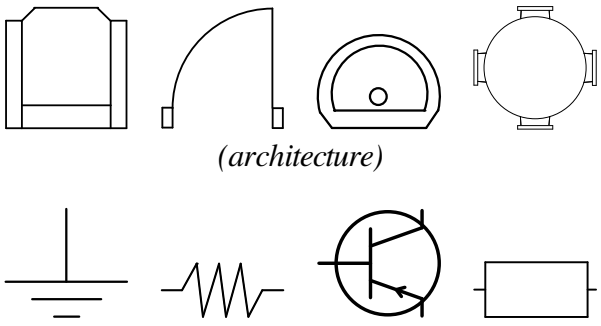
i



(architecture) *(électronique)*

FIGURE 6.5 – Les dessins techniques de deux domaines, l'architecture et l'électronique, qui contiennent les symboles à reconnaître [*cf.* Figure 6.6].

i



(architecture)

(électronique)

FIGURE 6.6 – Les symboles extraits depuis les dessins techniques de deux domaines, l'architecture et l'électronique [*cf.* Figure 6.5].

Les images de la collection sont générées depuis plus de 150 modèles de symboles différents. Il y a 4 ensembles d'images qui sont notés *setA, setB, setC* et *setC*. Le plus grand ensemble, le setC, contient 150 modèles, le setA contient 50 premiers modèles du setC, le setB contient 100 premiers modèles du setC (le setA est donc aussi un sous-ensemble du setB). Les images dans ces ensembles sont nommées par

Ensemble	Modèles	Modifications	Images
setA	50	50	2500
setB	100	50	5000
setC	150	50	7500
setD	36	50	1800

TABLEAU 6.1 – Contenus des ensembles d'images de GREC.

la numérotation (symbol001,symbol002,symbol003,etc.). Le setD contient 36 modèles nommés par leur objet contenu, par exemple, door, battery, table, etc. Il y a des nouveaux modèles dans le setD, par exemple, batterie, core-air n'appartenant pas au setC.

À partir des modèles de chaque ensemble, leurs images d'expérimentation sont générées via différentes modifications :

— Rotation
— Décalage
— Redimensionnement
— Floutage
— Dilatation
— Ajout de bruit

Ces modifications et leurs mélanges sont appliquées pour créer 50 versions différentes à partir chaque modèle. Le Tableau 6.1 résume les ensembles de GREC.

Une fois les images d'expérimentation sont générées, elles sont désordonnées aléatoirement et les ordres des ensembles sont indépendants. La Figure 6.7 montre les premières images de chaque ensemble.

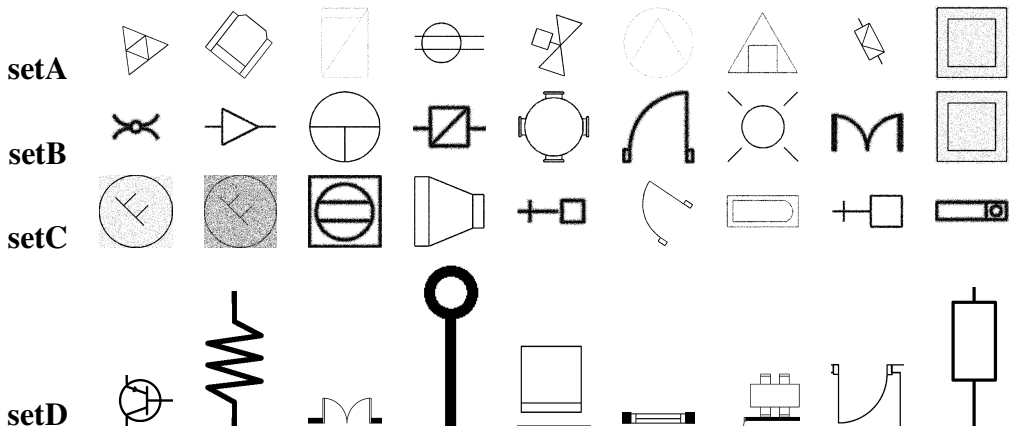


FIGURE 6.7 – Les premières images des ensembles de GREC. Les images sont aléatoirement ordonnées.

La collection de GREC contient donc 16800 images en total. Nous utilisons toutes ces images dans notre expérimentation.

MPEG contient différentes classes de formes solides qui représentent différents objets comme les fruits, les véhicules, les animaux, etc. Ce sont des images binaires, c'est-à-dire que le noir représente l'objet et le blanc représente le fond (ou inverse). Cette collection de forme est utilisée souvent dans les expérimentations des méthodes de caractérisation de formes [8]. La Figure 6.8 montre quelques exemples d'images de MPEG.

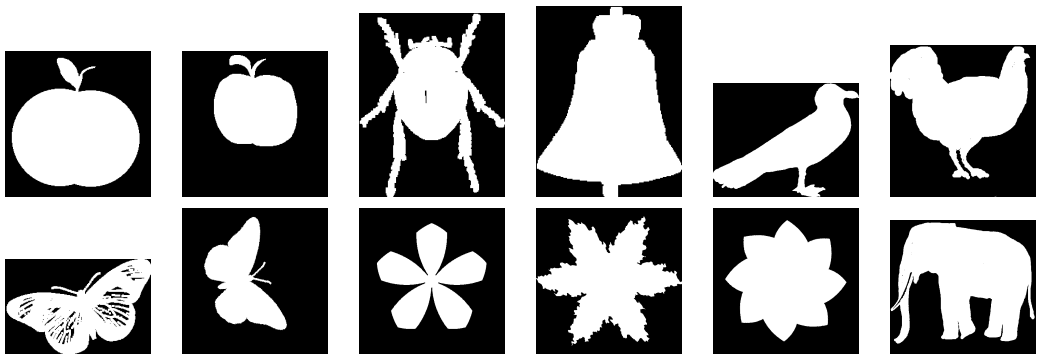


FIGURE 6.8 – Les images de formes d'expérimentation de la norme MPEG.

En effet, ces formes sont distribuées dans 70 catégories différentes : éléphant, pomme, téléphone, fourchette, etc. Chaque catégorie contient 20 formes différentes. Par exemple, la Figure 6.9 montre des formes différentes des chevaux. Ainsi, la collection contient totalement 1400 images de forme.

FIGURE 6.9 – Collection MPEG : les formes différentes des chevaux.

LEMS collecte des images ayant la même modalité des formes dans MPEG. La collection de LEMS répète plusieurs catégories de MPEG comme oiseau (bird), os

8. En utilisant le moteur de recherche Google, la collection de formes MPEG est trouvée dans plus de 170 recherches

Collection	Images	Nature	Taille moyenne
GREC	16800	Ligne droit, courbe	284×284
MPEG	1400	Forme solide	388×342
LEMS	1070	Forme solide	136×123

TABLEAU 6.2 – Les collections de formes

(bone), bouteille (bottle), etc. Des nouvelles catégories sont chat (cat), main (hand), dinosaure (dino), etc. Pour les catégories qu'elle répète la collection MPEG, leurs formes peuvent être différentes. De plus, le nombre de formes dans chaque catégorie est différent. Par exemple, la catégorie vache (cow) ne contient que 2 formes alors que la catégorie outil (tool) contient 63 formes.

La Figure 6.10 montre quelques images de cette collection.

FIGURE 6.10 – Exemples des images de formes de la collection LEMS.

Construction de la base d'images de tests Ainsi, nous avons 3 collections d'images de natures différentes, la Tableau 6.2 résume ces collections.

En total, nous avons 19270 images pour nos expérimentations. Les expérimentations sont :

— Construction de la base d'images
— Mise à jour de la base d'images
— Recherche des images

Les images sont d'abord mélangées. Ensuite, elles sont désordonnées aléatoirement. Enfin, 18000 premières images sont utilisées dans le test de construction de la base d'images, 1170 images suivantes seront ajoutées dans la base pour expérimenter la performance de la mise à jour. 100 dernières images sont les entrées pour la recherche d'images. En résumé, l'utilisation des images dans les expérimentations se décrit dans la Tableau 6.3

Expérimentation	Nombre d'images
Construction de la base d'images	18000
Mise à jour de la base d'images	1170
Recherche des images	100

TABLEAU 6.3 – La distribution des images dans les expérimentations

Caractérisation des images : les moments de Zernike

Comme nous disons dans la Sous-section 6.3.1, il existe plusieurs méthodes de description du contenu d'une image. Parmi les méthodes de caractérisation de formes, nous pouvons noter les moments (Hu, Zernike, etc), les chaînes de Freeman, le descripteur de Fourrier, les approximations polygonales. Ici, nous utilisons le moment Zernike, une méthode de référence pour les expérimentations.

À coté la description de formes, nous avons besoins aussi des étapes avant et après le calcul de la signature de forme comme l'homogénéisation de la couleur des objets (noir ou blanc, l'autre couleur est du fond), le débruitage, la normalisation des signatures.

Pré-traitements : Binarisation, débruitage aveugle Pour tous les traitements, nous devons d'abord garantir la conformité des images d'entrées. De plus, un pré-traitement facilite le traitement et augmente la performance. Dans nos expérimentations, nous souhaitons d'abord identifier la couleur des objets dans les images. Les images sont binaires : nous choisissons le blanc (valeur 1) pour les objets et le noir (valeur 0) pour le fond.

Ensuite, nous débruitons les images pour minimiser l'impact des bruits. Il n'y a que les images de GREC qui contiennent de bruits importants. Les moments de Zernike sont de bons descripteurs de forme mais ils sont sensibles quand ils traitent les images bruitées. Un simple débruitage est utilisé : nous enlevons les petits points de différentes couleurs par rapport leurs voisins qui sont potentiellement les bruits ``sel et poivre'', le type de bruits notables dans les images de MPEG. Nous testons plusieurs rayons, les points ayant le rayon plus petit ou égale au rayon testé étant supprimés. Le rayon 2 est choisi parce qu'il donne le bon résultat.

Les moments de Zernike Les moments de Zernike [83,84] résultent de l'application des polynômes de Zernike pour caractériser les formes. Les moments de Zernike sont très connus dans la caractérisation de forme [9]. On montre que les moments de Zernike sont parmi les meilleurs descripteurs de forme [16, 130].

9. Plus de 2700 articles trouver par Google Scholar pour le terme ``Zernike moments''

Les polynômes de Zernike sont un ensemble des polynômes complexes orthogonaux sur un disque d'unité :

$$Z_n^m(\rho, \phi) = R_n^m(\rho) e^{im\phi}$$

où m, n sont les nombres entiers naturels, ρ est la distance au centre du disque, ϕ est l'angle d'azimuth (en radians), les $R_n^m(\rho)$ sont les polynômes radiaux définis tels que :

$$R_n^m(\rho) = \sum_{i=0}^{n-|m|/2} \frac{(-1)^l (n-l)!}{l! \left[\frac{n+|m|}{2} - l\right]! \left[\frac{n-|m|}{2} - l\right]!} \rho^{n-2l}$$

pour $m, n \in \mathbb{N}, n - m$ pair. $R_n^m(\rho) = 0$ pour $n - m$ impair.

Les moments de Zernike sont les résultats des projections de la forme sur les axes établis depuis ces polynômes. Les caractéristiques remarquables des moments de Zernike sont :

— Comme les polynômes de Zernike sont orthogonaux, il n'y a pas de redondance d'information dans les moments de Zernike.
— Les moments de Zernike sont invariants à la rotation.
— La reconnaissance de forme avec les moments de Zernike accepte un niveau modéré de bruit.

Le calcul original des moments de Zernike est lent. Cependant, plusieurs améliorations sont proposées pour l'accélérer [3,41,75]. Nous implémentons la méthode proposée dans [73].

Avec chaque n donné, nous avons $\lfloor n/2 \rfloor + 1$ valeur de m pour que $n - m$ soit pair (qui peut potentiellement faire $R_n^m(\rho) \neq 0$). Autrement dit, pour chaque n, nous avons $\lfloor n/2 \rfloor + 1$ moments. Par conséquent, pour le calcul des moments jusqu'à n donné, le nombre de moments que nous avons est :

$$\begin{cases} \frac{(n+2)^2}{4} & \text{si } n \text{ est pair} \\ \frac{(n+1)(n+3)}{4} & \text{si } n \text{ est impair} \end{cases} \tag{6.3}$$

Dans l'application, tous les moments jusqu'à n choisi sont calculés. Ces moments sont considérés comme les éléments pour composer un vecteur. En conséquence, la dimension du vecteur augmente de puissance 2 avec n.

Avec les n petits, les moments de Zernike décrit la forme en gros. Quand n grand, la forme est décrite en détail. L'augmentation de n permet de mieux décrire la forme mais elle entre par conséquent dans les hautes dimensions. C'est la raison pour que nous appliquons la courbe remplissant l'espace, qui permet de bien indexer les vecteurs de haute dimension.

Les moments de Zernike sont des descripteurs standards dans la caractérisation de formes. Leur performance est éprouvée à travers plusieurs études compa-

ratives [16, 130]. Après des tests pour déterminer l'ordre optimal des moments de Zernike, $n = 9$ est choisie pour les expérimentations parce qu'elle est la valeur la plus petite qui donne les signatures suffisamment précises. En choisissant cette valeur, les signatures sont les vecteurs de 30 dimensions [10].

Post-traitement des signatures : normalisation Les courbes remplissant l'espace que nous proposons dans cette thèse n'agissent que sur des espaces discrètes. Par conséquent, pour l'indexation, les signatures doivent être discrétisées. De plus, elles doivent aussi être normalisées. Concrètement, nous utilisons 1 octet pour sauvegarder les valeurs. Elles sont donc limitées dans l'intervalle $[0, 255]$. Supposons que $\mathbf{min}_i$ et $\mathbf{max}_i$ sont les valeurs la plus petite et la plus grande des éléments i-ième des signatures $\forall i \in \{0, \mathbf{d} - 1\}$, nous appliquons la normalisation $\mathfrak{n}$ suivant sur toutes les signatures $s = (s_0, s_1, \dots, s_{\mathbf{d}-1})$:

$$\mathfrak{n}(s_i) = \frac{s_i - \mathbf{min}_i}{\mathbf{max}_i - \mathbf{min}_i}$$

L'extraction de la signature normalisée est l'étape nécessaire dans toutes les opérations : l'indexation, la recherche d'image, la mise à jour de la base d'images. Elle suit une série des traitements que nous décrivons ci-dessus.

6.3.3 Choix des courbes remplissant l'espace des caractéristiques

Dans le but de comparer les courbes issues de notre proposition [*cf.* Chapitre 5] avec les courbes existantes, dans les expérimentations, nous testons deux courbes remplissant l'espace :

— MAX : la courbe 30-dimensions maximisant la conservation de la localité issue de notre proposition dans le Chapitre 5 avec la répétition des isométries [*cf.* Sous-section 6.2.3],

— BUTZ : la courbe 30-dimensions générée par algorithme de Butz.

Comme la dimension est relativement élevée, l'indexation avec ces courbes est basée sur l'algorithme en ligne [*cf.* Section 6.2.3]. Dans ce cas, l'utilisation de l'algorithme hors ligne est impossible avec nos équipements informatiques [*cf.* Annexe B].

6.3.4 Système de recherche

Dans la reste du chapitre, la performance des courbes remplissant l'espace est expérimentée dans une application concrète : la recherche d'images dans une grande base. Le système de recherche est décrite par le Diagramme 6.1.

10. $\frac{(9+1)(9+3)}{4} = 30$

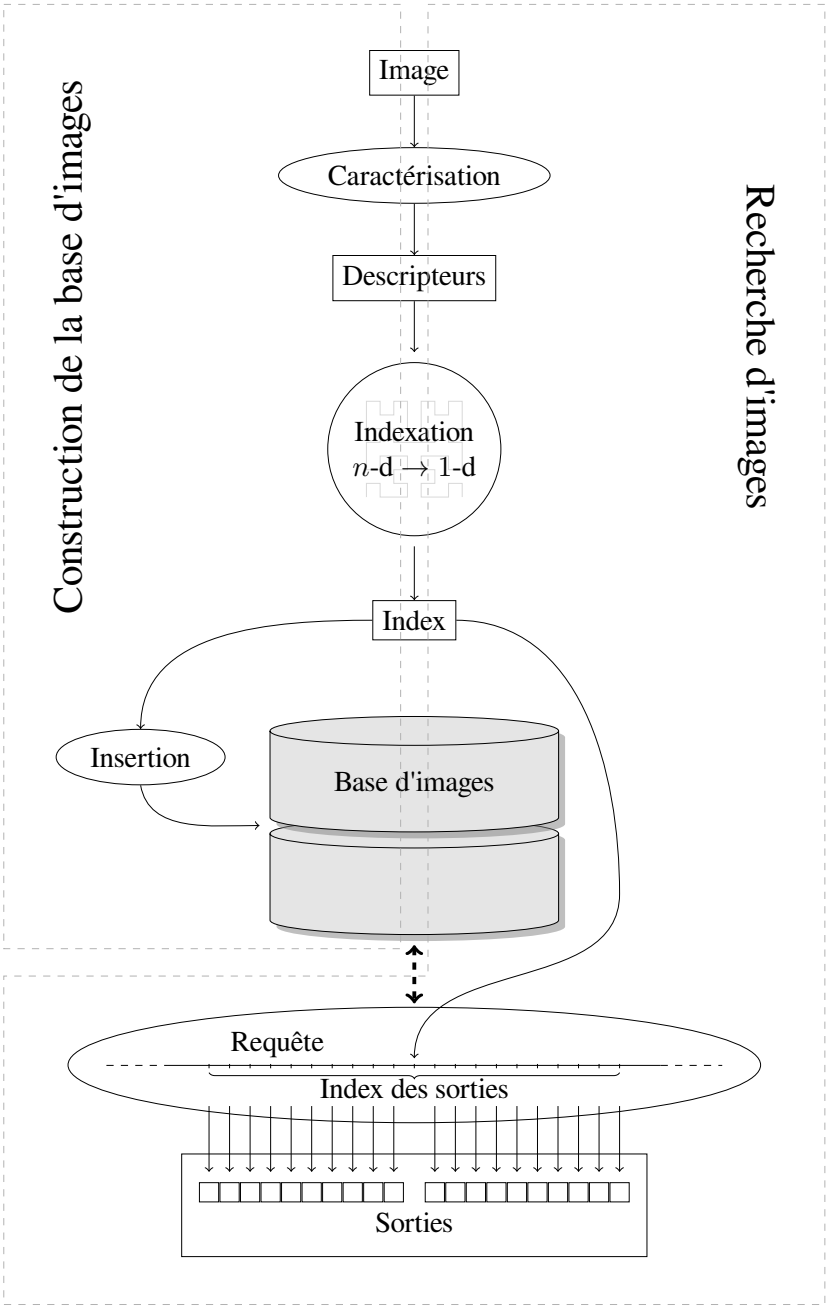


DIAGRAMME 6.1 – Le système de recherche s'appuyant sur les courbes remplissant l'espace.

Le système comprend 2 procédures principales : l'indexation et la recherche. L'indexation consiste à insérer les informations de l'image dans la base en respectant

l'ordre des index. L'indexation est utilisée pour construire et mettre à jour la base d'images. La recherche identifie les index les plus proches à l'index de l'entrée. La sortie associe la liste des images correspondant à ces index.

La correspondance d'une image à son index est réalisée dans les 2 procédures : l'extraction des caractéristiques pour établir le vecteur multidimensionnel et le calcul de l'index de ce vecteur.

6.3.5 Matériel et implémentation : standard

Les expériences ont été menées avec une machine PC dans une configuration standard. Concrètement, un ordinateur portable aux caractéristiques suivantes a été utilisé :

— Processeur : Intel Core 2 Duo T9800 2.93Ghz x 2
— Mémoire : 3.9 GB
— Système d'exploitation : Ubuntu 11.10 64-bit

Langage d'implémentation : C++

6.3.6 Indexation : construction de la base d'images

Rappelons que l'application de la courbe remplissant l'espace est basée sur la conservation de la localité : les index des voisins d'un point sont proches de son index. Par conséquent, les images similaires ont théoriquement des index proches.

Avec la signature normalisée, l'indexation est faite avec les méthodes proposées dans la Section 6.2. Pour une recherche rapide, nous ordonnons les index par un ordre choisi (croissance ou décroissance). L'arbre B+ [43], une variation de l'arbre B [13], est choisie pour sauvegarder l'indexation. Non seulement cette structure de données ordonne les index mais de plus, elle permet de construire efficacement et de mettre à jour facilement la base d'images.

Ainsi, l'indexation est résumée par le processus suivant :

1. Extraction de la signature normalisée de l'image
2. Correspondance entre la signature et son index
3. Insertion des informations de l'image dans l'arbre B+ en conservant l'ordre des index.

Pour faciliter l'évaluation de la performance de la courbe remplissant l'espace, nous précisons les informations suivantes dans une image :

— Lien vers image
— Signature (moments de Zernike)
— Description de forme (table, window, apple, elephant, etc.)

Courbe	Temps total (s)	Temps moyen par image (s)
Max	640.81	0.03560
Butz	641.03	0.03561

TABLEAU 6.4 – Le temps (en seconde) d'indexation de 18000 images de formes avec deux courbes remplissant l'espace différentes.

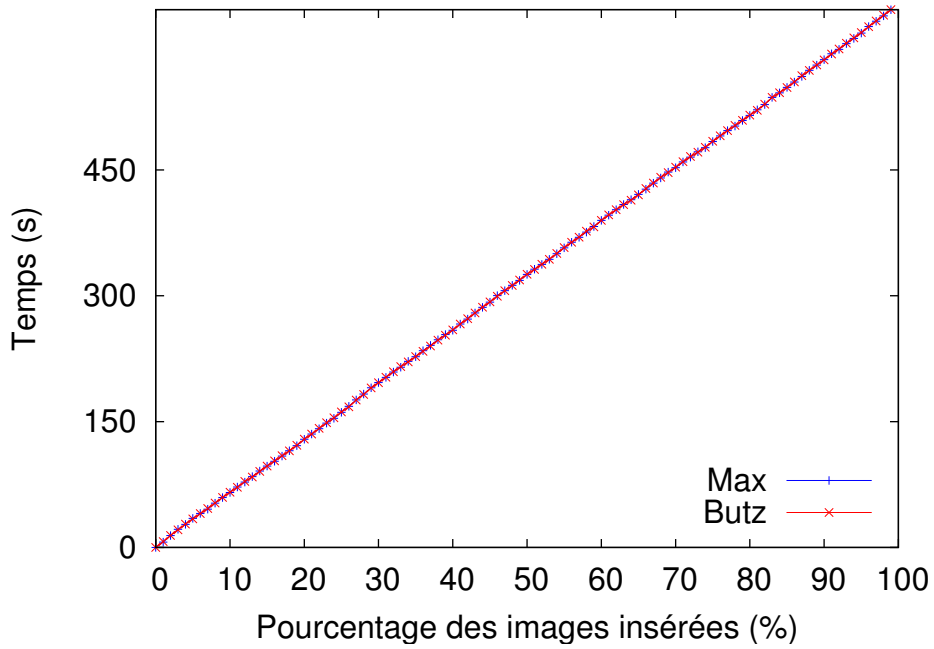


DIAGRAMME 6.2 – Le temps d'indexation de 18000 images avec deux courbes remplissant l'espace différentes.

Nous expérimentons l'indexation avec 18000 premières images de la collection de formes. Le résultat est montré dans le Tableau 6.4 et le Diagramme 6.2.

Le temps d'indexation est d'environ 641 secondes (10.7 minutes) pour 18000 images. Le temps d'indexation moyen pour chaque image est seulement 0.0356 seconde. Le temps d'indexation comprend le temps de l'ensemble du processus d'indexation à partir d'une image : calcul des moments de Zernike, correspondance avec son index et insertion de l'image dans l'arbre B+.

Mise à jour de la base d'images Les bases d'images ne sont pas statiques. Les insertions des nouvelles images sont fréquentes dans les bases. Dans certaines méthodes d'indexation, les mises à jours demandent parfois des traitements importants. Par exemple, la méthode des k-moyennes nécessite de refaire entièrement l'indexa-

tion à chaque mise à jour.

Nous expérimentons ici la mise à jour de la base d'images avec l'indexation par la courbe remplissant l'espace. La mise à jour a la même nature que la construction. Il s'agit simplement d'une insertion des images dans la base. Ainsi, le coût de la mise à jour est similaire à celui que nous avons montré dans la construction de la base. En effet, dans notre expérimentation de mise à jour, l'indexation de 1170 images dure 42.036 secondes, en moyenne, chaque image est indexée dans 0.0359 seconde.

Il y a par contre une augmentation légère du temps d'indexation de chaque image (en moyen de 0.0003 seconde). Cette augmentation peut être expliquée par l'augmentation de la hauteur de l'arbre B+. En effet, à hauteur h, l'arbre B+ peut contenir maximum $\#\mathbf{nœud}^h$ images, où $\#\mathbf{nœud}$ est le nombre d'image maximal peut être contenu dans chaque noeud. Dans notre expérimentation, $\#\mathbf{nœud} = 20$. Quand le nombre d'images ajoutées excède la limite actuelle de l'arbre B+, sa hauteur s'augmente automatiquement. Cette augmentation cause le plongement plus profond de chaque insertion d'image. Par conséquent, le temps d'insertion devient plus long.

Cependant, cette augmentation est presque négligeable parce qu'il n'y a un changement qu'aux images $\#\mathbf{nœud}^i$-ièmes avec $i \in \mathbb{N}$. L'augmentation de temps à chaque changement est faible. À noter que l'augmentation du coût d'insertion ne concerne que l'arbre B+. Il n'y a pas changement sur le temps de traitement pour l'extraction de signature et le calcul de l'index.

Nous voyons ici l'avantage de l'application de la courbe remplissant l'espace par rapport d'autres méthodes de réduction de dimension car la mise à jour ne modifie pas les données existantes dans la base. Elle ne donne lieu qu'à une insertion simple de l'image dans la base. En considérant une autre méthode de réduction de dimension comme k-moyenne, à chaque insertion de nouvelle donnée dans la base, il est nécessaire de faire un examen de toutes les données existantes. Cela coûte beaucoup de ressources (en temps et en mémoire).

Ainsi, non seulement l'application de la courbe remplissant l'espace construit rapidement la base de données mais aussi elle permet une mise à jour de la base de données rapide et dynamique.

6.3.7 Recherche des images : résultats et analyses

Dans l'expérimentation de recherche d'images, nous retirons simplement les images ayant les index les plus proches de l'index de l'image requête. Il peut y avoir des images qui ne contiennent pas la forme similaire de celle de l'entrée. Dans la perspective d'une application réelle, ces sorties peuvent être raffinées pour extraire les formes similaires.

Le processus de la requête contient aussi les étapes pour retrouver l'index de l'image d'entrée : pré-traitement d'image, extraction de la signature, normalisation de la signature, calcul de l'index.

Pour chaque image d'entrée, nous sélectionnons les images ayant les index les plus proches de son index. En donnant un nombre de sorties, les images similaires sont sélectionnées par l'Algorithme 6.

Algorithme 6 Chercher r images ayant les index les plus proches de l'index de l'image entrée.

```
/* Chercher r images ayant les index les plus proches de l'index de l'image
   d'entréé Image */
1:  procédure RECHERCHE(Image, r )

2:      /* Calculer l'index de l'image d'entrée (calculer les moments de Zernike de
           la forme contenue dans l'image et les correspondre à son index) */
3:      Index ← CALCULMOMENTSDEZERNIKE(Image)

4:      /* Chercher les index les plus proches à gauche et à droite de l'index de
           l'image d'entrée */
5:      Gauche ← VOISINGAUCHE(Index)
6:      Droite ← VOISINDROITE(Index)

7:      /* Sortir les r images ayant les index les plus proches de l'index de la forme
           contenue dans l'image d'entrée */
8:      /* SORTIR(i) : sortir l'image ayant l'index i */
9:      pour i ← 1 à r faire
10:         si Index − Gauche < Droite − Index alors
11:             SORTIR(Gauche)
12:             Gauche ← VOISINGAUCHE(Gauche)
13:         sinon
14:             SORTIR(Droite)
15:             Droite ← VOISINDROITE(Droite)
16:         fin si
17:     fin pour
18: fin procédure
```

Nous testons ci-dessous avec le nombre de sorties $r = 2, 5, 10$ et 20. Pour chaque r, 100 images requêtes sont aléatoirement extraites depuis la collection de formes.

Résultats : Illustration et analyse

Nous illustrons ici le cas de la recherche avec $r = 20$. La Figure 6.11 montre 10 parmi les images requêtes.

La Figure 6.12 et Figure 6.13 montre les résultats des recherches de la forme $\triangle$

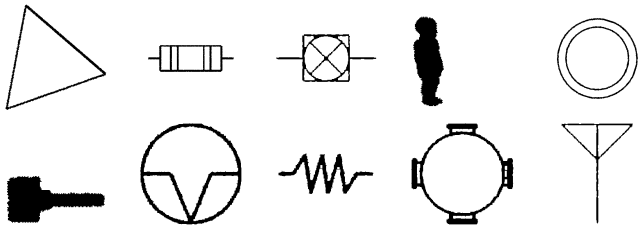


FIGURE 6.11 – Exemples d'images requêtes.

(le triangle, la première forme montrée dans la Figure 6.11) sorties par les applications de deux courbes remplissant l'espace mentionnées ci-dessus.

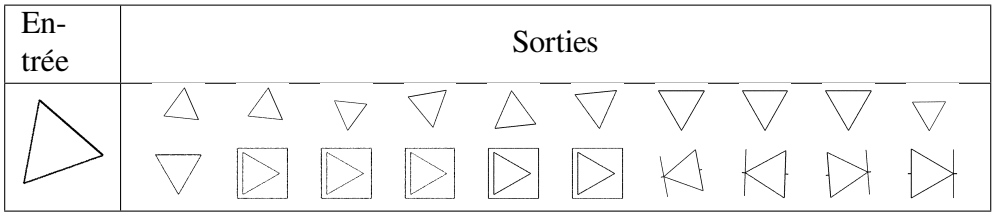


FIGURE 6.12 – Les sorties par l'application de la courbe remplissant l'espace générée par l'algorithme de Butz [*cf.* Sous-section 6.3.3].

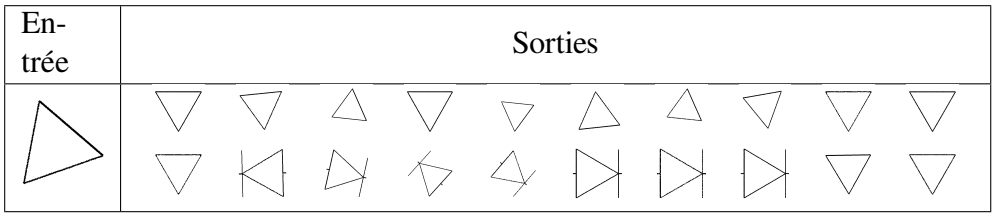


FIGURE 6.13 – Les sorties par l'application de la courbe remplissant l'espace MAX [*cf.* Sous-section 6.3.3].

En regardant ces résultats, nous reconnaissons bien les bonnes sorties sur les premiers candidats. Notons que les moments de Zernike que nous implémentons sont invariants par rotation, translation et changement d'échelle. Ainsi, nous pouvons considérer que ces formes sont de la même classe.

Pour les autres sorties, malgré qu'elles ne correspondent pas à la forme demandée (le triangle), il n'en est pas moins vrai qu'elles contiennent des formes visuellement similaires. Concrètement, les formes sont composées de triangle.

Cependant, les sorties associées pour une même requête sont différentes suivant la méthode utilisée. Notamment, les deuxièmes lignes de 2 sorties sont différentes. À titre d'exemple, l'image ▷ est dans les premières sorties [*cf.* Figure 6.12] mais elle n'appartient pas aux deuxièmes sorties [*cf.* Figure 6.13]. Même pour les identiques sorties, c'est-à-dire, les premières lignes, leur ordre n'est pas pareil. Ce phénomène peut être expliqué par la façon d'ordonnancement différente des points dans l'espace par des courbes différentes, qui cause des indexations différentes.

Figure 6.14 et Figure 6.15 illustrent les sorties par l'application de 2 courbes correspondantes à une autre forme d'entrée. Nous tirons également les mêmes remarques.

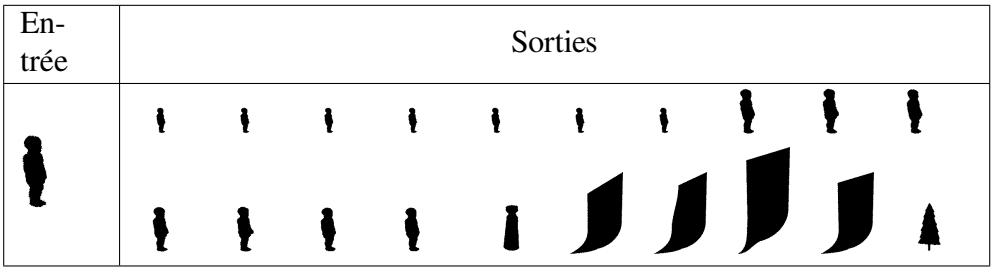


FIGURE 6.14 – Les sorties par l'application de la courbe remplissant l'espace générée par l'algorithme de Butz [*cf.* Sous-section 6.3.3].

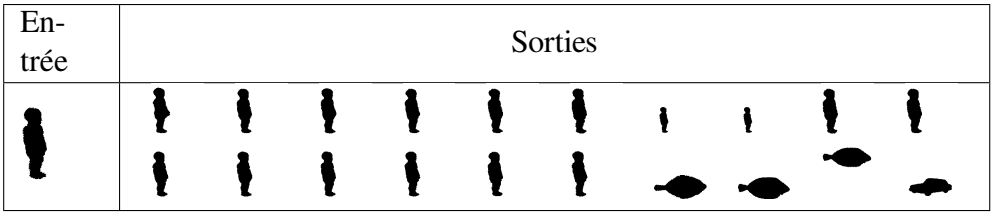


FIGURE 6.15 – Les sorties par l'application de la courbe remplissant l'espace MAX [*cf.* Sous-section 6.3.3].

Résultats à grande échelle

Critère 1 : Précision La performance des courbes remplissant l'espace est ensuite mesurée par la précision :

$$\text{précision} = \frac{\text{nombre d'images similaires}}{\text{nombre de sorties (}\ r\ \text{)}} \tag{6.4}$$

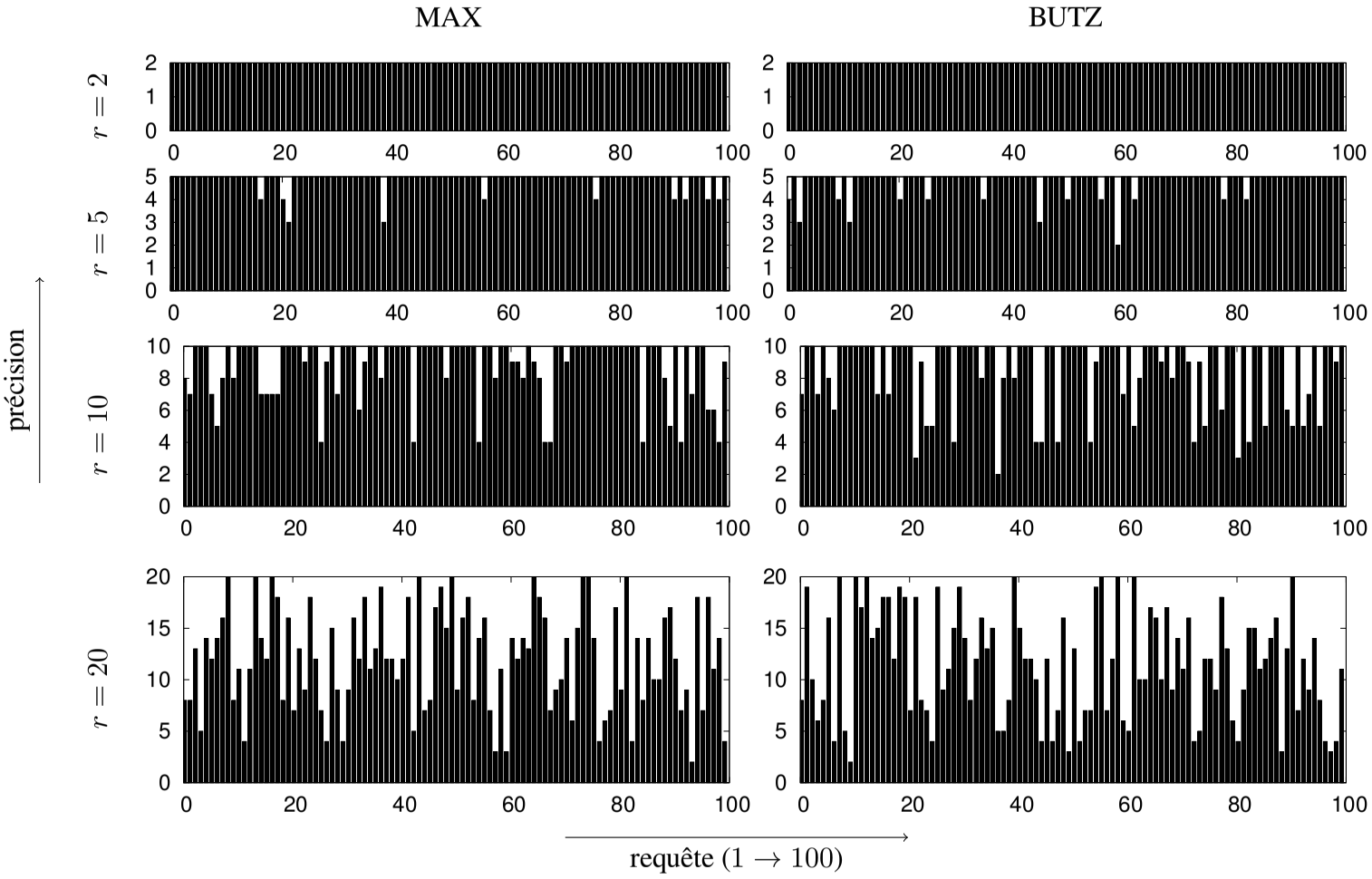


DIAGRAMME 6.3 – Précision de la recherche de 100 images réalisée par l'application de deux courbes remplissant l'espace. La précision est le nombre d'images similaires parmi 20 sorties. La recherche de la forme de triangle montrée ci-dessus est celle 71-ième.

Le Diagramme 6.3 montrée la précision des tests de recherches correspondants aux $r = 2, 5, 10$ et 20 en appliquant deux courbes remplissant l'espace. Chaque test est l'ensemble de 100 requêtes. Le tirage aléatoire des requêtes est refait après chaque changement de r. Le résultat en gros compris le nombre d'images simlaires (sur 100 requêtes), la précision et l'écart-type sont montrés dans Tableau 6.5.

Nombre de sorties	Total		Précision (%)		Écart-type	
	MAX	BUTZ	MAX	BUTZ	MAX	BUTZ
2	200	200	100	100	0	0
5	488	481	97,6	96,2	0,38	0,53
10	880	848	88	84,8	1,90	2,28
20	1208	1150	60,4	57,5	4,96	5,29

TABLEAU 6.5 – Chiffres des retours des recherches correspondant à $r = 2, 5, 10, 20$: nombres d'images similaires, précisions et écart-types.

Impact du rayon de recherche sur l'espace 1-D Nous voyons que la précision dépend du nombre de sorties r. Pour $r = 2$, 100% des retours sont les images similaires à celles des entrées. La précision décroit à chaque augmentation de r [*cf.* Tableau 6.5].

Diagramme 6.4 montre cette décroissance.

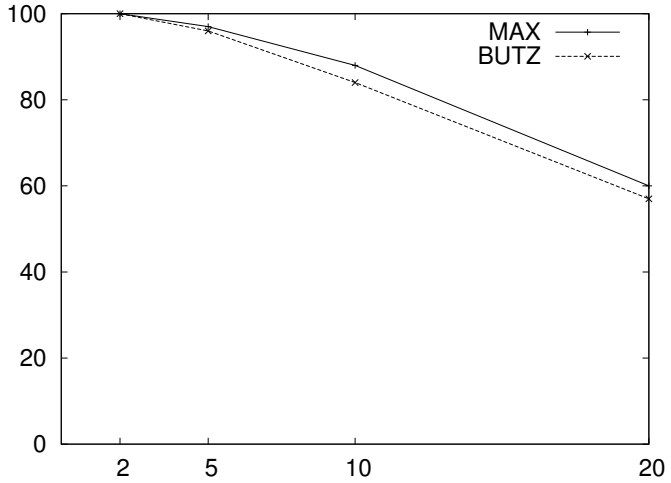


DIAGRAMME 6.4 – La précision de deux courbes à travers les nombres de sorties r différents.

Il est toutefois important de noter l'augmentation du nombre de sorties ``désirées'' avec l'augmentation du rayon r. Cette augmentation est illustrée par Diagramme 6.5.

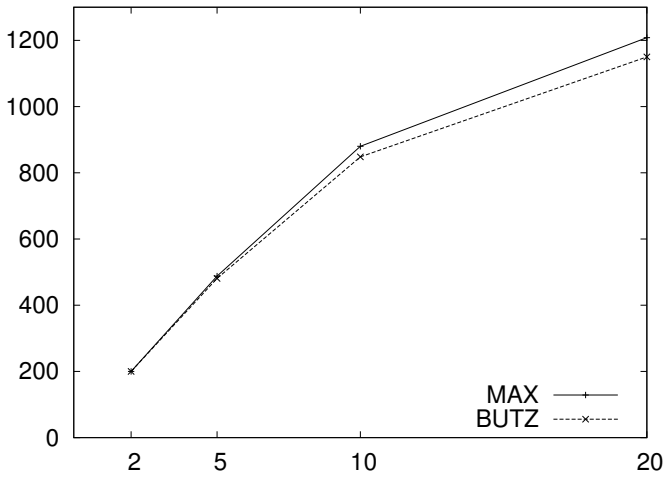


DIAGRAMME 6.5 – L'augmentation du nombre des images similaires à travers les nombres de sorties r différents.

Figure 6.16 illustre la décroissance de la précision et la croissance du nombre d'images similaires avec l'augmentation du nombre de sorties r. Si nous prenons 4 sorties, il y a 4/4 images similaires. Avec 8 sorties, nous avons 6/8 images similaires. Enfin, avec 14 sorties, nous avons 8/14 images similaires. Ainsi, à chaque augmentation de r, la précision est réduite mais le nombre d'images similaires augmente. Figure 6.13 illustre ce phénomène. Après les 11 premières sorties qui sont les images similaires, nous avons 7 formes qui ne sont pas similaires. Cependant, ces formes sont suivies par deux images similaires.

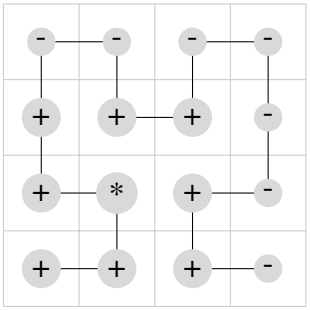


FIGURE 6.16 – Illustration de la distribution des voisins d'un point sur la courbe remplissant l'espace. Les voisins de (*) sont marqués par (+).

Fiabilité À partir du Tableau 6.5, nous voyons que la précision varie entre 100% et 60%. Pour une recherche rapide ([*cf.* Section 6.3.7]), cette précision peut paraître acceptable.

Dans tous les cas, la précision par l'application de la courbe MAX est supérieure à celle de BUTZ. Cela tend de confirmer la vérité des mesures de la conservation de la localité et aussi le processus d'optimisation des courbes.

De plus, l'écart-type de la précision de l'application de la courbe MAX est toujours inférieur à celui de BUTZ. Nous pouvons en déduire que les résultats de MAX est plus fiables.

Critère 2 : Rapidité Les temps de recherche en moyenne par l'application de 2 courbes MAX et BUTZ sont respectivement 0.03303 et 0.03421 seconde. Le Diagramme 6.6 montre le temps de recherche.

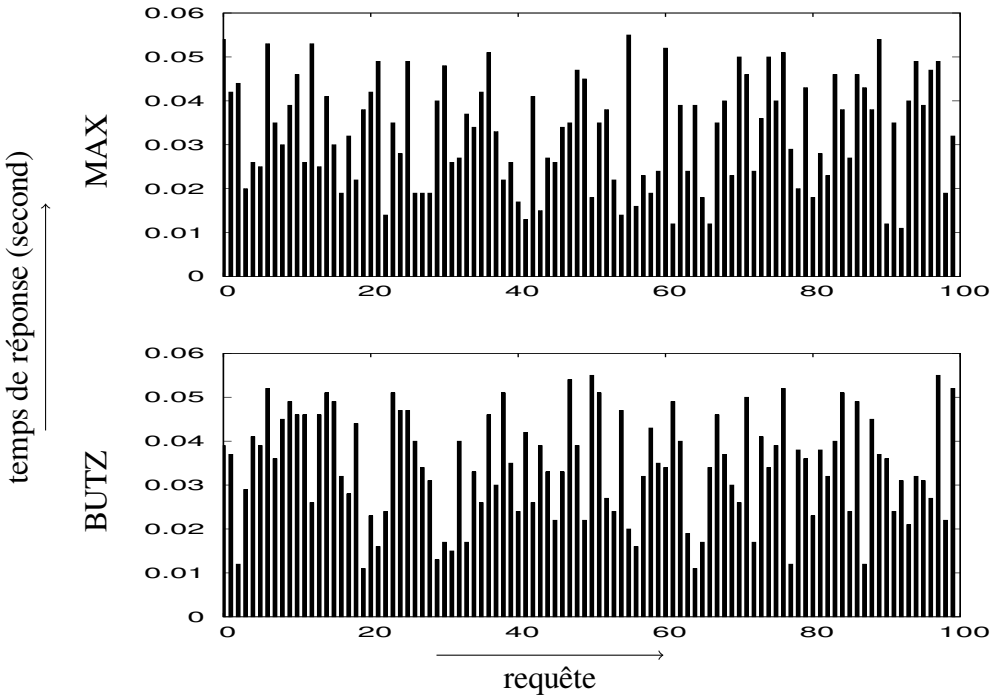


DIAGRAMME 6.6 – Temps de la recherche de 100 images réalisée par l'application de deux courbes remplissant l'espace.

Nous voyons que le temps de recherche de chaque image est très court, même dans le cas le plus défavorable. Le temps de recherche reste inférieur de 0.06 second dans tous les cas.

Discussion Notons que ces mesures de précision sont également influencées par les étapes précédentes notament la description de l'image. En effet, nous avons choisis les moments de Zernike qui même s'ils sont considérés comme des descripteurs fiables et pertinents dans la littérature. Or, ils ne sont pas de description parfaite. Les mesures présentées ci-dessus sont donc des résultats associant d'une part les courbes parcourant l'espace et son principe de la localité, et des descripteurs des images. Chacun de ces principes introduit une part d'erreur qu'il n'est pas possible de circonscrire à ce stade.

6.4 Conclusion

Dans ce chapitre nous avons montré les éléments nécessaires pour l'application de la courbe de Hilbert mais également une application concrète dans la recherche d'image :

— L'indexation par les courbes remplissant l'espace : nous développons une méthode permettant de faire correspondre des points multi-dimensionnels avec leurs index qui sont l'ordre sur la courbe remplissant l'espace choisie. Il s'agit d'une méthode flexible, adaptable à toutes les courbes définies dans la Chapitre 5. Une version optimisée est aussi proposée pour permettre de réduire la complexité en mémoire et en temps en s'appuyant sur la simplification des transformations en simples opérations,

— L'application à la recherche dans une grande base d'images : nous proposons un système de recherche s'appuyant sur les courbes remplissant l'espace. Les tests de ce système sont réalisés sur une base d'images de taille relativement grande ($\sim$20 000 images).

Par les résultats de ces tests, nous constatons les caractéristiques suivantes de ce système de recherches :

— Recherche rapide : les requêtes sont exécutées dans une durée d'environ 0.034 second. Le temps de requête reste inférieur de 0.06 seconde pour toutes les entrées testées,

— Indexation rapide : l'indexation revient à une simple insertion de l'image à sa position sur la courbe. Par conséquent, le temps de construction de la base image est court. 18000 images sont indexées dans 641 secondes (< 11 minutes).

— Mise à jour flexible : en principe, l'addition des nouvelles images revient également à l'indexation de ces images dans la base, correspond simplement à des insertions des images. Ainsi, la mise à jour ne modifie pas la partie existante de la base et la procédure est rapidement exécutée.

L'avantage de ce système de recherche est qu'il n'y a pas d'apprentissage. La construction et la mise à jour de la base d'images correspondent à simplement des

insertions des images dans la bases aux positions pertinentes. Le système est donc rapide et flexible.

Ces résultats encourageant tendent à confirmer l'utilité et la performance des courbes remplissant l'espace pour la recherche dans les grandes bases d'images : elles permettent de faciliter l'indexation de l'espace multidimensionnel et conservent bien la localité, qui représente la relation spatiale des points dans l'espace.

L'application de deux courbes différentes montre que la courbe qui conserve le mieux la localité a la meilleure performance. Elle démontre ainsi la précision de la mesure de la conservation de la localité.

Enfin, il est à noter que l'ensemble des tests s'appuient sur les descripteurs de Zernike sans discussion des résultats inhérents à cette méthode. D'autres descripteurs et/ou à la selection de caractérisation devra en tout état de cause être prise en compte afin de mesurer les influences des caractéristiques face à la méthode de parcours dans l'espace, les résultats finaux couplent les 2 aspects.

Chapitre 7

Conclusion

7.1 Contributions

Dans cette thèse, nous avons proposé une formalisation de ce qu'est une courbe remplissant l'espace et une généralisation de la courbe de Hilbert. Le résultat de la généralisation est une famille de courbes ayant le niveau de conservation de la localité comparable aux extensions multidimensionnelles connues de la courbe de Hilbert. Les algorithmes pour la construction des courbes multidimensionnelles et le calcul de l'index ont été fournis. Ces développements ont été appliqués dans une application de recherche d'image. Des tests comparatifs montrent les bonnes performances d'une courbe extraite de la famille proposée par rapport la courbe de Hilbert générées par l'algorithme de Butz.

7.1.1 Proposition d'une nouvelle formulation d'une courbe remplissant l'espace

Les courbes remplissant un espace sont utiles dans divers domaines d'applications notamment grâce à leur capacité à conserver la localité. Ces courbes permettent d'ordonner les points évoluant dans un espace multidimensionnel sur une ligne unidimensionnelle en conservant au mieux les relations de voisinage inter-points. Autrement dit, à deux points (espace original) proches (métrique donnée) sont attribués deux index (espace unidimensionnel cible) proches.

Toutefois, s'il existe plusieurs définitions d'une courbe remplissant l'espace, elles restent implicites et ne permettent pas la définition d'algorithmes pour leur construction au-delà de quelques dimensions. D'autre part, les courbes remplissant l'espace sont souvent appréhendées à travers des cas particuliers tels que les propositions de Hilbert, Lebesgue ou Peano.

Dans cette thèse nous tentons d'établir une nouvelle formulation de ce qu'est

une courbe remplissant l'espace rendant possible l'élaboration d'algorithmes pour la construction de la courbe et le calcul de l'index dans le cas multidimensionnel (**d**=30 dans notre application [*cf.* Chapitre 6]). Le coeur de cette tentative est une définition précise de l'autosimilitude.

L'autosimilitude telle que nous la définissons apporte deux avantages :

— un calcul rapide de l'index : en étant autosimilaire, une courbe est composée de plusieurs sous-courbes qui sont des similitudes d'un motif primitif unique. Ainsi, au lieu d'être définie par une énumération exhaustive des points, une sous-courbe est représentée par sa similitude correspondante. Avec ce modèle de construction compact, le calcul de l'index est rapide car il n'est plus nécessaire de déterminer la position d'un point d'entrée dans une grande liste de points mais simplement d'identifier les sous-courbes (processus hiérarchique) qui contiennent le point en question [*cf.* Section 6.2.3].

— contribue à définir des courbes conservant bien la localité. Localement, en étant une similitude du motif primitif, chaque sous-courbe ordonne consécutivement les points d'un sous-espace (localité source) dans un segment (localité cible). De plus, si la conservation de la localité du motif primitif est optimisée, alors, chaque sous-courbe jouit par duplication de cette conservation. Par l'héritage [*cf.* Définition 10], l'ordre des sous-espaces est imposé par la mise en relation de chaque sous-espace avec un point d'une courbe autosimilaire, dite d'ordre inférieur. Ainsi, l'héritage contribue à ce que le niveau de conservation de la localité obtenu lors de la construction d'une courbe aux premiers ordres est préservé à travers les ordres supérieurs.

7.1.2 Proposition d'une mesure générale de la conservation de la localité

L'estimation de la conservation de la localité d'une fonction s'appuie sur certaines mesures [*cf.* Sous-section 3.1.3]. Néanmoins, ces mesures estiment la conservation de la localité dans les contextes particuliers, par exemple, pour les courbes remplissant l'espace avec une métrique donnée.

Nous proposons dans cette thèse une mesure générale adaptée aux différents contextes par l'utilisation de paramètres. Par exemple, il est désormais possible de définir indépendamment les rayons de voisinages des espaces source et cible. Nous avons montré que la mesure proposée pour l'estimation de la conservation de la localité couvre trois mesures connues de la littérature [31, 60, 113]. En effet, nous avons décrit les paramètres pour les cas de 3 mesures connues [*cf.* Section 3.4]. Ainsi, notre proposition de mesure est aussi une généralisation de ces 3 mesures.

La mesure servira d'étalon pour estimer le niveau de conservation de la localité des courbes fournies dans cette thèse.

7.1.3 Proposition d'une famille de courbes multidimensionnelles conservant bien la localité

Parmi les courbes remplissant l'espace connues, la courbe de Hilbert est celle qui conserve le mieux la localité. À partir des travaux de Hilbert pour la courbe originale en 2-D, une extension est nécessaire pour des applications multidimensionnelles. Toutefois, les extensions existantes [23, 29] de cette courbe ne s'appuient que sur le seul motif RBG et fixe la jonction des sous-courbes, ce qui donne finalement naissance à une seule courbe à chaque dimension donnée.

Nous proposons une définition de la courbe de Hilbert dans le cas multidimensionnel en retenant seulement l'adjacence, qui est la propriété qui crée la différence de niveau de conservation de la localité de la courbe de Hilbert. Pour élargir les possibilités d'optimisation de la conservation de la localité, nous relâchons également la contrainte de l'autosimilitude et la remplaçons par l'héritage.

Le résultat de cette généralisation [*cf.* Définition 18] est une famille de courbes [*cf.* Sous-section 5.4.1] ayant une conservation de la localité comparable aux courbes générées par les méthodes connues (Butz [29], Bially [23]) [*cf.* Sous-section 5.4.2]. La diversité des courbes est une conséquence de l'augmentation de dimension de l'espace, le nombre de courbes augmentant avec la dimension de l'espace. La généralisation est étayée par des éléments de preuve et son application est illustrée à travers de nombreux exemples.

L'estimation de la conservation de la localité des courbes selon différentes mesures, y compris celle proposée dans le Chapitre 3, montre que les courbes produites conservent aussi bien et parfois mieux la localité par rapport aux courbes générées par les algorithmes connus de Butz et de Bially.

Courbe de Hilbert autosimilaire Pour les applications de grande dimension, nous proposons la courbe de Hilbert autosimilaire, qui optimise le calcul de l'index. Un seul motif primitif est utilisé. Elle ne demande pas beaucoup d'espace pour sauvegarder les informations nécessaires pour la construction et le calcul de l'index. La courbe utilisée dans notre application [*cf.* Chapitre 6] est choisie selon le critère de conservation de la localité.

7.1.4 La génération des courbes

Des réflexions sur la mise en œvre (implémentation) de la généralisation de la courbe de Hilbert multidimensionnelle sont menées ce qui se traduit par la proposition d'algorithmes [*cf.* Section 5.3]. Ces algorithmes s'appuient sur les analyses de la construction de la courbe.

L'avantage de la génération proposée par rapport des méthodes existantes est la flexibilité. Elle peut générer toutes les courbes produites par la définition de la

courbe de Hilbert multidimensionnelle.

Pour les courbes autosimilaires (y compris la courbe de Hilbert autosimilaire), en s'appuyant sur l'isométrie [*cf.* Sous-section 4.3.2] des sous-courbes, la génération des courbes n'a besoin que de type d'informations essentielles :

— le motif primitif,

— les isométries produisant les sous-courbes à partir le motif primitif.

L'isométrie de chaque sous-courbe est simplifiée par la décomposition en deux opérations élémentaires : la réflexion et la permutation des coordonnées. Ces opérations consommant peu d'espace mémoire sont exécutées dans une durée très courte. C'est pourquoi le calcul de l'index appliqué dans le Chapitre 6 est très rapide.

7.1.5 La recherche d'image, l'indexation multidimensionnelle

La proposition des nouvelles courbes remplissant l'espace et les algorithmes permettant la construction de courbes et le calcul de l'index qui fonctionne dans le cas de grande dimension ouvre la possibilité d'explorer les nouvelles applications des courbes remplissant l'espace.

Nous explorons dans le Chapitre 6 la performance des courbes remplissant l'espace par la recherche d'images dans de grandes bases. Dans cette application, chaque image est représentée par un vecteur de caractéristique de dimension $\mathbf{d}$=30. Une collection d'images est synthétisée par un ensemble de points $\mathbf{d}$-dimensionnels évoluant dans l'espace des caractéristiques. L'indexation ordonne les images selon leurs index. La localité borne combien de points proches sur la ligne correspondent à des images aux caractéristiques effectivement proches (métrique donnée) et donc visuellement compatibles. Elle peut donc inférer sur les résultats de la recherche et donc sur les performances notamment en termes de précision. Des tests à grandes échelles (19270 images), menés sur les bases de référence [*cf.* Section 6.3.2] semblent confirmer ces hypothèses.

La performance des courbes remplissant l'espace est expérimentée par l'application de la recherche d'image par le contenu qui montre que les courbes courbes remplissant l'espace, concrètement les courbes de Hilbert multidimensionnelles, donnent de bons résultats dans l'espace de 30 dimensions.

L'indexation est très rapide : par exemple, une image est indexée en moyenne en 0,03 secondes. Un autre élément marquant est que l'indexation avec les courbes remplissant l'espace est stable avec la mise à jour de la base d'images. À l'ajout des nouvelles images, la partie existante de la base reste inchangée. La mise à jour de la base d'image revient simplement à insérer des index des nouvelles images, aucune modification supplémentaire n'étant nécessaire.

Le résultat de la recherche révèle une précision correcte de l'indexation par les courbes remplissant l'espace. La sortie de chaque requête comporte une part importante des images visuellement similaires avec l'image d'entrée. Concrètement, il y a

en moyenne 12 images similaires à celle de l'entrée parmi 20 images de résultat. Le temps de réponse est très satisfaisant (0,033 seconde/image).

Notons que ces tests mettent en jeu plusieurs types d'images (naturelles, graphiques), aux modalités variées (binaire, couleur). Cette approche semble favorablement accueillie par la communauté [119, 120].

7.2 Quelques perspectives

7.2.1 Applications

La proposition de nouvelles courbes remplissant l'espace multidimensionnel accompagnée d'algorithmes pour leur construction et le calcul de l'index d'un point donné ouvre des perspectives en termes d'application. Le calcul de l'index léger et rapide permet d'appliquer ces courbes dans le cas de grande dimension, cas fréquent en informatique.

La diversité des courbes remplissant l'espace et particulièrement, des courbes généralisant la courbe de Hilbert crée la possibilité de choisir la courbe qui conserve bien la localité dans un contexte donné. Avec les localités données (source et cible), la mesure proposée dans le Chapitre 3 permet de réaliser les comparaisons pour déterminer la courbe qui s'adapte le mieux à un contexte applicatif donné.

7.2.2 Correction de la conservation de la localité

Malgré que ces courbes conservent bien la localité, les points perdent toujours des voisins parce que, dans tous les cas, une correspondance d'un point multidimensionnel avec un index unidimensionnel est une réduction de dimension. La correction de cette perte de voisins est souhaitable car elle permettrait de rendre plus précis les résultats des applications.

À titre d'exemple, la perte de voisins peut être corrigée par l'utilisation de différentes courbes afin de réduire l'impact des zones où la perte est la plus préjudiciable. La Figure 7.1*(a)* montre un exemple de ces zones. Une combinaison des courbes décalées en position permettraient de réduire l'impact néfaste de ces zones sur conservation de la localité [*cf.* Figure 7.1 *(b)*].

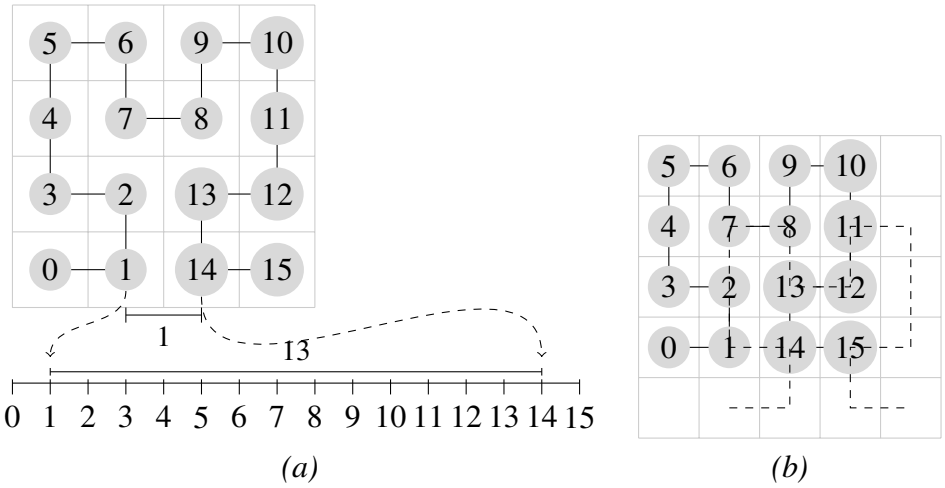


FIGURE 7.1 – La perte de voisins et sa correction. Dans *(a)*, les points 1 et 14 sont voisins (distance 1) mais leurs index sont éloignés (distance 13). Cette faute de voisinage est corrigée par une autre courbe (pointillée) dans *(b)*. Nous observons alors que par un décalage en position (pointillée), les index attribués aux points 1 et 14 sont alors proche (distance 1).

Annexe A

La génération des courbes Sweep et Scan

Donnant un point $x = (x_0, x_1, \ldots, x_{\mathbf{d}-1})$ dans un espace de $\mathbf{d}$ dimensions, l'index du point x sur les courbes Sweep f et Scan g de résolution N est calculé comme suit :

A.1 Sweep

$f(x) = \sum_{i=0}^{\mathbf{d}-1} N^i x_i$

A.2 Scan

Examiner la fonction $\delta(i)$ définie comme suit : $\delta(k) = 1$ si k pair, $\delta(k) = -1$ si k impair.

$g(x) = \sum_{i=0}^{\mathbf{d}-2} N^i((x_{i+1} \bmod 2)(N-1) + \delta(x_{i+1})x_i) + N^{\mathbf{d}-1} x_{\mathbf{d}-1}$

Annexe B

PC utilisé pour les tests

Les expériences ont été menées avec une machine PC dans une configuration standard. Concrètement, un ordinateur portable aux caractéristiques suivantes a été utilisé :

- Processeur : Intel Core 2 Duo T9800 2.93Ghz x 2
- Mémoire : 3.9 GB
- Système d'exploitation : Ubuntu 11.10 64-bit

Langage d'implémentation : C++

Bibliographie

[1] M. Al-Rousan, J.K. Archibald, and L. Bearnson. Evaluating the impact of locality on the performance of large-scalesci multiprocessors. *Performance Evaluation*, 46(4) :275--302, 2001.

[2] S. Albers, L.M. Favrholdt, and O. Giel. On paging with locality of reference. *Journal of Computer and System Sciences*, 70(2) :145--175, 2005.

[3] Gholamreza Amayeh, Ali Erol, George Bebis, and Mircea Nicolescu. Accurate and efficient computation of high order zernike moments. *Advances in visual computing*, pages 462--469, 2005.

[4] Jennifer M. Anderson and Monica S. Lam. Global optimizations for parallelism and locality on scalable parallel machines. *SIGPLAN Not.*, 28(6) :112--125, June 1993.

[5] M. Ankerst, G. Kastenmüller, H.P. Kriegel, and T. Seidl. 3d shape histograms for similarity search and classification in spatial databases. In *Advances in Spatial Databases*, pages 207--226. Springer, 1999.

[6] M. Ankerst, G. Kastenmüller, H.P. Kriegel, T. Seidl, et al. Nearest neighbor classification in 3d protein databases. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 34--43. AAAI Press, 1999.

[7] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A.Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6) :891--923, 1998.

[8] T. Asano, D. Ranjan, T. Roos, E. Welzl, and P. Widmayer. Space-filling curves and their use in the design of geometric data structures. *Theoretical Computer Science*, 181(1) :3--15, 1997.

[9] Tetsuo Asano. Digital halftoning algorithm based on random space-filling curve. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, 82(3) :553--556, 1999.

[10] B. Awerbuch, M. Luby, AV Goldberg, and S.A. Plotkin. Network decomposition and locality in distributed computation. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pages 364--369. IEEE, 1989.

[11] Fabrizio Baiardi, Sarah Chiti, Paolo Mori, and Laura Ricci. Integrating load balancing and locality in the parallelization of irregular problems. *Future Generation Computer Systems*, 17(8) :969 -- 975, 2001.

[12] John J Bartholdi and Loren K Platzman. Heuristics based on spacefilling curves for combinatorial problems in euclidean space. *Management Science*, 34(3) :291--305, 1988.

[13] Rudolf Bayer and Edward M. McCreight. Organization and maintenance of large ordered indexes. *Acta informatica*, 1(3) :173--189, 1972.

[14] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree : an efficient and robust access method for points and rectangles. *SIGMOD Rec.*, 19(2) :322--331, May 1990.

[15] J.S. Beis and D.G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 1000--1006. IEEE, 1997.

[16] Saeid O Belkasim, Malayappan Shridhar, and Majid Ahmadi. Pattern recognition with moment invariants : a comparative study and new results. *Pattern recognition*, 24(12) :1117--1138, 1991.

[17] J.L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9) :509--517, 1975.

[18] S. Berchtold, C. Böhm, D.A. Keim, and H.P. Kriegel. A cost model for nearest neighbor search in high-dimensional data space. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 78--86. ACM, 1997.

[19] Stefan Berchtold, Daniel A Keim, and Hans-Peter Kriegel. The x-tree : An index structure for high-dimensional data. *Readings in multimedia computing and networking*, page 451, 2001.

[20] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful ? *Database Theory—ICDT'99*, pages 217--235, 1999.

[21] Gregory Beylkin. Discrete radon transform. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 35(2) :162--172, 1987.

[22] T. Bially. Space-filling curves : Their generation and their application to bandwidth reduction. *Information Theory, IEEE Transactions on*, 15(6) :658 -- 664, November 1969.

[23] Theodore Bially. Space-filling curves : Their generation and their application to bandwidth reduction. *Information Theory, IEEE Transactions on*, 15(6) :658--664, 1969.

[24] S. Biswas. Hilbert scan and image compression. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 3, pages 207--210. IEEE, 2000.

[25] J.R. Bitner, G. Ehrlich, and E.M. Reingold. Efficient generation of the binary reflected gray code and its applications. *Communications of the ACM*, 19(9) :517--521, 1976.

[26] Stevens Le Blond, Arnaud Legout, and Walid Dabbous. Pushing bittorrent locality to the limit. *Computer Networks*, 55(3) :541 -- 557, 2011.

[27] M. Blume, M.A. Lazarus, L.S. Peranich, F. Vernhes, W.R. Caid, T.E. Dunning, G.R. Russell, K.L. Sitze, et al. Predictive modeling of consumer financial behavior using supervised segmentation and nearest-neighbor matching, January 4 2005.

[28] A.R. Butz. Convergence with hilbert's space filling curve. *Journal of Computer and System Sciences*, 3(2) :128--146, 1969.

[29] A.R. Butz. Alternative algorithm for hilbert's space-filling curve. *IEEE Transactions on Computers*, 20(4) :424--426, 1971.

[30] Arthur R Butz. Space filling curves and mathematical programming. *Information and Control*, 12(4) :314--330, 1968.

[31] Charles L Byrne. Block-iterative methods for image reconstruction from projections. *Image Processing, IEEE Transactions on*, 5(5) :792--794, 1996.

[32] D. Cai, X. He, and J. Han. Document clustering using locality preserving indexing. *Knowledge and Data Engineering, IEEE Transactions on*, 17(12) :1624--1637, 2005.

[33] Deng Cai, Xiaofei He, Xuanhui Wang, Hujun Bao, and Jiawei Han. Locality preserving nonnegative matrix factorization. In *Proceedings of the 21st international jont conference on Artifical intelligence*, pages 1010--1015. Morgan Kaufmann Publishers Inc., 2009.

[34] Georg Cantor. Une contribution à la théorie des ensembles. *Acta Mathematica*, 2(1) :311--328, 1883.

[35] C. Carson, M. Thomas, S. Belongie, J. Hellerstein, and J. Malik. Blobworld : A system for region-based image indexing and retrieval. In *Visual Information and Information Systems*, pages 660--660. Springer, 1999.

[36] J Lawrence Carter and Mark N Wegman. Universal classes of hash functions. *Journal of computer and system sciences*, 18(2) :143--154, 1979.

[37] José Castro, Michael Georgiopoulos, Ronald Demara, and Avelino Gonzalez. Data-partitioning using the hilbert space filling curves : Effect on the speed of convergence of fuzzy artmap for large database problems. *Neural networks*, 18(7) :967--984, 2005.

[38] D. Chase. Code combining--a maximum-likelihood decoding approach for combining an arbitrary number of noisy packets. *Communications, IEEE Transactions on*, 33(5) :385--393, 1985.

[39] Sibao Chen, Haifeng Zhao, Min Kong, and Bin Luo. 2d-lpp : A two-dimensional extension of locality preserving projections. *Neurocomputing*, 70(4–6) :912 -- 921, 2007.

[40] Y.H. Cho and J.K. Kim. Application of web usage mining and product taxonomy to collaborative recommendations in e-commerce. *Expert Systems with Applications*, 26(2) :233--246, 2004.

[41] Chee-Way Chong, P Raveendran, and R Mukundan. A comparative analysis of algorithms for fast computation of zernike moments. *Pattern Recognition*, 36(3) :731--742, 2003.

[42] François Combes. *Algèbre et géométrie : agrégation, CAPES, licence, maîtrise*. Bréal, 1998.

[43] Douglas Comer. Ubiquitous b-tree. *ACM Computing Surveys (CSUR)*, 11(2) :121--137, 1979.

[44] Jordan J Cox, Yasuko Takezaki, Helaman RP Ferguson, Kent E Kohkonen, and Eric L Mulkay. Space-filling curves in tool-path applications. *Computer-aided design*, 26(3) :215--224, 1994.

[45] Ricardo da Silva Torres and Alexandre Xavier Falcão. Content-based image retrieval : Theory and applications. *Revista de Informática Teórica e Aplicada*, 2(13) :161--185, 2006.

[46] M. Datar, N. Immorlica, P. Indyk, and V.S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253--262. ACM, 2004.

[47] G. David Forney. Convolutional codes ii. maximum-likelihood decoding. *Information and control*, 25(3) :222--266, 1974.

[48] Yuhui Deng. Exploiting the performance gains of modern disk drives by enhancing data locality. *Information Sciences*, 179(14) :2494 -- 2511, 2009.

[49] John M Dennis. Inverse space-filling curve partitioning of a global ocean model. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1--10. IEEE, 2007.

[50] Oliver Deussen, Stefan Hiller, Cornelius Van Overveld, and Thomas Strothotte. Floating points : A method for computing stipple drawings. *Computer Graphics Forum*, 19(3) :41--50, 2000.

[51] C. Ding and X. He. K-nearest-neighbor consistency in data clustering : incorporating local information into global optimization. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 584--589. ACM, 2004.

[52] C. Ding and X. He. Cluster aggregate inequality and multi-level hierarchical clustering. *Knowledge Discovery in Databases : PKDD 2005*, pages 71--83, 2005.

[53] R. Dorrigiv and A. López-Ortiz. List update with probabilistic locality of reference. *Information Processing Letters*, 2012.

[54] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1) :11--15, 1972.

[55] A.A. Efros, A.C. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 726--733. IEEE, 2003.

[56] C. Faloutsos and S. Roseman. Fractals for secondary key retrieval. In *Proceedings of the eighth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, PODS '89, pages 247--252, New York, NY, USA, 1989. ACM.

[57] Imola K Fodor. A survey of dimension reduction techniques. *Center for Applied Scientific Computing, Lawrence Livermore National Laboratory*, 9 :1--18, 2002.

[58] J.H. Friedman, J.L. Bentley, and R.A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3) :209--226, 1977.

[59] G. Giacinto. A nearest-neighbor approach to relevance feedback in content based image retrieval. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 456--463. ACM, 2007.

[60] C. Gotsman and M. Lindenbaum. On the metric properties of discrete space-filling curves. *IEEE Transactions on Image Processing*, 5(5) :794--797, 1996.

[61] A. Grama. *Introduction to parallel computing*. Addison Wesley, 2003.

[62] J. Grimshaw. Locality and extended projection. *AMSTERDAM STUDIES IN THE THEORY AND HISTORY OF LINGUISTIC SCIENCESERIES 4*, pages 115--134, 2000.

[63] Xiaoguang Gu, Lei Zhang, Yongdong Zhang, Dongming Zhang, and Jintao Li. An improved method of locality sensitive hashing for indexing large-scale and high-dimensional features. *Signal Processing*, 2012.

[64] Jie Gui, Wei Jia, Ling Zhu, Shu-Ling Wang, and De-Shuang Huang. Locality preserving discriminant projections for face and palmprint recognition. *Neurocomputing*, 73(13–15) :2696 -- 2707, 2010.

[65] Frank Günther, Miriam Mehl, Markus Pögl, and Christoph Zenger. A cache-aware algorithm for pdes on hierarchical data structures based on space-filling curves. *SIAM Journal on Scientific Computing*, 28(5) :1634--1650, 2006.

[66] Lei Guo, Song Jiang, Li Xiao, and Xiaodong Zhang. Fast and low-cost search schemes by exploiting localities in p2p networks. *Journal of Parallel and Distributed Computing*, 65(6) :729 -- 742, 2005.

[67] Antonin Guttman. R-trees : a dynamic index structure for spatial searching. *SIGMOD Rec.*, 14(2) :47--57, June 1984.

[68] MR Haji-Hashemi, H Mir-Mohammad Sadeghi, and VM Moghtadai. Space-filling patch antennas with cpw feed. *Session 1A1*, page 119, 2009.

[69] N.J.A. Harvey, M.B. Jones, S. Saroiu, M. Theimer, and A. Wolman. Skipnet : A scalable overlay network with practical locality properties. In *Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems-Volume 4*, pages 9--9. USENIX Association, 2003.

[70] X. He, D. Cai, H. Liu, and W.Y. Ma. Locality preserving indexing for document representation. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 96--103. ACM, 2004.

[71] D.B. Heras, J.C. Cabaleiro, and F.F. Rivera. Modeling data locality for the sparse matrix–vector product using distance measures. *Parallel Computing*, 27(7) :897 -- 912, 2001.

[72] D. Hilbert. Über die stetige abbildung einer linie auf ein flächenstück. *Mathematische Annalen*, 38(3) :459--460, 1891.

[73] Khalid M Hosny. Fast computation of accurate zernike moments. *Journal of Real-Time Image Processing*, 3(1) :97--107, 2008.

[74] P.G. Howard and J.S. Vitter. Fast and efficient lossless image compression. In *Data Compression Conference, 1993. DCC'93.*, pages 351--360. IEEE, 1993.

[75] Sun-Kyoo Hwang and Whoi-Yul Kim. A novel approach to the fast computation of zernike moments. *Pattern Recognition*, 39(11) :2065--2076, 2006.

[76] Piotr Indyk. Nearest neighbors in high-dimensional spaces, 2004.

[77] R. Jain. Characteristics of destination address locality in computer networks : a comparison of caching schemes. *Computer networks and ISDN systems*, 18(4) :243--254, 1990.

[78] S Jin and A Bestavros. Greedydual- web caching algorithm : exploiting the two sources of temporal locality in web request streams. *Computer Communications*, 24(2) :174 -- 183, 2001.

[79] Alexis Joly, Carl Frélicot, and Olivier Buisson. Robust content-based video copy identification in a large reference database. *Image and Video Retrieval*, pages 511--516, 2003.

[80] S. Kamata, M. Niimi, and E. Kawaguchi. A gray image compression using a hilbert scan. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 3, pages 905--909. IEEE, 1996.

[81] Ibrahim Kamel and Christos Faloutsos. On packing r-trees. In *Proceedings of the second international conference on Information and knowledge management*, pages 490--499. ACM, 1993.

[82] Mahmut Taylan Kandemir. Improving whole-program locality using intra-procedural and inter-procedural transformations. *Journal of Parallel and Distributed Computing*, 65(5) :564 -- 582, 2005.

[83] Alireza Khotanzad and Yaw Hua Hong. Invariant image recognition by zernike moments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(5) :489--497, 1990.

[84] Whoi-Yul Kim and Yong-Sung Kim. A region-based shape descriptor using zernike moments. *Signal Processing : Image Communication*, 16(1) :95--102, 2000.

[85] J.M. Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 599--608. ACM, 1997.

[86] F. Korn, N. Sidiropoulos, and C. Faloutsos. Fast nearest neighbor search in medical image databases. Technical report, Institute for Systems Research, 1996.

[87] Vladimir E Kuznetsov. Method for storing map data in a database using space filling curves and a method of searching the database to find objects in a given area and to find objects nearest to a location, February 1 2000.

[88] Gauthier Lafruit and Jan P Cornelis. Parallelization of the 2d fast wavelet transform with a space-filling curve image scan. In *SPIE's 1995 International Symposium on Optical Science, Engineering, and Instrumentation*, pages 470--482. International Society for Optics and Photonics, 1995.

[89] Nina Siu-Ngan Lam and Kam-biu Liu. Use of space-filling curves in generating a national rural sampling frame for hiv/aids research*. *The Professional Geographer*, 48(3) :321--332, 2005.

[90] C-H Lamarque and Frédéric Robert. Image analysis using space-filling curves and 1d wavelet bases. *Pattern Recognition*, 29(8) :1309--1322, 1996.

[91] A. Lapidoth. Nearest neighbor decoding for additive non-gaussian noise channels. *Information Theory, IEEE Transactions on*, 42(5) :1520--1529, 1996.

[92] J.K. Lawder and P.J.H. King. Querying multi-dimensional data indexed using the hilbert space-filling curve. *ACM Sigmod Record*, 30(1) :19--24, 2001.

[93] J.K. Lawder and P.J.H. King. Using state diagrams for hilbert curve mappings. *International Journal of Computer Mathematics*, 78(3) :327--342, 2001.

[94] Henri Lebesgue. *Leçons sur l'intégration*. Paris, Gauthier-Villars, 1904.

[95] C. Leopold. Arranging program statements for locality on the basis of neighbourhood preferences. *International journal of approximate reasoning*, 19(1) :73--90, 1998.

[96] A.S. Lewis and G. Knowles. Image compression using the 2-d wavelet transform. *Image Processing, IEEE Transactions on*, 1(2) :244--250, 1992.

[97] Jun-Bao Li, Jeng-Shyang Pan, and Shyi-Ming Chen. Kernel self-optimized locality preserving discriminant analysis for feature extraction and recognition. *Neurocomputing*, 74(17) :3019 -- 3027, 2011.

[98] Swanwa Liao, Mario A Lopez, and Scott T Leutenegger. High dimensional similarity search with space filling curves. In *Data Engineering, 2001. Proceedings. 17th International Conference on*, pages 615--622. IEEE, 2001.

[99] M. Lillibridge, K. Eshghi, D. Bhagwat, V. Deolalikar, G. Trezise, and P. Camble. Sparse indexing : large scale, inline deduplication using sampling and locality. In *Proccedings of the 7th conference on File and storage technologies*, pages 111--123. USENIX Association, 2009.

[100] Jae S Lim. Two-dimensional signal and image processing. *Englewood Cliffs, NJ, Prentice Hall, 1990, 710 p.*, 1, 1990.

[101] King Ip Lin, Hosagrahar V Jagadish, and Christos Faloutsos. The tv-tree : An index structure for high-dimensional data. *The VLDB Journal—The International Journal on Very Large Data Bases*, 3(4) :517--542, 1994.

[102] M.H. Lipasti, C.B. Wilkerson, and J.P. Shen. Value locality and load value prediction. *ACM SIGOPS Operating Systems Review*, 30(5) :138--147, 1996.

[103] Xian Liu. Four alternative patterns of the hilbert curve. *Applied mathematics and computation*, 147(3) :741--752, 2004.

[104] D.G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150--1157. Ieee, 1999.

[105] A. Mahanti, D. Eager, and C. Williamson. Temporal locality and its impact on web proxy cache performance. *Performance Evaluation*, 42(2) :187--203, 2000.

[106] Yossi Matias and Adi Shamir. Video scrambling apparatus and method based on space filling curves, October 15 1991.

[107] Yossi Matias and Adi Shamir. A video scrambling technique based on space filling curves. In *Advances in Cryptology—CRYPTO'87*, pages 398--417. Springer, 2006.

[108] John McVay, Ahmad Hoorfar, and Nader Engheta. Thin absorbers using space-filling-curve high-impedance surfaces. In *Antennas and Propagation Society International Symposium, 2005 IEEE*, volume 2, pages 22--25. IEEE, 2005.

[109] John McVay, Ahmad Hoorfar, and Nader Engheta. Space-filling curve rfid tags. In *Radio and Wireless Symposium, 2006 IEEE*, pages 199--202. IEEE, 2006.

[110] Yves Meyer. Wavelets-algorithms and applications. *Wavelets-Algorithms and applications Society for Industrial and Applied Mathematics Translation., 142 p.*, 1, 1993.

[111] Graeme Mitchison and Richard Durbin. Optimal numberings of an n*n array. *SIAM Journal on Algebraic Discrete Methods*, 7(4) :571--582, 1986.

[112] Baback Moghaddam, Kenneth J Hintz, and Clayton V Stewart. Space-filling curves for image compression. In *Proceedings of the SPIE*, pages 414--421, 1991.

[113] B. Moon, HV Jagadish, C. Faloutsos, and J.H. Saltz. Analysis of the clustering properties of the hilbert space-filling curve. *IEEE Transactions on Knowledge and Data Engineering*, 13(1), 2001.

[114] Guy M Morton. *A computer oriented geodetic data base and a new technique in file sequencing*. International Business Machines Company, 1966.

[115] F.J. Mulhern and R.J. Caprara. A nearest neighbor model for forecasting market response. *International journal of forecasting*, 10(2) :191--207, 1994.

[116] Beomseok Nam and Alan Sussman. A comparative study of spatial indexing techniques for multidimensional scientific datasets. In *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*, pages 171--180. IEEE, 2004.

[117] E Netto. Beitrag zur mannigfaltigkeitslehre. *Crelle J*, 86 :263--268, 1879.

[118] G. Nguyen, P. Franco, R. Mullot, and J. Ogier. Mapping high dimensional features onto hilbert curve : Applying to fast image retrieval. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 425--428, 2012.

[119] Giap Nguyen, Patrick Franco, Remy Mullot, and Jean-Marc Ogier. Mapping high dimensional features onto hilbert curve : Applying to fast image retrieval. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 425--428. IEEE, 2012.

[120] Giap Nguyen, Patrick Franco, and Jean-Marc Ogier. Space-filling curve for image dynamical indexing. In *Computer and Information Sciences III*, pages 311--319. Springer, 2013.

[121] R. Niedermeier, K. Reinhardt, and P. Sanders. Towards optimal locality in mesh-indexings. In *Fundamentals of Computation Theory*, pages 364--375. Springer, 1997.

[122] XH Niyogi. Locality preserving projections. In *Advances in neural information processing systems 16 : proceedings of the 2003 conference*, volume 16, page 153. The MIT Press, 2004.

[123] Michael G. Norman and Pablo Moscato. The euclidean traveling salesman problem and a space-filling curve. *Chaos, Solitons & Fractals*, 6(0) :389 -- 397, 1995.

[124] Ozcan Ozturk. Data locality and parallelism optimization using a constraint-based approach. *Journal of Parallel and Distributed Computing*, 71(2) :280 -- 287, 2011.

[125] Vivek S. Pai, Mohit Aron, Gaurov Banga, Michael Svendsen, Peter Druschel, Willy Zwaenepoel, and Erich Nahum. Locality-aware request distribution in cluster-based network servers. *SIGPLAN Not.*, 33(11) :205--216, October 1998.

[126] Ying Han Pang, Jin Teoh Andrew Beng, and Fazly Salleh Abas. Regularized locality preserving discriminant embedding for face recognition. *Neurocomputing*, 77(1) :156 -- 166, 2012.

[127] G. Peano. Sur une courbe, qui remplit toute une aire plane. *Mathematische Annalen*, 36(1) :157--160, 1890.

[128] A Perez, S Kamata, and E Kawaguchi. Peano scanning of arbitrary size images. In *Pattern Recognition, 1992. Vol. III. Conference C : Image, Speech and Signal Analysis, Proceedings., 11th IAPR International Conference on*, pages 565--568. IEEE, 1992.

[129] Carlo S Regazzoni and Andrea Teschioni. A new approach to vector median filtering based on space filling curves. *Image Processing, IEEE Transactions on*, 6(7) :1025--1037, 1997.

[130] Jérôme Revaud, Guillaume Lavoué, and Atilla Baskurt. Improving zernike moments comparison for optimal similarity and rotation angle retrieval. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(4) :627--636, 2009.

[131] F. Ricci and P. Avesani. Data compression and local metrics for nearest neighbor classification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(4) :380--384, 1999.

[132] Yong Rui, Thomas S Huang, and Shih-Fu Chang. Image retrieval : Current techniques, promising directions, and open issues. *Journal of visual communication and image representation*, 10(1) :39--62, 1999.

[133] H. Sagan. *Space-filling curves*, volume 2. Springer-Verlag New York, 1994.

[134] B.M. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Recommender systems for large-scale e-commerce : Scalable neighborhood formation using clustering. In *Proceedings of the Fifth International Conference on Computer and Information Technology*, pages 158--167, 2002.

[135] D. Saupe. Accelerating fractal image compression by multi-dimensional nearest neighbor search. In *Data Compression Conference, 1995. DCC'95. Proceedings*, pages 222--231. IEEE, 1995.

[136] D. Saupe. *Fractal image compression via nearest neighbor search*. Univ., Inst. für Informatik, 1996.

[137] A. Savary. Typographical nearest-neighbor search in a finite-state lexicon and its application to spelling correction. *Implementation and Application of Automata*, pages 451--455, 2002.

[138] IJ Schoenberg. On the peano curve of lebesgue. *Bull. Amer. Math. Soc*, 44 :519, 1938.

[139] Thomas Seidl and Hans-Peter Kriegel. Optimal multi-step k-nearest neighbor search. *SIGMOD Rec.*, 27(2) :154--165, June 1998.

[140] T. Sellis, N. Roussopoulos, and C. Faloutsos. The r+-tree : A dynamic index for multi-dimensional objects. In *International Conference on Very Large Data Bases*. VLDB endowments, 1987.

[141] X. Shen, Y. Zhong, and C. Ding. Predicting locality phases for dynamic memory optimization. *Journal of Parallel and Distributed Computing*, 67(7) :783--796, 2007.

[142] Ewa Skubalska-Rafajlowicz and Adam Krzyzak. Fast k-nn classification rule using metric on space-filling curves. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 2, pages 121--125. IEEE, 1996.

[143] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(12) :1349--1380, 2000.

[144] John R Smith and Shih-Fu Chang. Tools and techniques for color image retrieval. In *SPIE proceedings*, volume 2670, pages 1630--1639, 1996.

[145] Shannon Spires and Steven Goldsmith. Exhaustive geographic search with mobile robots along space-filling curves. *Collective robotics*, pages 1--12, 1998.

[146] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 2166--2176. IEEE, 2003.

[147] Y. Tao, D. Papadias, and Q. Shen. Continuous nearest neighbor search. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 287--298. VLDB Endowment, 2002.

[148] T.N. Tran, R. Wehrens, and L. Buydens. Knn-kernel density-based clustering for high-dimensional multivariate data. *Computational statistics & data analysis*, 51(2) :513--525, 2006.

[149] A. Vedaldi and B. Fulkerson. Vlfeat : An open and portable library of computer vision algorithms. In *Proceedings of the international conference on Multimedia*, pages 1469--1472. ACM, 2010.

[150] Luiz Velho and Jonas de Miranda Gomes. Digital halftoning with space filling curves. In *ACM SIGGRAPH Computer Graphics*, pages 81--90. ACM, 1991.

[151] Douglas Voorhies. Space-filling curves and a measure of coherence. *Graphics Gems II*, pages 26--30, 1991.

[152] K.Q. Weinberger, J. Blitzer, and L.K. Saul. Distance metric learning for large margin nearest neighbor classification. In *In NIPS*. Citeseer, 2006.

[153] H. Weingarten, Y. Steinberg, and S. Shamai. Gaussian codes and weighted nearest neighbor decoding in fading multiple-antenna channels. *Information Theory, IEEE Transactions on*, 50(8) :1665--1686, 2004.

[154] J. Wolf. Efficient maximum likelihood decoding of linear block codes using a trellis. *Information Theory, IEEE Transactions on*, 24(1) :76--80, 1978.

[155] M.E. Wolf and M.S. Lam. A data locality optimizing algorithm. *ACM Sigplan Notices*, 26(6) :30--44, 1991.

[156] Yong Xu, Ge Feng, and Yingnan Zhao. One improvement to two-dimensional locality preserving projection method for use with face recognition. *Neurocomputing*, 73(1–3) :245 -- 249, 2009.

[157] Liping Yang, Weiguo Gong, Xiaohua Gu, Weihong Li, and Yixiong Liang. Null space discriminant locality preserving projections for face recognition. *Neurocomputing*, 71(16–18) :3644 -- 3649, 2008.

[158] Liping Yang, Weiguo Gong, Xiaohua Gu, Weihong Li, and Yanfei Liu. Bagging null space locality preserving discriminant classifiers for face recognition. *Pattern Recognition*, 42(9) :1853 -- 1858, 2009.

[159] P.N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the fourth annual ACM-SIAM*

Symposium on Discrete algorithms, pages 311--321. Society for Industrial and Applied Mathematics, 1993.

[160] Man Lung Yiu, Yufei Tao, and Nikos Mamoulis. The b dual-tree : indexing moving objects by space filling curves in the dual space. *The VLDB Journal*, 17(3) :379--400, 2008.

[161] M. Zechner, M. Muhr, R. Kern, and M. Granitzer. External and intrinsic plagiarism detection using vector space models. *Proc. SEPLN'09*, pages 47--55, 2009.

[162] G. Zhang, G. Zhang, and S. Cheng. Lanc : locality-aware network coding for better p2p traffic localization. *Computer Networks*, 55(6) :1242--1256, 2011.

[163] H. Zhang, A.C. Berg, M. Maire, and J. Malik. Svm-knn : Discriminative nearest neighbor classification for visual category recognition. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2126--2136. IEEE, 2006.

[164] L. Zhang, Y. Zhuang, and Z. Yuan. A program plagiarism detection model based on information distance and clustering. In *Intelligent Pervasive Computing, 2007. IPC. The 2007 International Conference on*, pages 431--436. IEEE, 2007.

[165] Yuefeng Zhang and Robert E Webber. Space diffusion : an improved parallel halftoning technique using space-filling curves. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 305--312. ACM, 1993.

[166] Kun Zhou, Qiming Hou, Rui Wang, and Baining Guo. Real-time kd-tree construction on graphics hardware. *ACM Trans. Graph.*, 27(5) :126 :1--126 :11, December 2008.