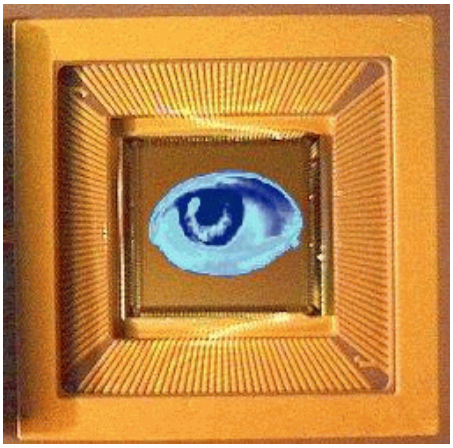
Vision artificielle rétinienne

THESE

présentée par **Antoine Manzanera**

pour obtenir le grade de docteur de
l'Ecole Nationale Supérieure des Télécommunications
spécialité : Signal et image



Soutenue le 24 Novembre 2000 devant le jury composé de :

Patrick GARDA	Président
Gilles BERTRAND	Rapporteur
Michel SCHMITT	Rapporteur
Françoise PRÊTEUX	Directrice de thèse
Thierry BERNARD	Examinateur
Bernard LONGUET	Examinateur
Yves SOREL	Examinateur

Vision rétinienne artificielle

Antoine Manzanera

Mots-clefs

Vision artificielle — Rétine programmable

Algorithmique — Architecture

Squelettisation — Ligne de Partage des Eaux

Champs de Markov — Géométrie discrète

Résumé

Dans un système de vision à base de rétine artificielle, les images sont traitées à l'endroit et à l'instant de leur acquisition. La rétine programmable est à la fois un capteur d'images et une machine SIMD, avec un processeur numérique logé dans chaque pixel. L'originalité de l'algorithmique vient d'une capacité de calcul limitée par une mémoire très faible (quelques bits par pixel). L'objectif de ce travail est de montrer qu'il est possible malgré ces limitations d'implanter des algorithmes complexes, motivant l'intégration de ce circuit dans un système de vision.

Dans une étude de complexité, nous proposons une représentation des opérateurs de rétine pour optimiser la mémoire. Nous présentons des limites théoriques fournies par la littérature. Enfin, nous montrons comment exploiter les notions de multigranularité et de composition.

Suit une étude d'implantation de deux classes d'algorithmes :

Les transformations homotopiques : nous proposons des algorithmes originaux de noyau homotopique et squelettisation adaptés à notre architecture. Outre la compacité, le squelette proposé presente une généricité géométrique et dimensionnelle. L'algorithme est prouvé et testé en 3D.

Les opérateurs géodésiques : Nous proposons une implantation compacte originale de la ligne de partage des eaux, incluant des notions de filtrage et de dynamique. Nous mettons en évidence les défauts du calcul synchrone dans ce cadre, et l'intérêt d'un fonctionnement asynchrone dans une topologie programmable, en tant que matériel dedié.

La partie applicative concerne le mouvement. Nous soulignons l'importance du codage dans ce cadre, et présentons une implantation realisée sur le prototype actuel. Puis, dans une approche alternative, nous proposons l'implantation d'un modèle markovien adapté conduisant à la mise en œuvre évidente d'une relaxation déterministe, et à une implantation très compacte du recuit simulé.

Retinal artificial vision

Antoine Manzanera

Keywords

Artificial vision — Programmable retina

Algorithmics — Architecture

Skeletonization — Watershed

Markov Random Fields — Discrete geometry

Abstract

In a vision system including an artificial retina, the images are processed where and when they are acquired. The programmable retina is both an image sensor and a SIMD machine, with a digital processor inside every pixel. The algorithmic is characterized by a computation capability limited by a tiny memory (only a few bits per pixel). The aim of this work is to show that it is possible to overcome these limitations to implant some complex algorithms on this circuit, justifying its conception as a vision device.

In a complexity study, we propose a representation of the retinal operators, for memory optimization purposes. We present theoretical limits provided by the litterature. Finally, we show how to exploit the principles of multigranularity and composition.

An implantation study for two classes of algorithms follows:

Homotopic operators: we propose original algorithms for the homotopic kernel and the skeleton adapted to our architecture. In addition to conciseness, the proposed skeleton proves to be versatile both geometrically and dimensionnally. The 3D skeleton is proved and programmed.

Geodesic operators: we propose an original concise implantation of the watershed algorithm, including filtering and dynamics notions. We emphasize the drawbacks of the synchronous computation in this context, and the interest of an asynchronous mode within a programmable topology, as a dedicated hardware.

The application-oriented part deals with movement. We show the importance of coding for temporal processing, and present an implantation realized on our prototype. Then, in an alternate approach, we propose the implantation of an adapted Markovian model leading to an easy setting of a deterministic relaxation, and to a very concise implantation of the simulated annealing.

Je tiens à exprimer ma profonde gratitude à Thierry Bernard, qui a défini les objectifs initiaux de cette thèse. Il a assuré un suivi scientifique allant bien au delà de l'encadrement, une partie importante de ce travail ayant été réalisée en étroite collaboration avec lui. Je le remercie pour l'enthousiasme qu'il m'a constamment manifesté et communiqué.

Ce travail a énormément bénéficié de l'influence de ma directrice de thèse, Françoise Prêteux. Elle a su m'accorder le temps nécessaire malgré un emploi du temps pour le moins chargé, et les pistes de recherche qu'elle m'a suggérées se sont toujours révélées heureuses. Je lui dois aussi de grands progrès dans l'expression écrite et orale d'idées scientifiques.

Bernard Longuet est à l'origine de cette thèse par son travail auprès de la direction d'Aérospatiale-Missiles pour assurer le financement dans le cadre d'un contrat CIFRE. Il m'a accordé une totale liberté dans mon travail, tout en sachant proposer au bon moment un cadre applicatif adapté. Merci à lui ainsi qu'à Nicole Cambou et Laurent Gallo pour l'intérêt qu'ils ont manifesté et diffusé au sein d'Aérospatiale.

Merci à Gilles Bertrand et Michel Schmitt, pour le lourd travail qu'ils ont réalisé en tant que rapporteurs bien sûr, mais surtout pour l'influence qu'ils ont eu sur ce travail par les idées apportées au cours de longues discussions ou réunions de travail.

Merci à Patrick Garda qui me fait l'honneur de présider le jury. J'en suis particulièrement heureux, la présente recherche s'inscrivant dans la lignée de ses travaux précurseurs.

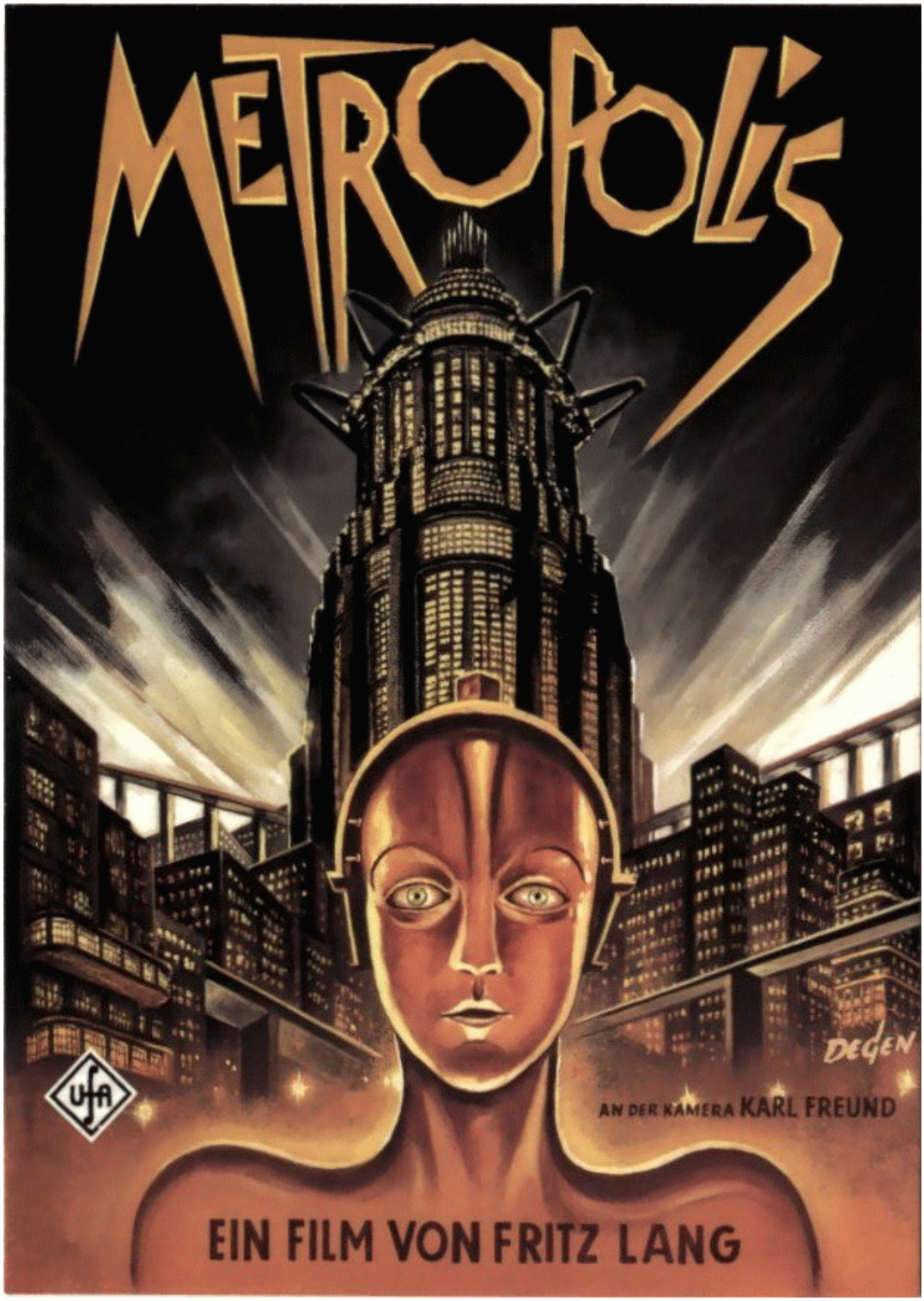
Merci à Yves Sorel pour sa participation au jury et son rôle en tant que responsable du groupe de travail A^3 du GDR ISIS, qui permet à ce travail, grâce à Michel Paindavoine également, de rayonner et de s'enrichir au sein de la communauté «rétine» française.

Je tiens à remercier les membres des différentes équipes qui m'ont accueilli dans le cadre d'une cotutelle tripartite qui fut riche sur les plans humains et scientifiques : le département Géographie Imagerie Perception du Centre Technique d'Arcueil, dirigé par Dominique Luzeaux, puis Frédéric Pradeilles ; l'unité de projets ARTEMIS de l'INT d'Evry, dirigé par Françoise Prêteux ; le service Vision de l'établissement de Chatillon d'EADS-Aéropatiale-Matra-Missiles, dirigé par Nicole Cambou.

Merci en particulier à Damien Mercier, Bertrand Collin et Fabrice Paillet, pour leur aide précieuse et leur travail qui a permis une expérimentation riche et excitante.

Enfin, je souhaiterais exprimer ma reconnaissance à tous ceux qui ont, plus ou moins volontairement, contribué à ce travail par leur réflexion. Outre les personnes déjà citées, merci à : Etienne Basset, Bernard Beauzamy, Raphaël Commaret, Gilles Darblade, Jean-Christophe Hohl, Longin Jan Latecki, Julie Vandenbusche, Luc Vincent, Jean Vuillemin, Uri Zwick... et ceux que j'oublie.

Table des matières


METROPOLIS
DEGEN
UFA
AN DER KAMERA KARL FREUND
EIN FILM VON FRITZ LANG

Introduction

Peut-on soutenir sans un certain malaise le regard du robot *Maria* de Metropolis (Fritz Lang - 1927), lorsqu'on achève une thèse en vision artificielle dans les derniers jours du deuxième millénaire et que l'on mesure l'énorme décalage entre le fantasme que suscitait cette année 2000 dans la littérature de science-fiction et la réalité de notre environnement cybernétique ?

Paradoxalement, le plus grand bouleversement de ces dernières décennies est bien l'invasion des calculateurs électroniques partout où ils sont susceptibles de seconder les humains dans l'exécution des tâches les plus complexes. Mais si la moindre calculatrice de poche est aujourd'hui capable d'effectuer un calcul d'une difficulté incommensurable pour un homme, un système de vision artificielle couplé à l'ordinateur le plus puissant ne parvient pas à approcher les performances du système visuel d'un insecte.

Aujourd'hui, un tel discours s'apparente à un truisme. Toutefois, la connaissance consciente de la véritable difficulté de la vision par ordinateur est relativement récente, puisque cette «première grande révélation» comme la nomme D. Marr [69] date des années 70. Il est probable qu'actuellement encore, la majorité des non-spécialistes considèrerait comme plus difficile que la vision artificielle de nombreuses tâches dont les ordinateurs s'acquittent depuis des années. Or, l'explication de cette ignorance fournit aussi l'une des raisons les plus fondamentales de cette difficulté : comment peut-on décrire algorithmiquement ce que l'on fait sans mettre en œuvre le moindre processus de réflexion consciente ?

C'est à ce titre que les études neurophysiologiques développées pour mieux comprendre les mécanismes de la vision chez les animaux ont suscité de grands espoirs. Sans fournir véritablement de clefs algorithmiques, ces études ont mis en évidence certaines propriétés dont la recherche en vision artificielle a pu s'inspirer.

Le concept de *rétine artificielle* qui constitue le cadre de ce travail, s'inscrit dans ce contexte. Il exploite trois caractéristiques fortes de la vision animale :

– L'information nerveuse produite au niveau des cellules photosensorielles

est traitée de manière *massivement parallèle*.

- Ce traitement fait intervenir des interactions *limitées à de petits voisinages* des informations contenues dans les cellules proches.
- L'information qui s'achemine de la surface photoréceptrice au centre nerveux le plus proche est d'une autre nature que celle produite par la phototransduction, et surtout d'un *volume beaucoup plus faible*.

Les recherches sur les rétines artificielles programmables ont débuté en France au Centre Technique d'Arcueil et à l'Université Paris XI sous l'impulsion de B. Zavidovique et de F. Devos dès le début des années 80, soit quelque temps avant d'autres travaux précurseurs menés aux Etats-Unis sur des rétines dédiées (M.I.T et C.I.T. principalement).

Dès les premiers temps, la recherche architecturale s'est accompagnée d'une réflexion algorithmique, avec les thèses de P. Garda [37] et A. Reichart [95]. L'algorithmique est alors caractérisée par un paradigme très fort lié à la notion de «B-codage» (ou codage en demi-teintes). L'intérêt de cette représentation est qu'elle permet d'envisager une large variété de traitements dans un contexte architectural alors particulièrement contraint. Mais elle confère à l'algorithmique un caractère quelque peu dogmatique.

Lorsque Th. Bernard intègre, puis prend la direction de l'équipe, il imprime une impulsion différente au projet. Il perçoit l'imminence d'une maturité technologique et l'opportunité de concentrer l'essentiel des efforts sur la recherche architecturale [7]. Ces efforts se traduisent par l'intégration de deux rétines de tailles records (65×76 en 1993 et 128×128 en 1997). Cependant, pendant près de dix ans, aucun travail algorithmique d'ampleur n'est entrepris.

Parallèlement, les responsables des services *Architecture* et *Vision* d'*Aérospatiale - missiles* (aujourd'hui *EADS*) décident de s'investir, dans le cadre d'une stratégie amont, pour financer des recherches sur les rétines. Après avoir participé à des travaux sur des rétines dédiées, ils manifestent leur intérêt au concept «tout programmable».

C'est dans ce contexte que débute cette thèse à la fin de l'année 1997. Les importants progrès architecturaux réalisés ouvrent une liberté algorithmique suffisante pour sortir du cadre des travaux précédents, et malgré tout encore assez limitée pour demeurer originale. En même temps, l'intérêt manifesté par *Aérospatiale* permet d'envisager la mise en œuvre de prochains circuits dans le cadre d'applications complètes, fournissant par là-même une saine dynamique au projet.

Le cadre contractuel de ce travail est celui d'une convention CIFRE (*ANRT - EADS*). L'encadrement est assuré par différents partenaires aux compétences complémentaires : architecture (Th. Bernard, *ENSTA/LEI*), al-

gorithmes (F. Prêteux, *INT/Unité de projets ARTEMIS*) et industrie (B. Longuet, *EADS/Aérospatiale/E/SCS/V*). Les travaux sont réalisés conjointement au département *Géographie Imagerie Perception* du Centre Technique d'Arcueil (*DGA/DCE*), et à ARTEMIS.

Les paramètres ayant principalement influencé la physionomie de ce travail sont les suivants :

- le caractère relativement vierge du sujet,
- la facette industrielle et la nécessité d'un cadre applicatif,
- la possibilité d'expérimenter sur un prototype réel.

Ce contexte nous a conduit à choisir une option plus exploratoire que fondamentaliste. Plutôt que de tenter de circonscrire une famille de traitements «adaptés» aux contraintes matérielles fortes des rétines programmables, nous avons voulu montrer qu'il était souvent possible de repenser les algorithmes dans le cadre de ces contraintes. L'objectif sous-jacent est de contribuer aux recherches sur les rétines artificielles par la démonstration de la pertinence des circuits actuels et à venir en tant que capteur de vision.

Le chapitre 1 présente le concept de rétine programmable, et détaille la problématique algorithmique abordée. Nos contributions sont développées dans les chapitres suivants. Le chapitre 2 consiste en une étude de la complexité. Les chapitres 3 et 4 concernent l'implantation de deux classes d'algorithmes, les transformations homotopiques et les opérateurs géodésiques. Enfin, le chapitre 5 expose le cadre applicatif et en développe l'enjeu scientifique principal qui est celui de la détection de mouvement.

Chapitre 1

La rétine numérique

1.1 Introduction

Dans ce chapitre sur le concept de rétine artificielle programmable, nous soulignons tout d'abord les avantages offerts par un tel circuit dans une machine de vision, puis nous indiquons les aspects technologiques qui rendent son émergence possible.

Nous présentons ensuite un état de l'art synthétique des circuits pouvant être qualifiés de «rétines artificielles», en situant la rétine artificielle programmable, objet de notre recherche, dans ce vaste panorama.

Enfin, nous ciblons notre propos sur l'algorithmique de la rétine, en spécifiant ses principes et son originalité, mettant l'accent sur les objectifs et les enjeux scientifiques de notre recherche.

1.2 Intérêt du concept de Rétine Artificielle

Dans une application de vision par ordinateur, les calculateurs doivent supporter des algorithmes de complexité élevée, appliqués à de gros volumes de données fournies à très haut débit par les capteurs (en général des caméras CCD). Dans une approche classique, le signal bidimensionnel acquis par le capteur est transmis séquentiellement via un convertisseur analogique-numérique, à une unité de calcul.

Le calculateur doit alors fournir une réponse, résultat de l'interprétation de la scène observée. Ce processus (figure 1.1) s'effectue à partir :

- d'opérations de *bas niveau*, qui transforment localement l'image en une autre image plus simple. Caractérisées par de grands volumes de données à traiter, elles se prêtent heureusement à une parallélisation potentiellement massive.

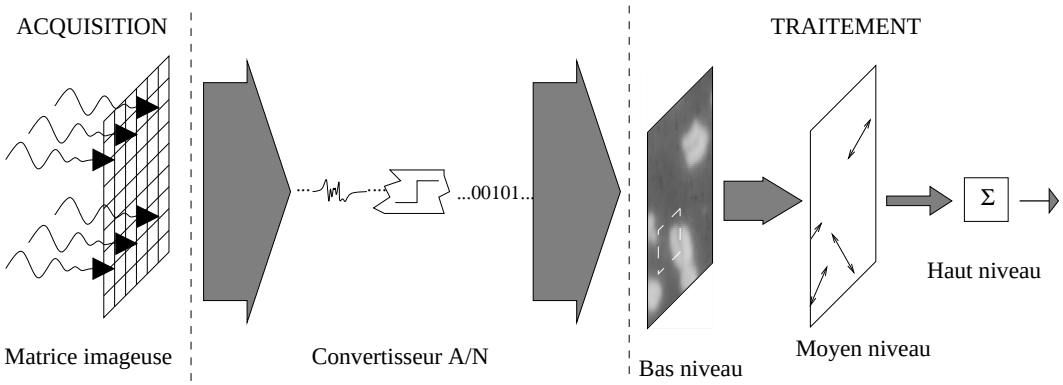


FIG. 1.1 – *Schéma synoptique d'une application de vision sur une architecture classique. L'épaisseur des flèches est proportionnelle au débit du volume de données entre les différentes phases.*

- d'opérations de *moyen niveau*, qui agissent sur des données plus structurées mais qui gardent une distribution bidimensionnelle. Le volume d'information traité est beaucoup moins important que précédemment, et le caractère local et parallèle se révèle plus critique.
- d'opérations de *haut niveau* enfin, qui agissent sur des données symboliques, de faible volume et dépourvues de structure spatiale régulière.

La figure 1.1 met en évidence un point faible de l'architecture d'un système de vision classique : le goulot d'étranglement subi par le flux d'information entre acquisition et traitement. L'image est «sérialisée» pour être transmise à travers un canal zéro-dimensionnel vers l'unité de traitement, laquelle lui redonne immédiatemment sa structure bidimensionnelle nécessaire aux opérations de bas-niveau. Cette limitation du système induit des performances faibles ou dégradées en terme de :

- *vitesse* d'exécution, indispensable en contrôle industriel,
- *compacité*, requise en robotique pour les systèmes embarqués,
- *consommation* d'énergie, primordiale pour les systèmes autonomes.

Cette constatation conduit naturellement à tenter d'effectuer le maximum de calculs sur l'image à l'endroit même où elle est acquise. C'est dans ce cadre que s'inscrivent les *rétines artificielles*, définies comme des circuits dans lesquels les fonctions d'acquisition, de codage et de traitement sont intimement rapprochées.

Bien que cette solution paraisse naturelle au vu des difficultés évoquées plus haut, il faut noter que les techniques d'intégration ne rendent possible (ou du moins raisonnable) une telle cohabitation entre fonctionnalités d'ac-

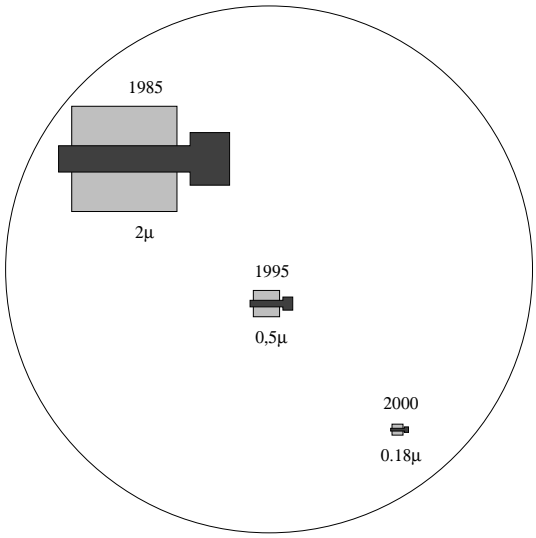


FIG. 1.2 – *Evolution des technologies standards du transistor MOS sur 15 ans. Le cercle de rayon* 10μ *permet de mesurer les possibilités offertes par les progrès de la microélectronique quant à l'avènement de rétines artificielles.*

quisition et de calcul que depuis quelques années. Aujourd'hui, les rétines artificielles jouissent d'une certaine maturité technologique : la taille du pixel des capteurs doit demeurer supérieure à quelques microns en raison des problèmes de diffraction optique, tandis que la miniaturisation des transistors CMOS se poursuit et évolue dans le domaine submicronique (figure 1.2), la technologie silicium $0,18\mu$ étant en voie de devenir le standard.

Il devient possible en conséquence de loger quelques dizaines de transistors dans le pixel sans dégrader sensiblement la résolution du capteur.

Le terme «rétine» suggère bien entendu une tendance bio-inspirée qui s'est dégagée depuis quelques années en vision artificielle, et dont notre approche n'est pas exempte. Il ne faudrait pas voir pour autant la biologie comme source d'inspiration privilégiée des rétines artificielles (même si certains circuits sont directement inspirés de la vision de la mouche [79] ou si certains capteurs «fovéaux» reproduisent la distribution des photocapteurs chez les vertébrés [35]). En revanche, l'étude de la vision chez les animaux est un guide riche d'enseignement dans le cadre de la recherche sur les capteurs intelligents : un certain nombre de traitements effectués au niveau de la rétine permet de réduire considérablement le flux d'information entre ce qui est reçu par l'œil et ce qui est acheminé par le nerf optique.

Cette réduction de flux fonde un principe essentiel des systèmes de vision à base de rétine artificielle, comme illustré sur la figure 1.3, où la largeur des

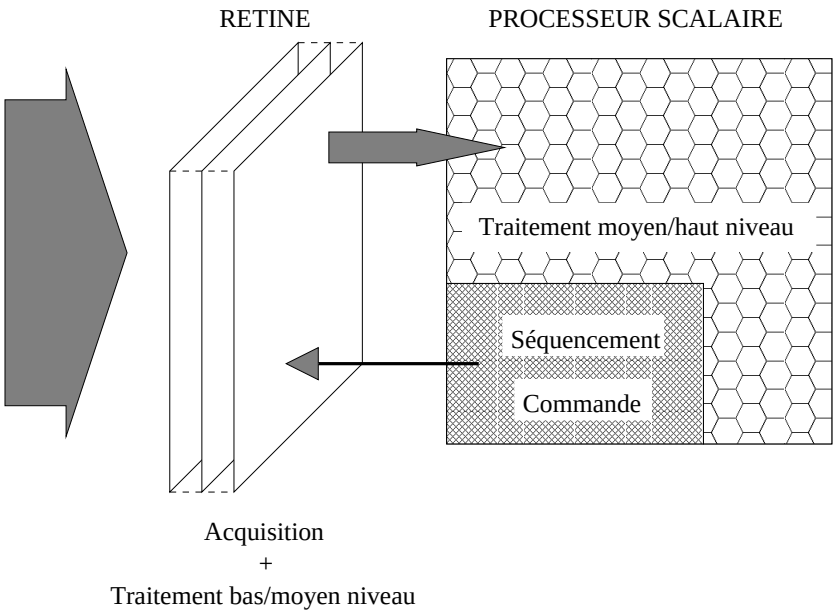


FIG. 1.3 – *Le monolithe rétinien et son partenaire scalaire. La largeur des flèches est proportionnelle au débit d'information.*

flèches est proportionnelle au débit d'information. Le traitement effectué sur la rétine constitue donc une aggrégation des données d'origine image en un ensemble de données scalaires qui seront interprétées et traitées par le processeur chargé de la vision haut niveau. Dans ce schéma, le même processeur est chargé de «piloter» la rétine en lui envoyant les séquences d'instructions qu'elle doit exécuter. Cette solution n'a cependant rien d'obligatoire, et le pilotage du circuit peut être effectué par une unité séparée.

Les rétines artificielles sont donc caractérisées par l'intégration sur le lieu même des photodétecteurs de fonctionnalités d'ordre supérieur, que l'on qualifiera d'*intelligence dans le plan focal*. Cette intelligence, nous allons le voir, peut prendre des formes très différentes et mener à des circuits à vocations peu comparables. Nous allons tenter de nous positionner au sein de ces diverses approches.

1.3 Tour d'horizon des rétines artificielles

Nous dégageons dans cette section les principales caractéristiques des différentes catégories de rétines artificielles, en situant dans ce cadre la rétine programmable et son originalité.

1.3.1 Intelligence dans le plan focal ou dans le pixel

Plusieurs circuits de type «capteurs intelligents» intègrent les fonctionnalités de calcul au bord de la matrice imageuse, le plus souvent sous la forme d'une ligne (resp. colonne) de processeurs traitant en parallèle les données extraites séquentiellement des colonnes (resp. lignes) de pixels [36], [41], [80].

La rétine artificielle programmable va beaucoup plus loin dans le rapprochement entre acquisition et calcul. Elle exploite de manière directe la distribution bidimensionnelle du signal capté en intégrant les fonctionnalités de traitement au niveau de chaque photocapteur. C'est le principe de l'*intelligence dans le pixel*, qui constitue une caractéristique fondamentale de notre approche.

1.3.2 Rétine analogique ou numérique

Si l'on recense les principales équipes de recherche travaillant sur des projets de rétine artificielles, force est de constater que la grande majorité concerne des circuits purement *analogiques*. Citons le California Institute of Technology, l'Université John Hopkins et le Massachussets Institute of Technology aux U.S.A., l'Université d'Adelaïde en Australie, le Centre Suisse d'Electronique et Microtechnique de Neuchâtel, l'Université de Séville et l'Université Catholique de Louvain en Europe, l'Institut d'Electronique Fondamentale d'Orsay et l'Institut National Polytechnique de Grenoble en France.

De fait, la physique du transistor MOS est d'une grande richesse, et offre souvent des solutions élégantes et compactes pour mettre en œuvre des opérations de bas niveau sur les images. Un circuit analogique utilise le transistor comme un élément complexe et exploite en particulier ses différents modes (faible/forte inversion), à l'inverse du circuit numérique qui utilise le transistor de manière binaire (comme un interrupteur commandé électriquement). Parce qu'il épouse certains comportements physiques du silicium, le circuit analogique utilise en général un moins grand nombre de transistors que le circuit numérique pour une même fonction spécifique.

Dans les approches *synthétiques*, des fonctions analogiques de base sont appliquées pour obtenir des traitements bas niveau sur les images. Des réseaux résistifs ou capacitifs sont utilisés pour du filtrage passe-bas. Par différenciation, on peut obtenir une image de gradient spatial, puis de contours par détection des passages par zéro. Les réseaux résistifs permettent également d'obtenir des informations sur l'orientation et la position d'objets par calcul de moments. Les circuits de corrélation servent dans l'appariement d'images de stéréovision, ainsi que dans de nombreux circuits spatio-temporels, où,

associés à des lignes à retard ou des mémoires analogiques, ils permettent de calculer le flot optique (analyse du mouvement), sur lequel d'autres traitements peuvent être effectués, comme la détection de singularités. Un exemple remarquable par sa simplicité, dû à Delbrück [31], est exposé figure 1.4.

Dans les approches *bio-mimétiques*, en revanche, la conception du circuit est entièrement fondée sur une tentative de reproduction en silicium de fonctions observées dans les rétines animales, principalement chez les insectes (perception du mouvement, inhibitions latérales, vision périphérique/fovéale).

En résumé, les circuits analogiques constituent des solutions peu consommatrice en ressources pour des traitements précis. Au vu des multiples exemples déjà réalisés, leur domaine d'application privilégié semble être la détection et l'analyse des variations spatio-temporelles dans le signal image.

1.3.3 Approche spécifique ou programmable

Les circuits analogiques évoqués précédemment assurent des fonctions spécifiques. Dans le contexte de réduction du flux d'information qui a principalement motivé l'apparition des rétines artificielles, cela soulève le problème de la polyvalence : d'une part, certaines opérations de réduction, telles que la représentation des formes ou la détection de points d'intérêt liés à la géométrie semblent faire défaut aux techniques analogiques, d'autre part, la résolution d'un problème de vision par un système spécifique n'est possible que si l'environnement associé est parfaitement maîtrisé, ce qui réduit fortement le domaine d'action de tels circuits.

Pour autant, rétine analogique n'est pas nécessairement synonyme de rétine dédiée. Les CNN (Cellular Non-linear Networks), ou réseaux de neurones cellulaires, une architecture de rétine analogique, prétendent à l'universalité. L'image de départ correspond à un état initial du réseau de neurones, l'image résultat à l'état de convergence du réseau sous l'action d'une équation d'évolution faisant intervenir un certain nombre de noyaux de convolutions [34]. Bien que plusieurs opérateurs de traitement d'image aient été ainsi implantés, il semble que certains problèmes matériels rendent critique la précision du calcul pour les circuits de grande taille.

Parmi les principes qui régissent notre approche, la *polyvalence*, objectif que nous cherchons à atteindre grâce à la *programmabilité*, est tout à fait fondamentale. Nous pensons qu'une véritable programmabilité nécessite une parfaite fiabilité quant à la manipulation et au stockage de l'information; celle-ci doit en particulier pouvoir «parcourir n'importe quelle distance» dans l'espace-temps sans être altérée. A l'heure actuelle, seul un traitement *numérique* de l'information permet d'obtenir ce résultat.

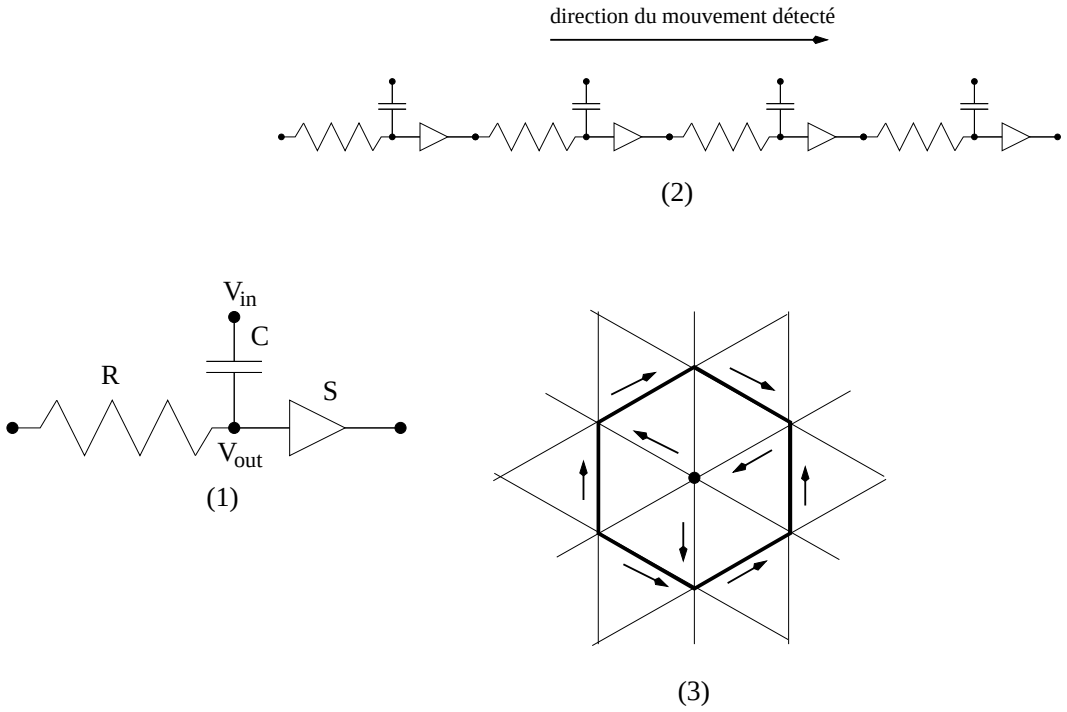


FIG. 1.4 – *La cellule de base (1) de Delbrück pour la détection de mouvement par filtrage spatio-temporel [31] est constituée d'une résistance (R), une capacité (C) et un suiveur (S). Une suite de tels montages identiques est assemblée pour former des* lignes à retard *(2) Pour chaque cellule, la tension d'entrée V_{in} provient d'un photocapteur. La tension de sortie V_{out} provient d'un filtrage passe-haut de la tension d'entrée et d'un filtrage passe-bas de la tension de sortie fournie par le suiveur du montage voisin. Finalement le signal constitué des tensions de sorties d'une ligne à retard détecte les mouvements qui se produisent dans la direction de la ligne, dans le sens des suiveurs, et à une vitesse correspondant à la constante de temps RC. Delbrück a assemblé une rétine dédiée au calcul du flot optique en croisant 3 lignes à retard sur chaque pixel (3). Le capteur est en maille hexagonale, le sens des lignes à retard est indiqué par les flèches.*

Notre approche est par conséquent résolument numérique. Cela constitue en soi une originalité, puisqu'à notre connaissance seules quatre équipes de recherche ont assemblé des rétines numériques : l'Université de Linköping [33], l'Université de Tokyo [51], [47], l'Institut d'Electronique Fondamental d'Orsay [39] et le CTA/GIP [12], [85].

Parmi ces réalisations, la rétine artificielle programmable du CTA/GIP se démarque encore par l'originalité de son architecture, moins «typiquement numérique» que celle de ses concurrents. Ceux-ci utilisent une architecture classique pour leur processeur élémentaire présent au sein de chaque pixel, avec des registres RAM statiques pour la mémoire. L'avantage d'un tel choix est de pouvoir adresser indépendamment les différents registres pour n'importe quel calcul, ce qui permet de gagner en rapidité d'exécution, en consommation d'énergie (peu de circulation d'information pour une opération élémentaire) et en facilité de programmation (code orthogonal). En revanche, cette solution impose de commander tous les registres indépendamment, soit de manière physique en leur attribuant à chacun un câble de commande, ce qui a pour effet d'envahir le circuit de rails de métal, soit de manière logique, en intégrant une unité de décodage au sein du processeur élémentaire, ce qui augmente dramatiquement le nombre de transistors...

La rétine du CTA/GIP utilise quant à elle les registres semi-statiques à décalage, qui conduisent à des registres spécialisés, une programmation plus complexe et une plus grande circulation d'information nécessaire à la réalisation d'une opération élémentaire, pouvant mener à une complexité de calcul et une consommation d'énergie accrue. En contrepartie, un très petit nombre de rails de métal suffisent à commander le processeur élémentaire, et aucun circuit de décodage n'est nécessaire.

Ce choix matériel a des répercussions importantes sur la densité d'intégration, comme le montre la figure 1.5. Ce graphique donne le nombre de pixels ainsi que le nombre de bits de mémoire par pixel pour quatre circuits numériques récents : *Bernard 93*, premier circuit conçu au CTA/GIP [12], *Eklund 96*, rétine de l'Université de Linköping [33], *Ishikawa 99*, dernier circuit de l'Université de Tokyo [47]. *Pvlsar2.2* est le dernier circuit conçu au CTA/GIP, objet principal de notre recherche [85]. La taille des points sur la figure 1.5 est proportionnelle à la mesure de compacité *aire/mémoire* : c'est l'aire de la surface utilisé par bit de mémoire exprimé en λ^2 (c'est-à-dire indépendamment de la technologie puisque le λ est égal à une demi-longueur de transistor minimal).

Bien que peu nombreuses encore, les différentes équipes citées ont mis en évidence un éventail assez important de possibilités techniques pour l'intégration de rétines numériques, grâce en particulier au caractère très diversifié de leurs approches. Cependant, seules les rétines du CTA/GIP, et en parti-

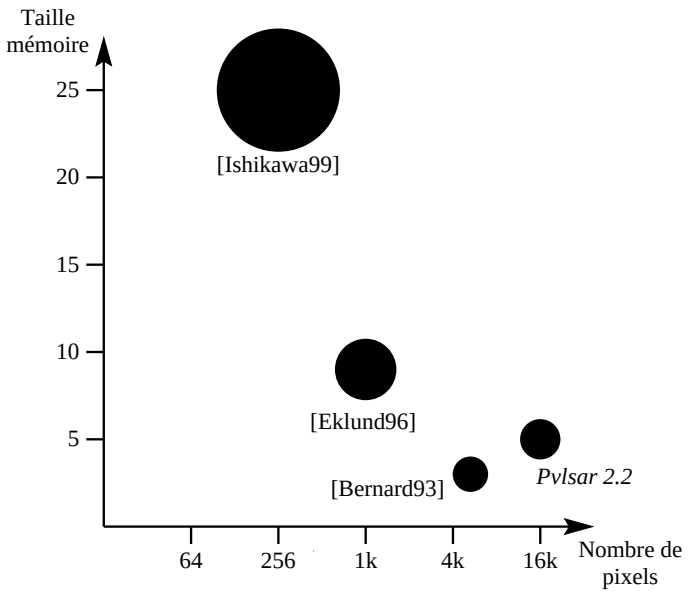


FIG. 1.5 – *Comparaison de la compacité des différentes rétines artificielles numériques de 1993 à 1999.*

culier *Pvlsar2.2* permettent à ce jour une expérimentation raisonnable pour un travail de recherche algorithmique tel que le nôtre.

1.3.4 Choix matériels et description

Pour un véritable état de l'art des rétines artificielles, on se reportera à [78] et [8]. Après le bref survol que nous venons de faire de ce domaine, il apparaît que la solution «idéale» pourrait résider dans la mixité analogique-numérique, comme l'illustre la figure 1.6. Puisque l'on ambitionne une véritable programmabilité, la plus grande partie du traitement doit être numérique, mais un certain nombre de fonctions pourrait être paramétré au niveau d'une «couche» analogique, qui soulagerait sensiblement la «couche» numérique. On peut ainsi envisager que l'information qui parvient sur la grille de processeurs numériques soit par exemple une image lissée, une image de gradient, ou encore une image d'intensité du flot optique...

Cependant la coexistence aussi intime de fonctions numériques et analogiques ne va pas sans d'importantes difficultés technologiques, jugées un peu lourdes au regard de la jeunesse du projet. Par conséquent, les deux rétines artificielles ayant été conçues au CTA/GIP sont des circuits essentiellement numériques. En particulier, les opérations de base effectuées sur les images sont exclusivement booléennes. La figure 1.7 montre l'architecture générale

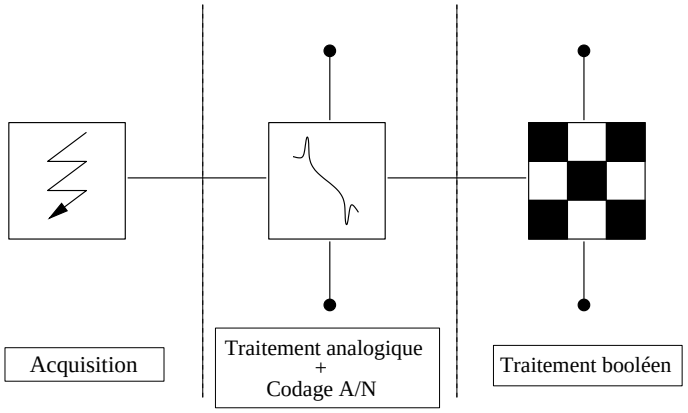


FIG. 1.6 – *Le pixel intelligent = Le processeur qui voit.*

de la Rétine Programmable : il s'agit d'une grille bidimensionnelle de processeurs élémentaires (PE) interconnectés selon une maille carrée. Chaque PE est équipé d'un photocapteur capable d'intégrer la lumière reçue par une surface photosensible et de la transformer en courant électrique, c'est la *phototransduction*. Ce courant est codé sur un certain nombre de registres binaires, c'est la *conversion analogique-numérique*. Le signal numérique bidimensionnel ainsi obtenu est traité grâce aux opérations logiques et aux registres-mémoire disponibles au sein du PE.

C'est essentiellement de cette dernière partie qu'il va être question tout au long de ce rapport. L'étude de l'*algorithmique rétinienne* consiste à tirer le meilleur de cette architecture pour le calcul d'opérateurs de vision. Or, bien que du point de vue algorithmique, un certain nombre de caractéristiques soient familières (parallélisme SIMD, réseau d'automates cellulaires...), nous verrons vite l'omniprésence d'une contrainte beaucoup plus exotique, et qui constitue l'enjeu scientifique de cette recherche : la gestion d'une mémoire de calcul extrêmement réduite. En effet, la mémoire disponible au sein de chaque PE correspond à quelques bits ! Par exemple *Pvlsar2.2* (figure 1.8) a cinq bits de mémoire par PE.

Nous arrivons à présent au cœur de notre recherche : la description de l'algorithmique rétinienne.

1.4 Le traitement booléen dans la rétine

Cette section a pour objet de présenter les principes de base de l'algorithmique rétinienne. Or, en tant qu'architecture parallèle SIMD (Single

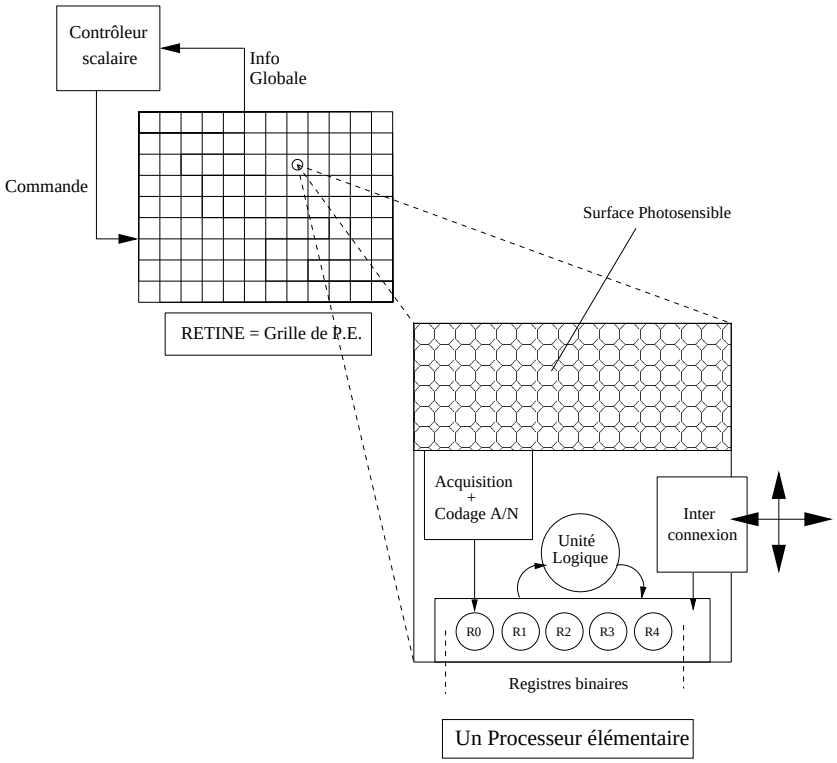


FIG. 1.7 – *Architecture symbolique de la rétine numérique.*

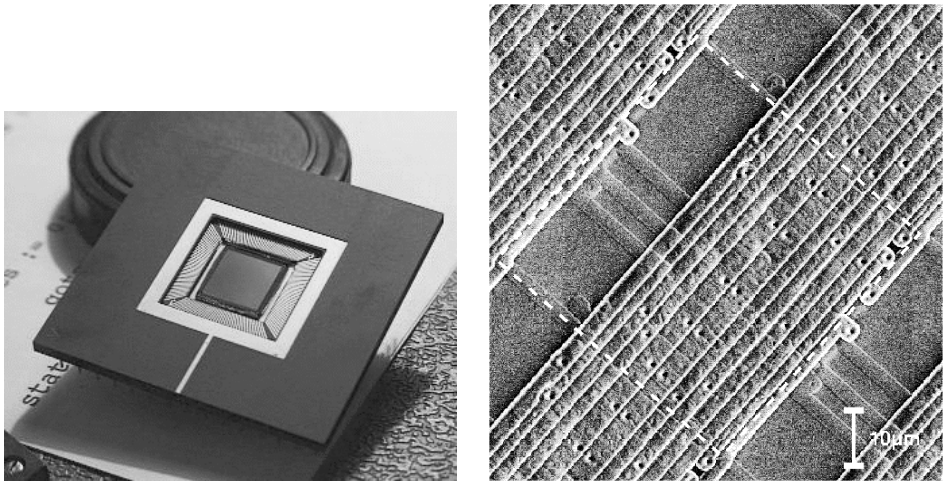


FIG. 1.8 – *Le circuit Pvlsar2.2 et un pixel vu au microscope électronique à balayage.*

Instruction Multiple Data), la rétine présente une caractéristique très conditionnante : la taille extrêmement réduite de la mémoire disponible au sein de chaque processeur.

Cette propriété implique une *représentation binaire de l'information*, par opposition aux représentations numériques multi-types (octet, entier, flottant, double). Pour la même raison, tandis que pour un algorithme manipulant l'information sur une architecture courante, les opérations de base (celles qui sont implantées matériellement) sont les opérations *arithmétiques*, sur la rétine programmable, les seules opérations de base sont les *opérateurs booléens*.

Par conséquent, le formalisme décrivant les algorithmes pour la rétine programmable est celui des *treillis booléens*. Ce principe fonctionnel de *granularité minimale* constitue en fait l'originalité majeure de l'algorithmique que nous allons présenter. Il implique d'importantes difficultés, bien sûr, pour repenser et adapter des algorithmes existants dans le but d'une implantation sur notre circuit. Mais il permet aussi une *polyvalence maximale*, principe même qui a motivé l'intégration de rétines programmables.

1.4.1 Formalisme

On représente l'information booléenne dans la rétine par la donnée de n *plans booléens* (n images binaires), n étant le nombre de bits de mémoire disponibles par processeur (figure 1.9).

Pour cette structure de données, on peut effectuer les actions élémentaires suivantes :

- Translater un plan d'un pixel dans l'une des quatre directions cardinales N, E, S, W,
- Effectuer le calcul d'une fonction booléenne à deux variables (ET, OU, OU exclusif, ...) entre deux plans.

Du point de vue algorithmique, la rétine est une machine parallèle SIMD, ce qui signifie que tous les processeurs effectuent de manière synchrone la même séquence d'actions élémentaires.

Un *Opérateur de Rétine* ou *Opérateur Rétinotopique* est défini par la composition d'un nombre fini d'actions élémentaires.

Tout au long de ce rapport, nous utiliserons indifféremment deux types de notation pour les opérations élémentaires : *ensembliste* ou *booléen*, selon la commodité de l'un ou l'autre. Le tableau 1.1 explicite l'équivalence de ces deux formalismes.

$\mathbb{Z}^2$ désigne le *plan discret*, auquel est identifiée la *rétine formelle*. Un *pixel* est un point de $\mathbb{Z}^2$.

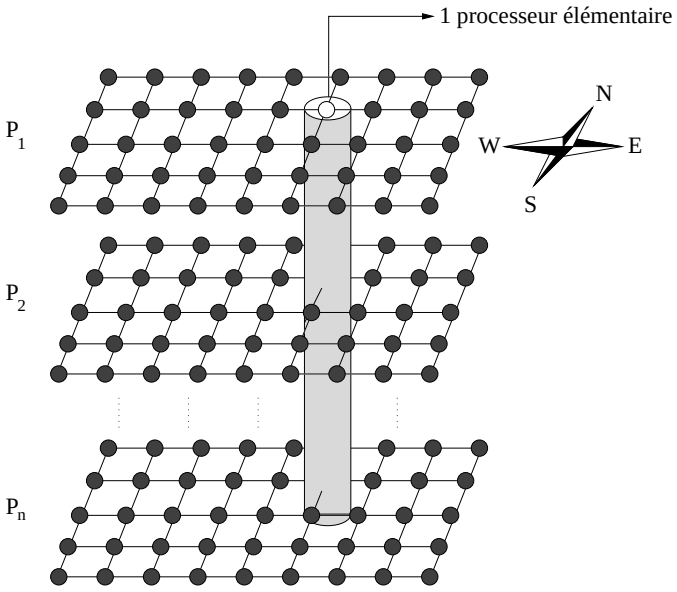


FIG. 1.9 – *L'information binaire dans la rétine.*

Du point de vue ensembliste, un *plan* ou une *image* binaire est une partie de $\mathbb{Z}^2$. Les opérations de base sur les images sont les opérations sur les ensembles.

Du point de vue booléen, un plan binaire est une fonction de $\mathbb{Z}^2$ dans {0,1}. Soient $p_1, \ldots, p_n$ n plans binaires et $\pi = (p_1, \ldots, p_n)$ la fonction vectorielle associée de $\mathbb{Z}^{2n}$ dans $\{0,1\}^n$. Si f est une fonction booléenne à n variables, on notera $f(p_1, \ldots, p_n) = f \circ \pi$.

Le tableau 1.1 montre le passage d'une notation à l'autre : si p est une image booléenne et I la même image en ensembliste, alors p est la fonction indicatrice de I, et I est l'image réciproque de 1 par p.

Quant aux translations élémentaires, nous les noterons par l'indexation d'une direction cardinale. Attention toutefois, nous utiliserons la convention suivante : p_{ouest} (ou p_W), représente la valeur du plan p à l'ouest, donc p_W est égale à p translaté d'un pixel... vers l'est !

Les bases de l'algorithmique rétinienne étant posées, nous allons voir dans les deux parties suivantes les conséquences sur le traitement d'images dans la rétine. Garda [37], [38] a montré les liens très étroits qui unissent l'algorithmique d'une rétine booléenne et la *morphologie mathématique* [106]. De fait, la majorité des algorithmes qui seront présentés ici sont issus de la morphologie.

Nous allons exposer la nature de ces liens, d'abord dans le cadre du traite-

FORMALISME	
BOOLEEN	ENSEMBLISTE
IMAGE	
$p \in \{0,1\}^{\mathbb{Z}^2}$	$I \in \mathcal{P}(\mathbb{Z}^2)$
$I = p^{-1}(1); p = \mathbf{1}_I$	
OPERATIONS DE BASE	
disjonction	
OU logique	Union
$p_1 \vee p_2$	$I_1 \cup I_2$
conjonction	
ET logique	Intersection
$p_1 \wedge p_2$	$I_1 \cap I_2$
complémentation	
NON logique	Complémentaire
$\bar{p}$	I^c
disjonction exclusive	
XOR logique	Différence symétrique
$p_1 \triangle p_2$	$I_1 \Delta I_2$
différence	
ET NON logique	Différence
$p_1 \bar{\rightarrow} p_2$	$I_1 \setminus I_2$

TAB. 1.1 – *Correspondance entre notations booléenne et ensembliste.*

ment binaire, puis dans celui des images en niveaux de gris. Cette séparation n'a pas seulement un caractère didactique, mais relève de deux philosophies différentes de l'adéquation algorithme-architecture. Dans le premier cas, une approche pragmatique (que *peut*-on faire?) conduit à constater que, disposant d'une telle architecture, les opérations les plus simples qu'on peut effectuer sur des images sont les opérateurs de base de la morphologie. Dans le deuxième cas, une approche idéaliste (que *veut*-on faire?) amène à penser que le principe de fusion des fonctions d'acquisition et de calcul suggère un traitement non linéaire, et ici encore la morphologie offre un cadre parfaitement adapté.

1.4.2 Traitement d'images binaires

A la faveur de l'abaissement constant du coût du transistor, les rétines programmables ravivent les liens forts qui ont uni il y a quelques années morphologie mathématique et architecture SIMD, avant que de nouvelles idées algorithmiques [103], [117] ne viennent en faciliter l'implantation sur machine standard.

Notons d'abord que les opérateurs de rétines vérifient les deux premiers principes de la morphologie mathématique :

- *Invariance en translation.* La nature du traitement est la même en tout point de $\mathbb{Z}^2$, ce qui correspond au fonctionnement SIMD de la rétine.
- *Connaissance locale.* La valeur calculée en un point ne dépend que d'un voisinage borné de ce point, comme dans la rétine, le calcul effectué par un processeur se réfère à un nombre fini de valeurs prises dans le voisinage du pixel.

De plus, les opérations de base de la morphologie mathématique sont définies, comme pour les opérateurs de rétines, sur des images binaires, et de ce fait, elles correspondent aux traitements d'images les plus simples pour la rétine.

Dans ce qui suit, nous donnons la définition des opérations de base en morphologie que sont l'érosion et la dilatation. X est une image binaire, B représente un voisinage de calcul appelé *élément structurant.*

Définition 1 *X et B deux sous-ensembles de $\mathbb{Z}^2$.*
*L'*addition de Minkowski *de X et B est l'ensemble :*

$$X \oplus B = \bigcup_{b \in B} X_b.$$

La soustraction de Minkowski *de X par B est l'ensemble :*

$$X \ominus B = \bigcap_{b \in B} X_b.$$

où X_b désigne le translaté de X par b.

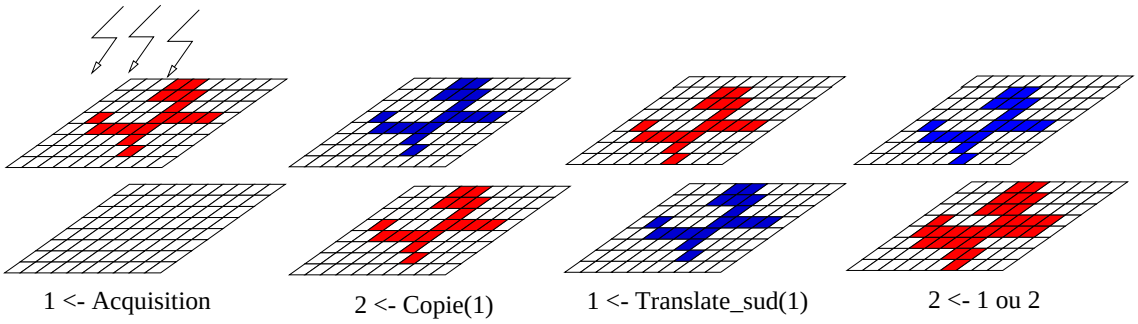


FIG. 1.10 – *Une dilatation sur la rétine.*

Définition 2 *On note* $\check{B} = -B$ *le symétrique de* B *par rapport à l'origine. L'*érodé *de* X *par* B *est défini par la soustraction de* X *par* $\check{B}$ *:*

$$X \ominus \check{B} = \{u \in \mathbb{Z}^2; B_u \subset X\}.$$

Le dilaté *de* X *par* B *est défini par l'addition de* X *et* $\check{B}$ *:*

$$X \oplus \check{B} = \{u \in \mathbb{Z}^2; B_u \cap X \neq \emptyset\}.$$

On utilisera en permanence les propriétés - immédiates - suivantes :

Propriété 1 *La dilatation et l'érosion sont deux opérations duales pour la complémentation :*

$$X \ominus B = (X^c \oplus B)^c.$$

La dilatation est associative *:*

$$(X \oplus B) \oplus B' = X \oplus (B \oplus B').$$

et donc :

$$(X \ominus B) \ominus B' = X \ominus (B \oplus B').$$

D'après le tableau 1.1, on voit que la dilatation (resp. érosion) se calcule au moyen d'un OU (resp. ET) sur le voisinage défini par l'élément structurant. Le calcul peut se faire sur la rétine avec deux plans binaires. De plus, la propriété d'associativité va permettre, comme nous le verrons au chapitre suivant, d'optimiser le calcul par décomposition des éléments structurants.

La figure 1.10 montre par exemple une dilatation par l'élément structurant $\{(0,0);(0,1)\}$ effectué sur la rétine, opération par opération.

La notion de *transformée en tout-ou-rien* (TTR) unifie et généralise les opérateurs d'érosion et de dilatation :

Définition 3 *Soient* H *et* M *deux sous-ensembles de* $\mathbb{Z}^2$ *tels que* $H \cap M = \emptyset$. *La* transformée en tout-ou-rien *(TTR) de* X *par le couple* (H, M) *est l'ensemble :*

$$X \circledast (H, M) = (X \ominus \check{H}) \cap (X^c \ominus \check{M}).$$

En termes d'images, la TTR correspond à une recherche de *configuration* dans une image binaire : l'ensemble H (resp. M) représente les points à 1 (resp. à 0) d'une imagette (configuration), dont l'origine prend la valeur 1 par la TTR.

En langage booléen, la TTR s'identifie á un *Monôme Conjonctif* f calculé sur deux voisinages distincts du pixel x, $H = \{h_i\}_{1 \leq i \leq n}$ et $M = \{m_j\}_{1 \leq j \leq p}$ tels que :

$$f(x) = (\bigwedge_{i=1}^{n} h_i) \wedge (\bigwedge_{j=1}^{p} m_j^c).$$

Elle est aussi calculée sur deux plans binaires dans la rétine.

Finalement, toute opération morphologique sur une image binaire s'exprime de manière canonique sous forme d'une union finie de transformées en tout ou rien (UTTR), donc définie par la famille $\{H_i, M_i\}_{i=1...N}$.

Dans le formalisme booléen, une UTTR correspond à une *forme disjonctive*, c'est-à-dire une disjonction de monômes conjonctifs :

$$f(x_1, \ldots, x_n) = \bigvee_{j=1}^{N} [\wedge_{i=1}^{n_j} \tilde{x_i}], \text{ avec } \tilde{x_i} = x_i \text{ ou } x_i^c.$$

Or, toute fonction booléenne peut s'écrire sous *forme normale disjonctive* (FND), qui est une forme canonique. Une forme canonique de fonctions booléennes a une importance particulière pour l'algorithmique rétinienne, car elle correspond à un «programme» qui peut être implanté de manière automatique sur la rétine. La figure 1.11 montre comment une FND est calculée sur la rétine : le plan du haut contient l'image binaire de départ. Pour chaque monôme, on lui fait décrire par des translations l'ensemble des variables qui constituent le monôme, en inversant l'image chaque fois que la variable correspondante apparaît complémentée. On calcule le ET logique de l'ensemble des variables de chaque monôme sur le plan du milieu, puis on calcule le OU logique de l'ensemble des monômes sur le plan du bas.

La figure 1.12 donne quelques exemples simples d'opérateurs de rétines représentés dans leur formalisme UTTR. Chaque élément géométrique représente un $H_i \cup M_i$. H_i est alors l'ensemble des points noirs, M_i l'ensemble des points blancs. Le point en gras est l'origine.

Les formalismes de l'algorithmique rétinienne et celui de la morphologie s'expriment tous deux dans le cadre des treillis booléens. Grâce à ce point commun, les traitements d'images rétinien de base coïncident avec les opérateurs fondamentaux de la morphologie.

Mais nous savons que les opérations morphologiques se généralisent au delà du cadre binaire. La rétine programmable, elle, est capable d'acquérir des images en niveaux de gris. Voyons à présent comment nous pouvons généraliser les traitements rétiniens aux images numériques.

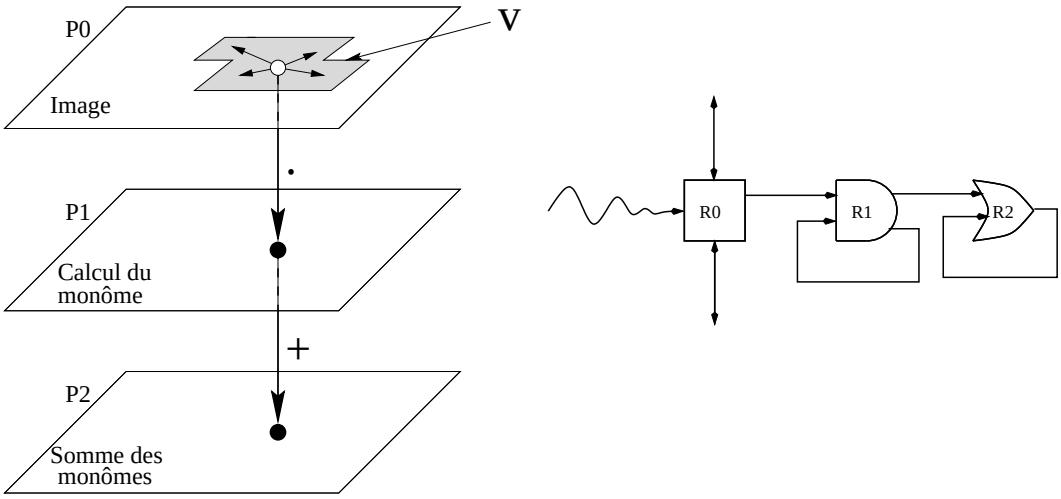


FIG. 1.11 – *Calcul d'une Forme Normale Disjonctive sur 3 plans binaires.*

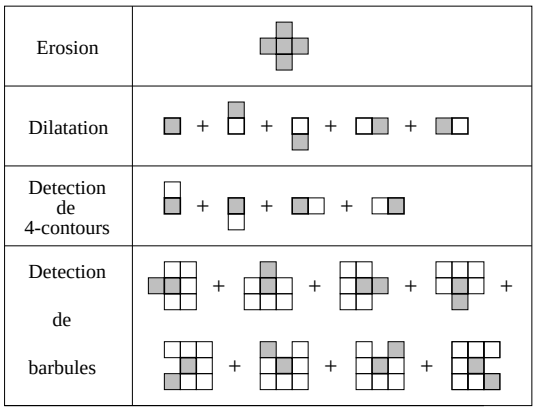


FIG. 1.12 – *Quelques opérateurs de rétines exprimés sous forme disjonctive.*

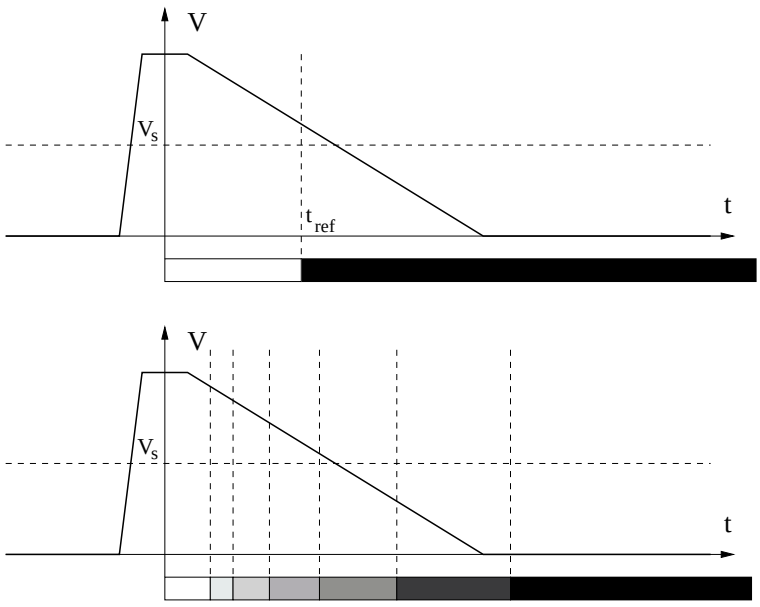


FIG. 1.13 – *Acquisition binaire et acquisition en niveaux de gris grâce au multiseuillage.*

1.4.3 Acquisition et traitement d'images en niveaux de gris

Le processus d'acquisition sur la rétine produit une image binaire représentant les niveaux d'éclairement des pixels vis-à-vis d'un seuil prédéfini. Dans le circuit, on utilise une *photodiode* préchargée à une certaine tension, qu'on laisse ensuite se décharger. C'est ici qu'intervient la photosensibilité : la tension aux bornes de la photodiode va décroître de manière plus ou moins linéaire, *à une vitesse proportionnelle au débit de photons captés*. On obtient par conséquent une image binaire de l'éclairement de la scène en mesurant si la tension aux bornes de la photodiode au bout d'un temps de référence t_{ref} est inférieure ou supérieure à une certaine tension de seuil V_s (figure 1.13).

Ce simple procédé de seuillage peut permettre d'obtenir une image en niveau de gris de la scène. En effet, il suffit qu'on soit capable de mesurer la tension aux bornes de la photodiode sans perturber la décharge de celle-ci. On peut alors effectuer n «lectures» à des temps de référence $t_1, \ldots, t_n$. Le premier instant t_i ou la tension de photodiode sera passée en deçà de V_s donnera le niveau de gris normalisé $(1 - i/n)$ (figure 1.13).

Du point de vue algorithmique, cela conduit à une représentation de l'image en niveau de gris par un ensemble de «coupes binaires», soit une

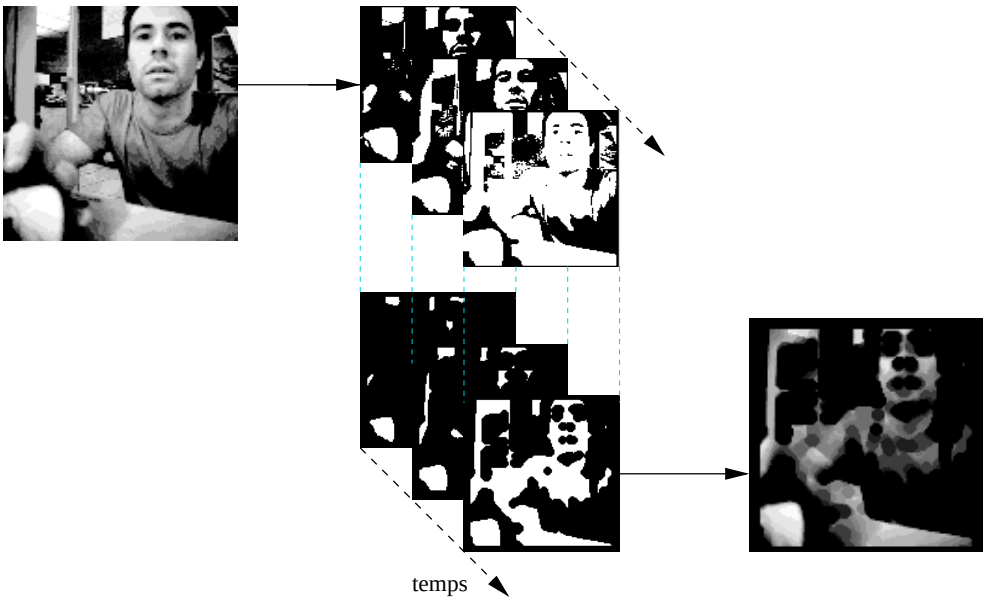


FIG. 1.14 – *Une érosion en niveaux de gris calculée sur la rétine programmable.*

suite croissante de n images binaires, représentant n seuillages successifs.

Ce type de représentation nous ramène à la morphologie, mais d'une façon plus fondamentale que dans la section précédente. En effet, en vertu du principe consistant à mêler de la manière la plus intime possible fonctions d'acquisition et de traitement, on va naturellement prolonger cette idée en une fusion spatio-temporelle de ces deux fonctions, c'est-à-dire traiter autant que possible l'image *pendant* l'acquisition.

Ce principe trouve un cadre formel idéal dans la définition des opérateurs de morphologie en niveau de gris. Une image binaire étant définie comme un sous-ensemble de $\mathbb{Z}^2$, on définit une image numérique en n niveaux de gris par une fonction de $\mathbb{Z}^2$ dans $\{0, ..., n-1\}$. Une image numérique X est équivalente à la donnée de $n-1$ images binaires :

$$X \equiv \{I_1, ..., I_{n-1}\}.$$
$$\text{avec } I_i = \{z \in \mathbb{Z}^2; X(z) \geq i\} \text{ et } X(z) = \sup_{0 \leq i \leq n-1} \{i; z \in I_i\}.$$

Le prolongement d'un opérateur morphologique Ω défini dans $\{0,1\}$ à l'opérateur $\hat{\Omega}$ défini dans $\{0, ..., n-1\}$ est simplement donné par :

$$\text{Si } X \equiv \{I_1, ..., I_{n-1}\}, \text{ alors } \hat{\Omega}(X) \equiv \{\Omega(I_1), ..., \Omega(I_{n-1})\}.$$

La figure 1.14 montre ainsi l'exemple d'une érosion en niveau de gris par un octogone.

En fait, tous les opérateurs morphologiques en niveaux de gris de base

peuvent se calculer par un petit nombre d'opérations booléennes effectuées sur les seuillages successifs. Nous allons présenter ces opérateurs, qui sont illustrés sur un exemple figure 1.15.

En présentant le calcul sur la rétine programmable de ces différents opérateurs, nous indiquerons le nombre de registres binaires nécessaires. Il faut préciser que ce nombre ne tient pas compte des registres utilisés pour coder le résultat, registres dont la quantité dépend bien sûr de manière logarithmique du nombre de niveaux de gris.

Nous avons déja présenté le calcul d'une érosion ou d'une dilatation (cf. figure 1.10) sur la rétine, qui est effectué sur deux plans binaires. Le résultat du traitement en niveaux de gris est montré figure 1.15, images (2) et (3). Si X est une image numérique, on a donc pour tout $z \in \mathbb{Z}^2$ et pour tout $B \subset \mathbb{Z}^2$:

$$(X \oplus \check{B})(z) = \sup_{b \in B}\{X(z-b)\} \text{ et } (X \ominus \check{B})(z) = \inf_{b \in B}\{X(z-b)\}.$$

Les opérateurs d'*ouverture* et de *fermeture* constituent la base du *filtrage morphologique*, notion qui sera développée dans les chapitres suivants. Ils sont respectivement définis par :

$$\gamma_B(X) = (X \ominus \check{B}) \oplus B \text{ et } \phi_B(X) = (X \oplus \check{B}) \ominus B.$$

Ces opérateurs sont croissants et idempotents. De plus, si B contient l'origine :

$$X \ominus \check{B} \leq \gamma_B(X) \leq X \leq \phi_B(X) \leq X \oplus \check{B}.$$

Ils sont calculés sur deux plans binaires (figure 1.15, images (4) et (5)).

Dans le cadre des opérateurs de type différentiel, on définit habituellement le *gradient morphologique* par $g_B(X) = (X \oplus \check{B}) - (X \ominus \check{B})$, calculable en effectuant sur chaque seuil la différence (ensembliste) entre le dilaté et l'érodé (figure 1.15 (6)). Il se calcule sur trois plans binaires. Sur la rétine, on calculera plus souvent les *gradients interne* ou *externe*, qui se calculent sur deux plans binaires seulement, étant respectivement définis par $g^i_B(X) = X - (X \ominus \check{B})$ et $g^e_B(X) = (X \oplus \check{B}) - X$ (figure 1.15, images (7) et (8)).

Le *chapeau haut-de-forme* (resp. *haut-de-forme conjugué*) est défini par $X - \gamma_B(X)$ (resp. $\phi_B(X) - X$). Cet opérateur détecte les parties fines et claires (resp. sombres) de l'image (figure 1.15, images (10) et (11)). Le calcul nécessite trois plans binaires.

L'ensemble des *maxima locaux* relativement à l'élément structurant B est défini par :

$$max_B(X) = \{z \in \mathbb{Z}^2; \forall y \in B, X(z-y) \leq X(z)\}.$$

L'*image des maxima locaux* est égal au produit de X par la fonction indicatrice de $max_B(X)$ (figure 1.15 (9)) :

$$\mu_B(X) = X.\mathbf{1}_{max_B(X)}.$$

Pour calculer cet opérateur, il suffit de remarquer que si $X \equiv \{I_1, ...I_{n-1}\}$,

alors on a :

$$\mu_B(X) = \sum_{i=1}^{n-1} I_i \setminus ((\bigcup_{j=i+1}^{n-1} I_j) \oplus \check{B}).$$

Le calcul nécessite trois plans binaires : si le plan binaire p_1 contient le seuil courant I_i, le plan p_2 contient un OU logique des seuils précédents, soit $\bigcup_{j \geq i} I_j$, le plan p_3 est utilisé pour les dilatations, on applique pour chaque seuil I_i la procédure suivante (le résultat est dans p_1) :

$$\begin{aligned} (p_2, p_3) &= (p_2, dilate_B(p_2)); \\ p_2 &= p_2 \vee p_1; \\ p_1 &= p_1 \bar{\rightarrow} p_2; \end{aligned}$$

Notons que nous aurons souvent recours à ce pseudo-code dans ce rapport. Lorsque l'algorithme utilise une procédure déja présentée, nous utiliserons la même convention que dans la première des trois lignes ci-dessus. Par exemple, si la procédure de calcul de la fonction F utilise 3 registres, on pourra noter par exemple $(a, b, c) = (a, F(a, b), \tilde{c})$. Cette notation signifie que le résultat, qui dépend des deux bits a et b, est contenu dans le registre qui contenait le bit b, que la valeur du bit a n'est pas modifiée, mais que celle du bit c a été perdue.

1.4.4 Extraction d'information

Nous avons présenté dans cette partie les principes de l'algorithmique rétinienne dans le cadre du traitement d'images. Ces principes ne concernent que la partie amont du processus dans lequel s'insère la rétine artificielle en tant que module d'un système de vision.

Comme nous l'avons exposé en introduction, l'objectif est de s'acquitter de la plus grande part du traitement d'images dans la rétine, pour ne sortir du circuit qu'un petit volume de données scalaires, compatibles avec l'utilisation d'un processeur conventionnel pour effectuer les tâches de plus haut niveau. A ce stade se posent deux questions fondamentales :

- **Quelle** information sortir du circuit ?
- **Comment** la sortir efficacement ?

La première question concerne l'informaticien, et la deuxième l'électronicien, mais la nécessité de mener de front ces réflexions en demeurant en interaction forte est évidente.

Etant donné l'importance de ce sujet pour la vision rétinienne, nous consacrons l'annexe A à une présentation des techniques d'extraction existantes

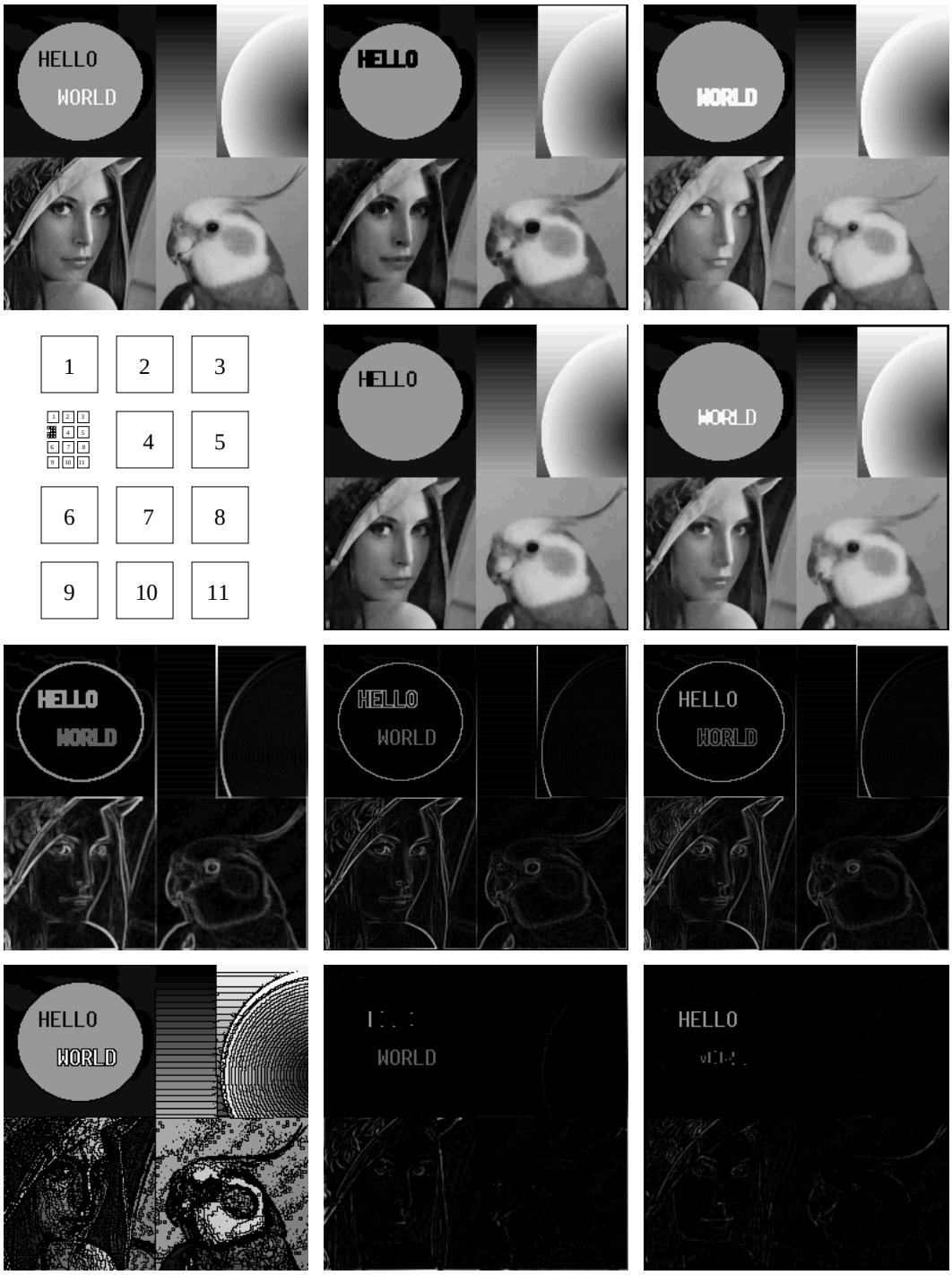


FIG. 1.15 – *Les opérations morphologiques de base en niveaux de gris. Dans toutes ces opérations, l'élément structurant est le voisinage* 3×3 *de l'origine. (1) Image originale (2) Erosion (3) Dilatation (4) Ouverture (5) Fermeture (6) Gradient morphologique (7) Gradient interne (8) Gradient externe (9) Maxima locaux (10) Chapeau haut-de-forme (11) Haut-de-forme conjugué.*

ou envisagées, ainsi qu'à quelques éléments de réflexion sur la nature des informations extraites. Cela étant, ce problème n'a pas été abordé de manière approfondie dans le cadre de cette thèse. Ce parti pris est loin d'être anodin en ce qui concerne l'apport de ce travail à la vision rétinienne, et appelle une explication.

Cette thèse aurait pu être structurée de manière verticale, en suivant la progression suivante :

- Acquisition et codage de l'information,
- Traitement d'images,
- Extraction d'information.

Nous avons choisi de structurer notre recherche de manière horizontale, en concentrant notre travail sur le point central de la vision rétinienne, qui concerne le domaine - déja immense - du traitement d'images. Nous nous proposons d'apporter quelques contributions fortes à l'algorithmique massivement parallèle de la rétine programmable, en examinant chaque fois que c'est nécessaire les aspects liés à la représentation et à l'extraction d'information.

1.5 Contributions et structure de la thèse

Nous avons présenté dans ce chapitre notre problématique de travail, liée aux spécificités architecturales et algorithmiques de la rétine programmable.

Les principes de l'algorithmique rétinienne et ses liens avec la morphologie mathématique, en grande partie établis par des travaux précurseurs [37], [95] ont été rappelés ou précisés ici.

Les chapitres suivants exposent nos contributions originales à l'algorithmique de vision pour rétine artificielle programmable.

Le chapitre deux traite de la *complexité* des opérateurs de rétine. Nous proposons un formalisme qui nous permet d'exploiter un certain nombre de résultats du domaine de la synthèse logique, et de présenter les principales techniques de minimisation des opérateurs rétiniens.

Le troisième chapitre présente l'ensemble de nos travaux sur l'implantation des tranformations *homotopiques* sur la rétine programmable. Nous montrons d'une part la puissance de cette architecture dans le cadre de ces opérateurs, et d'autre part l'intérêt fondamental de la minimisation logique.

Le chapitre quatre est consacré aux algorithmes *géodésiques*. Il met en évidence l'importance de la *reconstruction* en tant qu'opérateur de base, et montre une autre facette de ce travail, qui est de proposer des options architecturales.

Enfin, le chapitre cinq traite d'une application dans le cadre de la détection de *mouvement*. On y expose la problématique des traitements temporels dans notre architecture, et on présente une implantation réalisée sur prototype. Puis, on propose différentes solutions probabilistes dans le contexte des *champs de Markov*.

Chapitre 2

Complexité des opérateurs de rétine

2.1 Introduction

Dans ce chapitre, nous posons le cadre théorique et tentons de dégager quelques éléments de réponse au problème qui se trouve au cœur de notre travail de recherche : "Comment effectuer un traitement donné *le plus rapidement possible* sur la rétine?" Dans le chapitre précédent, nous évoquions l'adéquation de la morphologie mathématique à notre architecture, ce qui suggérait l'utilisation d'un formalisme basé sur les transformées en tout-ou-rien. Il est important cependant de ne pas demeurer dans un cadre formel aussi rigide, car dans notre architecture, la formalisation d'un opérateur est plus qu'un mode d'écriture, c'est *la description d'un procédé de calcul*, elle détermine entièrement l'efficacité d'un traitement.

Nous nous plaçons ici dans le cas le plus général. On souhaite calculer en chaque pixel, une certaine fonction booléenne d'un voisinage donné de ce pixel, en un temps minimum. Dans ce qui suit, nous expliquons le vocabulaire et les notations que nous allons utiliser dans ce chapitre.

Soit $N \in \mathbb{N}$. Considérons une rétine à N plans binaires $\{p^1, \ldots, p^n\}$. Les *opérations élémentaires* (OE) qu'il est possible de faire sur cette rétine sont :

- **translation élémentaire :** $p^i \leftarrow p^i_{dir}$, avec $1 \leq i \leq N$, et dir prenant la valeur d'une des quatre directions cardinales, Nord, Ouest, Sud ou Est.
- **fonction booléenne élémentaire :** $p^i \leftarrow F(p^i, p^j)$, avec $1 \leq i \leq N$. $1 \leq j \leq N$, et F prenant la valeur d'une des seize fonctions booléennes à deux variables. Ces fonctions booléennes sont répertoriées dans le tableau 2.1 par leurs tableaux de vérité et le symbole que nous utili-

0	0	1	1	0	0	0	1	1	1	1	0	0	1	1	0
0	0	1	1	1	1	0	1	0	0	1	0	1	0	0	1
0		**1**		$\curvearrowleft$		$\curvearrowright$		$\bar{\curvearrowleft}$		$\bar{\curvearrowright}$		$\triangle$		$\bar{\triangle}$	
0	0	0	1	1	1	1	0	1	0	1	1	0	1	0	0
0	1	1	1	1	0	0	0	1	1	0	1	0	0	1	0
$\wedge$		$\vee$		$\bar{\wedge}$		$\bar{\vee}$		$\leftarrow$		$\rightarrow$		$\bar{\leftarrow}$		$\bar{\rightarrow}$	

TAB. 2.1 – *Les 16 fonctions booléennes à deux variables et leur notation.*

sons pour les désigner. Les tableaux de vérité se lisent de la manière suivante :

F(0,0)	F(0,1)
F(1,0)	F(1,1)
F	

Définition 4 *Un* Opérateur de rétine *(OR) est défini par une séquence finie d'OE. Soit Ω un OR.*
La Complexité en temps *de Ω est le nombre d'OE de la séquence. On la note $\mathcal{C}^t(\Omega)$. Elle représente le temps d'éxécution de l'OR.*
La Complexité en espace *de Ω est le nombre de plans différents utilisés dans la séquence. On la note $\mathcal{C}^e(\Omega)$. Elle représente le nombre de registres binaires utilisés pour le calcul de l'OR.*

Dans la première partie de ce chapitre, nous nous plaçons dans le cadre restreint des *Opérateurs de rétine primitifs* (ORP) que nous définissons. Ces opérateurs particuliers correspondent au cas où l'on n'a pas d'hypothèse sur la forme du voisinage, et sont donc dépourvus de «structure spatiale». Cette restriction nous permet de nous placer dans un formalisme existant, celui des *circuits booléens*. En effet, la complexité des ORP est équivalente à la complexité des fonctions booléennes. Nous présentons dans une deuxième partie les principaux résultats connus dans ce domaine, qui fondent les limites théoriques de notre recherche.

Les deux parties suivantes tentent de tirer les enseignements de ces résultats et d'apporter des éléments pour augmenter l'efficacité de l'algorithmique rétinienne. La première de ces parties présente les avantages de la *multigranularité*, qui permet un gain virtuel de mémoire. La seconde introduit le *principe de composition* de l'algorithmique rétinienne, qui constitue la structure spatiale des opérateurs de rétine, et permet de diminuer la complexité en réduisant la taille des voisinages. Nous y généralisons le formalisme des circuits booléens, un OR étant égal à la composition de plusieurs ORP.

Finalement, dans la dernière partie nous illustrons l'ensemble des résultats présentés dans ce chapitre dans le cadre d'un exemple, qui est celui des fonctions «seuil».

2.2 Opérateurs primitifs

2.2.1 ORP et circuits booléens

Dans cette partie, nous considérons le cas particulier des opérateurs primitifs. Un opérateur de rétine est dit *primitif* si toutes les translations qui le composent sont effectuées sur le même plan booléen. Un ORP calcule donc une fonction booléenne sur un voisinage en accédant à chaque variable par une translation du plan contenant les données. Un ORP peut se formaliser par une représentation classique de graphe appelée *circuit booléen.*

Nous présentons ce formalisme en suivant essentiellement la syntaxe adoptée par Zwick [124].

Définition 5 *Un graphe orienté est défini par la donnée d'un ensemble X de* sommets *et d'un ensemble $V \subset X \times X$ d'* arcs.

Définition 6 *Un sommet u admet un ensemble de* Successeurs $SUCC(u) = \{s_i \in X; (u, s_i) \in V\}$ *et un ensemble de* Prédécesseurs $PRED(u) = \{p_i \in X; (p_i, u) \in V\}$. *On appelle* degré d'entrée *de u le cardinal de $PRED(u)$ et* degré de sortie *de u le cardinal de $SUCC(u)$.*

Définition 7 *On définit une relation d'ordre partiel sur les nœuds par $u < v \iff \exists (v_i)_{i=0}^{k}$, suite dans X tq $(u, v_0) \in V, \forall i (v_i, v_{i+1}) \in V$ et $(v_k, v) \in V$.*

Définition 8 *Un* circuit booléen γ *est un graphe connexe (X, V) orienté sans cycle tel que tout sommet u de degré d'entrée $n, n > 0$, est étiqueté par une fonction booléenne $f_u : \{0,1\}^n \longrightarrow \{0,1\}$. Un tel sommet est appelé* nœud *du circuit. Les sommets de degré d'entrée 0 ou* entrées *du circuit sont étiquetés par une variable. On distingue un unique nœud particulier s tel que le degré de sortie est nul. Ce nœud est appelé* sortie *du circuit.*

Définition 9 *Un nœud* calcule *une fonction booléenne en calculant le* résultat *de sa fonction-étiquette appliquée sur le n-uplet des résultats calculés par les nœuds qui le précèdent, le résultat d'une entrée étant la valeur de sa variable-étiquette. Etant donné cette définition très naturelle, un circuit* calcule *la même chose que son nœud de sortie.*

Définition 10 *Soit $B_n = \{f; f : \{0,1\}^n \longrightarrow \{0,1\}\}$. Un circuit booléen est dit* de base B_n *si tous ses nœuds ont un degré d'entrée égal à n. On note C_{B_n} l'ensemble des circuits de base B_n.*

Définition 11 *La* taille $\mathcal{T}$ *d'un circuit booléen est égale au nombre de ses nœuds.*

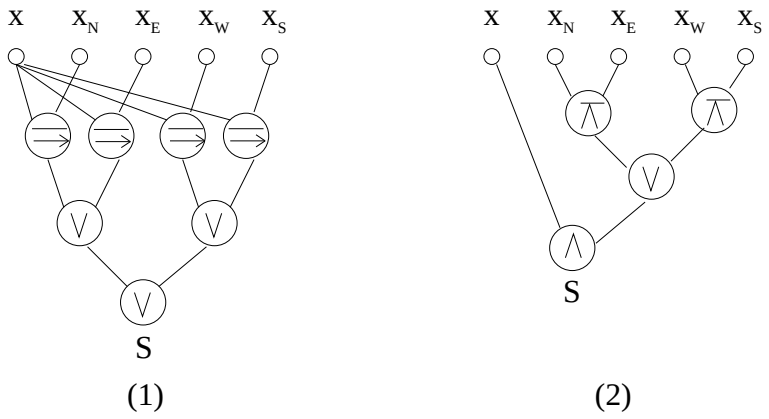


FIG. 2.1 – *Deux ORP calculant la même fonction : contours binaires.*

Un circuit booléen représente donc un mode de calcul d'une fonction boolénne par une certaine décomposition en opérations élémentaires, lesquelles correspondent aux fonctions qui étiquettent les nœuds du circuit. Dans le cas des ORP, ces opérations élémentaires étant les fonctions booléennes de deux variables, nous ne considérerons que des nœuds à deux entrées, c'est à dire des circuits de base B_2.

La figure 2.1 montre deux exemples d'ORP représentés sous forme de circuits booléens. Ils calculent la même fonction, (contours d'une image binaire). Ici x est la valeur du pixel à l'origine, x_N, x_E, x_W, x_S les voisins respectivement au nord, est, ouest et sud. Mais ils représentent des modes opératoires distincts. L'ORP (1) correspond à la forme disjonctive $(x \wedge \bar{x_N}) \vee (x \wedge \bar{x_E}) \vee (x \wedge \bar{x_W}) \vee (x \wedge \bar{x_S})$. L'ORP (2) correspond à l'expression factorisée $x \wedge (\bar{x_N} \vee \bar{x_E} \vee \bar{x_W} \vee \bar{x_S})$. Dans la suite, nous allons voir comment s'interprètent ces écritures en terme de complexité.

2.2.2 Mesures de complexité

Temps

L'intérêt principal de ce formalisme, est, on l'aura compris, de fournir un indicateur du *temps de calcul*. Il est clair que la taille du circuit de base B_2 correspondant à un ORP est égale au nombre d'opérations booléennes entre deux plans. Reste cependant à tenir compte du deuxième type d'actions élémentaires : les translations. Dans le cas des ORP, les translations correspondent à l'*accès aux variables*, elles ne sont effectuées que sur le plan contenant les données initiales.

Pour mesurer le coût des translations, il faut donc pondérer chaque arc

sortant d'une variable par un nombre représentant le *coût d'accès* à la variable, c'est à dire le nombre de translations élémentaires nécessaires pour l'«atteindre». La complexité en temps $\mathcal{C}^t$ de l'ORP Ω peut donc se calculer grâce au circuit γ :

$$\mathcal{C}^t(\gamma) = \mathcal{T}(\gamma) + \sum_{i=1}^{m} w_i$$

où m est le nombre d'accès aux variables.
où w_i est le coût du $i$ème accès à une variable.

Soient l et h les dimensions du plus petit rectangle englobant les n variables, il est facile de voir qu'on a la majoration :

$$\mathcal{C}^t(\Omega) \leq \mathcal{T}(\gamma) + m(l + h - 2).$$

En effet, $l + h - 2$ est la longueur du plus long chemin (en terme de translations élémentaires) qu'on peut parcourir dans un rectangle de taille $l \times h$. De plus, $m \leq \mathcal{T}(\gamma)$. On a donc :

$$\mathcal{C}^t(\Omega) \leq (l + h - 1)\mathcal{T}(\gamma).$$

Le résultat important relatif aux ORP est bien sûr que $\mathcal{C}^t(\Omega) \leq O(\mathcal{T}(\gamma))$.

Le temps de calcul d'un Opérateur Rétinien Primitif dépend essentiellement de la complexité du mode opératoire de la fonction booléenne associée.

On assimilera par la suite la complexité en temps de l'ORP à la taille du circuit qui le représente. Par exemple, sur la figure 2.1, l'ORP (1) a une complexité en temps de 7, l'ORP (2) de 4.

Il faut à présent considérer le problème de la complexité en espace. Sur la figure 2.1, l'ORP (1) se calcul sur 3 plans binaire, l'ORP (2) sur 2 plans binaires seulement. Comment connaître le nombre de registres binaires nécessaire pour le calcul d'un ORP en général? C'est l'objet de la section suivante.

Espace

Pour calculer à partir d'un circuit γ le nombre de plans-mémoire que nécessite le mode de calcul de l'ORP correspondant, il faut imaginer que le processus de calcul sur la rétine correspond à un élagage du circuit; c'est-à dire qu'on retire un nœud du circuit chaque fois que l'on a calculé sa fonction-étiquette. On considère d'autre part une *pile* qui représente la mémoire. On empile un nouveau registre lorsque c'est nécessaire, et on libère les registres non indispensables dès que possible. Lorqu'on parvient au nœud de sortie, la fonction est alors calculée et le nombre de plans mémoire nécessaire à son

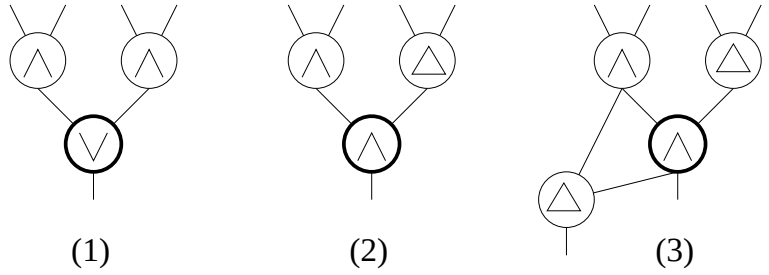


FIG. 2.2 – *Le calcul des nœuds en gras nécessite l'utilisation d'un registre supplémentaire dans les cas (1) et (3), mais pas dans le cas (2).*

calcul (complexité en espace) est égal à la hauteur maximum atteinte par la pile.

A présent, nous allons répondre à la question suivante : Quand est-il nécessaire d'empiler un nouveau registre ? Pour fixer les idées, examinons la figure 2.2. On cherche à savoir dans ces trois exemples si le calcul du nœud en gras va nécessiter un nouveau registre ou non. Dans le cas (1), la réponse est oui, puisque ses deux prédécesseurs doivent être calculés sur deux registres différents. Dans le cas (2), en revanche, on calcule d'abord la sortie c du prédécesseur de droite, et dans le même registre, si a et b sont les deux entrées du prédécesseur de gauche, on calcule $c \leftarrow c \wedge a$ puis $c \leftarrow c \wedge b$. Le nœud en gras ne coûte donc pas de mémoire ici. Dans le cas (3), on pourrait calculer la sortie du nœud en gras dans le registre de son prédécesseur de droite comme dans le cas (2), mais il existe un autre nœud qui nécessite le calcul explicite de la sortie du prédécesseur de gauche, le nœud en gras coûtera donc un registre mémoire dans ce cas.

Il faut maintenant recenser de manière exhaustive les cas où le calcul des nœuds peut être ainsi «cascadé». Nous introduisons pour cela la notion d'*associabilité* entre opérateurs booléens.

Définition 12 *Soit un nœud de circuit booléen étiqueté par $F \in B_2$. Supposons que son prédécesseur de gauche (resp. de droite) est étiqueté par $P \in B_2$. F est dit* associable *à P si et seulement s'il existe deux opérateurs G et H dans B_2 tels que $F(P(x,y),z) = G(x,H(y,z))$ (resp. $F(x,P(y,z)) = G(H(x,y),z)$).*

Exemples :

- $F = P = \Box$ et l'opérateur est associatif.
  $(a \Box b) \Box c = a \Box (b \Box c)$
- l'un est $\triangle$, l'autre est $\bar{\triangle}$.
  $(a \triangle b) \bar{\triangle} c = a \bar{\triangle} (b \triangle c)$

- l'un est $\vee$ (resp. $\wedge$), l'autre est $\leftarrow$ (resp. $\bar{\rightarrow}$).
  $(a \vee b) \leftarrow c = a \vee (b \leftarrow c)$
  et $(a \leftarrow b) \vee c = a \vee (b \rightarrow c)$
- $F = \vee$ (resp. $\wedge$), et $P = \bar{\wedge}$ (resp. $\bar{\vee}$).
  $(a \bar{\wedge} b) \vee c = a \rightarrow (b \rightarrow c)$
  et $(a \bar{\vee} b) \wedge c = a \bar{\rightarrow} (b \bar{\rightarrow} c)$

On peut à présent recenser précisément les cas pour lesquels le calcul d'un nœud ne nécessite pas d'empiler un nouveau registre. Nous admettrons qu'on doit toujours conserver le plan contenant les variables. *Supposons dans un premier temps que le degré de sortie maximum de chaque nœud est égal à* 1, *c'est-à dire que les circuits sont des* arbres orientés. On se trouve dans un cas particulier de circuits booléens qu'on appellent *formules* booléennes.

Proposition 1 *Soit γ une formule sur B_2. Le calcul d'un nœud de γ étiqueté par un opérateur $F \in B_2$ ne nécessite pas de registre mémoire supplémentaire si et seulement si l'une des deux conditions suivantes est vérifiée :*

- *L'un de ses prédécesseurs est un opérateur, et l'autre une variable.*
- *F est associable avec l'un de ses prédécesseurs.*

Dans le premier cas, en effet (figure 2.3 (1)), on calcule $c \leftarrow P(a, b)$ puis $c \leftarrow F(c, x)$ dans le même registre.

Dans le deuxième cas (figure 2.3 (2)), supposons que F est associable à P_1. On veut calculer $s = F(P_1(a, b), P_2(c, d))$ On calcule d'abord $z \leftarrow P_2(c, d)$. Donc $s = F(P_1(a, b), z)$. Comme F est associable à P_1, il existe deux opérateurs G et H tels que $s = G(a, H(b, z))$. On calcule donc $z \leftarrow H(b, z)$, puis $z \leftarrow G(a, z)$, tous les calculs étant faits dans le même registre.

Réciproquement, si les prédécesseurs de F sont deux variables, il faudra nécessairement un registre supplémentaire car on doit conserver la valeur des variables. Si P_1 et P_2 sont deux opérateurs, et que F n'est associable ni à l'un ni à l'autre, si l'on calcule d'abord par exemple P_2, on ne peut faire interagir aucune des entrées de P_1 avec la sortie de P_2 sans perdre d'information, et donc le calcul de F *nécessite le calcul préliminaire explicite de ses deux prédécesseurs*, et par conséquent coûte un registre binaire de mémoire □

Pour passer au cas général des circuits booléens, il suffit de voir que la règle précédente n'est valable que si le prédécesseur du nœud n'a pas d'autre successeur qui n'est pas encore calculé.

Pour un circuit quelconque, la complexité en espace est calculée par l'algorithme de coloriage des nœuds suivant :

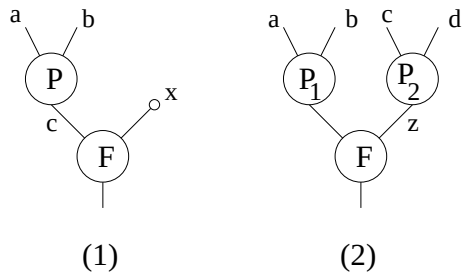


FIG. 2.3 – *Preuve de la proposition 1*

```
procédure COLORIER(F : nœud)
    si F est une variable STOP
    sinon
        (P1, P2) ← prédécesseurs de F
        pour (i, j) dans {(1, 2), (2, 1)}, faire :
            si Pi est un opérateur et Pj une variable
            ou si F est associable à Pi
            alors
                si Pi n'est pas marquée
                    marquer Pi
                    colorier F en blanc
                sinon
                    colorier F en noir
        fin_pour
        COLORIER(P1)
        COLORIER(P2)
fin_procédure
```

Le coloriage complet d'un circuit de sortie S est obtenu par la commande COLORIER(S). Il permet d'obtenir la complexité en espace des ORP représentés par des circuits booléens grâce à la formule récursive de la proposition suivante :

Proposition 2 *Soit un ORP* Ω *représenté par une* formule γ.
Si γ *est une variable,* $\mathcal{C}^e(\Omega) = 1$
Sinon, $\exists F, \gamma_1, \gamma_2$ *tels que* $\gamma = F(\gamma_1, \gamma_2)$*, et on note* $\mathcal{C}^e(\gamma_1) = n$ *et* $\mathcal{C}^e(\gamma_2) = m$
alors : si F *est noir ET si* $(m = n\ OU\ max(m, n) \leq 2)$, $\mathcal{C}^e(\Omega) = max(m, n) + 1$
sinon $\mathcal{C}^e(\Omega) = max(m, n)$

En effet, si le nœud est noir et si $m < n$ (voir figure 2.4), on commencera à calculer le nœud qui recquiert n registres de mémoire puis l'autre. Si $n > 2$

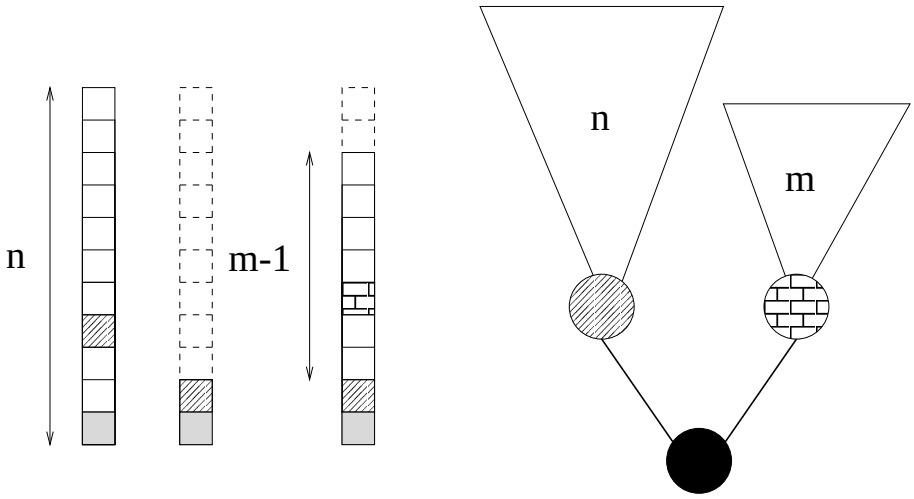


FIG. 2.4 – *Preuve de la proposition 2*

alors le deuxième calcul pourra se faire sur $n-2$ points restants (La valeur des variables est toujours conservée; ici le point du bas de la pile.) Si $n = 2$ alors $m = 1$, mais comme à la fois la valeur du nœud de gauche et celle des variables doivent être conservées, puisque le nœud est noir, on aura $\mathcal{C}^e(\Omega) = 3$. □

La figure 2.5 montre un exemple simple de calcul sur une formule coloriée.

Pour terminer, nous considèrerons l'exemple d'une fonction booléenne particulière. Il s'agit de la fonction «seuil» T_5^2. Nous retrouverons de telles fonctions à la fin de ce chapitre. Les deux circuits représentés sur la figure 2.6 calculent cette même fonction T_5^2, fonction de cinq variables qui prend la valeur 1 si et seulement si au moins deux des cinq variables valent 1.

Le circuit de gauche fait la disjonction entre les C_5^2 produits possibles de deux variables. Celui de gauche effectue une addition sur deux bits et teste le passage à 1 du bit de poids fort. La complexité en temps de ces circuits vaut respectivement 19 et 13, il suffit de compter le nombre de nœuds. Pour connaître la complexité en espace, il faut utiliser la procédure de coloriage, puis faire le calcul récursif de la proposition 2. Ce calcul est illustré sur la figure 2.7. Le circuit de gauche a donc une complexité en espace de 3, celui de droite de 4.

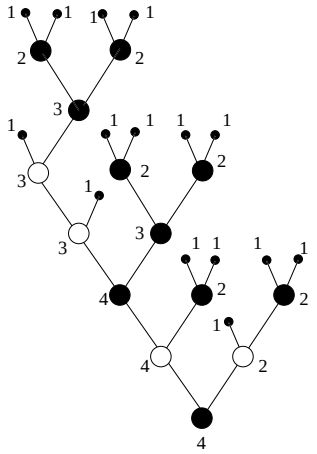


FIG. 2.5 – *Le calcul de la complexité en espace sur un circuit colorié.*

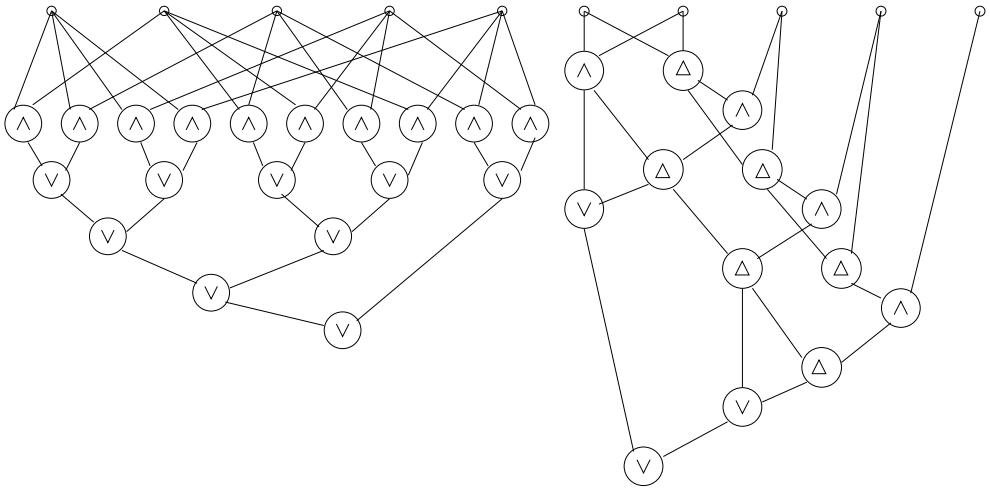


FIG. 2.6 – *Deux circuits booléens calculant la même fonction seuil* T_5^2.

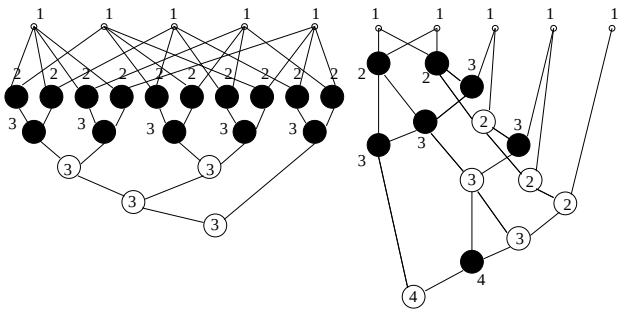


FIG. 2.7 – *Coloriage des circuits et calcul de complexité en espace*

2.3 Bornes de complexité

Le formalisme que nous avons présenté dans la section précédente fournit donc une représentation des opérateurs de rétine primitifs qui permet de connaître à la fois le temps de calcul (complexité en temps) et le nombre de registres nécessaire (complexité en espace). Il permet également de se ramener au problème de la *complexité des fonctions booléennes*, ce qui signifie que l'optimisation d'un traitement rétinien est principalement un problème de *minimisation logique*, auquel est consacré une large littérature [118], [20], [89], [90]. Le domaine d'application est celui de la *synthèse logique*. Il s'agit dans ce cadre de définir une architecture de calcul d'une certaine fonction numérique en minimisant l'utilisation des ressources matérielles.

Les résultats que nous allons présenter maintenant concernent les bornes dites «universelles», c'est-à-dire que nous considérons l'ensemble des fonctions booléennes, par opposition à une certaine classe de fonctions (par exemple les fonctions booléennes symétriques que nous évoquerons dans la dernière section). La présentation de ces résultats a pour but de répondre aux deux questions suivantes :

- Quels ordres de grandeur pouvont nous espérer en général pour le temps de calcul d'un opérateur de rétine quelconque?
- Comment évolue le temps de calcul d'un opérateur de rétine en fonction de la capacité mémoire dont nous disposons?

2.3.1 Bornes inférieures

Le premier résultat que nous présentons est le théorème de Shannon. Il donne une borne inférieure pour la complexité du calcul de la majorité des fonctions booléennes. Il est basé sur un argument de comptage que nous détaillons.

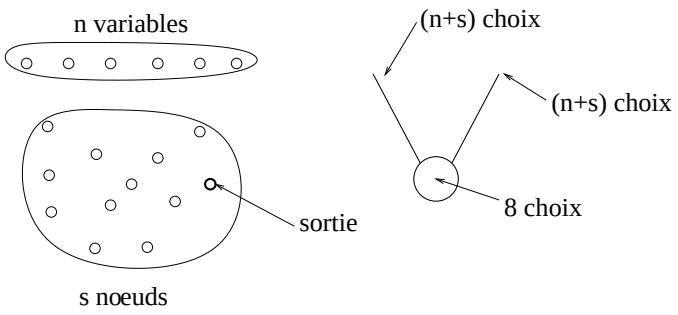


FIG. 2.8 – *L'argument de comptage de Shannon.*

Si A est un ensemble, on note $Card(A)$ son cardinal.
Soit $f \in B_n$. On appelle *complexité* de f la quantité $\kappa(f)$ définie par $\kappa(f) = \min_{\gamma \in C_{B_2}} \{\mathcal{T}(\gamma); \gamma \text{ calcule } f\}$, la taille minimum d'un circuit de base B_2 calculant f.

Théorème 1 *[109, Shannon (1949)]*
Soit n et s deux entiers naturels tels que $n \leq s$. Si l'on pose $\mathcal{F}(n,s) = Card\{f; f \in B_n, \kappa(f) \leq s\}$, alors on a :

$$\mathcal{F}(n, \tfrac{2^n}{n}) = o(Card(B_n))$$

Ce théorème signifie que *la plupart des fonctions booléennes à n variables ont une complexité supérieure ou égal à $\frac{2^n}{n}$*. La démonstration est basée sur la majoration du nombre $C(n,s)$ de circuits à n entrées, et de taille s.
Etant donné n variables et s nœuds, fabriquons un graphe tel que tous les nœuds aient 2 entrées, et qu'un seul nœud (la sortie) n'ait pas de successeur (voir figure 2.8). Notons que nous n'imposons pas l'absence de cycle, ce qui affaiblit la majoration.

- choix d'une sortie $\longrightarrow s$ choix.
- pour chaque nœud :
  - choix d'une entrée $\longrightarrow (n+s)$ choix.
  - pour chacune des deux entrées donc $\longrightarrow (n+s)^2$ choix.
  - choix d'une fonction étiquette (8 choix car on peut éliminer les constantes et les projections, et ne prendre qu'un opérateur non commutatif sur deux) donc $\longrightarrow 8(n+s)^2$ choix.
- et ceci pour les s nœuds donc $\longrightarrow (8(n+s)^2)^s$
- l'ordre induit par le choix successif des nœuds n'a pas d'importance donc on divise par $s!$.

Finalement, $C(n,s) \leq \frac{s(8(n+s)^2)^s}{s!}$

la formule de Stirling donne $s! \geq (\frac{s}{e})^s$ donc $C(n,s) \leq \frac{s(8(n+s)^2)^s}{(\frac{s}{e})^s}$

en prenant le logarithme, $\log C(n,s) \leq s \log 8(n+s)^2 + \log s - s(\log s + \log e)$

Soit :

$$\begin{array}{rcll} \log C(n,s) & \leq & 3s + 2s\log(n+s) + \log s - s(\log s + \log e) & \\ & \leq & 5s + s\log s + \log s + s\log e & \text{puisque } (n \leq s) \\ & \leq & (s+1)\log s + 7s & \end{array}$$

et avec $s = \frac{2^n}{n}$, $\log C(n,s) \leq (\frac{2^n}{n} + 1)\log(\frac{2^n}{n}) + 7(\frac{2^n}{n})$

c'est-à-dire $\log C(n,s) \leq 2^n - (1 - o(1))(\frac{2^n \log n}{n})$

donc finalement $C(n,s) \leq \frac{2^{2^n}}{2^{\frac{2^n \log n}{n}(1-o(1))}}$

Le nombre de circuits à n entrées de taille $\leq s$ vaut $K(n,s) = \sum_{i=n}^{s} C(n,i)$.
Or $K(n-1,s) = o(C(n,s))$ donc $K(n,s) = O(C(n,s))$.

Il est clair d'autre part que $\mathcal{F}(n,s) \leq K(n,s)$.

D'autre part $Card(B_n) = 2^{2^n}$ (Dénombrement des tables de vérité de B_n, qui contiennent chacune 2^n valeurs booléennes).
Donc finalement, $\frac{\mathcal{F}(n,\frac{2^n}{n})}{Card(B_n)} \leq O(2^{-\frac{2^n \log n}{n}(1-o(1))})$

et donc $\mathcal{F}(n,\frac{2^n}{n}) = o(Card(B_n))$ □

2.3.2 Bornes supérieures

Dans cette section, nous présentons plusieurs méthodes de représentation permettant de garantir une complexité maximale pour le calcul de n'importe quelle fonction booléenne.

Une borne en $O(n2^n)$

Théorème 2 *$\forall f \in B_n$, $\exists \gamma$, γ calcule f, tq : $\mathcal{T}(\gamma) = O(n2^n)$.*

En effet, toute fonction booléenne booléenne peut se décomposer sous forme normale disjonctive (FND). Si l'on convient de noter, pour x et a deux variables booléennes, $x^a = x$ si $a = 1$ et $\bar{x}$ si $a = 0$, alors toute fonction

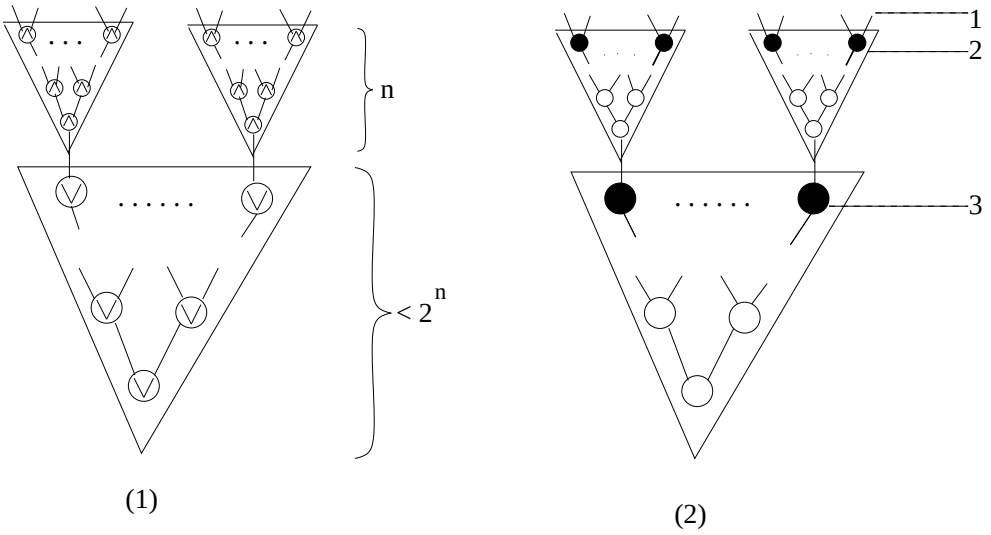


FIG. 2.9 – *Complexité de la FND.*

booléenne à n variables peut s'écrire :

$$f(x_1, \ldots, x_n) = \bigvee_{\{(\alpha_1,\ldots,\alpha_n)\in\{0,1\}^n ; f(\alpha_1,\ldots,\alpha_n)=1\}} (x_1^{\alpha_1} \wedge \ldots \wedge x_n^{\alpha_n}).$$

Soit au plus 2^n monômes de n variables (voir figure 2.9 (1)), elle peut donc être calculée par un arbre γ sur B_2 de moins de $n2^n$ portes. □

* Notons qu'on a alors : $C^e(\gamma) = 3$. (voir figure 2.9 (2)).

Cette décomposition fournit la complexité en temps de calcul maximal pour les fonctions booléennes. Dans les pires cas, le temps de calcul est évidemment catastrophique. Cependant, la FND reste pour la rétine une représentation intéressante. D'une part, elle est minimale en complexité en espace. D'autre part, elle se révèle dans certains cas beaucoup plus compacte que dans son écriture étendue, grâce à la minimisation de Karnaugh [49]. Cette méthode, suffisamment classique pour ne pas s'y attarder, permet de réduire le nombre et la taille des monômes, par la mise en facteur d'expressions tautologiques.

Enfin, il va de soi que ces résultats de complexité s'appliquent pour toutes les décompositions canoniques «à deux niveaux» («Somme» associative de monômes). C'est le cas par exemple de la *Ring Sum Expansion* (RSE), somme par l'opérateur XOR de monômes conjonctifs. Cependant, pour une fonction booléenne en particulier, l'une ou l'autre de ces représentations pourra s'avérer plus intéressante.

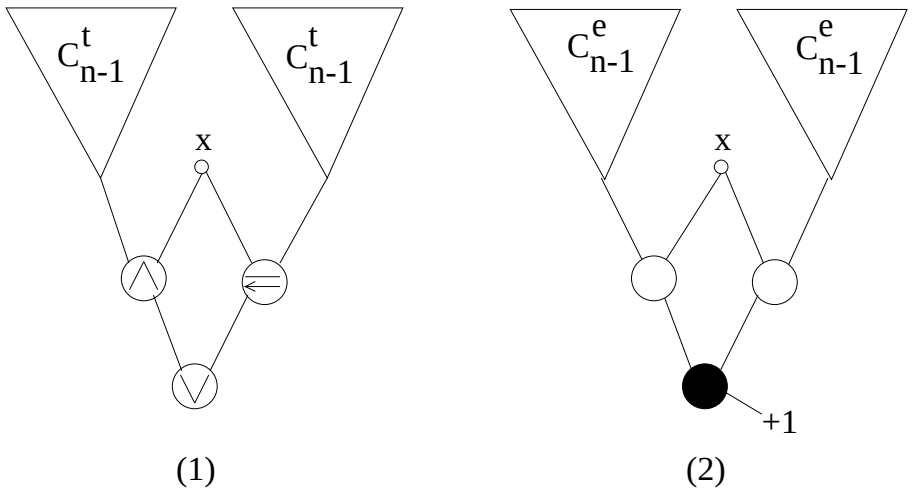


FIG. 2.10 – *Complexité de l'expansion de Shannon.*

Une borne en $O(2^n)$

Théorème 3 $\forall f \in B_n$, $\exists \gamma$, γ *calcule* f, *tq:* $\mathcal{T}(\gamma) = O(2^n)$.

La preuve se fait par récurrence sur le nombre de variables: Si $f \in B_2, \exists \gamma, \gamma$ calcule $f, \mathcal{T}(\gamma) = 1$

et $\forall f \in B_n, f(x_1, \ldots, x_n) = (f(0, x_2, \ldots, x_n) \wedge \bar{x_1}) \vee (f(1, x_2, \ldots, x_n) \wedge x_1)$.

donc si C_n^t est le nombre de nœuds d'un arbre calculant une fonction dans B_n, $C_n^t = 2C_{n-1}^t + 3$, pour $n \geq 2$ (voir figure 2.10 (1)).

D'où : $C_n^t = 2^n - 3$. □

* Un raisonnement par récurrence analogue permet de voir que $C^e(\gamma) = O(n)$. (voir figure 2.10 (2))

Ce type d'écriture correspond à l'*expansion de Shannon*. On la rencontre également sous une autre forme, en remplaçant les OU par des XOR. Comme dans le cas FND/RSE, l'une ou l'autre des représentations peut être bien plus intéressante pour une fonction particulière, mais dans un cadre universel, les résultats de complexité sont les mêmes.

Bien que cette borne soit exponentielle, on peut, pour certaines fonctions, obtenir des décompositions beaucoup plus efficaces. En effet, l'*ordre* dans lequel les variables interviennent joue un rôle déterminant dans la complexité, et un choix judicieux de cet ordre peut conduire à des expressions très compactes [22].

Une borne en $O(\frac{2^n}{n})$

Théorème 4 *$\forall f \in B_n$, $\exists \gamma$, γ calcule f, tq : $\mathcal{T}(\gamma) = O(\frac{2^n}{n})$.*

Précisons que le résultat le plus fin dans cet ordre de grandeur est donné par le théorème de Lupanov [59] (voir [124] pour une preuve en anglais). Il dit que $\forall f \in B_n, \exists \gamma, \gamma$ calcule $f, tq \mathcal{T}(\gamma) = (1 + o(1))\frac{2^n}{n}$.. C'est un théorème très fort puisque d'après le théorème 1, il s'agit de la borne universelle définitive. Dans le cadre des ORP, puisque $C^t(\Omega) = O(\mathcal{T}(\gamma))$, on se contentera de démontrer le résultat sensiblement moins fort :

$\forall f \in B_n, \exists \gamma, \gamma$ calcule $f, \mathcal{T}(\gamma) = (4 + o(1))\frac{2^n}{n}$..

Cette borne correspond à la représentation par *diagrammes de décision binaires* (BDD) [22], qui sont obtenus en compactant les arbres produit par l'expansion de Shannon (théorème 3), soit $\forall f \in B_n, f(x_1, \ldots, x_n) = (f(0, x_2, \ldots, x_n) \wedge \bar{x_1}) \vee (f(1, x_2, \ldots, x_n) \wedge x_1)$.

Au niveau de récursion $n-k$, on calcule 2^{n-k} fonctions à k variables (voir figure 2.11 (1)). Or il y a 2^{2^k} fonctions à k variables.

Supposons qu'au delà d'un certain niveau $n - k$, on calcule toutes les fonctions à k variables (voir figure 2.11 (2)). On aura : $\mathcal{T}(\gamma) = T_1(\gamma)(n-k) + T_2(\gamma)(k)$.

où $T_1(\gamma)(n - k)$ est la taille de l'arbre élagué jusqu'au niveau $n - k$.

et $T_2(\gamma)(k)$ est la taille totale nécéssaire au calcul des 2^{2^k} fonctions à k variables.

il est facile de voir que $T_1(\gamma)(n-k) = 3(1+2+\ldots+2^{n-k-1}) = 3(2^{n-k}-1)$.

Pour calculer toutes les fonctions de B_k, on calcule toutes les fonctions de B_{k-1}, puis toutes les expressions de la forme : $(f_i(x_1, \ldots, x_{k-1}) \wedge \bar{x_k}) \vee (f_j(x_1, \ldots, x_{k-1}) \wedge x_k)$ (voir figure 2.12).
Et donc $T_2(\gamma)(k) = 2^{2^k} + 2 \times 2^{2^{k-1}} + T_2(\gamma)(k-1)$.

Or $2 \times 2^{2^{k-1}} = o(2^{2^k})$ et $T_2(\gamma)(k-1) = o(2^{2^k})$.

Donc $T_2(\gamma)(k) = 2^{2^k}(1 + o(1))$.

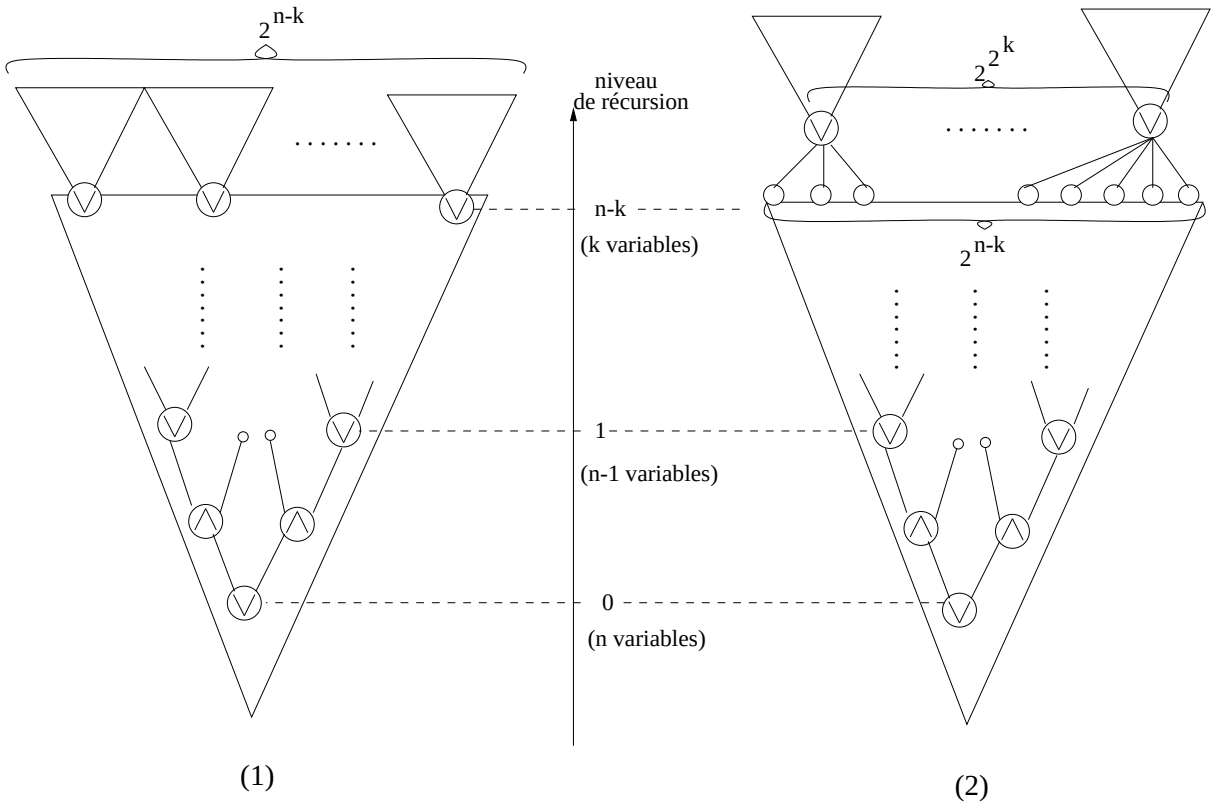


FIG. 2.11 – *Principe de la décomposition par BDD.*

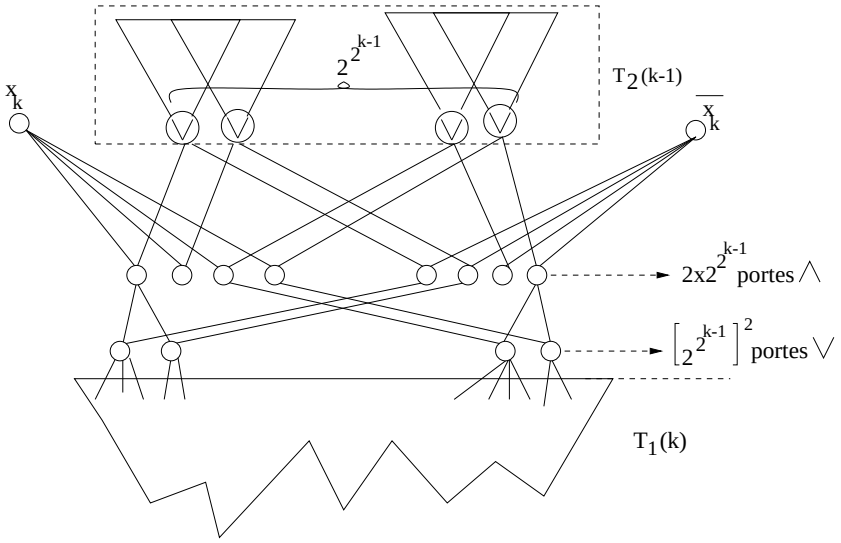


FIG. 2.12 – *Calcul de la taille du circuit pour la décomposition par BDD.*

Finalement, $\mathcal{T}(\gamma) = 3(2^{n-k} - 1) + 2^{2^k}(1 + o(1))$.

en prenant $k = \log(n - \log(n))$,

$$\begin{aligned} \mathcal{T}(\gamma) &= 3\,\frac{2^n}{n-\log(n)} + \frac{2^n}{n}(1 + o(1)). \\ &= \frac{2^n}{n}(4 + o(1)). \ \square \end{aligned}$$

* Complexité en espace : $C^e(\gamma) = O(2^n)$. Cette décomposition est bien sûr extrêmement prodigue en mémoire, puisque son principe est de ne calculer qu'une fois, et de garder en mémoire, toutes les fonctions booléennes de k variables.

2.3.3 Conclusions

Les résultats que nous venons de présenter nous amènent à tirer un certain nombre de conclusions.

Clairement, on ne peut espérer disposer d'une méthode universelle qui fournirait automatiquement le programme de la rétine à partir de la donnée d'une fonction booléenne (par sa table de vérité par exemple). Le théorème de Shannon nous apprend en effet que le temps de calcul pour une fonction de n variables sera presque toujours supérieur à $\frac{2^n}{n}$. Quant aux bornes inférieures, les décompositions canoniques qu'il est possible d'effectuer avec un petit nombre de plans mémoire ne nous garantissent que des limites inférieures de complexité qui sont plus qu'exponentielles.

Mais on se gardera bien de l'interprétation fataliste qui consiterait à conclure qu'il est vain d'espérer implanter un algorithme de traitement d'images en général ! Les chapitres suivants fournissent au contraire plusieurs exemples encourageants. Il est important de préciser deux points à ce sujet :

- Si la complexité d'une fonction booléenne est exponentielle en général, il est tout à fait possible que la plupart des fonctions qui nous intéressent échappent à cette règle. Et, les bornes supérieures correspondant aux pires cas, il est également possible qu'on puisse trouver, pour chaque fonction intéressante, une décomposition qui soit particulièrement compacte.
- Les résultats sont données sous forme asymptotique, or bien souvent les fonctions booléennes utilisées ont des supports très limités, correspondant à des petits voisinages.

Ces résultats fournissent néanmoins un enseignement fort, et observable en permanence dans la pratique : le manque de mémoire constitue la contrainte majeure dans l'optimisation des traitements rétiniens. En cas d'explosion combinatoire, il n'y aura souvent pas d'autre alternative que de disposer de plus de mémoire ou de renoncer au traitement envisagé.

Nous allons voir dans la section suivante comment il est possible, grâce à la *multigranularité*, de disposer d'une mémoire par pixel dynamiquement accrue, au prix d'une perte partielle de parallélisme, et le gain qui peut en résulter en temps de calcul.

Enfin, s'il n'est pas possible, par les opérateurs primitifs, d'échapper à des temps de calcul exponentiels en fonction du nombre de variables, la solution pourra être de *réduire le nombre de variables*. Dans la section 2.5, nous verrons qu'un opérateur de rétine en général est en fait défini par une composition plus ou moins complexe d'ORP. Dans ce cadre, le gain en temps de calcul consiste à exploiter dès que possible la *forme* du voisinage contenant les variables de la fonction booléenne à calculer pour décomposer son calcul par des sous-fonctions utilisant des voisinages beaucoup plus petits.

2.4 La multigranularité

Le modèle architectural que nous avons utilisé pour étudier la complexité des OR est celui d'une machine au parallélisme SIMD massif. Tous les processeurs de la rétine effectuent la même instruction au même instant. Avec les mesures de complexité que nous avons définies dans ce cadre, l'optimisation d'un traitement sur la rétine programmable revient à effectuer la minimisation suivante. Si l'on souhaite calculer une fonction booléenne f d'un certain voisinage sur une rétine comportant N registres binaires de mémoire par processeur, on cherchera l'OR Ω qui minimise $\mathcal{C}^t(\Omega)$, avec les contraintes (1) Ω calcule f et (2) $(C)^e(\Omega) \leq N$.

Or la contrainte (2) joue de toute évidence un rôle déterminant dans notre problème, où N est très petit. Précisons ce qualificatif, à la lumière des résultats présentés plus haut : pour une fonction booléenne calculée sur un voisinage 3×3, et avec $N = 8$, on dispose de moins d'un registre binaire par variable, cela signifie qu'une complexité en espace *linéaire* est déjà rédhibitoire.

C'est principalement dans le but de s'affranchir de cette contrainte qu'une architecture de rétine programmable à granularité variable a été proposée [85]. Le principe est de gagner de la mémoire au détriment du parallélisme. D'un point de vue architectural, la *multigranularité* implique la capacité à commander lignes et/ou colonnes de manière différente selon une certaine loi périodique. Ces différents niveaux de commande sont symbolisés dans la figure 2.13 : le niveau 0 correspond au mode SIMD total, où tous les processeurs reçoivent la même commande et effectuent la même action. Si l'on utilise les niveaux de commande 0 à k, les processeurs reçoivent des commandes différentes selon leur rang modulo 2^k, et ont donc des actions différentes.

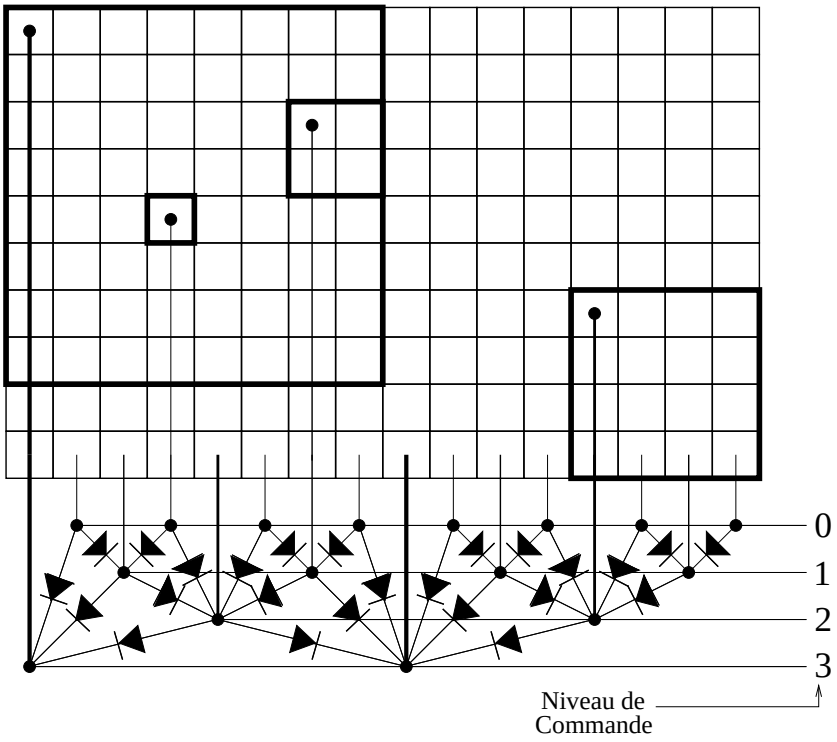


FIG. 2.13 – *La rétine programmable à granularité variable. Représentation symbolique d'un mode de commande multi-niveaux.*

Cette différenciation implique un regroupement des processeurs en super-processeurs qu'on appelle *clusters*. En quelque sorte, on est toujours en mode SIMD, mais la cellule de base est le cluster, qui peut être de taille et de puissance variable.

La multigranularité offre de nombreux avantages, en particulier celui de pouvoir faire de la *multirésolution*. Nous détaillerons ces applications dans les chapitres suivants, le but de cette section étant de présenter la multigranularité en tant que réponse au problème de pénurie de mémoire.

En effet, si les processeurs au sein d'un cluster peuvent avoir une action différente, on peut alors calculer un opérateur de rétine *en plusieurs passes*. L'algorithme est alors sérialisé, et le calcul est fait en autant de passes qu'il y a de processeurs dans le cluster. Le point fondamental est que chaque processeur «actif» peut alors disposer de la mémoire libre des processeurs «inactifs» de son cluster. Le processus de calcul en multigranularité est symbolisé sur la figure 2.14.

Grâce à ce principe, notre problème d'optimisation s'émancipe de la contrainte de complexité mémoire constante. En effet, soit Ω un OR, dont le calcul nécessite $\mathcal{C}^e(\Omega)$ registres binaires. Si l'on dispose de N registres binaires par processeurs, alors le regroupement de processeurs dans un cluster peut permettre au processeur actif de récupérer $N-2$ registres de mémoire par processeur inactif dans son cluster (On doit toujours conserver le registre contenant les variables, et un autre registre pour le calcul du résultat). Pour pouvoir calculer Ω, on doit donc regrouper les processeurs dans des clusters de taille δ, avec :

$$\delta = \frac{\mathcal{C}^e(\Omega)-2}{N-2}$$

Comme le traitement est sérialisé dans le cluster, le coût réel du traitement devient $\mathcal{C}^t(\Omega) \times \delta$, et finalement le problème d'optimisation est de trouver Ω qui minimise le produit $\mathcal{C}^t(\Omega) \times \mathcal{C}^e(\Omega)$, sous la contrainte : Ω calcule f.

Le gain en complexité n'est bien sûr effectif que, dans la mesure où, l'espace d'optimisation devenant beaucoup plus grand (à noter qu'il est néanmoins limité par la capacité de «clusterisation» du circuit, *Pvlsar2.2*, par exemple, ne permet que des regroupements de processeurs par quatre), on espère qu'entre un OR Ω calculable sur N bits et un OR Ω' qui nécessite plus de N bits, on puisse obtenir des gains de complexité $\frac{\mathcal{C}^t(\Omega)}{\mathcal{C}^t(\Omega')}$ qui soient beaucoup plus grands que le nombre de passes du semi-parallélisme $\frac{\mathcal{C}^e(\Omega')-2}{N-2}$. C'est heureusement souvent le cas. Nous en verrons un exemple remarquable dans la dernière section de ce chapitre.

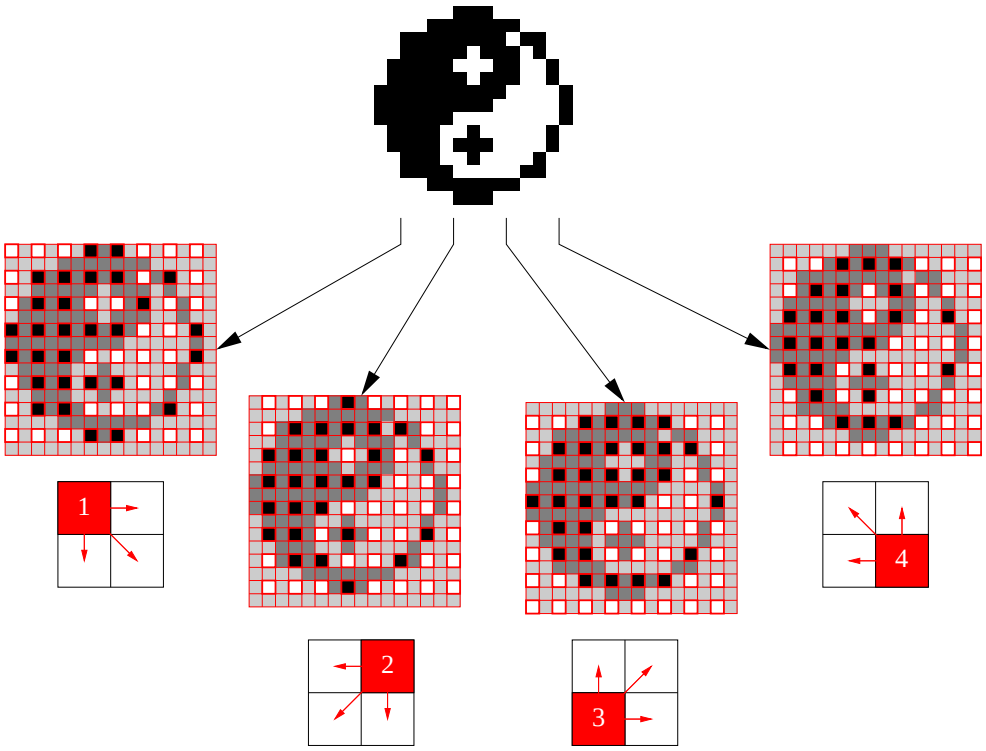


FIG. 2.14 – *Principe du calcul en plusieurs passes en utilisant la multigranularité. Les quatre processeurs composant le* cluster *effectuent* l'un après l'autre *le calcul pour leur pixel, en utilisant les registres mémoire des autres processeurs.*

2.5 Le principe de composition

C'est dans cette section que nous allons caractériser les opérateurs de rétine au sens général. Dans le cas particulier des ORP, nous n'autorisions les translations que sur un seul des plans binaires de la rétine. Cette supposition nous a permis de nous placer dans un cadre théorique assez riche, celui de la complexité des fonctions booléennes, et d'exploiter un certain nombre de résultats indépendamment de la forme du voisinage que constituent les variables de la fonction booléenne.

Cette restriction joue néanmoins un rôle fondamental dans les résultats de complexité que nous avons présentés. En effet, si l'on autorise les translations sur tous les plans binaires, le nombre de fonctions booléennes qu'on peut calculer avec un nombre donné d'opérations élémentaires *devient beaucoup plus important*. Cela signifie en particulier que l'argument de comptage de Shannon n'est plus valide. De plus, le nombre d'OE n'est plus limité inférieurement par le nombre de variables.

Cette propriété offre l'une des possibilité de gain en complexité les plus importantes pour les OR, c'est le *principe de composition*. L'exemple le plus simple, qui constitue la base de ce principe, nous vient encore une fois de la morphologie mathématique.

Il s'agit du principe des *polyèdres de Steiner*. Un tel polyèdre (de dimension n) est défini par une composition de dilatation par des segments. En exploitant simplement la propriété d'associativité de la dilatation, on peut gagner un très grand nombre d'opérations, car si d est le «diamètre» du polyèdre, une érosion (ou une dilatation) par ce polyèdre sans décomposition nécessite de l'ordre de d^n opérations, tandis qu'en décomposant par des érosions (ou dilatations) par les segments, on descend à environ $n \times d$.

Bien entendu, cette propriété s'applique à toute opération associative. La figure 2.15 illustre le procédé élémentaire de composition sur la rétine programmable pour l'application d'un opérateur associatif $\Box$ quelconque sur un voisinage de taille 3×3. On passe ainsi d'une complexité en temps de 8 pour l'ORP, à 4 pour l'OR égal à la composition de deux ORP. Sur le schéma 2.15(2), un rectangle représente un plan binaire sur lequel on effectue des translations. La valeur des registres de ce plan est celle calculée par le nœud qui se trouve juste au dessus.

L'utilisation de cette technique impose bien entendu des conditions sur la géométrie du voisinage. Cependant, tout élément structurant «octogonal» (convexe au sens trivial), peut se décomposer en une suite de dilatations par des éléments structurants *primitifs* [3] (c'est-à-dire contenus dans un carré 3×3. De plus, tout élément structurant peut s'écrire comme union d'éléments structurants octogonaux. Le calcul de l'union nécéssite un plan mémoire en

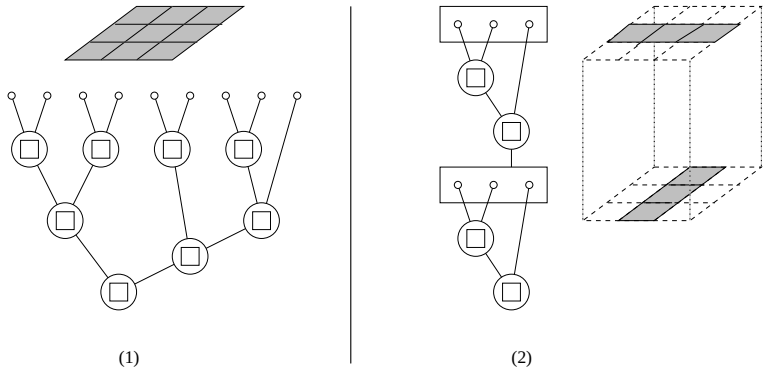


FIG. 2.15 – *Principe élémentaire de composition des OR. Calcul d'une «somme» associative sur un voisinage* 3×3 *par (1) un ORP de taille 8 (2) la composée de deux ORP de taille 2.*

plus sur la rétine, il n'en reste pas moins que le gain en nombre d'opérations pourra être très grand pour une large variété de géométrie de voisinage.

Le principe de composition n'est pas seulement valable pour les opérations morphologiques. En effet, toute opérateur numérique «séparable» relève de ce principe : l'application de deux noyaux de convolution de dimension un dans des directions orthogonales, ou le calcul de filtres à réponse impulsionnelle infinie par itération d'un filtre à support fini constituent deux exemples propres au domaine du traitement linéaire. Dans les chapitre suivants, le recours à la composition apparaîtra chaque fois que possible, et pas seulement dans les cas d'érosions et dilatations. La section suivante en fournira un exemple remarquable dans le cadre des fonctions seuil.

Enfin, il faut préciser que l'intérêt de l'étude de complexité menée au long de ce chapitre n'est absolument pas remise en question. Tous les résultats sont valides dans le cas des ORP, et ils doivent être considérés lorsqu'on ne peut faire aucune hypothèse sur la géométrie du voisinage.

En fait, un OR au sens général est caractérisé par une certaine composition d'ORP. La complexité en temps d'un OR est simplement définie par *la somme des complexités en temps des ORP qui le composent.* Comme la complexité en temps des ORP est toujours plus que linéaire, on a toujours intérêt à décomposer au maximum les OR par des ORP ayant le plus petit support possible.

Quant au calcul de la complexité en espace pour les OR, cela ne présente pas de difficultés. Le calcul se fait de la même façon en reportant sur les variables d'un plan translaté le nombre de plans binaires nécessaires au calcul du nœud qui le précède.

2.6 Un exemple : le cas des fonctions «seuil»

Nous présentons dans cette dernière section un exemple d'étude de complexité d'opérateurs de rétine dans le cadre d'une classe de fonction booléenne particulière. Cette étude se rattache à une réflexion menée par un groupe de travail [5] auquel nous avons collaboré, sur la complexité de l'implantation sur la rétine programmable des fonctions booléennes symétriques. Nous ne présentons ici que notre contribution aux résultats de cette réflexion, dont les conclusions, beaucoup plus larges, peuvent être trouvées dans [5].

Une fonction booléenne symétrique est une fonction booléenne invariante par toute permutation de ses variables. Par conséquent, la valeur prise par une telle fonction ne dépend que du *nombre* de variables booléennes à 1. Si l'on désigne par S_n le sous-ensemble de B_n composé des fonctions booléennes à n variables symétriques, il est possible [118] de représenter tout f de S_n par un $(n+1)-$vecteur binaire $(v_0, \ldots, v_n)$ tel que $x_1 + \ldots + x_n = i \Rightarrow f(x_1, \ldots, x_n) = v_i$. Les fonctions égalité E_n^i, qui sont les fonctions booléennes à n variables qui valent 1 si et seulement si la somme des n variables vaut i, constituent une base de S_n, puisque pour f représentée par $(v_0, \ldots, v_n)$, $f(x) = \bigvee_{i=0}^{n} E_n^i(x) \wedge v_i$.

Les *fonctions seuil* sont un autre type de fonctions symétriques. On notera T_n^k la fonction booléenne à n variables qui vaut 1 si et seulement si la somme des n variables *est supérieure ou égale à* k. Cette classe de fonction, plus intéressante en elle-même que les fonctions égalité pour le traitement d'image, permet également de calculer toute fonction symétrique, grâce à l'égalité :

$$E_n^k(x) = T_n^k(x) \bar{\rightarrow} T_n^{k+1}(x)$$

Comment calculer les fonctions seuil le plus rapidement possible ? Nous allons passer en revue différentes méthodes. A titre d'illustration, nous utiliserons l'exemple de la fonction T_9^2.

2.6.1 Forme disjonctive

La première méthode à examiner est la moins coûteuse en mémoire : la forme disjonctive, qui se calcule sur trois registres binaires. Pour calculer la fonction T_n^k par cette méthode, il suffit de voir que $T_n^k(x) = 1$ si et seulement si il existe au moins un monôme de taille k extrait de x qui vaut 1. La forme dijonctive consiste donc en la disjonction de tous les monômes de k variables parmi n. Le nombre total d'opérations élémentaires est très simple à calculer (voir figure 2.16). Il y a C_n^k monômes à k variables parmi n. Le calcul de ces monômes utilise $(k-1)$ ET. Il faut ensuite faire la disjonction entre tous les monômes, soit $C_n^k - 1$ OU. La complexité en temps du calcul de T_n^k par forme

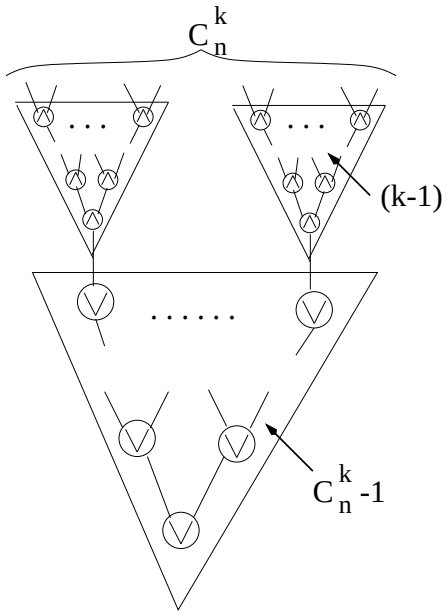


FIG. 2.16 – *Complexité du calcul des fonctions seuil par forme disjonctive.*

disjonctive est donc : $kC_n^k - 1$. Dans l'exemple de T_9^2, qui est celui que nous utilisons dans la suite, cela représente 71 opérations élémentaires.

2.6.2 Compteur saturant

La décomposition précédente fournissant une complexité en temps exponentielle, elle est inappliquable en général. A l'inverse, la méthode que nous présentons maintenant est beaucoup plus économe en opérations élémentaires, tout en restant dans le cadre des ORP. Il s'agit du calcul par *compteur saturant*. Le principe est tout simplement de calculer itérativement au moyen d'un compteur la somme en binaire des n variables, et de «basculer» à 1 un registre dès que la valeur k est atteinte. L'algorithme général est le suivant :

pour $i = 1 \ldots n$
    $c = p_0 \wedge x_i$
    $p_0 = p_0 \triangle x_i$
propagation de la retenue...
    pour $j = 1 \ldots \lfloor log(k) \rfloor$
        $p_j = p_j \triangle c$
        $c = c \vec{\rightarrow} p_j$
    fin_pour
saturation...

$c = p_{i_1} \wedge \ldots \wedge p_{i_t}$
$r = r \vee c$
fin_pour

Dans cet algorithme, on code la valeur de la somme sur les s registres $p_0 \ldots p_{s-1}$. La valeur de cette somme ne nous intéressant pas si elle dépasse k, on a $s = \lfloor log(k) \rfloor + 1$, $\lfloor . \rfloor$ désignant la partie entière. Un registre supplémentaire, noté c, est utilisé pour coder la retenue propagée. Signalons au lecteur pointilleux que l'algorithme utilise l'égalité $a \wedge b = a \bar{\rightarrow} (a \triangle b)$. Pour chaque variable, après addition, on teste si la valeur k est atteinte. Pour cela, on fait la conjonction de tous les bits parmi les s bits de la somme qui doivent être à 1 dans l'écriture binaire de k. Dès que c'est le cas, le registre noté r est mis à 1, et conservera cette valeur jusqu'à la fin.

On peut majorer facilement le nombre d'opérations élémentaires : Pour chacune des n variables, on effectue 2 OE par registre sur un nombre de registres au plus égal à s, soit $2n(\lfloor log(k) \rfloor + 1)$ opérations pour le calcul de la somme. Pour la saturation, le nombre d'opérations dépend du nombre de bits à 1 dans l'écriture binaire de k. Il y en a au plus s. Cela fait donc un nombre maximal d'OE égal à $3n(\lfloor log(k) \rfloor + 1)$. La complexité des fonctions T_n^k est maximale pour $k = \lceil \frac{n}{2} \rceil$ ($\lceil . \rceil$: partie entière supérieure), puisqu'au delà on raisonne par dualité (en comptant les 0). On a donc montré la proposition suivante :

Proposition 3 *La complexité en temps du calcul d'une fonction seuil à n variables par la méthode du compteur saturant est inférieure à $3n\lfloor log(n) \rfloor$.*

Complexité en espace :
La méthode utilise $\lfloor log(k) \rfloor + 1$ registres pour le calcul de la somme. 1 registre pour les variables. 1 registre pour la retenue. Pour le coût en mémoire de la saturation, il faut distinguer deux cas : Si l'écriture binaire de k comporte plus d'un chiffre à 1, alors il faudra calculer le résultat sur un registre r supplémentaire. Sinon, aucun registre supplémentaire n'est nécessaire, le résultat est lu dans le bit de poids fort de la somme.

La figure 2.17 présente un exemple d'utilisation du compteur saturant pour le calcul de la fonction T_9^2. La complexité en temps de cet ORP est de 29, la complexité en espace de 4.

2.6.3 Méthode par composition

Nous terminons en proposant la méthode de calcul des fonctions seuil *par composition*. Comme pour toute méthode par composition, elle n'est valable que sous certaines conditions concernant la géométrie du voisinage. Dans la

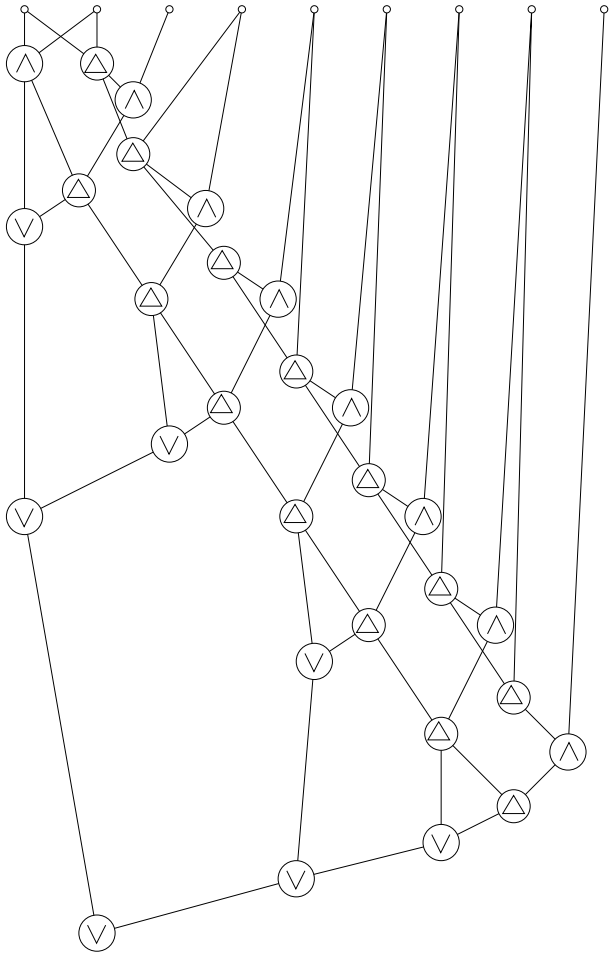


FIG. 2.17 – *Calcul de la fonction seuil T_9^2 par la méthode du compteur saturant.*

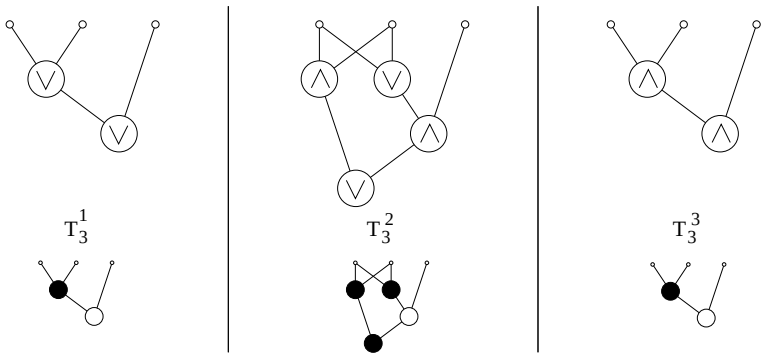


FIG. 2.18 – *Les briques de base pour le calcul des fonctions seuil sur un carré* 3×3 *sont les ORP calculant les fonctions seuil à 3 variables.*

suite, nous supposerons que le voisinage formé par les variables est un carré.

Le principe de cette méthode est basé sur le fait que les fonctions seuil T_n^k constituent une généralisation des opérateurs morphologiques de base. En effet T_n^1 n'est autre qu'une dilatation, tandis que T_n^n est une érosion morphologique. On a également la généralisation du principe de dualité :

$$T_n^k(x) = \overline{T_n^{n+1-k}(\bar{x})}.$$

Appliquer la décomposition des polyèdres de Steiner dans le contexte des fonctions seuil revient à faire la remarque suivante, par exemple pour un voisinage 3×3 : «dire qu'il y a au moins un point à 1 dans un carré 3×3 équivaut à dire qu'il y a au moins une colonne 1×3 contenant au moins un point à 1».

C'est ce principe que nous appliquons dans notre méthode. Si nous nous plaçons dans le cas où les variables forment un carré 3×3, en décomposant le calcul en colonnes, on réduit à 3 (au lieu de 9) le nombre de variables des ORP qui vont intervenir dans le calcul des fonctions T_9^k. En fait les OR calculant les T_9^k sont définis par composition des 3 ORP représentés sur la figure 2.18, qui calculent les fonctions T_3^k, pour $k = 1, 2, 3$. La complexité en temps de ces ORP est respectivement 2, 4 et 2. La complexité en espace est respectivement 2, 3 et 2 (voir coloriage en bas de la figure).

La figure 2.19 montre l'exemple du calcul par la méthode de composition de la fonction T_9^2. Le rectangle du haut correspond à trois variables se trouvant sur la même ligne, les deux rectangles du bas à trois variables se trouvant sur la même colonne. En regardant attentivement la figure, on retrouve les ORP de la figure 2.18. On retrouve alors facilement le principe de composition pour le calcul de T_9^2 : «Dire que la somme des variables est supérieure à 2 dans le carré 3×3 équivaut à dire qu'il y a au moins 1 colonne

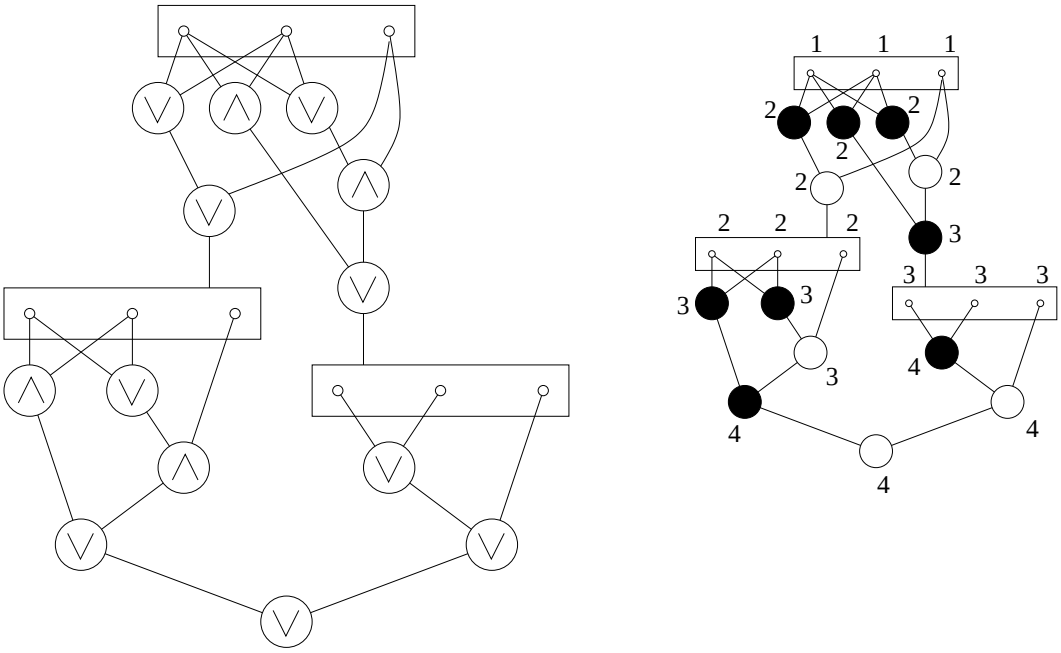


FIG. 2.19 – *Calcul de la fonction seuil T_9^2 sur un voisinage 3×3 par la méthode de composition.*

dont la somme vaut 2 ou au moins 2 dont la somme vaut 1». La complexité en temps de l'OR correspondante est de 13, la complexité en espace, 4 (voir coloriage et calcul figure 2.19, à droite).

Ce procédé peut être généralisé pour toute valeur de n et k tant que le voisinage est un carré. Sur la figure 2.20, on montre les procédés de calcul pour toutes les fonctions seuil calculées sur un carré 3×3. Les ORP de bases (voir figure 2.18) étant parfaitement caractérisés, leur complexité en temps et espace connues, on les représentera pour simplifier par des macro-nœuds.

2.6.4 Résultats et conclusion

Finalement nous résumons dans les tableaux 2.2 et 2.3 les performances des trois méthodes présentées. La méthode par composition s'avère sensiblement plus rapide que la méthode du compteur saturant, qui est certainement l'un des opérateurs primitifs les plus efficaces, sans utiliser plus de mémoire.

Dans [20], il est prouvé que pour tout $n \geq 2$, un circuit booléen calculant T_n^2 comporte au moins $2n-4$ nœuds. Dans le calcul de T_9^2, on est en dessous de cette limite théorique. Ce résultat est remarquable car il signifie que grâce au principe de composition, on peut faire mieux qu'avec n'importe quel ORP.

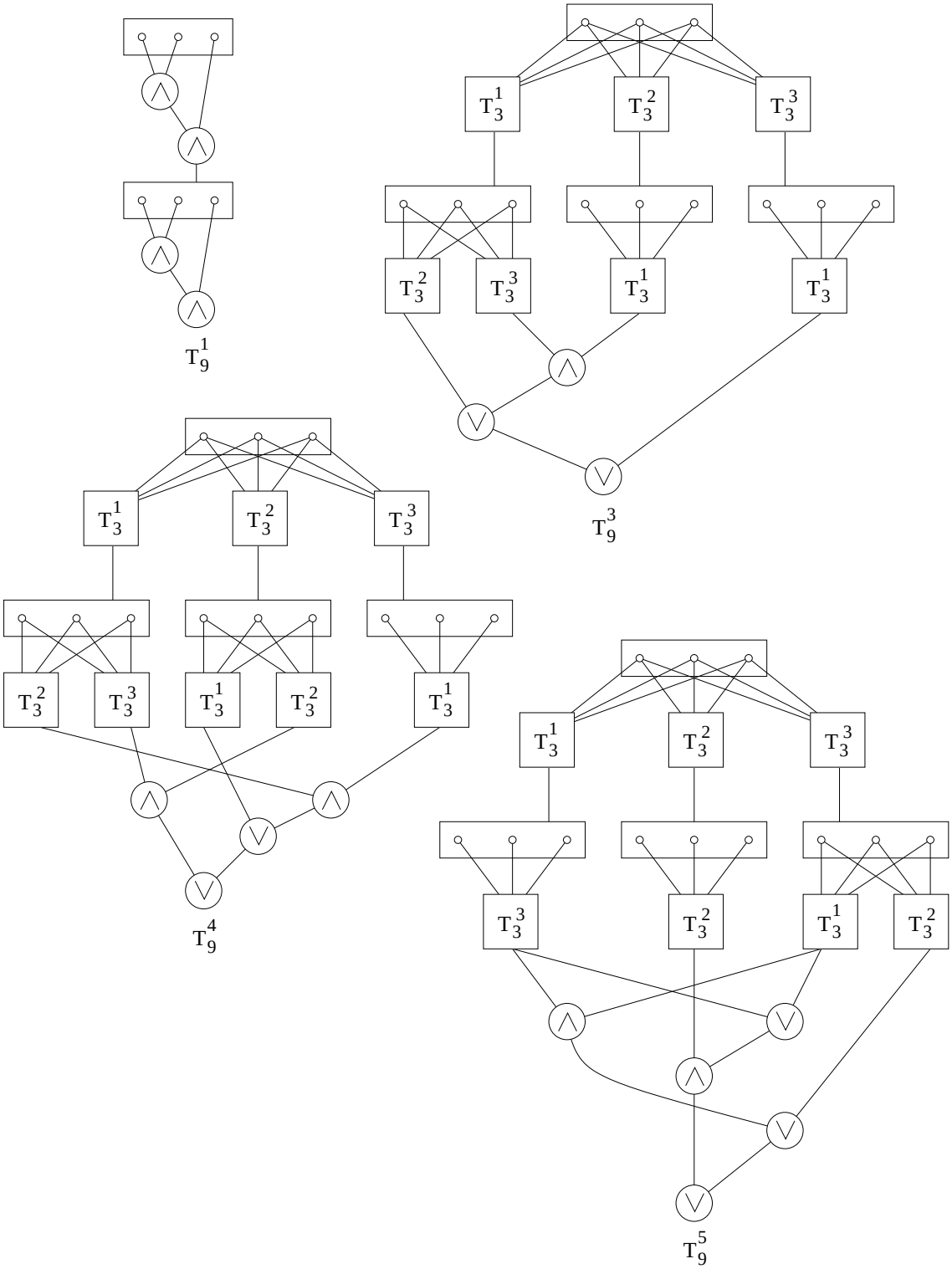


FIG. 2.20 – *Calcul des autres fonctions seuil T_9^k sur un voisinage 3×3 par la méthode de composition.*

Fonction	T_9^1	T_9^2	T_9^3	T_9^4	T_9^5
Forme disjonctive	8	71	251	503	629
Compteur saturant	8	29	37	41	45
Méthode par composition	4	13	21	26	25

TAB. 2.2 – *Complexité en temps du calcul des fonctions seuil par les différentes méthodes présentées.*

Fonction	T_9^1	T_9^2	T_9^3	T_9^4	T_9^5
Forme disjonctive	2	3	3	3	3
Compteur saturant	2	4	5	5	6
Méthode par composition	3	4	5	5	5

TAB. 2.3 – *Complexité en espace du calcul des fonctions seuil par les différentes méthodes présentées.*

Comment exploiter ces résultats dans le cadre de la multigranularité ? Nous avons vu dans la section 2.4 que le but était de minimiser le produit $\mathcal{C}^t(\Omega) \times \mathcal{C}^e(\Omega)$. Le tableau 2.4 donne la valeur de la complexité en temps à capacité mémoire fixée (3 ou 4 ici). Sachant que le processeur actif récupère $(N-2)$ bits de chaque processeur passif de son cluster, on voit que pour appliquer la méthode par composition lorsqu'on a trois registres, un regroupement par 2 est nécessaire lorsque $k = 2$, et par 4 lorsque $k \geq 3$. Les temps de calcul augmentent en conséquence puisque l'opération est sérialisé dans le cluster. Il faut cependant mettre en comparaison ces chiffres avec ceux obtenus avec les ORP sur 3 bits, tels que la forme disjonctive.

2.6.5 Application : les filtres de rang

Même si les fonctions T_9^2 et T_9^3 sont très utiles pour implanter *le jeu de la vie* du mathématicien Conway, nous présentons la principale application

Fonction	T_9^1	T_9^2	T_9^3	T_9^4	T_9^5
3 registres	4	26	84	104	100
4 registres	4	13	42	52	50

TAB. 2.4 – *Complexité en temps du calcul des fonctions seuil en utilisant 3 ou 4 registres, avec recours éventuel à la multigranularité.*

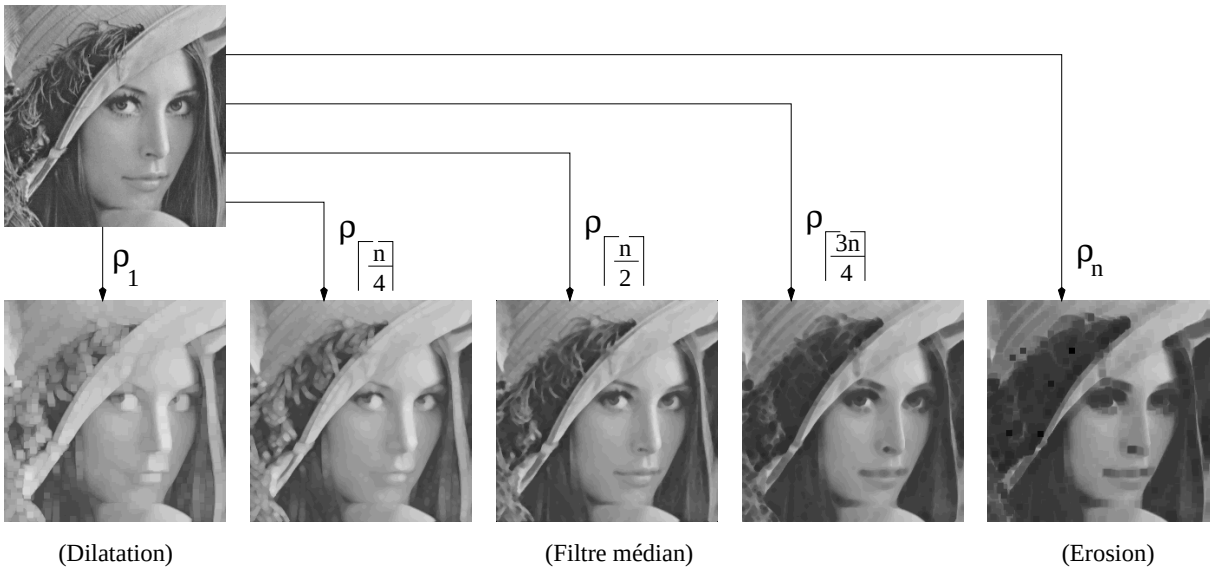


FIG. 2.21 – *Application des fonctions seuil au calcul des filtres de rang.*

des fonctions seuil dans le cadre du traitement d'image. Il s'agit du calcul des *filtres de rang* pour une image numérique.

Soit $x = (x_1, \ldots, x_n)$ un vecteur numérique (les x_i sont des entiers positifs majorés par un entier M). On peut le représenter (voir chapitre précédent) par un ensemble de M coupes binaires $(a^j)_{0 \leq j \leq M-1}$, où les $a^j = (a_1^j, \ldots, a_n^j)$ sont les vecteurs binaires définis par $a_i^j = 1$ si et seulement si $x_i \geq j$. On a alors $x = \sum_{j=0}^{M-1} a^j$.

Notons $\rho_k(x)$ l'opérateur défini par $\rho_k(x) = \sum_{j=0}^{M-1} T_n^k(a^j)$.

Soit $x = (x_1, \ldots, x_n)$. Si σ est une permutation de $(1, \ldots, n)$ telle que $x_{\sigma(1)} \leq \ldots \leq x_{\sigma(n)}$, on montre que :

$$\rho_k(x) = x_{\sigma(k)}$$

Les opérateurs ρ_k forment une importante catégorie de *filtres non linéaires*, celle qui consiste à remplacer un ensemble de n valeurs numériques par celle qui occupe un certain rang lorsqu'on les classe par ordre croissant. La figure 2.21 montre quelques exemples de filtrage de rang pour une image numérique.

Considérant ici une fenêtre de taille n, on retrouve pour les valeurs extrêmes les versions numériques des opérateurs morphologiques de base (dilatation pour le rang 1, correspondant au *max*, et érosion pour le rang n, correspon-

dant au *min*). On peut voir également les traitements correspondant aux rangs intermédiaires. Un cas particulier important est celui du rang $\lceil \frac{n}{2} \rceil$, qu'on appelle *filtre médian*, très utile en analyse d'images, et que nous retrouverons dans les chapitres suivants.

Chapitre 3

Transformations homotopiques

3.1 Introduction

Ce chapitre est consacré à une étude d'implantation des transformations homotopiques sur la rétine numérique, c'est-à-dire des opérateurs qui préservent les relations de connexité dans les images binaires. Nous avons choisi d'étudier et de présenter cette classe d'algorithmes pour deux raisons que nous exposons maintenant.

La première est liée au fait que l'architecture de la rétine programmable présente une adéquation certaine pour ces algorithmes, qui se prêtent parfaitement au parallélisme SIMD de la rétine, se calculent par des interactions sur des petits voisinages, qu'on peut traduire par des expressions booléennes calculables avec très peu de mémoire. Une telle adéquation, si elle était pressentie, n'était pas acquise; nous la mettons en évidence par la proposition d'algorithmes originaux dans les sections 3.4, 3.5 et 3.6. L'efficacité de ces algorithmes sur notre architecture, ainsi que leurs qualités intrinsèques sont démontrés section 3.7.

La seconde raison qui motive le choix de cette étude concerne l'intérêt des transformations homotopiques pour le traitement d'image et la reconnaissance de formes. L'existence d'opérateurs de segmentation basés sur la topologie, comme nous le verrons dans le chapitre suivant, justifie notre intérêt pour le noyau homotopique et le SKIZ, présentés section 3.4. La squelettisation, que nous présentons dans les sections suivantes, est quant à elle connue comme une opération fondamentale de prétraitement dans les algorithmes de reconnaissance de formes.

Le principe de base régissant les algorithmes présentés dans ce chapitre est le principe d'homotopie, c'est-à-dire de préservation de la topologie. Avant de la formaliser, nous pouvons donner une intuition de cette propriété par

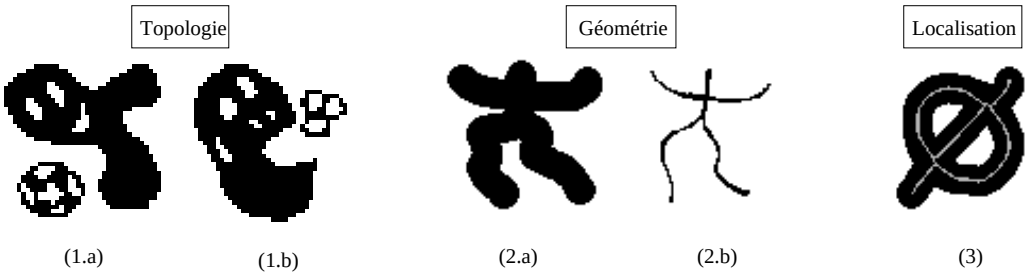


FIG. 3.1 – *Les principes de la squelettisation.*

la première illustration de la figure 3.1. Quel point commun y a-t-il entre un fantôme qui joue au football (1.a) et un gorille avalant un bretzel (1.b)? La topologie ! On peut en effet passer d'une image à l'autre en enlevant ou ajoutant des points *sans jamais* séparer ni fusionner de composantes connexes, ni créer ou percer de trous.

Cet exemple montre que la préservation de la topologie ne suffit pas à représenter fidèlement une forme. La squelettisation impose en plus que la *géométrie* de la forme soit respectée. Sur les images (2.a) et (2.b) de la figure 3.1, par exemple, les caractéristiques de formes tels que ramifications, élongations, courbures sont très proches. De plus, le *squelette* doit rendre compte de la localisation de l'objet, comme sur la figure 3.1(3), où les deux images sont superposées.

Historiquement, la représentation de forme par squelette date du début des années 60, avec la notion d'*Axe médian* de Blum [19]. Plus tard, la préservation de la géométrie est formalisée par Calabi [23] avec la notion de *réunion des centres des boules maximales*. Les premiers travaux sur l'homotopie et la semi-continuité du squelette sont principalement dûs à Matheron [108], et les recherches sur le SKIZ à Lantuéjoul [56].

Les recherches sur les algorithmes de calcul du squelette numérique se sont intensifiées au cours des deux dernières décennies, en 2D et en 3D, avec plusieurs centaines d'articles publiés sur le sujet. On peut grossièrement regrouper ces algorithmes dans trois classes :

- Ceux qui sont basés sur la *fonction distance*.
- Ceux qui sont basés sur le concept de *feux de prairies*.
- Ceux qui consistent à produire une suite décroissante d'images homotopes par une opération appelée *amincissement*.

Les premiers sont en général des algorithmes non itératifs, qui calculent en chaque point de l'image sa distance au complémentaire, puis raisonnent sur les configurations des valeurs dans le voisinage pour décider de l'appartenance

HOMOTOPIE	Le squelette doit posséder la même topologie que l'image initiale.
MÉDIALITÉ	Le squelette doit se trouver «au milieu» de l'image initiale.
ISOTROPIE	Le squelette doit être invariant à une rotation de $\pi/2$ de l'image initiale.
ÉPAISSEUR UNITÉ	Le squelette doit être fait de courbes fines.
IMMUNITÉ AU BRUIT	Le squelette doit être peu modifié par le retrait ou l'ajout aléatoire de points au contour initial.

TAB. 3.1 – *Les principales caractéristiques recherchées dans les algorithmes de squelette numérique.*

au squelette.

Les seconds utilisent le concept le plus ancien dans la détermination des squelettes: en imaginant qu'un feu allumé sur les contours de l'objet se propage à vitesse constante, le squelette est constitué des points de rencontre de différents fronts à l'intérieur de l'image. Ces algorithmes utilisent généralement le support des contours actifs (snakes en anglais), sont formalisés de façon continue et implantés par des schémas numériques standards.

Les troisièmes sont des algorithmes par relaxation, ils consistent à retirer itérativement des points du contour de l'image sans modifier la topologie et, autant que possible, la géométrie, et ceci jusqu'à convergence.

Les algorithmes que nous présentons dans ce chapitre appartiennent à cette dernière classe, parce qu'elle semble la mieux adaptée au parallélisme de données et à la pénurie de mémoire. Il serait cependant fort intéressant de disposer d'arguments plus formels, en étudiant d'autres types d'implantation [116].

Le tableau 3.1 liste les 5 principales caractéristiques que l'on attend d'un algorithme de squelettisation. Comme nous le verrons par la suite, certaines de ces propriétés sont incompatibles, et c'est dans la recherche d'un meilleur compromis que doit se situer un algorithme de squelettisation.

Les sections 3.2 et 3.3 ont pour objet de formaliser les différentes propriétés que nous venons d'évoquer, et de permettre leur expression sous forme booléenne.

Dans les 3.4 à 3.6, nous présentons nos transformations homotopiques, dont nous étudions les propriétés et comportements de manière comparative dans la section 3.7.

Les sections 3.8 et 3.9 ont pour objet de démontrer l'intérêt de l'algorithme de squelettisation proposé indépendamment de notre architecture. Nous montrons en effet que l'algorithme s'adapte facilement à la maille hexagonale, et surtout qu'il peut s'exprimer en maille cubique dans n'importe quelle dimension. Nous illustrons ce fait par une présentation des résultats et des propriétés de l'algorithme en 3D.

3.2 Topologies et distances discrètes

Les algorithmes qui sont présentés dans ce chapitre sont basés sur la *topologie*, c'est-à-dire que le calcul fait toujours intervenir des critères de *connexité*. Dans le cas de la squelettisation, un autre critère joue également un rôle important, qui n'est plus topologique, mais *métrique*, ce qui signifie qu'il fait intervenir des notions de *distance*.

L'objet de cette section est de présenter les quelques notions mathématiques nécessaires à la compréhension de l'ensemble du chapitre, à l'exclusion des deux dernières sections qui auront leur propre formalisme. Ici, nous nous intéressons exclusivement à l'implantation d'opérateurs homotopiques sur la rétine programmable, ce qui impose une certaine structure discrète : dimension deux, maillage carré.

3.2.1 Topologies dans la maille carrée

Soit $\mathbb{Z}^2$ le plan discret. On notera $x = (x^1, x^2)$ et $y = (y^1, y^2)$ deux *points* de $\mathbb{Z}^2$ représentés par leurs coordonnés cartésiennes. On définit les relations de *voisinage* dans la maille carrée de la façon suivante :

Définition 13 *x et y sont* 4-voisins *si et seulement si :*

$$\delta_4(x, y) = |x^1 - y^1| + |x^2 - y^2| \leq 1.$$

x et y sont 8-voisins *si et seulement si :*

$$\delta_8(x, y) = max(|x^1 - y^1|, |x^2 - y^2|) \leq 1.$$

La figure 3.2 illustre ces relations de voisinage. Les chiffres 4 et 8 font référence aux nombres de voisins pour chaque relation. On utilisera pour désigner les voisins d'un points x l'indexation par la direction cardinale, comme indiqué sur la figure 3.2(3).

Les opérateurs δ_4 et δ_8 de la définition 13 sont des distances qu'on dit *induites* par la $4-$ et la $8-$connexité respectivement.

Définition 14 *Soit A et B deux sous-ensembles de $\mathbb{Z}^2$. A et B sont dits* $K-$voisins *($K = 4$ ou 8) si et seulement si :*
Il existe $x \in A$ et $y \in B$ tels que x et y sont K-voisins.

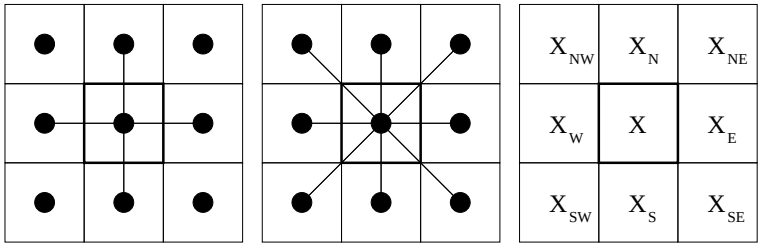


FIG. 3.2 – *Les deux relations classiques de connexité dans la maille carrée. (1) 4-connexité (2) 8-connexité (3) Noms utilisés pour désigner les 8 voisins d'un point x.*

Définition 15 *Soit $X \subset \mathbb{Z}^2$ X est une* K-composante connexe *(K-cc) si et seulement si :*
Il n'existe pas de partition de X en deux sous-ensembles qui ne soient pas K-voisins.

Définition 16 *Soit $x \in X \subset \mathbb{Z}^2$. Le point x est dit* K-intérieur *à X si et seulement si tous les K-voisins de x appartiennent à X.*

3.2.2 Fonction distance - Maxima locaux

Soit $x \in \mathbb{Z}^2$, et $r \in \mathbb{N}$.
On note $\mathcal{B}_K(x,r) = \{y \in \mathbb{Z}^2; \delta_K(x,y) \leq r\}$ la boule de centre x de rayon r de la distance δ_K.

Définition 17 *Soit $X \subset \mathbb{Z}^2$. La boule $\mathcal{B}_K(x,r)$ est dite* maximale *pour X si et seulement si :*

$$\forall y \in X,\ \forall q \in \mathbb{N},\ \mathcal{B}_K(x,r) \subset \mathcal{B}_K(y,q) \subset X \Rightarrow (y,q) = (x,r).$$

Définition 18 *Soit $X \subset \mathbb{Z}^2$. On définit $\mathcal{S}_K(X)$ l'*axe médian *de X relativement à la distance δ_K comme la réunion des centres des boules maximales de la distance δ_K.*

Définition 19 *$X \subset \mathbb{Z}^2$. On note $X^c = \mathbb{Z}^2 \setminus X$ le complémentaire de X. La* fonction K-distance *associée à X est la fonction :*

$$\begin{array}{rccc} \Phi_K : & \mathbb{Z}^2 & \longrightarrow & \mathbb{N} \\ & x & \longmapsto & \Phi_K(x) = \delta_K(x, X^c). \end{array}$$

Propriété 2 $\mathcal{S}_K(X) = \bigcup\{x \in X; \forall y \in \mathcal{B}_K(x,1), \Phi_K(y) \leq \Phi_K(x)\}$ *:*
La réunion des centres des K-boules maximales est égale à l'ensemble des maxima locaux de la fonction K-distance.

$p^0 = \text{image};$	$p^0 = image;$
$p^1 = p^0 \wedge p^0_{SW};$	$p^1 = p^0 \wedge p^0_N \wedge p^0_S;$
$p^0 = p^0 \wedge p^1_N \wedge p^1_E;$	$p^0 = p^1 \wedge p^1_E \wedge p^1_W;$

TAB. 3.2 – *Calcul des érosions élémentaires en 4 et 8-connexité*

3.2.3 Calcul sur la rétine programmable

Voyons à présent comment effectuer le calcul des différents ensembles que l'on vient de définir sur la rétine programmable.

Si l'on note $B_K \in \mathbb{Z}^2$ l'ensemble des K-voisins de l'origine, alors l'ensemble des points K-intérieurs de l'ensemble X est égal à $X \ominus B_K$, l'érodé de X par la K-boule élémentaire. Le calcul des érosions élémentaires est donné par le tableau 3.2. On y utilise le principe de composition au maximum. Le nombre d'opérations est de 3 pour la 4-connexité et 4 pour la 8-connexité. Les calculs se font sur 2 plans binaires si le plan contenant les données image n'est pas conservé, et 3 s'il doit l'être.

Si l'on note $X^0 = X$ et $X^{n+1} = X^n \ominus B_K$, l'ensemble X^n est l'ensemble des points de X dont la valeur de la fonction K-distance est strictement supérieur à n. $x \in X$ est un maximun local de la K-distance si et seulement si il existe n tel que $x \in X^n$ et $x \notin X^{n+1} \oplus B_K$ ($\oplus$ désignant la dilatation). L'axe médian pour la K-distance est donc donné par :

$$\bigcup_{n \geq 0} X^n \setminus (X^{n+1} \oplus B_K)$$

Pour calculer l'axe médian sur la rétine, on applique donc l'algorithme du tableau 3.3 (5 plans mémoire sans conserver l'image initiale). Le plan p_0 contient les érodés successifs X^n, dans le plan p_1, on calcule les points non K-intérieurs à X^n qui ont un K-voisin K-intérieur à X^n. La différence, calculée dans le plan p_3, est constituée des points qu'on appelle *résidus d'ouverture*. Le résultat, calculé dans p_4 est l'union sur toutes les itérations des résidus d'ouverture. Les lignes comprenant deux plans à gauche de l'affectation correspondent à des macro-instructions qui utilisent deux plans binaires. La notation $\tilde{p}$ signifie que la valeur du plan p a été modifiée.

Le calcul de l'axe médian nécessite 8 opérations élémentaires à chaque passe pour la 4-distance, 10 pour la 8-distance. Chaque passe correspond à une érosion par une K-boule de rayon 1 et la convergence à la disparition de l'objet, le nombre de passe est donc égal au maximum sur X de la fonction K-distance. Donc si ρ_K est le rayon de la plus grande K-boule contenue dans X, la complexité du calcul est : $\rho_K(\frac{K}{2} + 6)$.

$p_0 =$ image;
$p_4 = \emptyset$;
répéter {
$p_3 = p_0$;
$(p_0, p_1) = (erode_K(p_0), \tilde{p_1})$;
$p_1 = p_0$;
$(p_1, p_2) = (dilate_K(p_1), \tilde{p_2})$;
$p_3 = p_3 \vec{\neg} p1$;
$p_4 = p_4 \vee p_3$;
} tant que $p_0 \neq \emptyset$;

TAB. 3.3 – *Calcul de l'axe médian*

3.3 Homotopie

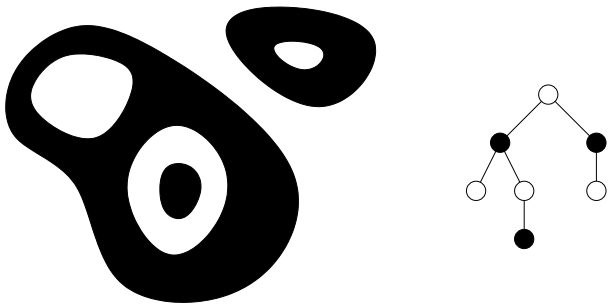
3.3.1 Ensembles homotopes

Dans le plan continu $\mathbb{R}^2$, la notion d'*homotopie* concerne les *arcs*, c'est-à-dire les fonctions continues de l'intervalle $[0, 1]$ dans $\mathbb{R}^2$. Soit A une partie de $\mathbb{R}^2$. Deux arcs γ_1 et γ_2 sont homotopes dans A si et seulement si il existe une fonction θ *bicontinue* de $[0, 1] \times [0, 1]$ dans A telle que :

$$\forall t \in [0, 1] : \quad \theta(t, 0) = \gamma_1(t), \text{ et}$$
$$\theta(t, 1) = \gamma_2(t).$$

L'homotopie étant une relation d'équivalence, on montre que pour un ouvert A quelconque de $\mathbb{R}^2$, l'ensemble des classes d'équivalence des arcs de A forment un groupe pour la loi de concaténation qu'on appelle *groupe fondamental* de A. Ce groupe caractérise la *topologie* des ouverts du plan. On peut prolonger la notion d'homotopie aux ouverts de $\mathbb{R}^2$ en disant que deux ouverts sont *homotopes* si et seulement si leurs groupes fondamentaux sont isomorphes; cela signifie qu'ils ont la même topologie. Etant donné F une fonction du plan dans lui-même, on dira que F *préserve* la topologie si pour tout ouvert A, A et $F(A)$ sont homotopes.

Dans le cas d'un sous-ensemble borné X du plan discret, on peut représenter son groupe fondamental par un *arbre orienté* (voir figure 3.3), dont chaque nœud représente une composante de X (les noirs) ou de X^c (les blancs). La racine de l'arbre représente la composante infinie de X^c. Chaque nœud admet autant de successeurs que la composante qu'il représente *entoure* de composante connexe du complémentaire. Avec cette représentation, des ensembles homotopes auront des arbres isomorphes.



(1) (2)

FIG. 3.3 – *Représentation du groupe fondamental par l'arbre de connexion.*

Cette caractérisation de la topologie peut paraître assez intuitive, en réalité, le fait qu'elle soit bien définie n'est pas du tout évident. En effet, dire qu'une composante connexe A de X «entoure» une composante connexe B de X^c (ou inversement) signifie qu'il existe une *courbe simple fermée* (courbe de Jordan) γ appartenant à A telle que B appartient à l'*intérieur* de γ. On utilise donc implicitement le théorème de Jordan, qui dit que toute courbe de Jordan sépare le plan en deux parties distinctes, une finie (qu'on appelle intérieur) et une infinie (l'extérieur).

Or dans la maille carrée, cette propriété n'est vraie que dans des conditions particulières. Examinons la figure 3.4. Si l'on considère la 8-connexité (1), l'objet X, en noir, est une courbe de Jordan, mais X^c forme une unique composante connexe. En 4-connexité (2), au contraire, X n'est pas une courbe de Jordan, mais est formé de plusieurs composantes connexes. Cependant, il sépare X^c en deux composantes connexes. En fait, le problème est résolu en choisissant pour X et pour X^c deux types de connexité différentes. Ainsi, dans la maille carrée, on considérera toujours soit la (8,4)-topologie (3), soit la (4,8)-topologie (4).

Dans la suite, on notera $\tilde{K} = 12 - K$ pour désigner la connexité duale. La notion d'homotopie que nous utiliserons dans la maille carrée est donc la suivante :

Définition 20 *A et B sont K-homotopes si et seulement si : il existe une bijection entre l'ensemble des K-cc de A et celles de B et une bijection entre les $\tilde{K}$-cc de A^c et celles de B^c, qui préservent la relation d'entourage.*

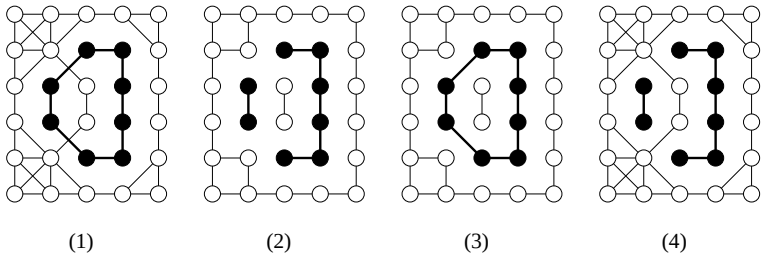


FIG. 3.4 – *Le théorème de Jordan dans la maille carrée n'est valable que si l'on choisit des connexités différentes pour X et pour X^c.*

3.3.2 Les points simples et leur caractérisation

L'homotopie de la définition 20 est la notion la plus générale concernant la préservation de la topologie. C'est une relation d'équivalence, ce qui est conforme à la notion d'homotopie dans le plan continu. Elle est parfois désigné par *homotopie symétrique* [63].

Pour caractériser les transformations homotopiques, il nous faudra cependant restreindre la notion précédente à des propriétés plus fortes. La première étape est celle de la *sous-homotopie* [63]. Si A et B sont homotopes et que $A \subset B$, on dira que A est sous-homotope à B. C'est maintenant une relation d'ordre, mais plus simple à caractériser, grâce à la propriété suivante, due à Stefanelli et Rosenfeld [110] :

Propriété 3 *A est K-sous-homotope à B si et seulement si les deux conditions suivantes sont vérifiées :*

- *Toute K-cc de B contient exactement une K-cc de A.*
- *Toute $\tilde{K}$-cc de A^c contient exactement une $\tilde{K}$-cc de B^c.*

Ceci nous amène à la notion clef, celle qui va fournir les critères explicites, la *simplicité* :

Définition 21 *Soit X un sous-ensemble de $\mathbb{Z}^2$. $A \subset X$ est dit K-*simple *dans X si et seulement si $X \setminus A$ est K-sous-homotope à X.*
Soit $x \in X$. On dira que x est un point K-simple *de X si $\{x\}$ est K-simple dans X.*

Sur, la figure 3.5, par exemple, le point x est à la fois 8-simple et 4-simple, le point y est 8-simple mais pas 4-simple car sa destruction crée une 4-cc supplémentaire. z est 4-simple mais pas 8-simple car sa destruction fait disparaître un trou. t n'est ni 8-simple ni 4-simple.

Un résultat très important [53] est que la simplicité d'un point peut être caractérisée en examinant son 8-voisinage seulement. Soit $x \in X$. Si l'on note $V_K(x)$ l'ensemble des K-voisins de x, alors on a le résultat suivant :

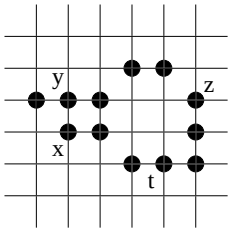


FIG. 3.5 – *x et y sont 8-simples. x et z sont 4-simples.*

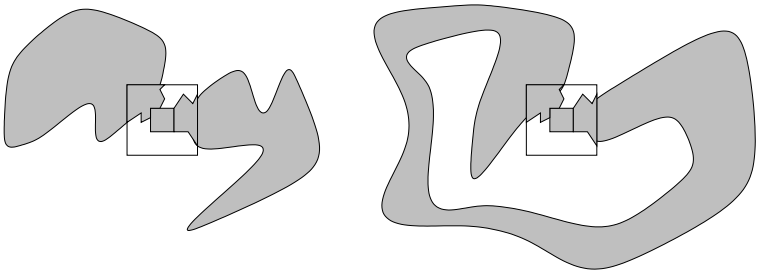


FIG. 3.6 – *Pourquoi la simplicité peut être déterminée localement.*

Théorème 5 *Soit $x \in X$ tel que $V_{\tilde{K}}(x) \cap X^c \neq \emptyset$. Alors x est K-simple si et seulement si x est K-voisin d'exactement une K-cc de $(V_8(x) \setminus \{x\}) \cap X$.*

Sans détailler la preuve de ce théorème, expliquons-en les deux enseignements majeurs : d'abord, si x est K-simple alors x est nécessairement un point du $\tilde{\text{K}}$-contour (c'est-à-dire que x a un $\tilde{\text{K}}$-voisin dans X^c), car sinon la destruction de x crée une nouvelle $\tilde{\text{K}}$-cc de X^c (un trou). Ensuite, pour comprendre pourquoi x n'est pas simple si sa destruction crée plus d'une K-cc de X dans le 8-voisinage de x, il suffit d'examiner la figure 3.6. S'il existe au moins deux K-cc dans $(V_8(x) \setminus \{x\}) \cap X$, alors soit ces deux K-cc appartiennent au niveau global à deux K-cc différentes (1), et alors la destruction de x déconnecte X, soit ce n'est pas le cas (2), mais alors la destruction de x connecte deux $\tilde{\text{K}}$-cc de X^c (perce un trou). Dans les deux cas, le fait d'enlever x entraîne bien un changement de topologie.

On sait à présent qu'on peut caractériser, dans le voisinage 3×3 la simplicité d'un point, il nous reste à savoir *comment* le calculer.

La première formule booléenne dont l'objet était de calculer la simplicité est due à Rutowitz [101]. Etant antérieure aux principaux travaux sur l'homotopie discrète, elle ne tient pas compte cependant du problème concernant le théorème de Jordan dans la maille carrée. Le *crossing number* de Rutowitz

utilise un seul type de connexité pour X et pour X^c. Il est défini par la formule ci-dessous, où les x_i représentent les 8-voisins de x de la façon suivante : $x_0 = x_E$, et ainsi de suite dans le sens direct jusqu'à $x_7 = x_{SE}$. Le signe + dans les indices est l'addition modulo 8.

$$\chi_R(x) = \sum_{i=0}^{7} x_i \bar{\to} x_{i+1}.$$

L'utilisation brute de la formule de Rutowitz «$\chi_R(x) = 1$» produit des *courbes imparfaites* [113]. Cependant le fait qu'un nombre important d'algorithmes de squelettisation parmi les précurseurs soit basé sur le crossing number témoigne de son importance «historique».

Ce n'est que quelques années plus tard qu'Hilditch [46] introduit la formule calculant la 8-simplicité, que Yokoi [122] généralise par les *nombres de connexité*. Ceux-ci sont donnés dans la définition 22, en utilisant les mêmes notation que pour le «crossing number».

Définition 22 Nombres de connexité

$$\mathcal{N}c_8(x) = \sum_{i=0}^{3} (x_{2i+1} \vee x_{2i+2}) \bar{\to} x_{2i} \ . \qquad \mathcal{N}c_4(x) = \sum_{i=0}^{3} x_{2i} \bar{\to} (x_{2i+1} \wedge x_{2i+2}) \ .$$

Le nombre de K-connexité $\mathcal{N}c_K(x)$ est exactement égal au nombre de K-cc de $V_8(x) \setminus \{x\}$, sauf pour les points K-intérieurs. Par conséquent les nombres de connexité fournissent un moyen simple de calculer la simplicité, puisque *un point x est K-simple si et seulement si* $\mathcal{N}c_K(x) = 1$.

La 8-simplicité peut ainsi être calculée en utilisant la procédure du tableau 3.4. Elle calcule explicitement la somme correspondant à la définition 22. La complexité en temps est de 15 OE; le calcul utilise 4 registres binaires.

Bien entendu, la 4-simplicité se calcule de manière similaire. La question qui nous préoccupe maintenant est de savoir si on ne peut pas calculer la simplicité de façon plus efficace en l'exprimant différement.

Examinons pour cela la figure 3.7, qui représente les 52 configurations de 8-voisinage différentes, à une rotation de $\pi/2$ près. On a représenté sur cette figure l'ensemble des configurations correspondant aux points 8-simples et aux points 4-simples (c'est le point central qui est concerné). On peut voir aussi le lien existant entre les nombres de connexité de Yokoi et le crossing number de Rutowitz : Les points vérifiant l'égalité $\chi_R(x) = 1$ sont les points qui sont à la fois 4-simples et 8-simples, en plus des deux configurations (κ) et (ξ), que nous allons retrouver un peu plus loin.

La figure 3.7, en fournissant une visualisation des configurations de points simples, permet de mettre en évidence des caractéristiques communes aux configurations de points simples, sous la forme de sous-configurations identiques, dans un but de minimisation logique.

(1)	$a = x \vee x_E$	
(2)	$d = a_N \overset{-}{\rightarrow} x_E$	
(3)	$a = a_{SW} \overset{-}{\rightarrow} x_W$	
(4)	$c = d \wedge a$	
(5)	$d = d \vee a$	
(6)	$a = x \vee x_N$	
(7)	$b = a_W \overset{-}{\rightarrow} x_N$	
(8)	$c = c \vee (d \wedge b)$	
(9)	$d = d \vee b$	
(10)	$b = a_{SE} \overset{-}{\rightarrow} x_S$	
(11)	$c = c \vee (d \wedge b)$	$// \; c = 1 \Longleftrightarrow \mathcal{N}c_8 > 1$
(12)	$d = d \vee b$	$// \; d = 1 \Longleftrightarrow \mathcal{N}c_8 \geq 1$
(13)	$a = d \overset{-}{\rightarrow} c$	$// \; a = 1 \Longleftrightarrow \mathcal{N}c_8 = 1$

TAB. 3.4 – *La procédure d'Hilditch.*

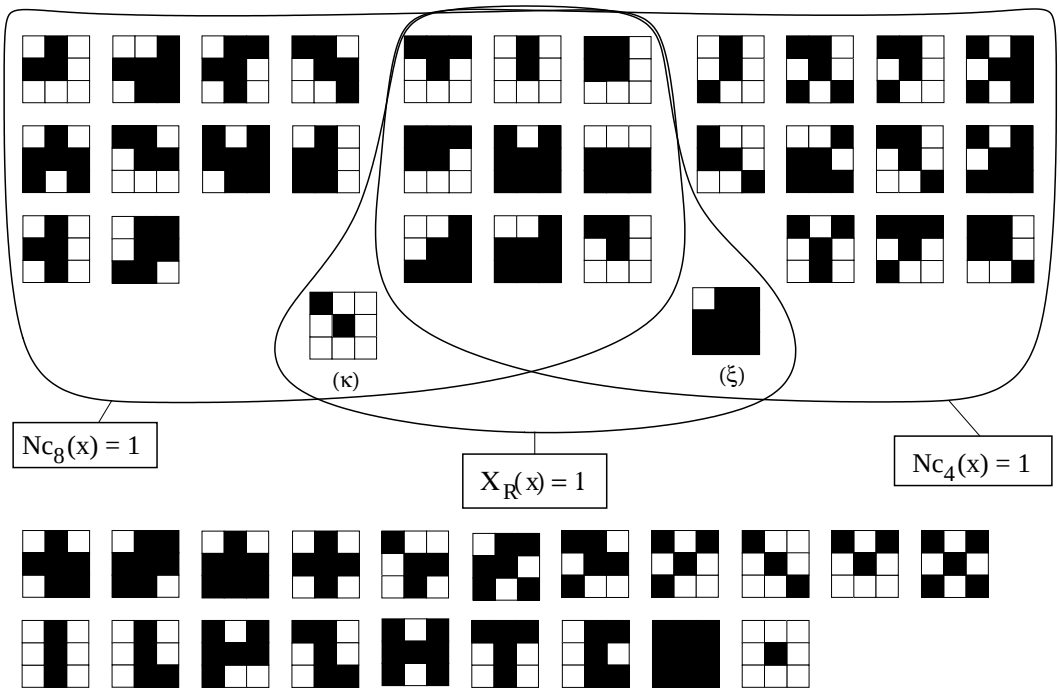


FIG. 3.7 – *Classification des configurations de voisinages, en fonction de la valeur du nombre de connexité et du crossing number.*

Cette représentation par configurations (*pattern matching* en anglais), appelle l'utilisation des tranformées en tout-ou-rien (TTR) de la définition 3, et des notations associées. Sur la figure 3.8, une autre classification est proposée, où chaque classe correspond à l'inclusion d'un motif simple : les points «de type» β (resp. β', γ), sont les points dont le 8-voisinage contient le motif β (resp. β', γ) de la figure 3.9. Les points correspondant aux patterns ι et ζ sont respectivement les points 8-intérieurs et les points 8-isolés. Les points de type σ sont tous les autres. Le lien avec la simplicité est le suivant : Aucun des points de type β (resp. β') n'est 8-simple (resp. 4-simple), sauf pour le pattern (κ) (resp. ξ). Les points de type γ ne sont ni 4-simples ni 8-simples, et les points de type σ sont à la fois 4-simples et 8-simples. De plus, les seuls points de type β (resp. β') qui ne soient pas 4-simples (resp. 8-simples) sont les points 4-isolés (resp. 4-intérieurs).

Finalement, la simplicité peut être exprimée dans le formalisme des TTR sous la forme réduite représentée figure 3.9, de la façon suivante : Si un point $x \in X$ n'est ni un point 8-simple (resp. 4-simple) ni un point 4-intérieur (resp. 4-isolé), alors $x \in X \circledast \beta$ ou $x \in X \circledast \gamma$ (resp. $x \in X \circledast \beta'$ ou $x \in X \circledast \gamma$). Notons qu'il faut prendre en compte toutes les rotations de $\pi/2$ de chaque configuration.

La réciproque est vraie, sauf si $V_8(x) \cap X$ contient deux points ou moins. Notons au passage que cette propriété est celle qu'a utilisé Arcelli pour le calcul de son squelette dans [1].

3.3.3 Ensembles simples et transformations homotopiques parallèles

Disposer d'une expression booléenne de la simplicité pour les points ne nous suffit pas pour caractériser les transformations homotopiques sur notre architecture parallèle. En effet, *une réunion de points simples ne forme pas un ensemble simple en général.* Il suffit de considérer la figure 3.5, où les points x et y sont tous les deux 8-simples. Cependant, la destruction simultanée de x et de y sépare l'objet en deux 8-cc.

Les techniques pour enlever en parallèle des points formant des ensembles simples sont cependant bien connues; en fait, on peut classer les transformations homotopiques parallèles en trois catégories :

- algorithmes en plusieurs passes *directionnelles.*
- algorithmes en plusieurs passes *semi-parallèles.*
- algorithmes *complètement parallèles* (en une seule passe).

Les premiers sont basés sur un résultat assez ancien de Stefanelli [110] qui montre que le retrait de points simple en parallèle ne modifie pas la topologie

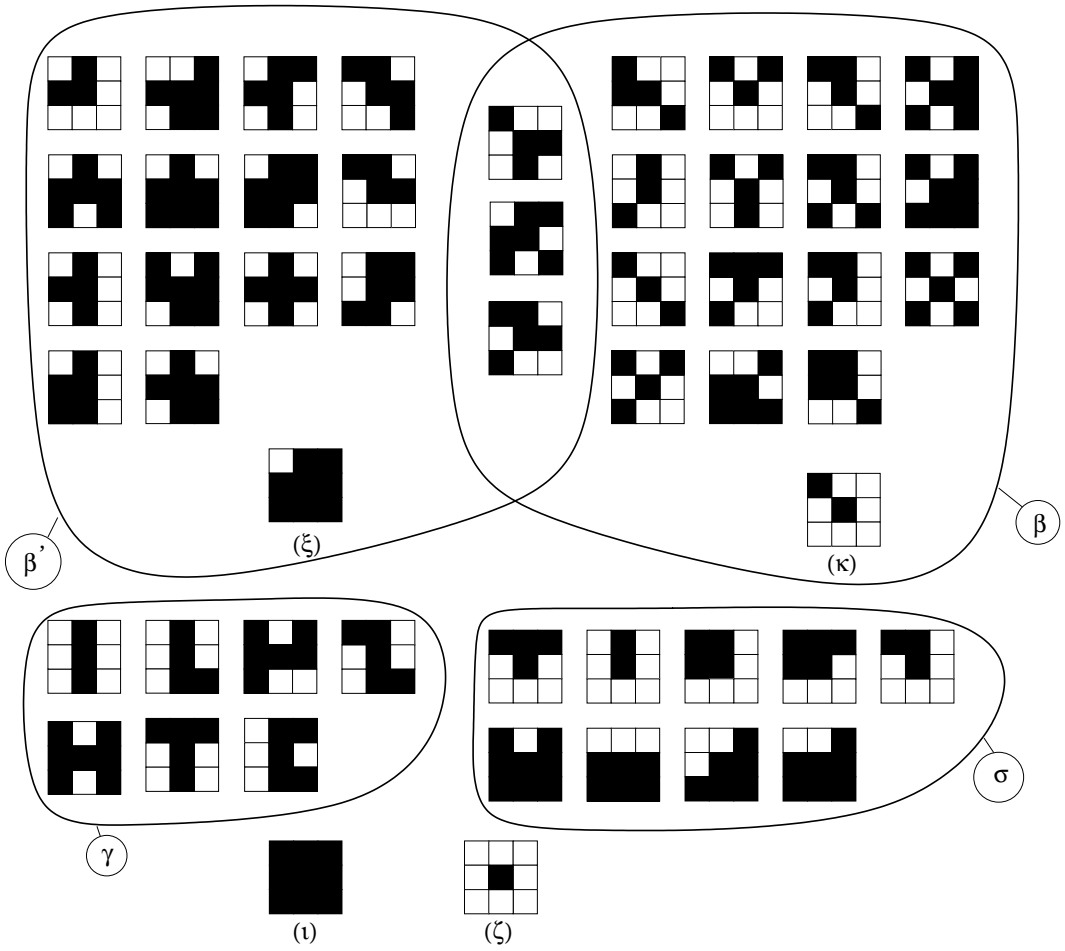


FIG. 3.8 – *Classification des configurations de voisinages en fonction de sous-configurations communes.*

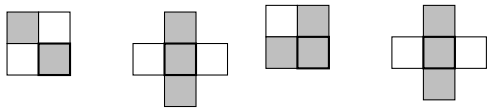


FIG. 3.9 – *Les configurations de déconnexion (1) en 8-connexité (2) en 4-connexité (à une rotation de $\pi/2$ près).*

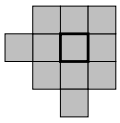


FIG. 3.10 – *La forme du plus petit support pour le calcul entièrement parallèle de la simplicité.*

si l'on se limite aux points de la frontière Nord (resp. Est, Sud, Ouest), i.e. les points dont le voisin N (resp. E, S, W) appartient au complémentaire. Le principe des algorithmes directionnels est donc de retirer les points simples de la frontière en alternant les directions.

Les seconds consistent à dégénérer le parallélisme en découpant régulièrement l'image en blocs de 2 ou 4 [84], et de sérialiser l'algorithme à raison d'un point traité par bloc et par passe. Ce type de techniques peut être appliqué sur la rétine programmable grâce à la multigranularité, nous les avons donc expérimentées. Il semble qu'elles n'apportent pas de gain en efficacité par rapport aux sous-itérations directionnelles, et elles produisent de plus des comportements en zigzag, comme il est souligné dans [55].

Dans le cas des algorithmes complètement parallèles, le fait d'éviter les déconnexions implique l'utilisation d'un voisinage plus important que le seul 8-voisinage. Hall [44] a montré que la taille minimum du voisinage à examiner par un algorithme sans sous-itération était de 11, sa forme étant, à une rotation de $\pi/2$ près, représentée par la figure 3.10.

Ce sont les travaux de Ronse [97], [98] qui ont permis de développer des algorithmes parallèles préservant la topologie. La première étape consiste à caractériser les ensembles simples. Dans [97], Ronse montre qu'un ensemble A est simple dans X si et seulement si A peut être ordonné par $A = \{a_1, \ldots, a_n\}$ de telle sorte que pour tout i dans $\{1, \ldots, n\}$, le point a_i est simple dans $X \setminus \{a_1, \ldots, a_{i-1}\}$.

Il n'est pas possible de déduire de cette propriété une caractérisation locale du point formant un ensemble simple. En effet, si une réunion de points simples ne forment pas un ensemble simple en général, réciproquement un ensemble simple peut contenir des points non simples. Il faut donc des conditions plus restrictives pour obtenir une caractérisation. Dans [98], Ronse donne des conditions suffisantes qui permettent la conception de transformations homotopiques parallèles. Nous présentons à présent le théorème que nous utiliserons plus loin pour démontrer le caractère homotopique des transformations proposées.

Théorème 6 *Ronse 88*
Soit F un opérateur de $\mathcal{P}(\mathbb{Z}^2)$ dans lui-même, tel que pour tout $X \subset \mathbb{Z}^2$,

$F(X) \subset X$. Alors F est K-homotopique si pour tout $X \subset \mathbb{Z}^2$, les 3 conditions suivantes sont vérifiées :

- *si $x \in X \setminus F(X)$, alors x est K-simple.*
- *si x et y sont $\tilde{K}$-voisins, et que $\{x, y\} \in X \setminus F(X)$, alors $\{x, y\}$ est K-simple.*
- *si K=8, alors aucune 8-cc contenue dans un carré 2×2 ne peut appartenir à $X \setminus F(X)$.*

Le caractère très restrictif de ces conditions est lié à une notion d'indépendance mutuelle de destruction pour certains groupes de points, et par suite, d'indépendance de l'ordre dans lequel les points seraient détruits s'ils l'étaient séquentiellement. Cela signifie que les notions de *parallélisme* et de *localité*, qui constituent la base de notre environnement algorithmique, sont incompatibles avec l'existence d'*un ordre donné* de destruction. Cette idée a été concrétisée par Bertrand [13] avec les définitions de *sous-homotopie forte* et de *points P-simples* : $A \subset B \subset \mathbb{Z}^2$, A est fortement sous-homotope à B si et seulement si pour tout C tel que $A \subset C \subset B$, C est sous-homotope à B. Les points de $B \setminus A$ sont dits P-simples dans B. Toute réunion de points P-simples forme un ensemble simple.

3.4 Noyau homotopique et SKIZ

Dans cette section nous traitons le problème suivant, comment retirer itérativement des ensembles simples d'une image X, jusqu'à rendre X *irréductible* (c'est-à-dire ne contenant plus de points simples), le plus vite possible? Lorsqu'un opérateur homotopique H converge pour toute image X vers une image irréductible $Y = H^{\infty}(X)$, on dit que Y est le *noyau homotopique* de X. Bien sûr, ce noyau n'a pas de définition formelle indépendante de l'opérateur H, cependant, sous certaines conditions sur H, il possède des propriétés métriques intéressantes.

La littérature traitant du calcul du noyau homotopique est très limitée par comparaison aux centaines d'articles traitant de squelettisation. Il s'agit cependant d'un opérateur très utile pour la segmentation d'images, et nous le retrouverons dans les chapitres suivants.

On peut obtenir le noyau homotopique en utilisant les nombres de connexité de la définition 22. Par exemple l'utilisation de la procédure d'Hilditch présentée tableau 3.4, en utilisant 4 sous-itérations, mène à une forme 8-irréductible. On peut en voir un exemple sur la figure 3.12. L'image (0) est l'originale, le noyau homotopique 8-connexe obtenue par la procédure d'Hilditch est l'image (2). On voit que la forme obtenue est 8-homotope à la forme originale (cinq 8-cc et quatre 8-trous). De la même façon en utilisant

Erosions	Configurations inhibitrices
Est	
Ouest	
Nord	
Sud	

TAB. 3.5 – *Les configurations intervenant dans le calcul du noyau homotopique en 4-connexité.*

le nombre de 4-connexité, on obtient l'image (3), 4-irréductible et 4-homotope à la forme originale (cinq 4-cc et deux 4-trous).

Le calcul de ces procédures utilise 15 OE par itérations. Nous allons voir qu'en utilisant les configurations introduites dans la section précédente, nous pouvons réduire sensiblement cette complexité.

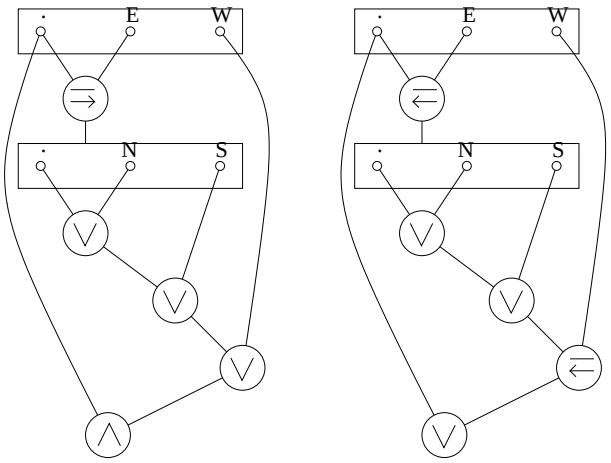
Nous proposons dans ce qui suit un algorithme de noyau homotopique en 4-connexité. Il est représenté sur le tableau 3.5 sous la forme de configurations d'érosions conditionnelles. C'est un algorithme qui utilise 4 sous-itérations directionnelles. Le principe est d'effectuer tour à tour les quatre érosions dans chaque direction cardinale (représenté par les quatre lignes de la colonne de gauche du tableau 3.5). Mais en chaque point, les érosions ne sont effectuées que si le point ne correspond à aucune des configurations «à éviter», représentées sur la colonne de droite.

L'opérateur de rétine correspondant au calcul d'une itération de cette algorithme est représenté figure 3.11(1). Il correspond à l'itération qui enlève les points de la frontière EST. Les autres OR se déduisent par simple rotation. Le calcul utilise 3 registres binaires, et chaque itération est effectuée en 5 OE.

Un exemple de noyau homotopique calculé par cette procédure est visible sur la figure 3.12(1). On voit que le résultat est conforme à celui obtenu avec le nombre de 4-connexité (image (3)), sans être rigoureusement identique toutefois.

Pour prouver la correction de cet algorithme, puisqu'il procède par sous-itérations directionnelles, il suffit, grâce au théorème de Stefanelli de montrer qu'il ne retire que des points 4-simples.

Considérons par exemple l'érosion EST, correspondant à la première ligne du tableau 3.5. Un point $x \in X$ détruit par une érosion EST est un point qui



(1) (2)

FIG. 3.11 – *L'opérateur de rétine correspondant à l'itération EST du noyau homotopique (1) et du noyau homotopique dual (2).*

a son voisin EST dans X^c. Si x n'est pas 4-simple, d'après les configurations de la figure 3.9, alors $x \in X \circledast \beta'_1$ ou $x \in X \circledast \beta'_2$ ou $x \in X \circledast \gamma_1$, où β'_1, β'_2, et γ_1 sont les trois configurations représentées sur la figure 3.13. Or si tel est le cas, le point n'est pas retiré, car il correspond aussi à une configuration à éviter de la première ligne du tableau 3.5 (respectivement, la deuxième, la troisième et la première). La transformation proposée est donc 4-homotopique.

Le principal intérêt du noyau homotopique, de notre point de vue, est de permettre le calcul d'une approximation du SKIZ (SKeleton by Influence Zone) [56]. Le SKIZ est une extension du diagramme de Voronoï aux composantes connexes. Si l'on note $CC_K(X)$ l'ensemble des K-cc de X, on peut définir le SKIZ de X comme suit :

Définition 23 *Soit $Y \in CC_K(X)$. La* zone d'influence *de Y est l'ensemble :*

$$IZ_K(Y) = \{z \in \mathbb{Z}^2; \forall Y' \in CC_K(X), \delta_K(z, Y) < \delta_K(z, Y')\}.$$

et le SKIZ (en K-connexité) de X est défini par :

$$SKIZ_K(X) = (\bigcup_{Y \in CC_K(X)} IZ_K(Y))^c.$$

Le SKIZ est donc constitué des frontières des zones d'influence des K-cc de X, la zone d'influence d'une K-cc Y est l'ensemble des points plus proches de Y (au sens de la K-distance) que de n'importe quelle autre K-cc.

En appliquant le noyau homotopique de manière duale, ce qui revient à appliquer l'OR représenté sur la figure 3.11(2), on obtient une approxi-

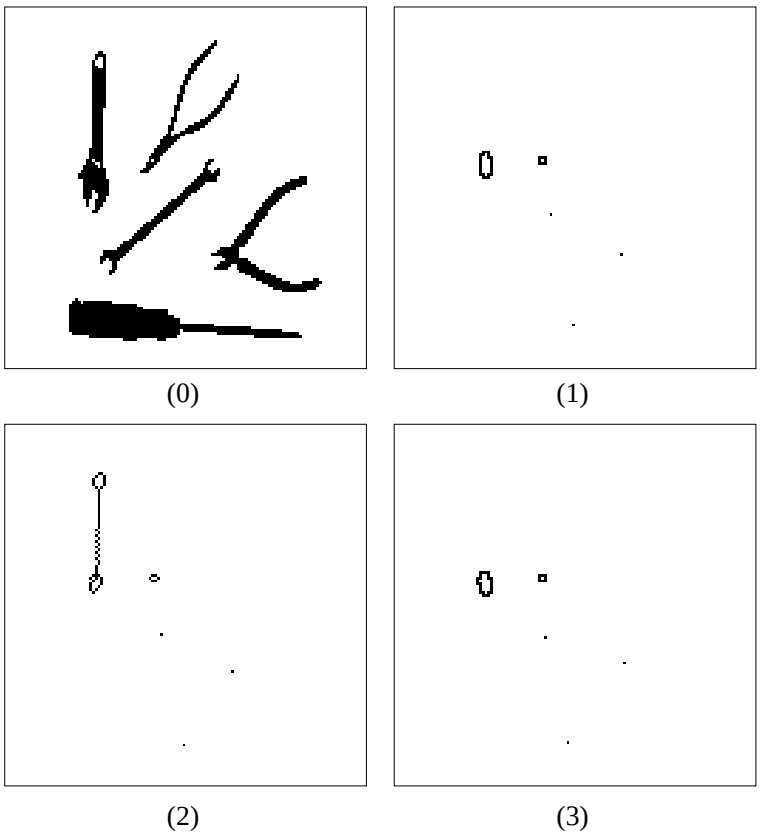


FIG. 3.12 – *Le noyau homotopique : résultat (1). A titre de comparaison, on peut voir en dessous les noyaux homotopiques obtenus en utilisant les nombres de 8-connexité (2) et de 4-connexité (3)*

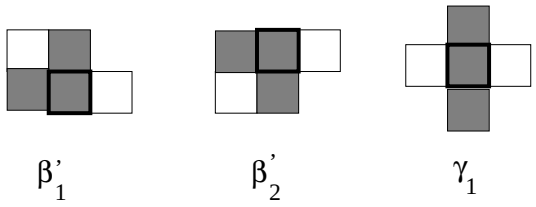


FIG. 3.13 – *Preuve de l'homotopie de l'algorithme de noyau homotopique. Un point qui a son voisin EST dans le complémentaire et qui n'est pas simple correspond à l'une de ces trois configurations*

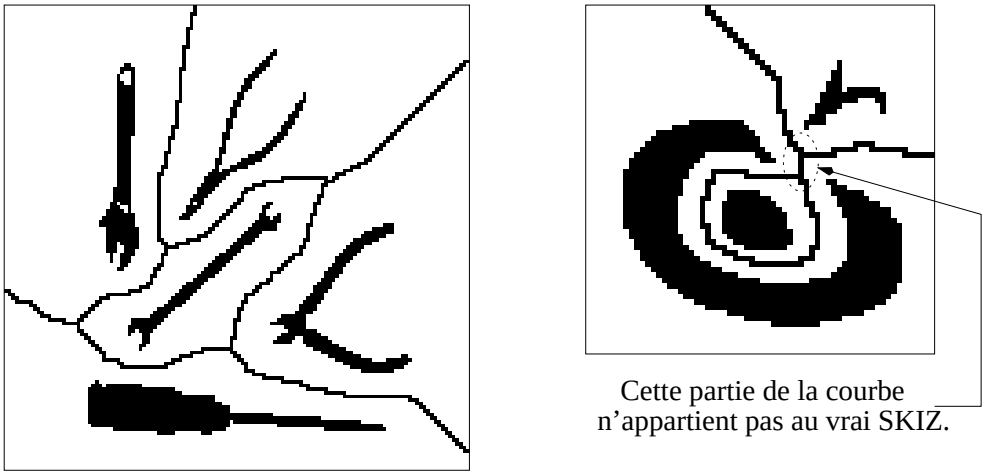


FIG. 3.14 – *Les résultats du SKIZ et ses limitations.*

mation du SKIZ en 8-connexité (en calculant le noyau homotopique sur le complémentaire, on change de topologie). On peut voir des résultats sur la figure 3.14, superposés aux images originales. Notons que les 8-trous produisent des points isolés qu'on élimine par un post-traitement car il n'appartiennent pas au SKIZ. Dans le vrai SKIZ, les dilatations seraient parfaitement isotropes, ce qui n'est pas le cas ici, puisqu'on alterne les directions de dilatation, et donc le résultat dépend (à un pixel près) de l'ordre choisi. En particulier, le vrai SKIZ peut présenter des courbes de deux pixels de large, ce qui n'est pas le cas ici. Enfin, une limitation connue du calcul du SKIZ par cette méthode est représenté sur la figure 3.14(2) : Lorsqu'une K-cc est entièrement contenue dans une concavité d'une autre K-cc, l'algorithme génère une courbe qui n'appartient pas au vrai SKIZ.

3.5 Le squelette MB

3.5.1 Introduction

Dans le cas du squelette, comme nous l'avons vu en introduction, la contrainte de médialité implique qu'en plus de l'homotopie, on impose la présence dans le squelette d'un certain nombre de points significatifs du point de vue de la géométrie et de la localisation de l'objet.

Une idée naturelle est donc de calculer *indépendamment* un ensemble de points que l'on définit comme étant l'axe médian, et de calculer ensuite

le noyau homotopique en imposant la présence de cet ensemble à chaque itération (par un simple OU). C'est le principe du *squelette par points d'ancrage* tel que celui proposé par Vincent [117]. Ce type de squelette offre l'avantage d'être à la fois défini proprement et polyvalent, on peut en effet faire varier la forme du squelette selon les besoins. Par exemple, dans [117], Vincent définit le squelette «classique» par le squelette ancré sur l'ensemble des centres de boules maximales, et le squelette «minimal» par le squelette ancré sur l'ensemble des érodés ultimes (voir chapitre suivant).

On peut appliquer ce type d'algorithme grâce aux procédures que nous avons décrites dans les sections précédentes. On calcule alors l'axe médian par la procédure du tableau 3.3, puis en conservant l'image de l'axe médian, on calcule le noyau homotopique, en effectuant la disjonction des deux images à chaque itération.

L'algorithme que nous allons proposer par la suite procède différemment. Puisque nous sommes animés par une volonté de concision logique, la solution du squelette par points d'ancrage ne nous satisfait pas complètement car elle est assez coûteuse en nombre d'OE. Dans la suite, nous exposons et justifions nos choix algorithmiques, puis nous proposons notre procédure.

3.5.2 Choix algorithmiques

La littérature offre, nous l'avons vu une profusion d'algorithmes de squelettisation, dont beaucoup s'implantent assez facilement sur la rétine programmable. La première question qui se pose alors est : pourquoi proposer de nouveaux algorithmes ? La réponse est que parmi tous les algorithmes qui ont été porté à notre connaissance, pratiquement aucun ne se pose le problème de la concision logique, et très peu même, fournissent une expression booléenne de la procédure. Nous expliquerons pourquoi dans la conclusion de ce chapitre, toujours est-il que, malgré nos efforts de minimisation dans l'implantation sur la rétine, ces algorithmes s'avéraient plutôt complexes en temps de calcul. Pour l'algorithme que nous présentons, à l'inverse, le principe a été de traquer la redondance logique dans les différentes contraintes qui régissent le calcul du squelette.

Indépendamment du problème de minimisation logique, le premier choix algorithmique est justement basé sur le choix de ces contraintes. Nous avons vu en introduction que les caractéristiques attendues d'un algorithme de squelettisation était en général incompatibles, ce qui donne un caractère «mal posé» au squelette et explique, dans une certaine mesure, la vaste littérature à son sujet. Nous avons décidé de nous limiter aux trois première contraintes du tableau 3.1, qui sont compatibles : *Homotopie, Médialité, Isotropie.*

Un certain nombre de choix algorithmiques restent à faire, principalement

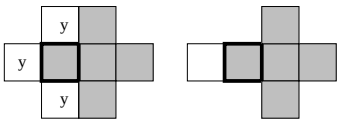


FIG. 3.15 – *La configuration correspondant à un point qui n'est pas un maximum local (1), et ce que l'on calcule dans l'algorithme MB (2).*

ceux (arbitraires?) liés à la maille carrée. Tout d'abord, la contrainte d'isotropie nous impose un algorithme complètement parallèle. En effet, toute séparation de l'algorithme en sous-itérations implique un choix dans l'ordre des «lieux de calcul», qu'ils soient directionnels ou semi-parallèles, ce qui entraîne une anisotropie. Il faut faire aussi le choix de la topologie. Squelette 8-connexe ou 4-connexe? Dans [55], Lam explique que pour des raisons d'épaisseur minimale, les squelettes 8-connexes sont préférés, et c'est le cas de la très grande majorité des algorithmes. Nous considèrerons donc toujours la 8-connexité pour l'image, et la 4-connexité pour le complémentaire (Notons que nous avons proposé le noyau homotopique en 4-connexité parce que nous étions principalement intéressés par son dual, le pseudo-SKIZ, qui est donc défini en 8-connexité). Reste la nature de l'axe médian. D'après le théorème 5, un point 8-simple appartient nécessairement au 4-contour (et réciproquement), donc un algorithme complètement parallèle n'enlèvera, à chaque itération, que des points du 4-contour. Il ne peut donc être basé que sur la 4-distance. L'axe médian que nous choisissons donc pour le squelette que nous présentons maintenant est par conséquent l'ensemble des maxima locaux de la 4-distance.

3.5.3 L'algorithme

Le calcul de la procédure du tableau 3.3 pour la 4-distance revient à itérer jusqu'à convergence la transformation consistant à retirer de l'image tous les points correspondant (à une rotation de $\pi/2$ près) à la configuration représentée sur la figure 3.15(1), où au moins l'un des points marqué y est un point blanc. Ce que l'on calcule réellement est plus restrictif, puisqu'on impose que le point qu'on enlève se trouve «entre» un point blanc et un point intérieur, c'est à dire les points correspondant (à une rotation de $\pi/2$ près) à la configuration représentée sur la figure 3.15(2).

Ces points sont des *points parfaits* selon la dénomination due à Eckhardt [32]. Celui-ci a remarqué que lorsqu'on n'enlevait que des points parfaits simples, on pouvait le faire de manière complètement parallèle (sans sous-

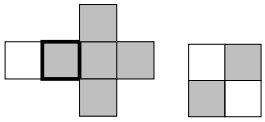


FIG. 3.16 – *Les configurations intervenant dans le calcul du squelette MB. Le principe de l'algorithme est de retirer simultanément tous les points correspondant à la configuration (α), tant qu'ils n'appartiennent pas à la configuration (β), à une rotation de $\pi/2$ près.*

itération). L'algorithme que nous présentons ici, que nous avons appelé MB au moment où nous l'avons publié [65] est en fait strictement équivalent à l'algorithme d'Eckhardt proposé dans [32], mais dont nous ignorions l'existence à cette époque.

L'intérêt de notre contribution, pour cet algorithme, réside dans la mise en évidence de la redondance logique entre le calcul des points parfaits et des points 8-simples. Si nous examinons la figure 3.9(1), il est facile de voir qu'un point $x \in X$ qui est parfait peut être tel que $x \in X \circledast \beta$, mais jamais tel que $x \in X \circledast \gamma$. Pour la 8-simplicité des points parfaits, il suffit donc de tester la configuration (β).

L'algorithme MB est donc entièrement défini par les deux configurations qui apparaissent dans la figure 3.16.

On retire ainsi itérativement, tous les points $x \in X$ tels que $x \in X \circledast \alpha$ ET $x \notin X \circledast \beta$, et ceci pour toutes les rotations de α et β de $\pi/2$. Le calcul complet est entièrement explicité dans la procédure du tableau 3.6. On utilise la notation du parallélisme de données : les lettres minuscules correspondent aux variables booléennes (les pixels), et les lettres majuscules aux images.

La représentation graphique de cette opérateur de rétine est l'OR (MB) représenté sur la figure 3.17. Cette OR utilise un certain nombre d'OR de base qui sont détaillés sur la même figure. On voit immédiatement que le calcul du squelette MB est effectué en 18 OE. On peut calculer aussi la complexité en espace de cet OR, qui est de 5 plans-mémoire.

Nous prouvons à présent que l'algorithme MB préserve la 8-topologie. Nous utilisons pour cela le théorème 6. Nous venons de montrer qu'on ne retirait que des points simples, donc la première condition est vérifiée. La troisième condition est évidemment vérifiée, puisqu'un point ne peut être détruit que s'il est 4-voisin d'un point 4-intérieur. Pour prouver la condition (2), il suffit de remarquer que deux points 4-voisins qui sont simultanément détruits correspondent nécessairement à la configuration de la figure 3.18. On voit bien qu'aucun des deux points n'appartient à la configuration (α) de

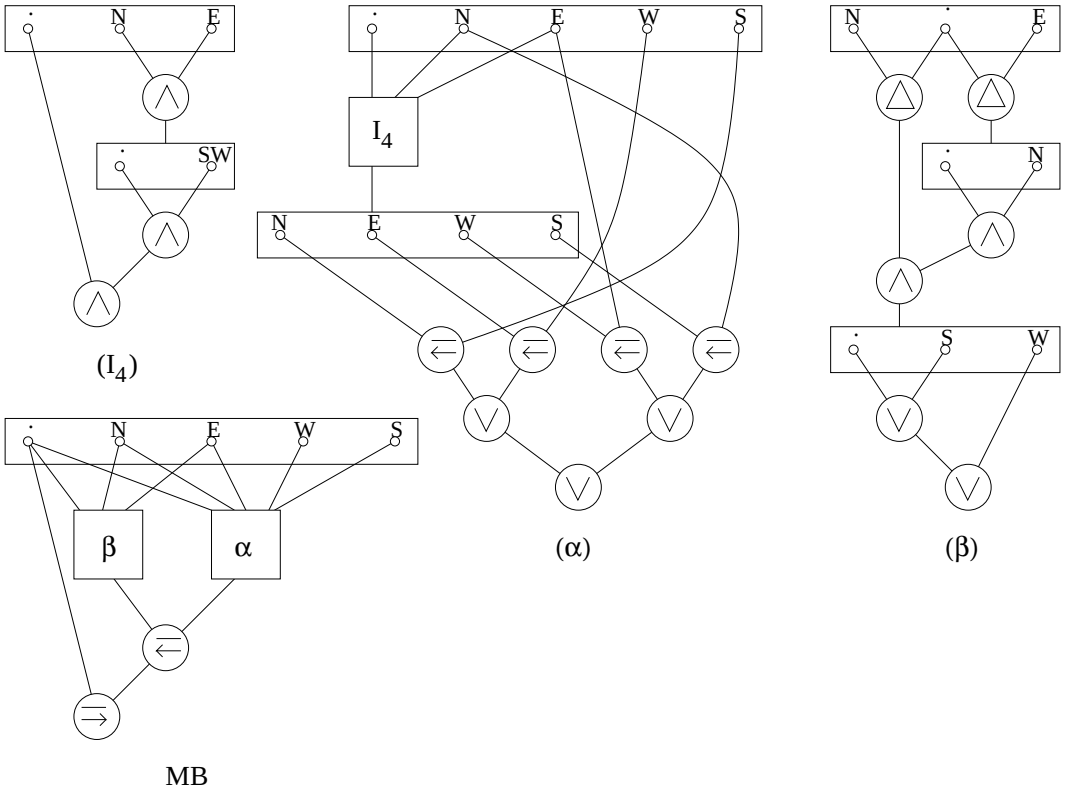


FIG. 3.17 – *Les opérateurs de rétines intervenant dans le calcul du squelette MB. Calcul des points 4-intérieurs (I_4), des points parfaits (α), des points appartenant à la configuration β (β), et finalement calcul d'une itération du squelette (MB).*

Répéter la séquence suivante pour tous les pixels en parallèle, jusqu'à stabilité :

(1) $a = x_N \wedge x_E$
(2) $a = a \wedge a_{SW} \wedge x$
//A = points 4-intérieurs de X
(3) $b = (a_S \bar{\rightarrow} x_N) \vee (a_N \bar{\rightarrow} x_S) \vee (a_E \bar{\rightarrow} x_W) \vee (a_W \bar{\rightarrow} x_E)$
//B = ensembles des points parfaits
(4) $c = x \triangle x_E$
(5) $c = c \wedge c_N \wedge (x \triangle x_N)$
(6) $c = c \vee c_W$
(7) $c = c \vee c_S$
//C = ensemble des pixels appartenant à la configuration (β)
(8) $x = x \bar{\rightarrow} (b \bar{\rightarrow} c)$
//On enlève les pixels qui appartiennent à B mais pas à C

TAB. 3.6 – *La procédure MB.*

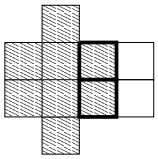


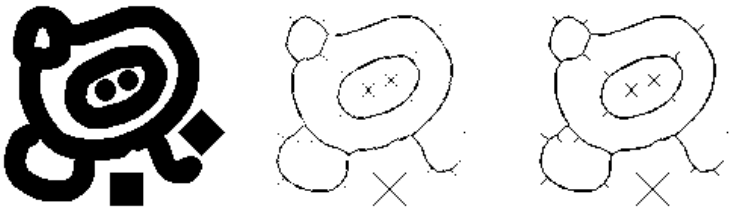
FIG. 3.18 – *Le cas de deux pixels 4-voisins.*

l'autre. Cela signifie que les points auraient été détruits tous les deux même s'ils avaient été traités séquentiellement, donc, *a fortiori* la paire est simple. Donc MB préserve la 8-topologie.

3.5.4 Conclusion

Comme nous le verrons dans la section 3.7, cet algorithme a une complexité en temps qui est presque deux fois moindre par rapport aux plus rapides des algorithmes comparables. La figure 3.19 montre un exemple de résultat de l'algorithme MB. On voit que le squelette est conforme à nos attentes, en particulier qu'il est basé sur les maxima locaux de la 4-distance. Dans la section 3.7, nous préciserons cette propriété.

Le fait que le squelette MB soit basé sur l'axe médian de la 4-distance entraîne une grande différence dans le comportement de l'algorithme vis-à-vis des directions de type $k\pi/2$ d'une part, et celle de type $(2k+1)\pi/4$ d'autre



(1) (2) (3)

FIG. 3.19 – *Résultat de l'algorithme MB (1) Forme originale (2) Axe médian de la 4-distance (3) Squelette MB.*

part. Qu'on en juge sur la figure 3.19 par la forme du squelette des disques, et par la différence entre les squelettes des deux carrés, qui sont identiques à une rotation de $\pi/4$ près.

C'est dans le but d'obtenir un comportement plus homogène par rapport aux deux types de distance de la maille carrée que nous avons proposé un deuxième algorithme [9], que nous appelons MB2, présenté dans la section suivante.

3.6 Le squelette MB2

3.6.1 Introduction

Le principe de l'algorithme MB2 est de considérer comme base du squelette un nouveau type d'axe médian, que nous définissons comme *axe médian mixte* de la maille carrée.

Définition 24 *Soit $X \subset \mathbb{Z}^2$. Soit $K = 4$ ou 8, $K' = 4$ ou 8, tels que $K \leq K'$. On note $\mathcal{S}_K^{K'}(X)$ l'ensemble défini comme suit :*

$$\mathcal{S}_K^{K'}(X) = \bigcup\{x \in X; \forall y \in \mathcal{B}_{K'}(x,1), \Phi_K(y) \leq \Phi_K(x)\}$$

L'ensemble $\mathcal{S}_K^{K'}(X)$ est donc constitué des maxima locaux de la K-distance, dans le K'-voisinage. Lorsque $K = K'$, on retrouve la notion d'axe médian de la K-distance introduite par la définition 18. Mais c'est sur l'axe médian mixte $S_4^8(X)$ que sera basé le squelette MB2.

L'axe médian $\mathcal{S}_K^{K'}(X)$ se calcule sur la rétine programmable par la procédure du tableau 3.3, en remplaçant «$dilate_K$» par «$dilate_{K'}$».

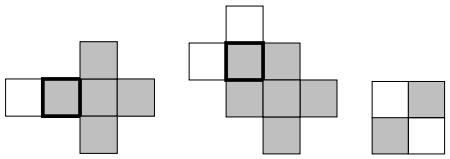


FIG. 3.20 – *Les configurations intervenant dans le calcul du squelette MB2. Le principe de l'algorithme est de retirer simultanément tous les points correspondant à la configuration ($\alpha 1$) OU à la configuration ($\alpha 2$), tant qu'ils n'appartiennent pas à la configuration (β), à une rotation de $\pi/2$ près.*

3.6.2 L'algorithme

Pour le squelette MB2, on étend la notion de points parfaits, en ajoutant une configuration de points à retirer. Le squelette MB2 est donc entièrement défini par les trois configurations de la figure 3.20.

On retire ainsi itérativement, tous les points $x \in X$ tels que ($x \in X \circledast \alpha_1$ OU $x \in X \circledast \alpha_2$) ET $x \notin X \circledast \beta$, chaque configuration étant comptée avec toutes ses rotations de $\pi/2$. Le calcul est entièrement explicité dans la procédure du tableau 3.7, en utilisant les mêmes conventions que pour le tableau 3.6.

La complexité en temps de cette procédure est de 28 OE par itération. L'algorithme reste très compact, et toujours complètement parallèle, comme nous le prouvons ci-dessous.

On prouve l'homotopie de MB2 grâce au théorème 6. La première condition impose que tous les points retirés soient simples. Or on ne retire que des points du 4-contour, parmi lesquels les points non 8-simples correspondent nécessairement à l'une des deux configurations β et γ de la figure 3.9(1). Un point $x \in X$ détruit par MB2 est tel que $x \in X \circledast \alpha_1$ ou $x \in X \circledast \alpha_2$. dans les deux cas, on a bien $x \notin X \circledast \gamma$. Comme par définition de MB2, $x \notin X \circledast \beta$, x est bien 8-simple. Comme tout point détruit par MB2 est nécessairement 8-voisin à un point 4-intérieur, la troisième condition est aussi respectée. Pour la condition (2), il suffit de voir que deux points $\{x, y\}$ détruits simultanément par MB2 correspondent obligatoirement à l'une des trois configurations représentées sur la figure 3.21.

En effet si x et y sont tous les deux dans $X \circledast \alpha_1$ (resp. $X \circledast \alpha_2$), alors la seule possiblité est celle de la figure 3.21(1) (resp. 3.21(2)). Dans ces deux cas, aucun des deux points n'appartient à la configuration de l'autre, on est dans la même situation que pour la preuve de MB, et donc la paire $\{x, y\}$ est simple. Si $x \in X \circledast \alpha_2$ et $y \in X \circledast \alpha_1$, alors la seule possibilité est celle de la

Répéter la séquence suivante pour tous les pixels en parallèle, jusqu'à stabilité :

(1) $a = x_N \wedge x_E$
(2) $a = a \wedge a_{SW} \wedge x$
//A = points 4-intérieurs de X
(3) $b_1 = (a_S \overset{-}{\rightarrow} x_N) \vee (a_N \overset{-}{\rightarrow} x_S) \vee (a_E \overset{-}{\rightarrow} x_W) \vee (a_W \overset{-}{\rightarrow} x_E)$
//B_1 = ensembles des points de type (α_1)
(4) $d = x \wedge x_{SE}$
(5) $b_2 = (a_{SW} \overset{-}{\rightarrow} d_N) \vee (a_{NE} \overset{-}{\rightarrow} d_W)$
(6) $d = x \wedge x_{SW}$
(7) $b_2 = b_2 \vee (a_{NW} \overset{-}{\rightarrow} d_E) \vee (a_{SE} \overset{-}{\rightarrow} d_N)$
//B_2 = ensembles des points de type (α_2)
(8) $b = b_1 \vee b_2$
(9) $c = x \triangle x_E$
(10) $c = c \wedge c_N \wedge (x \triangle x_N)$
(11) $c = c \vee c_W$
(12) $c = c \vee c_S$
//C = ensemble des pixels appartenant à la configuration (β)
(13) $x = x \overset{-}{\rightarrow} (b \overset{-}{\rightarrow} c)$
//On enlève les pixels qui appartiennent à B mais pas à C

TAB. 3.7 – *La procédure MB2.*

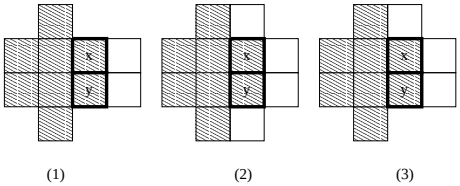


FIG. 3.21 – *Les trois cas possibles de deux pixels 4-voisins retirés simultanément par MB2.*

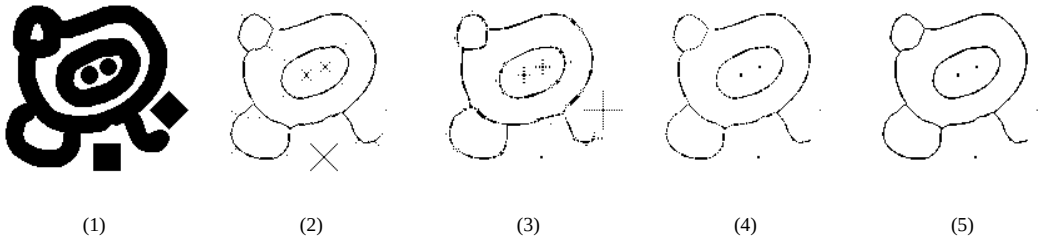


FIG. 3.22 – *Résultat de l'algorithme MB2 (1) Forme originale (2) Axe médian de la 4-distance (3) Axe médian de la 8-distance (4) Axe médian mixte S_4^8 (5) Squelette MB2.*

figure 3.21(3). Dans ce cas, x n'appartient pas à la configuration (α_1) de y, donc y reste «destructible» par MB2 dans $X \setminus \{x\}$, donc *a fortiori* simple, et finalement la paire $\{x, y\}$ est 8-simple. On a donc prouvé que MB2 préservait la 8-topologie de manière complètement parallèle.

3.6.3 Conclusion

La figure 3.22 donne un exemple de résultat du squelette MB2. Il apparaît clairement que ce squelette est basé sur l'axe médian mixte $S_4^8(X)$ de la définition 24. Cet axe médian, comme on peut le voir sur la figure 3.22, où l'on présente à coté les deux types d'axes médians «classiques» de la maille carrée, possède l'intéressante propriété de moins discriminer les directions diagonales des directions horizontales et verticales.

Cette propriété, comme nous allons le voir dans la section suivante, entraîne un comportement très intéressant de MB2 vis-à-vis des formes bruitées.

3.7 Résultats comparatifs et propriétés

Dans cette section, nous étudions en détail les propriétés et le comportements des algorithmes proposés relativement aux caractéristiques classiques des algorithmes de squelettisation. Cette étude est comparative, car nous présentons simultanément les propriétés correspondantes d'un certain nombre d'algorithmes de squelettisation du même type (c'est-à-dire parallèle et itératif). Tous les algorithmes concurrents dont nous présentons les squelettes ont été implémentés par nos soins sur le simulateur rétinien. Ils ont donc été expérimentés et leur coût de calcul en nombre d'OE a pu être calculé précisément.

La figure 3.23 représente les résultats des différents algorithmes sur la même image. Les noms utilisés pour référencer les algorithmes seront toujours ceux qui apparaissent sous le squelette correspondant. La référence bibliographique complète de chaque algorithme est donné dans la légende de la figure 3.23.

3.7.1 Homotopie, isotropie et parallélisme complet

Nous avons déjà prouvé que MB et MB2 préservaient la 8-topologie de manière complètement parallèle dans les sections qui leur sont consacrées. Le parallélisme complet, ainsi que le caractère entièrement symétrique du traitement garantissent l'*isotropie* telle qu'elle est définie dans le tableau 3.1.

Une conséquence de l'isotropie est la présence de courbes «épaisses» dans les squelettes MB et MB2, comme on peut le voir sur les résultats. La propriété d'épaisseur unité, imposée par certains algorithmes, nous semble moins fondamentale, dans la mesure où l'on peut obtenir le même résultat que ces algorithmes par un post-traitement anisotrope, consistant en un nombre constant d'itérations d'un algorithme tel que Jang et Chin, ou AFP3. De plus, certaines formes pathologiques connues comme celle qui figure en haut à droite de l'image de la figure 3.23 sont irréductibles quelle que soient leur épaisseur : on ne peut leur enlever aucun point sans modifier la 8-topologie.

3.7.2 Complexité

Nous avons donné plus haut la complexité en temps de calcul de nos algorithmes, il nous faut à présent la comparer à d'autres algorithmes récents de la littérature, que nous avons pour la plupart implantés, en particulier tous ceux qui sont présentés dans les résultats de la figure 3.23. Le tableau 3.8 présente les différents algorithmes et leur complexité.

Le type d'algorithme indique s'il s'agit d'un algorithme à sous itérations directionnelles (D), ou complètement parallèle (CP). La complexité, qui apparaît dans la dernière colonne, correspond au *nombre total* d'opérations élémentaires nécessaires au calcul du squelette. C'est donc le nombre d'OE d'une itération du squelette multiplié par le nombre d'itérations. Dans le cas des algorithmes CP, comme on enlève à chaque itération des points du 4-contour dans toutes les directions, le nombre d'itérations est supérieur ou égal à r, grandeur qui représente le rayon de la plus grande 4-boule contenue dans l'image. Pour les algorithmes MB et MB2 (ainsi que Latecki et Guo & Hall), ce nombre est exactement égal à r ($r+1$ pour Guo et Hall, à cause de l'épaisseur unité). En effet, pour ces algorithmes, on retire *tous* les points du 4-contour qui peuvent être retiré par la procédure, c'est-à-dire

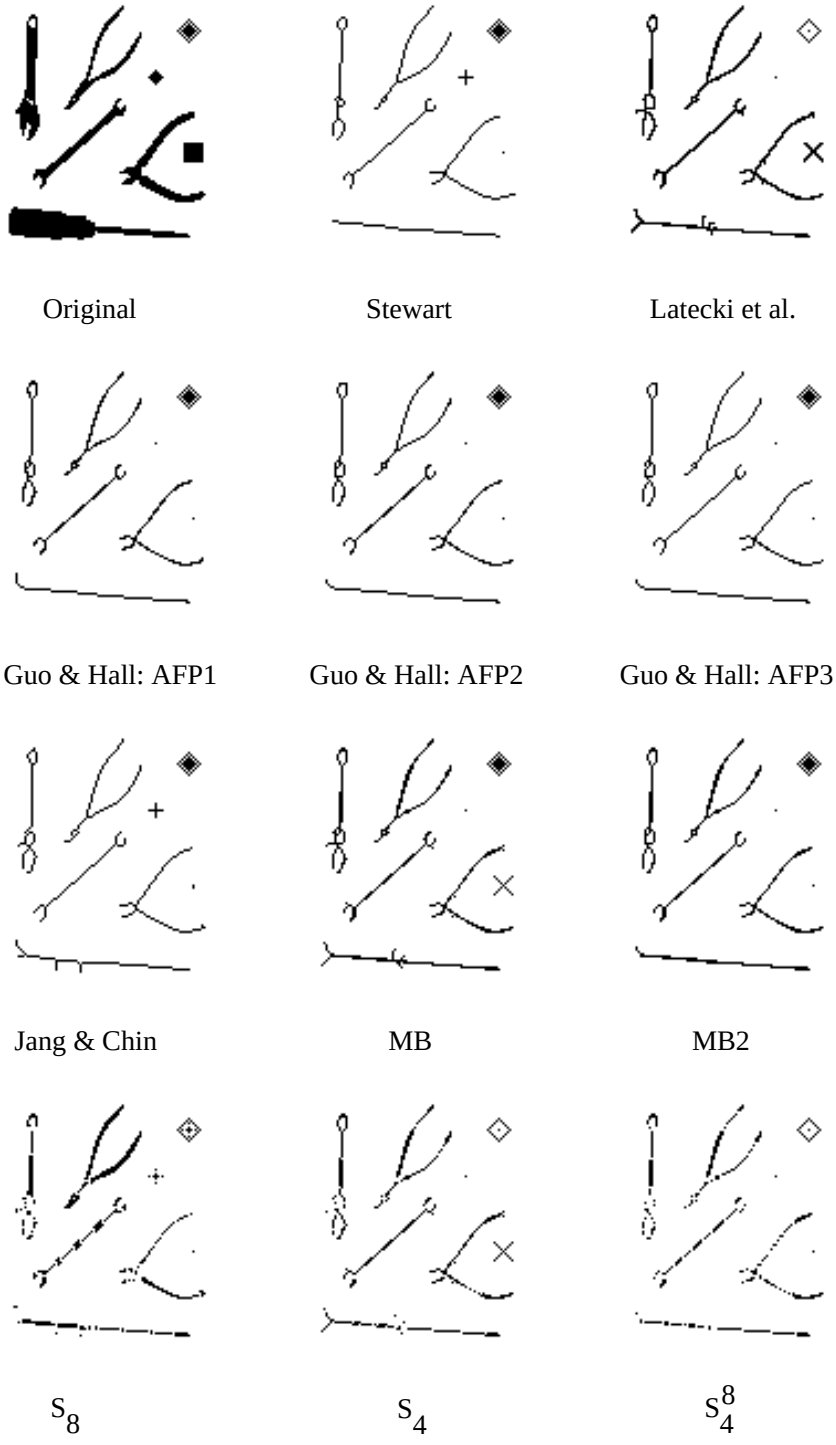


FIG. 3.23 – *Résultats comparatifs. A coté des deux algorithmes proposés sont affichés le résultats d'autres algorithmes itératifs : Stewart[111], Latecki [58], les trois algorithmes de Guo et Hall[43], et Jang et Chin[48]. Sur la dernière ligne sont affichés les différents axes médians correspondant à la définition 24.*

qu'un point du 4-contour qui n'est pas retiré à l'itération n ne le sera pas plus à l'itération $n+1$. Ce n'est pas le cas d'autres algorithmes, principalement celui de Stewart, qui effectue un nombre d'itérations plus grand que l'épaisseur de l'objet, dans une proportion qu'il est difficile d'évaluer. Dans ce cas, la complexité dépend aussi de la forme de l'image, et la valeur indiquée ne représente qu'un minimum.

Dans le cas des algorithmes directionnels, il faut faire toutes les sous-itérations une fois pour compléter une itération complète (c'est-à-dire qui retire des points dans les 4 directions). Le nombre total d'OE est donc 4 fois le nombre d'OE d'une sous-itération multiplié par le nombre d'itérations complètes. Chaque sous-itération correspond à une érosion conditionnelle par un élément structurant à deux éléments, donc une itération complète correspond à une érosion conditionnelle par la 8-boule élémentaire. Le nombre d'itérations complètes pour les algorithmes directionnels est donc égal à r', quantité qui représente le rayon de la plus grande 8-boule contenue dans l'image. Notons que $r' \leq r \leq 2r'$.

Algorithme	Type	Connexité	Taille du support	Complexité en temps
Jang & Chin[48]	D	8	7	$32 \times r'$
Cardoner & Thomas[27]	D	8	7	$40 \times r'$
Latecki & al.[58]	CP	4	13	$16 \times r$
Stewart[111]	CP	8	19	$64 \times r$
Wu & Tsai[119]	CP	8	11	$60 \times r$
Guo & Hall (AFP1)[43]	CP	8	11	$73 \times r$
Guo & Hall (AFP2)[43]	CP	8	11	$81 \times r$
Guo & Hall (AFP3)[43]	CP	8	11	$91 \times r$
MB	CP	8	13	$18 \times r$
MB2	CP	8	21	$28 \times r$

TAB. 3.8 – *Comparaison de la complexité en temps des algorithmes de squelettisations itératifs parallèles.*

Dans leur catégorie (squelettes 8-connexes), MB et MB2 sont les plus rapides. MB2 produit des résultats semblables à ceux de Guo et Hall (la différence entre leurs trois algorithmes ne concerne que l'épaisseur du squelette obtenu), en étant plus de deux fois plus rapide.

Le seul algorithme plus rapide que MB porté à notre connaissance est dû à Latecki [58], mais ce n'est pas le même type de squelette car son domaine de validité est restreint. Pour Latecki, en effet, une image telle que celle de la

figure 3.23 est une image *mal formée*. Le domaine de validité de l'algorithme est en fait l'ensemble des images qui ne contiennent pas la configuration (β) de MB. Il est facile de voir que, pour ces images bien formées (*well composed sets*), la 8-topologie et la 4-topologie coïncident. L'algorithme de squelettisation de Latecki consiste alors à retirer itérativement les points parfaits (les points de type (α) de MB), tant que cette destruction ne fait pas apparaître de configuration (β), les images devant bien entendu rester bien formées. Les configurations intervenant dans le calcul de son squelette sont donc exactement les mêmes, mais en raison d'une anisotropie nécessaire à l'obtention d'un squelette fin, la présence de la configuration (β) n'est testée que dans deux directions sur quatre, et on gagne donc deux opérations. La forme du squelette obtenu est comme on pouvait s'y attendre assez proche de MB, mais un squelette bien formé n'est autre qu'un squelette 4-connexe, ce qui diminue la souplesse de la représentation.

Enfin, le tableau 3.8 indique la taille du voisinage examiné par les algorithmes, c'est à dire le nombre de variables de la fonction booléenne calculée par l'opérateur correspondant. Nous reviendrons dans la conclusion de ce chapitre sur l'importance qu'accorde certains auteurs à cette donnée. Notons simplement que puisque Hall [44] a montré que la taille minimum du support pour les algorithmes de squelettisation complètement parallèles était de 11, MB, avec un support de taille 13, utilise le support isotrope minimal.

3.7.3 «Médialité»

Venons-en aux aspects non topologiques de la squelettisation, qui concernent la conservation de la géométrie. Nous avons dit dans la présentation de MB et MB2, qu'ils étaient basés respectivement sur l'axe médian 4-connexe $S_4(X)$ et sur l'axe médian mixte $S_4^8(X)$. La proposition suivante précise cette notion.

Proposition 4 *Soit $X \in \mathbb{Z}^2$. Si tous les points de X détruits par MB (resp. MB2) le sont dans l'ordre induit par Φ_4, la fonction 4-distance, alors le squelette produit par la procédure MB (resp. MB2)* contient *l'ensemble $S_4(X)$ (resp. $S_4^8(X)$).*

En effet grâce à la configuration (α) (resp. (α_1) et (α_2)), tout point x détruit par MB (resp. MB2) est 4-voisin (resp. 8-voisin) d'un point 4-intérieur. Donc si les points sont examinés dans l'ordre de la 4-distance, $\Phi_4(y) > \Phi_4(x)$, et donc x n'est pas un maximum local de la 4-distance dans son 4-voisinage (resp. 8-voisinage). Finalement le squelette doit contenir tous les points de $S_4(X)$ (resp. $S_4^8(X)$). □

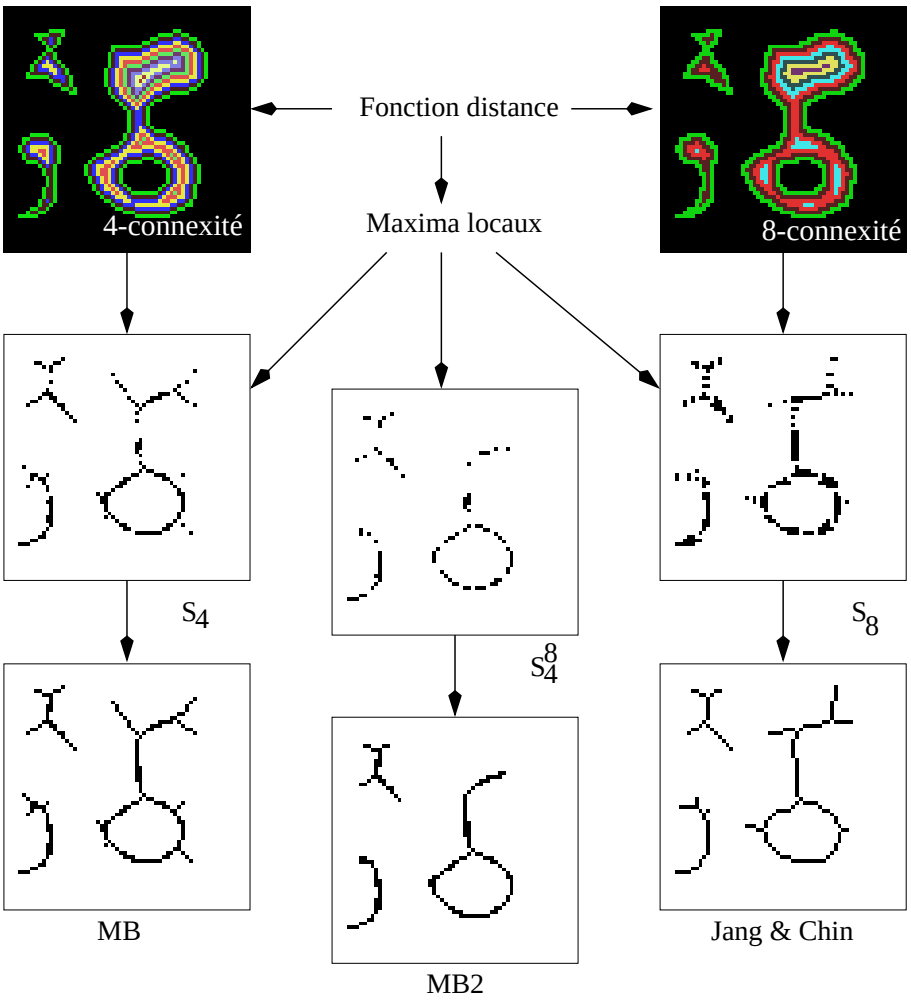


FIG. 3.24 – *Différents type d'axe médian conduisant à différents squelettes.*

Cette propriété est illustrée sur la figure 3.24, où l'on a placé à titre de comparaison le squelette de Jang et Chin, qui est clairement basé sur l'axe médian 8-connexe $S_8(X)$, même si, contrairement à MB et MB2 pour la même image, il ne le contient pas entièrement.

Il est utile de préciser que, pour la grande majorité des images, les points sont examinés dans l'ordre de la fonction 4-distance. C'est le cas pour toutes les images de résultats qui sont présentés ici, à l'exclusion de la figure 3.25. Cette figure montre deux exceptions, correspondant à des images contenant des configurations qui «protègent» un contour, l'empêchant d'être aminci.

La nature de l'axe médian sur lequel sont basés MB et MB2 a une importance fondamentale, c'est elle qui détermine le comportement de ces deux

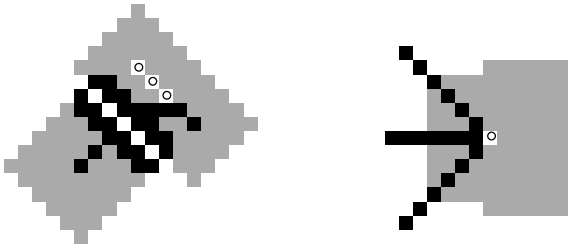


FIG. 3.25 – *Images pathologiques : Le squelette (en noir) ne contient pas entièrement l'axe médian (la différence est l'ensemble des points marqués par un cercle).*

algorithmes en tant que *descripteur de formes*. Nous allons en voir tout de suite une conséquence très importante.

3.7.4 Sensibilité au bruit et reconstructibilité

Le squelette MB2 est, par opposition à MB, basé sur un type particulier d'axe médian dont le principe est de traiter de manière plus homogène les deux distances canoniques de la maille carrée. En particulier, comme on a pu le voir sur les résultats, l'opérateur de squelettisation ne fait pas de différence entre une 4-boule et une 8-boule, dont les squelettes sont toujours réduits à un point. Il produit aussi un seul point pour toute une famille d'ensembles qui sont compris entre une 4-boule et une 8-boule de même rayon. Appelons ces ensembles des *boules floues*. On peut voir sur la figure 3.26 quelques boules floues de même rayon, et leur squelette réduit à leur centre (en blanc). Les contours blancs épais représentent les frontières entre la 4-boule et la 8-boule qui encadrent la boule floue.

Si l'on définit formellement une boule floue comme étant un ensemble B tel que $S_4^8(B)$ est réduit à un point qu'on appelle son *centre* c, on peut montrer que B est un ensemble compris entre une 4-boule et une 8-boule de même rayon, tel que pour tout x dans B, B contient le plus petit rectangle englobant c et x.

Le comportement de MB2 vis-à-vis des boules floues est à l'origine de ses bonnes propriétés concernant l'immunité au bruit et l'invariance en rotation,

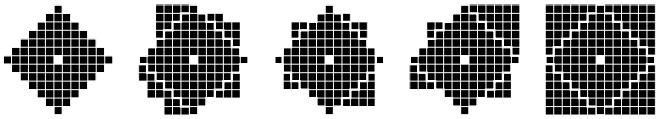


FIG. 3.26 – *Quelques boules floues de rayon 7, et leurs squelettes, réduits à leurs centres.*

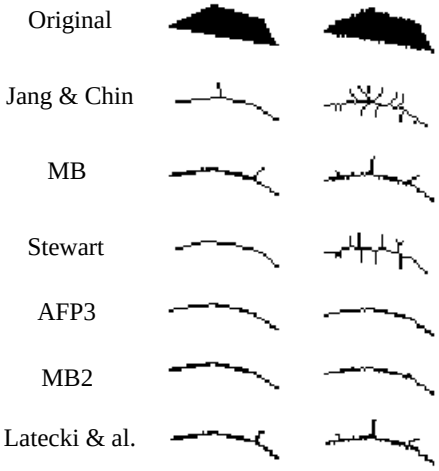


FIG. 3.27 – *Le comportements des algorithmes de squelettisation vis-à-vis du bruit.*

comme on peut s'en rendre compte sur les figures 3.27 et 3.28, où le résultat est comparé à d'autres algorithmes déjà référencés figure 3.23.

En revanche, une forme X ne peut être reconstruite à partir de son squelette MB2 que de manière floue : on sait que chaque point du squelette est centre d'une boule floue de rayon connu, de telle sorte que l'union de toutes les boules floues est égale à X.. Ce fait est illustré par la figure 3.29, où la forme est reconstruite à partir du squelette pondéré (la valeur de la fonction distance sur le squelette est ici représenté par le niveau de gris). Contrairement à MB (à gauche) qui est basé sur l'axe médian $S_4(X)$, et qui permet par conséquent une reconstruction *exacte*, MB2 (à droite) ne permet qu'une reconstruction *approximative*. Cette reconstruction peut être effectuée comme sur la figure par une dilatation par des octogones de taille correspondante, qui constituent un bon choix de boules floues «moyennes».

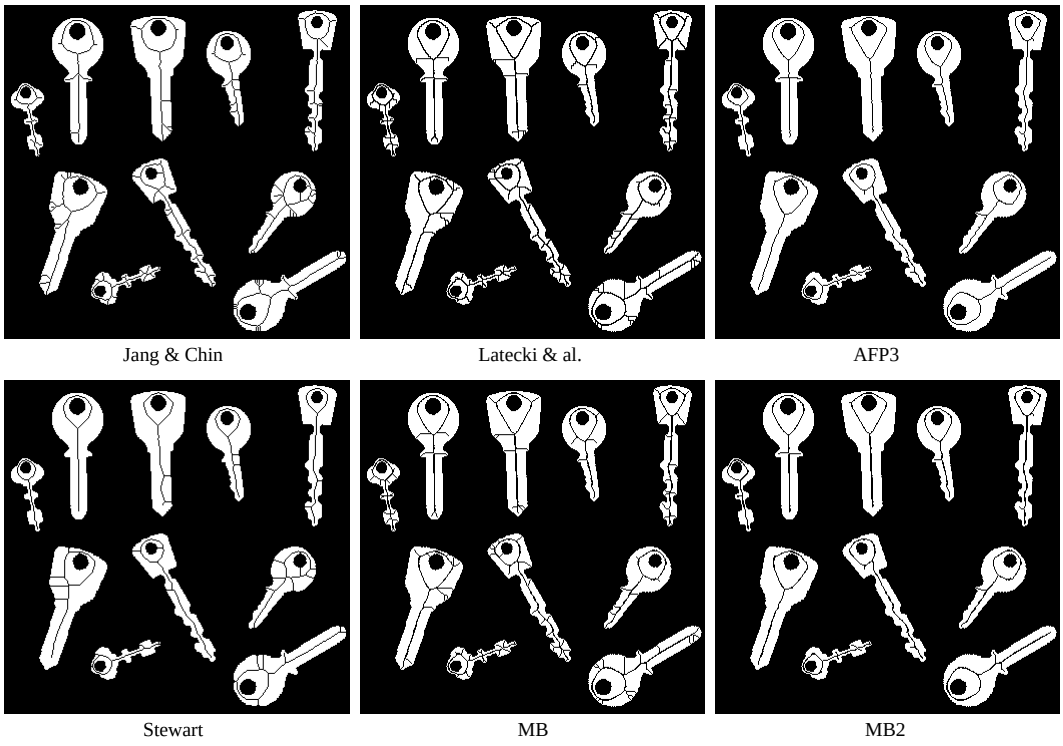


FIG. 3.28 – *Le comportements des algorithmes de squelettisation vis-à-vis de la rotation. Ici, le squelette est superposé à l'image originale.*

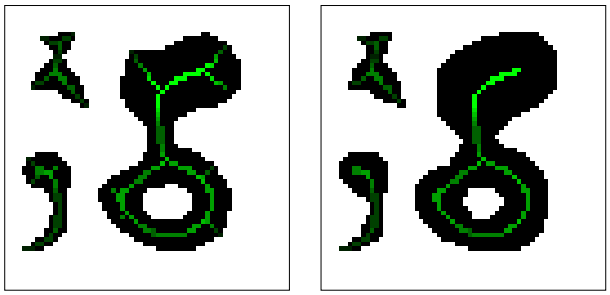


FIG. 3.29 – *Reconstruction à partir du squelette pondéré: exacte pour MB (à gauche) et approximative pour MB2 (à droite).*

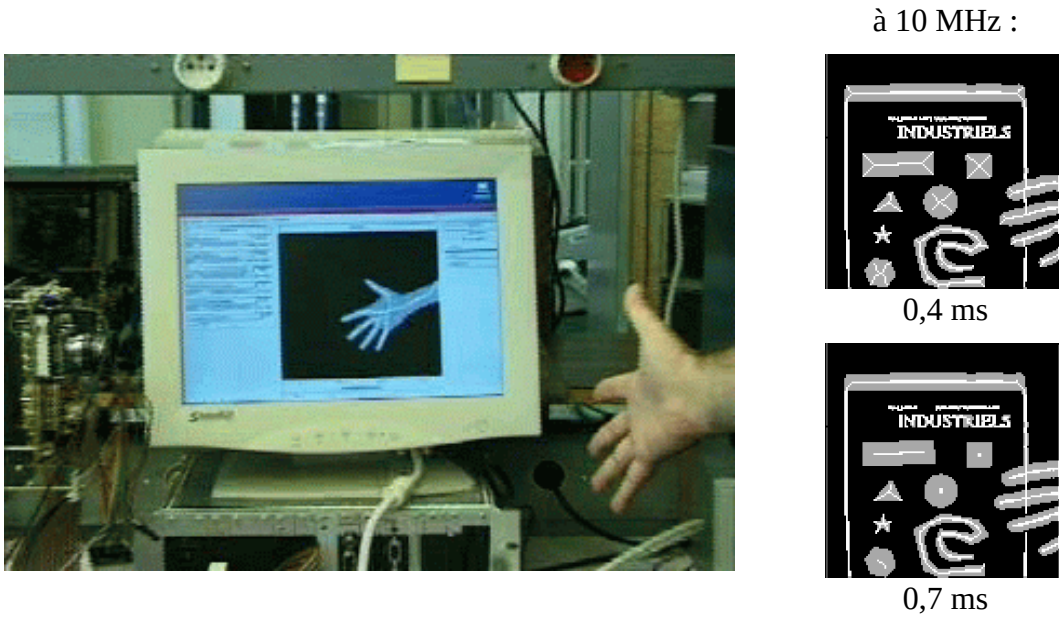


FIG. 3.30 – *Implantation sur* Pvlsar2.2 *(5 registres binaires) des algorithmes MB et MB2. Expérimentation sur le banc d'essai (à gauche). Temps de calcul total des squelettes (à droite) pour une fréquence de 10 MHz.*

3.7.5 Conclusion

Les deux algorithmes que nous venons de présenter permettent de calculer le squelette d'une image binaire de manière extrêmement efficace sur la rétine programmable. Ils produisent un résultat équivalent à celui de squelettes connus dans la littérature en au moins deux fois moins d'opérations. De ce point de vue, ils satisfont à nos objectifs de concision logique pour une implantation efficace sur la rétine programmable (voir figure 3.30).

Mais les résultats comparatifs que nous avons présentés, le fait que la concision logique implique aussi concision conceptuelle, et donc facilité d'implantation, enfin, le caractère bien défini des squelettes, qui permet de comprendre leur comportement, montrent l'intérêt de ces algorithmes indépendamment de notre architecture.

Les caractéristiques proches de MB et MB2 nous suggérent qu'il s'agit d'un seul algorithme, générique quant à la forme de l'axe médian, ce qui permet une polyvalence susceptible d'enrichir la représentation de la forme.

Nous allons en fait plus loin, en montrant que MB s'adapte facilement dans un autre type de maillage régulier, tel que la maille hexagonale, dans la section 3.8. Mais surtout, nous montrons qu'il peut s'exprimer dans la maille

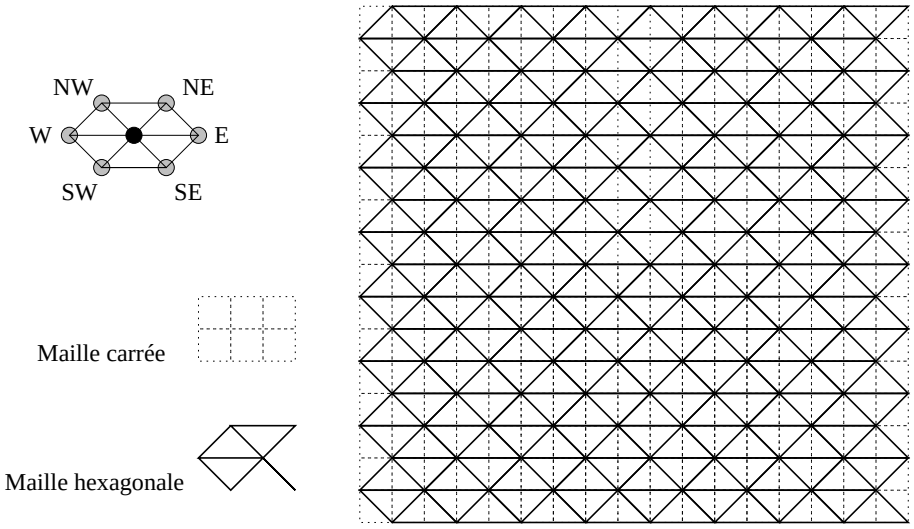


FIG. 3.31 – *La maille hexagonale.*

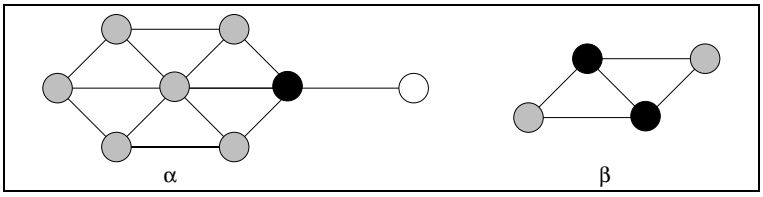


FIG. 3.32 – *Les patterns MB-hexa.*

(hyper)cubique en n'importe quelle dimension, dans la section 3.9.5.

3.8 MB en maille hexagonale

Si $\mathbb{Z}^2$ est le plan discret représenté par la maille carrée nous posons $\mathcal{H}$ le plan discret en maille hexagonale tel que $\mathcal{H} \subset \mathbb{Z}^2$ et $\forall (i,j) \in \mathbb{Z}^2, (i,j) \in \mathcal{H} \Longleftrightarrow i \not\equiv j(2)$ (voir Figure 3.31(b). La topologie est définie par la relation de voisinage : (i,j) adjacent à (i',j') si et seulement si $|i-i'| = 2$ ou $|i-i'| = |j-j'| = 1$ (voir Figure 3.31(a)).

L'algorithme MB-hexa est le suivant, il consiste à retirer les points correspondant à la configuration α mais pas à la configuration β (à une rotation de $\pi/3$ près), de la figure 3.32. Dans cette dernière configuration, les deux points noirs sont des points qui correspondent à la configuration α.

L'algorithme complet est explicitement donné par la table 3.8.

Répéter la séquence suivante pour tout $x \in X$ en parallèle, jusqu'à stabilité :
$a = x \wedge x_E$
$a = a \wedge a_{NW} \wedge a_{SW} \wedge x_W$
// A est l'érodé de X.
$b = (a_W \backslash x_E) \vee (a_E \backslash x_W) \vee (a_{SW} \backslash x_{NE}) \vee (a_{NE} \backslash x_{SW}) \vee (a_{SE} \backslash x_{NW}) \vee (a_{NW} \backslash x_{SE})$
// B est l'ensemble des points "de type α".
$c = b \wedge b_E \wedge x_{NE} \wedge x_{SE}$
$c = c \vee c_W$
$d = b \wedge b_{SE} \wedge x_{SW} \wedge x_E$
$c = c \vee (d \vee d_{NW})$
$d = b \wedge b_{SW} \wedge x_{SE} \wedge x_W$
$c = c \vee (d \vee d_{NE})$
// C est l'ensemble des points "de type β".
$x = x \setminus (b \setminus c)$

TAB. 3.9 – *La procédure MB-hexa.*

Le nombre d'opérations élémentaires effectuées au cours d'une itération est de 31. L'algorithme est en une seule passe. Un exemple de résultat est montré figure 3.33.

3.9 Squelettisation MB nD

Dans cette section, nous présentons un algorithme pour calculer le squelette d'un objet binaire n-dimensionnel. La procédure que nous proposons permet en efffet d'obtenir par relaxation d'un opérateur d'amincissement sur un sous-ensemble de $\mathbb{Z}^n$, la représentation discrète d'une variété de dimension $(n-1)$, tout en préservant la topologie et la géométrie, dans le sens communément accepté en 2D et en 3D.

La réduction de la quantité d'information nécessaire à la représentation d'une forme constitue un enjeu majeur pour diverses disciplines, qui ne se limite pas aux objets bidimensionnels. L'intérêt de la squelettisation 3D, en particulier, manifesté depuis une dizaine d'année par un nombre grandissant de publications sur le sujet, réside dans la modélisation d'objets complexes comme par exemple des réseaux sanguins ou bronchiques en imagerie médicale. La squelettisation nD semble un problème plus exotique, mais elle est utile par exemple en robotique, pour définir les trajectoires les plus sûres dans un espace multi-paramètres, dans lequel l'«image» binaire représente le sous-espace des contraintes satisfaites.

Les premières parties de cette section sont consacrées à la présentation du

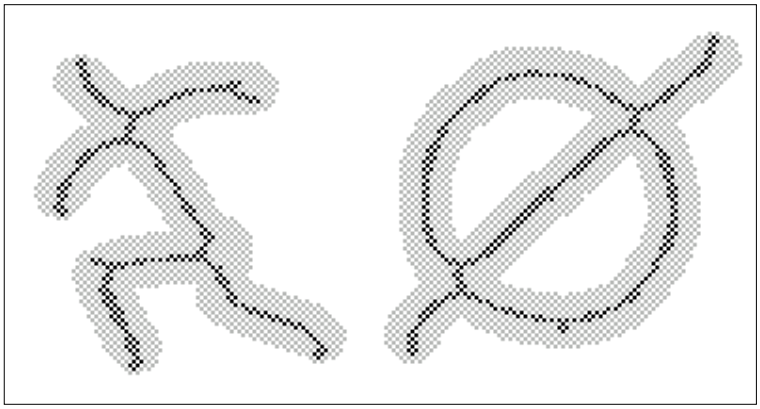


FIG. 3.33 – *Un exemple de squelette MB-hexa.*

formalisme nécessaire à la définition de la procédure d'une part et à l'étude de ses propriétés d'autre part. Notons que nous le présentons de manière totalement unifiée, sans propriété spécifique à telle dimension. Dans les parties suivantes, nous présentons notre algorithme nD, puis nous développons ses propriétés, en nous concentrant sur les aspects 3D, puisque le cas 2D, dont cet algorithme constitue une généralisation, a déjà été traité dans les sections précédentes.

3.9.1 La grille hypercubique

Soit $\mathbb{Z}$ l'ensemble des entiers, $\mathbb{N}$ l'ensemble des entiers naturels, $\mathbb{R}$ l'ensemble des nombres réels. Soit $\mathcal{P}(A)$ l'ensemble des parties de A, $\mathcal{P}^*(A) = \mathcal{P}(A) \setminus \emptyset$. Soit $n \in \mathbb{N}$, $\mathbb{Z}^n$ est l'espace discret n-dimensionnel. La *grille (hyper)cubique* est definie par le plongement de $\mathbb{Z}^n$ dans l'espace affine attaché à $\mathbb{R}^n$ par l'application suivante :

$$\begin{array}{rccl} \Phi : & \mathbb{Z}^n & \longrightarrow & \mathcal{P}(\mathbb{R}^n) \\ & z & \longmapsto & \Phi(z) = \prod_{i=1}^{n} [z_i - \frac{1}{2}; z_i + \frac{1}{2}] \end{array}$$

Ainsi un *point* z de l'espace discret est identifié à un *hypercube* de l'espace affine quantifié, c'est-à-dire au produit cartésien des segments fermés centrés sur z, z_i étant la i-ème coordonnées de z dans la base canonique.

3.9.2 Topologies discrètes hybercubiques

Soit Ψ la fonction définie sur $\mathcal{P}^*(\mathbb{R}^n)$ par $\Psi(P) = \mathcal{V}$ tel que $\mathcal{V}$ est la variété affine engendrée par P. On peut définir n types de relations d'adjacence (et donc de topologies) dans la maille hypercubique, comme suit :

Définition 25 *Soient z et z' deux points de $\mathbb{Z}^n$ tels que $\Phi(z) \cap \Phi(z') \neq \emptyset$. alors z et z' sont* k-adjacents $(0 \leq k \leq n-1)$ *si et seulement si :*

$$dim(\Psi(\Phi(z) \cap \Phi(z')) = k.$$

Définition 26 *Soient z et z' deux points de $\mathbb{Z}^n$. z et z' sont* k-voisins *s'il existe j, $k \leq j \leq n-1$, tel que z et z' sont j-adjacents.*

L'adjacence de deux points correspond donc à une intersection non vide de deux hypercubes, et le niveau d'adjacence, à la dimension de la variété affine engendrée par l'intersection. On notera la différence avec la notion de voisin, qui correspond à la définition usuelle de la littérature. Un exemple en 2D est donné Figure 3.34.

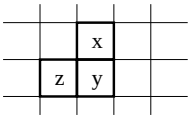


FIG. 3.34 – *Relations de connexité en dimension 2: x et y sont 1-adjacents (et pas 0-adjacents). x et z sont 0-adjacents. x et y sont 1-voisins et par conséquent 0-voisins.*

Soit une *image binaire* I un sous-ensemble de $\mathbb{Z}^n$.

Définition 27 *Points* Intérieurs *d'une image binaire : $I \subset \mathbb{Z}^n$. $z \in I$, z est un point* k-intérieur *de I si et seulement si : $\forall z'$, z et z' sont k-voisins $\Rightarrow z' \in I$.*

Définition 28 *Soient A et B deux sous-ensembles de $\mathbb{Z}^n$. A et B sont dits k-connectés s'il existe $a \in A$ et $b \in B$ tels que a et b sont k-voisins. $X \subset \mathbb{Z}^n$ est une k-composante connexe (notée k-cc) s'il n'existe pas de partition de X en deux sous-ensembles qui ne soient pas k-connectés.*

Dénombrement des k-voisins : Considérons l'origine O de $\mathbb{Z}^n$, un point lui est k-adjacent si et seulement si il a $n-k$ coordonnées dans l'ensemble $\{-1, +1\}$, et les autres égales à 0. Donc un point a $A(n,k)$ points k-adjacents, avec :

$$A(n,k) = 2^{n-k} C_n^{n-k}.$$

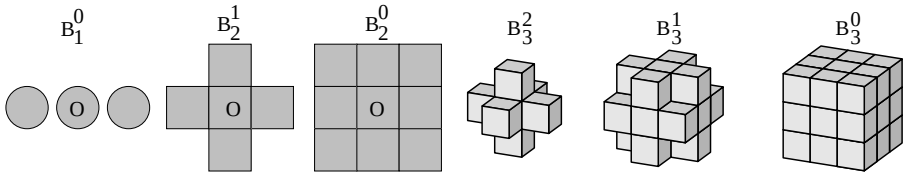


FIG. 3.35 – *Les voisinages élémentaires de l'origine pour les dimensions 1 à 3.*

et donc $V(n,k)$ voisins (à l'exclusion de lui-même), avec :

$$V(n,k) = \sum_{i=k}^{n-1} A(n,i).$$

Le tableau 3.10 donne les nombres $V(n,k)$ pour n inférieur à 4. Pour $k=0$, on retrouve bien sûr le nombre total de voisins en dimension n :

$$W_n = V(n,0) = \sum_{i=0}^{n-1} 2^{n-i} C_n^{n-i} = 3^n - 1.$$

k \ n	1	2	3	4
3	-	-	-	8
2	-	-	6	32
1	-	4	18	64
0	2	8	26	80

TAB. 3.10 – *Nombres de k-voisins en nD, pour* $n \leq 4$.

Ces nombres sont présentés pour insister sur le fait que, dans la littérature, comme dans la première partie de ce chapitre, les termes usuellement employés sont $V(n,k)$-connexité, $V(n,k)$-voisins, etc. Néanmoins, dans toute cette section, dans un but à la fois d'homogénéité et de concision, nous utiliserons toujours le simple préfixe "k-".

La figure 3.35 montre les boules élémentaires pour les dimensions 1 à 3. On notera par la suite, comme sur la figure B_n^k l'ensemble des k-voisins de l'origine en dimension n.

3.9.3 Distances discrètes et hypersurfaces médianes

Soit δ_k^n la distance induite par la k-topologie en dimension n. (ou δ_k lorsqu'il n'y a pas d'ambiguïté quant à la dimension). Soit $X \subset \mathbb{Z}^n$. Les

définitions métriques utilisés en nD sont équivalentes à celle que nous avons présentées en 2D :

La *fonction distance* associée à X et relative à δ_k est la fonction qui à x dans $\mathbb{Z}^n$ associe $\delta_k(x, X^c)$.

$\mathcal{B}_k(x, r)$ désigne la boule de centre $x \in \mathbb{Z}^n$ et de rayon $r \in \mathbb{N}$.

$S_k(X)$, l'ensemble des centres de boules maximales pour la distance δ_k est appelée *hypersurface médiane* de X associée à la distance δ_k. C'est aussi l'ensemble des maxima locaux de la fonction distance correspondante.

On généralise également la notion d'axe médian mixte de la définition 24 par la notion de *(m,p)-surface médiane* :

Définition 29 *Soit $p \leq m \leq n$, la (m,p)-(hyper)surface médiane est définie par:*

$$S_m^p(X) = \{x \in X; \forall y \text{ } p\text{-voisin de } x, \delta_m(y, X^c) \leq \delta_m(x, X^c)\}.$$

La (m,p)-surface médiane est donc l'ensemble des maxima de la fonction m-distance, dans le p-voisinage.

3.9.4 Préservation de la topologie

Le problème de la conservation de la topologie en maille cubique est lié comme en dimension 2 à la vérification de la propriété de Jordan. Il est par exemple souhaitable qu'une composante connexe du complémentaire ne traverse la surface de l'objet que si celle-ci comporte un trou. Pour retrouver cette propriété, il est nécessaire de considérer deux niveaux de connexité différents pour l'objet et pour son complémentaire. Pour l'algorithme proposé, nous considérons toujours le niveau de connexité le plus large pour l'objet : la 0-connexité (deux hypercubes de X sont connectés dès qu'ils ont un sommet en commun), et le niveau le plus strict pour le complémentaire : la (n-1)-connexité (deux hypercubes de X^c sont connectés seulement s'ils ont une hyperface en commun). Nous utiliserons le préfixe «(0,n-1)-» pour rappeler ce choix de topologie.

Précisons qu'en nD, la notion de *trou* n'est pas clairement définie. En dimension 3, on distingue les *cavités*, qui sont des composantes connexes bornées du complémentaire, et les *trous* au sens de Jordan, qui peuvent «bloquer» un lacet fermé. Une transformation homotopique en 3D doit préserver à la fois cavités et trous.

Dans cette section nous ne débattrons pas du problème de préservation de la topologie en nD pout tout n, puisqu'à notre connaisance, il n'existe pas de caractérisation reconnue en nD par la communauté scientifique travaillant sur le thème de la topologie discrète. Néanmoins le formalisme qui est proposé dans les résultats qui suivent est complètement générique quant

à la dimenson, et les résultats sont vrais pour les dimensions 2 et 3.

Il faut d'abord donner un sens à «préserver la topologie» en dimension supérieure à 2. En dimension 3, le groupe fondamental de X n'est plus suffisant pour caractériser sa topologie. Il faut considérer aussi le groupe fondamental de X^c (qui ne dépend pas que de celui de X comme en 2D). Ainsi, en 3D déjà, on ne peut plus représenter la topologie par l'arbre de connexion. Dans les dimensions suivantes, les groupes d'homotopie d'ordre supérieur sont nécessaires pour caractériser la topologie.

Comme en 2D, on utilise la notion de *simplicité* pour désigner un ensemble ou un point dont le retrait ne modifie pas la topologie. En 2D et 3D, il existe des caractérisations *explicites* et locales de la simplicité des points. Donnons à présent cette caractérisation pour la (0,n-1)-connexité :

Théorème 7 *Soit $X \subset Z^n$ une image binaire. Soit $x \in X$. On note X_0^x l'ensemble des 0-voisins de x, sauf x lui-même, qui appartiennent à X et $\overline{X_{n-2}^x}$ l'ensemble des (n-2)-voisins de x qui appartiennent à X^c. x est* simple *dans X pour la (0,n-1)-connexité si et seulement si les deux conditions suivantes sont vérifiées :*

- *$\{x\}$ est 0-connecté à une seule 0-cc de X_0^x.*
- *$\{x\}$ est (n-1)-connecté à une seule (n-1)-cc de $\overline{X_{n-2}^x}$.*

Ce théorème est dû à Bertrand et Malandain[15] qui l'ont énoncé en 3D. Dans notre formulation unifiée, il correspond pour n=2 au théorème 5 avec K=8.

La caractérisation des ensembles simples proposée par Ronse dans [97] a été généralisée aux images de dimension supérieure par Kong [52]. Il montre que comme en 2D, un sous-ensemble de X de $\mathbb{Z}^n$ est simple si et seulement si il peut être ordonné en une séquence $\{x_1, \ldots, x_n\}$ telle que pour tout i dans $\{1, \ldots, n\}$, x_i est simple dans $X \setminus \{x_1, \ldots, x_{i-1}\}$.

Comme Ronse [98] l'avait fait en 2D, Ma [60] propose des conditions suffisantes pour prouver la correction d'algorithmes d'amincissement parallèles en 3D en ne testant qu'un nombre limité de configurations. Grâce à la définition 30, nous pouvons proposer ici encore une expression unifiée de ces résultats dans le théorème 8.

Définition 30 *On appelle* élément unité de dimension k *de la grille hypercubique, $0 \leq k \leq n$, tout ensemble U de 2^k points de $\mathbb{Z}^n$, tel que toute paire de points de U est une paire de (n-k)-voisins.*

Théorème 8 *(Généralisation de Ronse 88 et Ma 94)*
Soit $X \subset Z^n$ une image binaire. Un algorithme qui retire en parallèle des points d'une image binaire n-dimensionnelle X préserve la (0,n-1)-connexité

si les deux conditions suivantes sont satisfaites :

- *Tout sous-ensemble de X contenu dans un élément unité de dimension $(n-1)$ et qui est enlevé par l'algorithme est simple.*
- *Aucune composante connexe de X contenue dans un élément unité de dimension n ne peut être entièrement enlevée.*

3.9.5 L'algorithme MB-nD

La procédure MB-nD est définie au moyen de deux fonctions binaires de $\mathbb{Z}^n$: la fonction *Alpha* et la fonction *Beta*. Ces deux fonctions sont respectivement définies dans les tableaux 3.11 et 3.12. L'algorithme est précisément énoncé dans le tableau 3.13.

Soit $I \subset Z^n$. La fonction $\alpha : I \longrightarrow \{0,1\}$ est définie comme suit :
Soit $z \in I$. SI il existe $k, 0 \leq k \leq n-1$, tel que :
(1) Il existe z' point $(n-1)$-intérieur, tel que z et z' sont k-adjacents
(2) Tout point t $(n-1)$-adjacent à z et tel que :
$\mathcal{S}_z(\Phi(z) \cap \Phi(z')) \subset (\Phi(z) \cap \Phi(t))$
appartient à I^c.
($\mathcal{S}_z$ désigne la symétrie de centre z)
ALORS $\alpha(z) = 1$.
SINON $\alpha(z) = 0$.

TAB. 3.11 – *Fonction Alpha.*

Exprimons la définition donnée dans le tableau 3.11 de manière moins formelle : un point z de I tel que $\alpha(z) = 1$ doit être k-adjacent à un point $(n-1)$-intérieur . Si tel est le cas, les deux hypercubes correspondants ont une intersection non vide, qui engendre une variété affine de dimension k. Considérons alors l'image de cette variété par la symétrie de centre z. C'est une variété affine parallèle de dimension k, égale à l'intersection de $n-k$ hyperplans affines. Alors $\alpha(z) = 1$ si tous les hypercubes $(n-1)$-adjacents à z et dont l'intersection avec z engendre l'un de ces hyperplans appartiennent au complémentaire de I.

Exprimons aussi informellement la définition du tableau 3.12 : un point z de I tel que $\beta(z) = 1$ doit contenir dans son k-voisinage deux couples de points k-adjacents qui ont la même intersection, et tels que l'un des deux couples appartient à l'image, et l'autre au complémentaire.

Le tableau 3.14 montre les configurations qui représentent les fonctions Alpha et Beta de l'algorithme MB-nD pour les dimensions 1 à 3. Un point

Soit $I \subset Z^n$. La fonction $\beta : I \longrightarrow \{0,1\}$ est définie comme suit :
Soit $z \in I$. SI il existe $k, 0 \leq k \leq n-2$, tel que :
Il existe deux couples de points k-adjacents (a,b) et (c,d) tels que a,b,c,d sont tous k-voisins de z, et tels que les deux conditions suivantes sont vérifiées :
(1) $\Phi(a) \cap \Phi(b) = \Phi(c) \cap \Phi(d)$
(2) $\{a,b\} \subset I$ et $\{c,d\} \subset I^c$
ALORS $\beta(z) = 1$.
SINON $\beta(z) = 0$.

TAB. 3.12 – *Fonction Beta.*

Répéter pour tous les points $z \in Z^n$ *en parallèle*, jusqu'à convergence:
SI $\alpha(z) = 1$ ET $\beta(z) = 0$, ENLEVER z.

TAB. 3.13 – *L'algorithme MB-nD.*

est enlevé si : (1) La configuration de son voisinage est identique à l'un des motifs α_i, où les points gris appartiennent à l'image, les points blancs au complémentaire, le point noir est l'origine du motif. (2) Il ne contient aucun des motifs β_i dans son i-voisinage.

Les motifs représentant la fonction Alpha correspondent aux configurations de transformée en tout-ou-rien (TTR) de la définition 3.

La fonction Beta peut aussi être définie formellement à l'aide des motifs comme les TTR. Un motif de $\mathbb{Z}^n$ étant un couple (H,M) de sous-ensemble finis de $\mathbb{Z}^n$ tels que $H \cap M = \emptyset$, nous définissons les *transformée de voisinage en tout-ou-rien* (TVTTR) [64] comme suit :

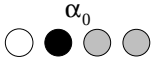
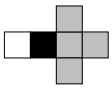
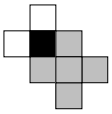
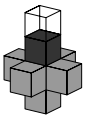
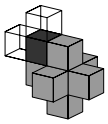
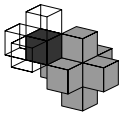
Si l'on note B_n^k l'ensemble des k-voisins de l'origine, comme sur la figure 3.35, si γ est un motif de $\mathbb{Z}^n$, la TVTTR de X par γ, relativement au k-voisinage, notée $X \circledcirc_k \gamma$ est définie par l'ensemble :

$$X \circledcirc_k \gamma = (X \circledast \gamma) \oplus (B_n^k \ominus (H \cup M)).$$

$x \in X \circledcirc_k \gamma$ signifie donc que x contient dans son k-voisinage la configuration γ. Dans cette définition, l'origine de γ peut être choisie arbitrairement, c'est pourquoi elle n'apparaît pas dans les motifs β représentés sur le tableau 3.14. Remarquons que quelque soit k, $X \circledast \gamma \subset X \circledcirc_k \gamma$

Grâce à ces notations, on peut exprimer la procédure MB-nD dans le formalisme en tout-ou-rien, de la façon suivante :

- $\alpha(z) = 1 \Longleftrightarrow \exists i,\ x \in X \circledast \alpha_i$.
- $\beta(z) = 1 \Longleftrightarrow \exists i,\ x \in X \circledcirc_i \beta_i$.

dimension	fonction Alpha	fonction Beta
1	α_0 	—
2	α_0  α_1 	β_0 
3	α_2  α_1  α_0 	β_1  β_0 

TAB. 3.14 – *Les configurations intervenant dans le calcul de MB-nD pour les dimensions 1 à 3.*

3.9.6 Résultats et propriétés

L'algorithme MB-nD correspond en 2D à l'algorithme MB2, que nous avons déjà présenté. N'ayant pas expérimenté au delà de la troisième dimension, nous nous concentrerons dans cette section sur le cas 3D.

La figure 3.36 montre quelques résultats de MB-3D sur de petites formes tridimensionnelles. La figure 3.37 donne un exemple d'application dans le cadre de l'imagerie médicale, par la squelettisation d'une image binaire de poumon obtenue par segmentation [93] des coupes 2D fournies par un scanner.

Concernant l'homotopie, la propriété fondamentale de l'algorithme MB-nD est donnée par la proposition suivante :

Proposition 5 *MB-nD préserve la (0,n-1)-topologie.*

Cette proposition a été prouvée plus haut pour n=2. Nous allons à présent fournir la preuve pour n=3. En accord avec le théorème 8 pour n=3, il faut prouver les deux assertions suivantes :

(1) Tout partie d'un ensemble retiré par MB-3D et contenue dans un carré 2×2 est simple.

(2) Aucune composante connexe contenue dans un cube $2 \times 2 \times 2$ ne peut être complètement retirée par MB-3D.

Soit $X \subset \mathbb{Z}^3$, on notera $mb(X)$ le sous-ensemble de X des points retirés

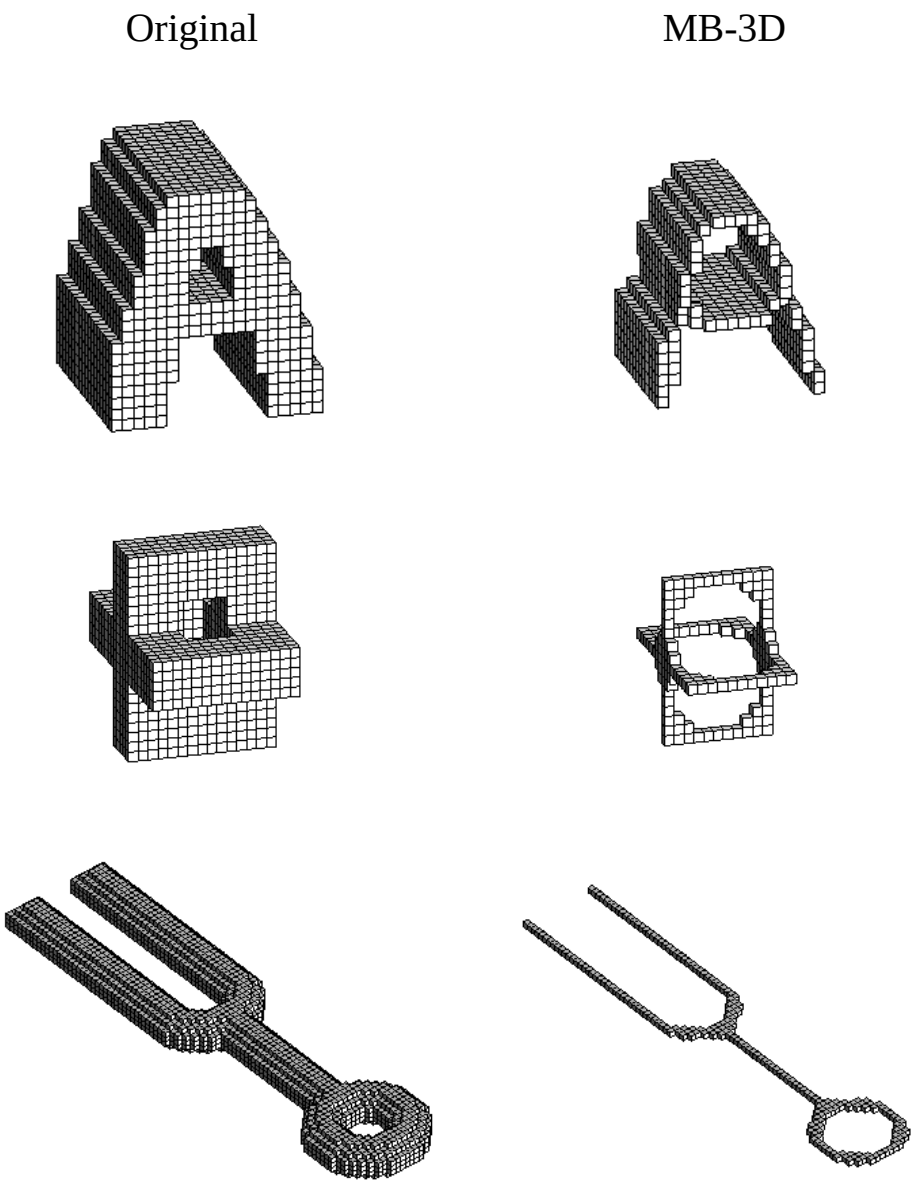


FIG. 3.36 – *Résultats de MB3D. La colonne de gauche contient les images originales, celle de droite les squelettes MB-3D.*

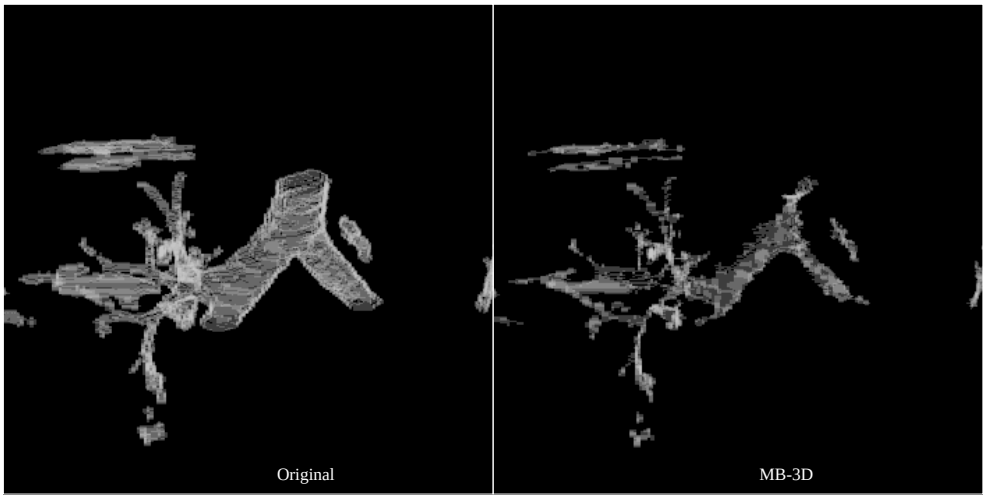


FIG. 3.37 – *Résultat de MB-3D sur une image segmentée de poumon.*

par MB-3D. Pour $x \in X$, nous utiliserons les deux ensembles X_0^x et $\overline{X_1^x}$ du théoreme 7 pour n=3. La preuve repose sur cinq lemmes. Les lemmes 1 à 3 traitent de la préservation de la 0-topologie de l'objet tandis que les lemmes 4 et 5 concernent la préservation de la 2-topologie du complémentaire. Les lemmes 1 et 4 montrent qu'une itération de l'algorithme MB-3D ne retire que des points simples. Les lemmes 1 et 2 sont utilisés pour prouver le lemme 3. Le lemme 4 est utilisé pour prouver le lemme 5. Les lemmes 3 et 5 montrent que toute paire de points 2-adjacents retirée par MB-3D est un ensemble simple. La preuve est finalement complétée dans la proposition 6.

Lemme 1 *Soit $x \in X$, entre deux 2-voisins a et b, tels que $a \notin X$ et $b \in X$ (cf figure 3.39). Si $\{x\}$ est 0-connecté à plus d'une 0-cc de X_0^x, alors soit $x \in X \circledast \beta_1$, soit $x \in X \circledast \lambda$, la configuration λ étant représentée sur la figure 3.38.*

PREUVE

Si $\{x\}$ est 0-connecté à plus d'une 0-cc de X_0^x, alors il existe un point y de X_0^x qui n'est pas 0-voisin de b. y ne peut pas être un 2-voisin de x, mais il peut être un 1-voisin, comme l'illustre le point c sur la figure 3.39(1). Dans ce cas, puisque c et b n'appartiennent pas à la même 0-cc, $x \in X \circledast \beta_1$. Si un tel point c n'existe pas, alors y est seulement un 0-voisin de x, comme l'illustre le point d sur la figure 3.39(2). Dans ce cas, $x \in X \circledast \lambda$ □

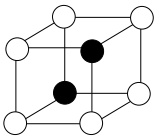


Fig. 3.38 – Config. λ

Corollaire 1 *Tout point $x \in mb(X)$ vérifie la première condition du théorème 7 pour n=3.*

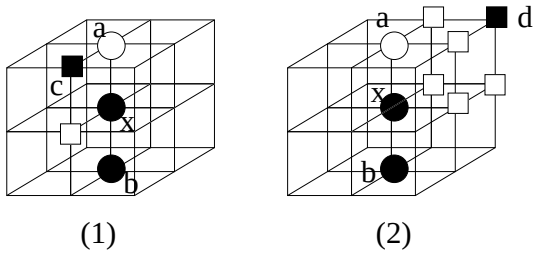


FIG. 3.39 – *Preuve du lemme 1.*

En effet, tout point contenue dans une configuration α_1 ou α_2 est nécessairement entre deux 2-voisins, l'un dans X, l'autre dans X^c. Il en va de même pour un point contenue dans α_3 et pas dans β_1. Le lemme 1 s'applique donc, et puisque la configuration λ est un cas particulier de la configuration β_2, le point est 0-voisin d'une seule 0-cc de X_0^x.

Lemme 2 *Soit $x \in X$. Soit Y un sous-ensemble de X tel que $Y \subset mb(X)$ et $Y \cup \{x\}$ soit contenue dans un carré 2×2. Alors $x \in (X \setminus Y) \circledcirc_0 \lambda$ implique $x \in X \circledcirc_0 \beta_0$.*

PREUVE

Considérons $x \in (X \setminus Y) \circledcirc_0 \lambda$. Si $x \in X \circledcirc_0 \lambda$, alors $x \in X \circledcirc_0 \beta_0$. Sinon, on se trouve dans la situation de la figure 3.40(1), où $Y \subset \{y_1, y_2, y_3\}$. Notons que les trois points représentés par des carrés appartiennent à Y ou à X^c. Si $y_1 \in X^c$ ou $y_3 \in X^c$, alors évidemment $x \in X \circledcirc_0 \beta_0$. Sinon, $\{y_1, y_3, z\} \subset X$. Dans ce cas, y_2 doit être contenue dans un α_i pour lequel le point intérieur apartient au cube de la figure 3.40(1). Mais pour chacune des sept possibilités, on vérifie facilement que ce n'est pas possible. Ainsi $y_2 \notin Y$, et donc $y_2 \in X^c$, et les quatre points $\{x, t, y_2, z\}$ forme une configuration β_0 □

Lemme 3 *Soit x et y deux 2-voisins tels que $\{x, y\} \subset mb(X)$. Alors $\{x\}$ est 0-connecté à une seule 0-cc de $(X \setminus \{y\})_0^x$.*

PREUVE

Sous les hypothèses du lemme 3, on peut facilement vérifier que quelque soit la configuration α_i dans laquelle il est contenu, x est entre deux 2-voisins, l'un dans $X \setminus \{y\}$, et l'autre dans X^c. Supposons maintenant que $\{x\}$ est 0-connecté à plus d'une 0-cc of $(X \setminus \{y\})_0^x$. D'après le lemme 1, x doit être contenu dans β_1 ou λ relativement à l'ensemble $(X \setminus \{y\})$. Mais le lemme 2 montre que cela ne peut être λ puisque x aurait alors contenu la configuration β_0 dans son 0-voisinage, et ce avant la destruction de y, ce qui est en contradiction avec sa destruction par MB-3D. Donc x est contenu dans β_1 relativement à $(X \setminus \{y\})$; précisément, la situation de x est celle de la figure 3.39(1), avec c et b dans deux 0-cc distinctes. Puisque x n'est pas

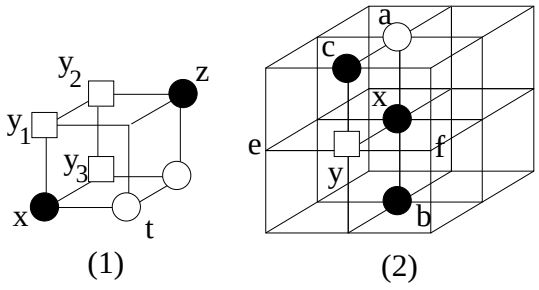


FIG. 3.40 – *Preuve des lemmes 2 et 3.*

contenu dans β_1 relativement à X, y, après sa destruction, fait partie d'une configuration β_1, comme le montre la figure 3.40(2). En outre, e et f doivent tous deux appartenir à X^c. Mais alors, y ne peut avoir appartenu à aucun α_i, ce qui contredit sa destruction par MB-3D □

Lemme 4 *Soit $x \in X$, entre deux 2-voisins a et b, tels que $a \notin X$ et $b \in X$. Si $\{x\}$ est 2-connecté à plus d'une 2-cc de $\overline{X_1^x}$, alors x est contenu dans une configuration β_1.*

PREUVE

Voir la figure 3.41(1). S'il existe $c \notin X$ tel que a et c appartiennent à deux 2-cc distinctes de $\overline{X_1^x}$, alors le point d tel que $d \neq x$, et d 2-adjacent à la fois à a et à c doit appartenir à X. Et donc $x \in X \circledast \beta_1$ □

Corollaire 2 *Tout point $x \in mb(X)$ vérifie la deuxième condition du théorème 7 pour n=3.*

Lemme 5 *Soit x et y deux 2-voisins tels que $\{x, y\} \subset mb(X)$. Alors $\{x\}$ est 2-connecté à une seul 2-cc de $\overline{(X \setminus \{y\})_1^x}$.*

PREUVE

Les hypothèses du lemme 5 (identiques à celles du lemme 3), impliquent que x est entre deux 2-voisins, l'un dans $X \setminus \{y\}$ et l'autre dans X^c. Supposons maintenant que $\{x\}$ est 0-connecté à plus d'une 2-cc de $\overline{(X \setminus \{y\})_1^x}$. D'après le lemme 4, x doit être contenu dans β_1 relativement à $(X \setminus \{y\})$. Voir figure 3.41(2), où a et y appartiennent à deux 2-cc distinctes de $\overline{(X \setminus \{y\})_1^x}$. Si b et c appartiennent tous deux à X, alors y n'aurait pu appartenir à aucun α_i, donc b ou c appartiennent à X^c. Supposons que ce soit b. Puisque $x \in mb(X)$, $x \notin X \circledcirc_0 \beta_0$, et donc $d \in X^c$. Puisque $x \notin X \circledcirc_1 \beta_1$, $e \in X^c$ également, et finalement a et y appartiennent à la même 2-cc, ce qui conduit à une contradiction □

Venons-en à présent à la proposition principale.

Proposition 6 *MB-3D préserve la (0,2)-topologie.*

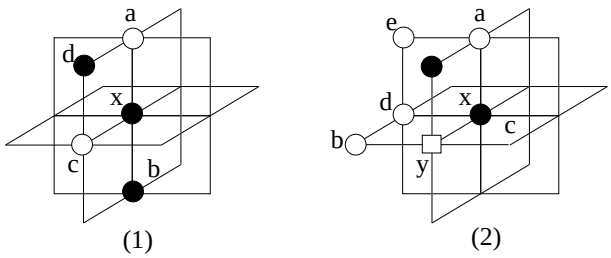


FIG. 3.41 – *Preuve des lemmes 4 et 5.*

PREUVE

Comme nous l'avons mentionné plus haut, le lemme 1 et le lemme 4 prouvent que tous les points retirés par une itération de MB-3D sont des points simples. Considérons maintenant $\{x_1, x_2\}$ une paire de 2-voisins, détruits simultanément par MB-3D. Le lemme 3 et le lemme 5 montrent que $\{x_1, x_2\}$ est un ensemble simple.

Plus généralement, soit Y un ensemble de points tel que $Y \subset mb(X)$ et Y est contenu dans un carré 2×2. Soit $x \in Y$ tel que x n'est pas simple dans $(X \setminus (Y \setminus \{x\}))$. Alors les lemmes 1 et 4, montrent que x est contenu dans une configuration β_1 ou λ, mais cette dernière est interdite par le lemme 2. Donc $x \in (X \setminus (Y \setminus \{x\})) \circledcirc_1 \beta_1$.

Considérons maintenant $\{x_1, x_2\} \subset mb(X)$ une paire de points 1-adjacents. Il est facile de voir que si $x_1 \notin X \circledcirc_1 \beta_1$, alors $x_1 \notin (X \setminus \{x_2\}) \circledcirc_1 \beta_1$. Donc x_1 est simple dans $(X \setminus \{x_2\})$, et donc $\{x_1, x_2\}$ est un ensemble simple.

Soit $\{x_1, x_2, x_3\} \subset mb(X)$ un triplet de points contenu dans un carré 2×2 tel que x_1 et x_2 sont 2-adjacents. Alors $\{x_1, x_2\}$ est simple, et on voit facilement que si $x_3 \notin X \circledcirc_1 \beta_1$, alors $x_3 \notin (X \setminus \{x_1, x_2\}) \circledcirc_1 \beta_1$, donc $\{x_1, x_2, x_3\}$ est un ensemble simple. Soit $\{x_1, x_2, x_3, x_4\} \subset mb(X)$ les quatre sommets d'un carré 2×2. $\{x_1, x_2, x_3\}$ est un ensemble simple, et si $x_4 \notin X \circledcirc_1 \beta_1$, alors $x_4 \notin (X \setminus \{x_1, x_2, x_3\}) \circledcirc_1 \beta_1$, donc $\{x_1, x_2, x_3, x_4\}$ est un ensemble simple.

Nous avons ainsi prouvé que tout ensemble retiré par MB-3D contenu dans un carré 2×2 est un ensemble simple. Enfin, il est évident qu'une itération de MB-3D ne peut pas retirer complètement une composante connexe contenue dans un cube $2 \times 2 \times 2$, puisqu'aucune configuration α_i n'est contenue dans ce cube élémentaire. Nous avons ainsi prouvé que MB-3D est un opérateur de réduction parallèle qui vérifie les deux conditions du théorème 8 pour n=3. Donc MB-3D preserve la (0,2)-topologie □

En ce qui concerne la préservation de la géométrie, le comportement de l'algorithme MB-nD est fondé sur le fait que le squelette MB-nD est basé sur la (n-1,0)-surface médiane de la définition 29, plus précisément :

Proposition 7 *Soit $X \in \mathbb{Z}^n$. Si tous les points de X détruits par MB-nD*

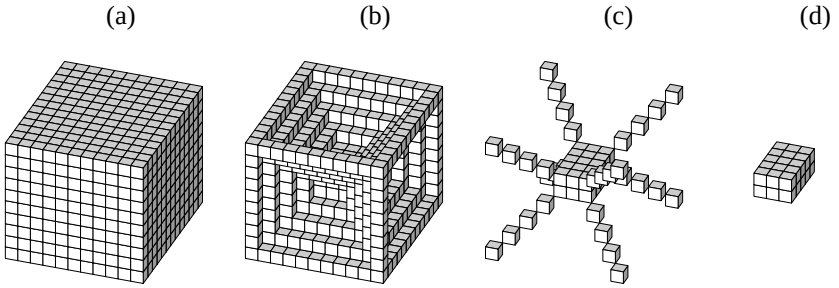


FIG. 3.42 – *Les différentes surfaces médianes en 3D: (a) Original (b)* S_2^2 *(c)* S_2^1 *(d)* S_2^0.

le sont dans l'ordre induit par la fonction (n-1)-distance, alors le squelette produit contient *l'ensemble* $S_{n-1}^0(X)$.

Il suffit en effet de considérer les motifs α_i du tableau 3.14 pour voir qu'un point x ne peut être enlevé que s'il a un 0-voisin y (n-1)-intérieur. Si les points sont examinés dans l'ordre de la fonction (n-1)-distance, alors $\delta_{n-1}(y, X^c) > \delta_{n-1}(x, X^c)$, et donc x n'est pas un maximum local de la (n-1)-distance dans son 0-voisinage, et donc $x \notin S_{n-1}^0(X)$.

Cette propriété, déjà illustrée en 2D sur la figure 3.24 l'est en 3D sur la figure 3.42, où l'on voit que différents squelettes peuvent être obtenus selon le type de surface médiane. Sur la figure 3.42(b), le squelette est obtenu par la restriction de la fonction Alpha au seul motif α_2, ce qui conduit à un squelette basé sur les maxima locaux de la 2-distance. Sur la figure 3.42(c), en se restreignant aux motifs α_2 et α_1, on obtient un squelette basé sur la (2,1)-surface médiane. Finalement, la fonction alpha complète est appliquée pour la figure 3.42(d), et le squelette qui en résulte est basé sur la (2,0)-surface médiane.

On retrouve ainsi dans toutes les dimensions une généricité sur la forme de la surface médiane, comme en 2D avec MB/MB2. En 3D, cette propriété peut servir à une représentation polyvalente de certains objets complexes, tels que les ventricules en imagerie médicale.

L'algorithme est entièrement parallèle, les points sont enlevés dans les 2n directions cardinales en même temps. Le nombre total d'itérations est donc égal au rayon r de la plus grosse (n-1)-boule contenue dans l'image. Le nombre total d'OE que nécessite le calcul du squelette MB-nD est de $(28 \times r)$ OE en 2D, et $(148 \times r)$ OE en 3D.

3.10 Conclusion

Les algorithmes proposés dans ce chapitre montrent à notre sens les vertus des contraintes imposées sur l'algorithmique par une architecture telle que la rétine programmable et ses problèmes de pénurie de mémoire. Nous avons en effet, par un travail de minimisation logique identique à notre démarche du précédent chapitre, proposé une procédure qui a mené à l'implantation sur notre architecture d'un algorithme de squelettisation qui est certainement l'une des plus efficaces qui existent.

Précisons le qualificatif d'efficacité utilisé ici. Pour comparer la vitesse d'éxécution de deux algorithmes au parallélisme de données, on estime comme nous l'avons fait le nombre total d'opérations élémentaires par pixel que nécessite le calcul. Mais peut-on alors comparer deux algorithmes sur des architectures différentes, puisque la nature de ces opérations élémentaires peut alors être aussi différente ? Nous prétendons que la mesure que nous utilisons pour notre complexité en temps, qui est le nombre total d'opérations booléennes de deux variables, est d'une part une mesure générale, puisque ces opérations sont les plus élémentaires qu'on puisse effectuer dans un algorithme numérique, et d'autre part une mesure qui a du sens pour n'importe quelle architecture. En effet, pour tout algorithme numérique, le coût de chaque opération booléenne élémentaire se retrouve, soit à un niveau logiciel, où elle influe sur le temps de calcul, soit à un niveau matériel, sous la forme d'une porte logique à deux entrées. Sur notre architecture, où les seules opérations cablées sont les opérations logiques à deux variables, le nombre d'OE est une mesure du temps de calcul; sur toute architecture en général, c'est une mesure de complexité «Surface $\times$ Temps». L'efficacité qu'amène la minimisation logique est par conséquent liée à une mesure d'énergie, où le quantum correspond à l'énergie consommée par une opération entre deux bits. Pour résumer, le nombre d'opérations booléennes élémentaires *total* (resp. *par pixel*) permet de comparer l'efficacité d'un calcul sur deux architectures quelconques (resp. SIMD).

On peut alors se demander pourquoi le problème de minimisation logique est aussi largement ignoré dans l'abondante littérature consacrée aux squelettes. La réponse est liée à la puissance des ordinateurs actuels. Comme l'explique Lam [55], le problème du temps de calcul n'est plus vraiment à l'ordre du jour pour les squelettes 2D, où l'effort s'est plus porté ces dernières années sur la préservation de la géométrie. Pour donner un ordre d'idée, la machine qui sert à la rédaction de ce rapport, PC de bureautique standard, est pourvue d'un microprocesseur dont la mémoire cache s'élève à 512 Ko. On peut par conséquent y «tabuler» toute fonction booléenne (c'est-à-dire entrer la liste de tous les n-uplets de valeurs booléennes correspondant à la valeur 1

ou 0 de la fonction), de support inférieur ou égal à 22 ! On appelle ce type de liste des *look-up tables* (LUT). Dans ce cas, la minimisation logique n'a pas d'intérêt, on s'efforce simplement de limiter la taille de la LUT pour accélérer son examen, d'où l'intérêt porté par certains auteurs à la minimisation du support.

Mais il ne faut pas en conclure que la concision logique de notre algorithme ne présente pas d'intérêt en dehors de la rétine programmable. En effet ce déséquilibre entre puissance disponible et puissance requise pour le calcul du squelette 2D est pratiquement inversé dans le cas de la 3D. A titre d'exemple, on peut citer les travaux de minimisation par BDD pour le calcul de la simplicité en 3D [96]. Le support étant alors supérieur ou égal à 27, le recours à une simple LUT présente un coût exorbitant. Le principe est alors de séparer le calcul en sous-expressions correspondant aux «alternatives» des BDD (voir chapitre précédent), récursivement jusqu'à ce que le calcul «rentre» dans la mémoire cache de la machine. En poussant ces techniques à l'extrême, on retrouve notre démarche de minimisation systématique des expressions logiques. Il est légitime de penser que le gain que représente MB en terme de concision logique en 2D se retrouve en 3D, et que par conséquent MB-3D puisse se prêter à une implantation très efficace sur une architecture standard.

Une étude comparative précise sur l'efficacité de MB-3D comme celle que nous avons menée en 2D, reste à faire (mais sort largement du cadre de notre recherche). Cependant, les algorithmes d'amincissement 3D disponibles dans la littérature possèdent des caractérisations plutôt compliquées, avec un grand nombre de configurations de retrait ou d'exception [61], [62], [102], [88], ou des règles particulières pour éviter la déconnexion due au retrait parallèle [42]. Dans ce cadre, MB-3D présente en tout état de cause l'avantage d'une grande simplicité d'implantation due à la compacité de sa caractérisation.

Finalement il faut souligner que c'est la recherche de la concision logique, qui, en dépouillant à l'extrême les expressions booléennes des différentes contraintes, a permis une définition suffisament pure d'une procédure d'amincissement, pour que sa généricité apparaissent naturellement, aussi bien au niveau de la polyvalence liée à la forme de la surface médiane que de l'expression n-dimensionnelle. Notons que MB-nD est à notre connaissance le premier algorithme de squelettisation proposé qui soit valide à la fois en 2D et en 3D. De futures recherches en topologie discrète dans les mailles de dimension supérieure permettront peut-être d'en savoir plus sur sa validité générale, puisque l'algorithme en nD est déjà disponible. Plus largement, ce travail peut suggérer des pistes de recherches. Nous songeons en particulier à notre expression unifiée des théorèmes de simplicité (théorèmes 7 et 8). En les présentant en nD [66], nous souhaitions soulever la question de leur

validité pour $n \geq 4$. Nous savons à présent que le théorème 7 n'est pas vrai pour tout n, mais ceci n'empêche *a priori* ni la validité du théorème 8, ni celle de l'algorithme MB-nD...

Chapitre 4

Opérateurs géodésiques

4.1 Introduction

Ce chapitre constitue la deuxième partie de l'étude sur l'implantation dans la rétine programmable d'algorithmes de morphologie mathématique. Comme dans le chapitre précédent, les problèmes de topologie discrète restent au cœur de nos préoccupations, mais nous soulevons en outre un problème clef pour notre architecture cellulaire, celui des opérations de base ayant un comportement global.

Nous entendons par «opération de base» une fonction intervenant de manière intensive dans le calcul d'un grand nombre d'algorithmes complexes, au même titre que la dilatation ou le calcul d'un contour binaire, par exemple. Le problème de l'efficacité de son calcul est par conséquent tout à fait fondamental.

L'opération qui se trouve au cœur de nos préoccupations dans ce chapitre est la *reconstruction géodésique*. Nous regroupons l'ensemble des algorithmes qui l'utilisent comme opération de base sous la dénomination «opérateurs géodésiques». Ces algorithmes sont caractérisés par une topologie qui n'est pas fixe, mais qui est déterminée par une image de référence. Nous montrons l'importance de la fonction de reconstruction, exposons les méthodes algorithmiques développées et concluons sur l'intérêt de consacrer des structures architecturales dédiées à son calcul.

4.2 Définitions et notations

Dans un espace métrique (E, δ), la géodésie sur un ensemble $X \subset E$ est liée à la notion de plus court chemin dans X. Pour $(x, y) \in E^2$, on note $\Gamma(x, y)$ l'ensemble des arcs reliant x et y dans E. Pour $\gamma \in \Gamma(x, y)$, on note

$\mathcal{L}(\gamma)$ la longueur de l'arc γ relativement à δ.

Définition 31 *Soit $X \subset E$. Pour tout $(x, y) \in X^2$, la* distance géodésique *de x à y dans X est définie par :*

$$\delta_X(x, y) = \inf\{\mathcal{L}(\gamma); \gamma \in \Gamma(x, y), \gamma \subset X\}.$$

Un arc $\gamma \in \Gamma(x, y)$ est une *géodésique* de x à y dans X si $\mathcal{L}(\gamma) = \delta_X(x, y)$.

Définition 32 (Mesures géodésiques [57]) *Soit X un ensemble connexe.* La fonction de propagation *sur X est définie, pour tout $x \in X$, par :*

$$\Pi_X(x) = \sup_{y \in X} \delta_X(x, y).$$

Le diamètre géodésique *de X est défini par :*

$$\Delta_X = \sup_{(x,y) \in X^2} \delta_X(x, y).$$

Définition 33 *La* boule géodésique *dans X, de centre $x \in X$ et de rayon r est définie par :*

$$\mathcal{B}_X(x, r) = \{y \in X; \delta_X(x, y) \leq r\}.$$

Définition 34 *Soit X et Y deux sous-ensembles de E. La* dilatation géodésique *de X dans Y, de rayon r, est l'ensemble :*

$$d_Y^r(X) = \{y \in Y; X \cap \mathcal{B}_Y(y, r) \neq \emptyset\}.$$

L'opération géodésique de base pour les algorithmes de ce chapitre est la *reconstruction géodésique*, définie de la manière suivante :

Définition 35 La reconstruction géodésique *de X dans Y est l'ensemble :*

$$R_Y(X) = \bigcup_{r>0} d_Y^r(X).$$

On notera en outre :

$$E_Y(X) = R_{Y \cup X}(X)$$

Le principe de cet opérateur est illustré sur la figure 4.1. Notons que si $X \subset Y$, alors $E_Y(X) = R_Y(X)$.

4.3 Premières applications

4.3.1 Calcul sur la rétine

Dans la rétine, on utilise une distance de la maille carrée δ_K, avec $K = 4$ ou 8. Si l'on note $B_K = \mathcal{B}_K(O, 1)$, la boule de rayon 1 pour la distance δ_K centrée sur l'origine, alors, si X et Y sont deux images binaires, la dilatation géodésique élémentaire (de rayon 1) de X dans Y est donnée par :

$$d_Y^1(X) = (X \oplus B_K) \cap Y.$$

De plus, pour $n > 1$, $d_Y^n(X) = (d_Y^{n-1}(X) \oplus B_K) \cap Y$. Cela nous amène au calcul dans la rétine. L'algorithme est effectué sur 3 plans mémoire. Le

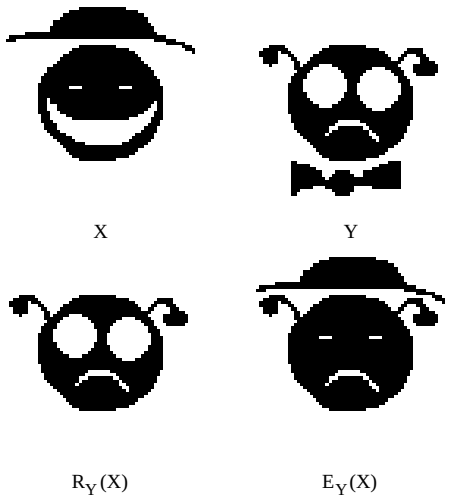


FIG. 4.1 – *Les opérateurs géodésiques de base.*

$p_0 = X$;
$p_1 = Y$;
répéter {
$(p_0, p_2) = (dilate_K(p_0), p_0)$;
$p_0 = p_0 \wedge p_1$;
$p_2 = p_2 \bar{\rightarrow} p0$;
} tant que $p_2 \neq \emptyset$;

TAB. 4.1 – *Calcul de la reconstruction géodésique en K-connexité.*

tableau 4.1 explicite le calcul de la reconstruction géodésique de X dans Y (le résultat du calcul est dans le plan p_0).

Si la dilatation nécessite n opérations, la complexité maximum de la reconstruction géodésique est $(n + 2) \times \Delta$, Δ étant le diamètre géodésique maximal de l'image de départ.

4.3.2 Comptage de composantes connexes

On peut calculer très efficacement le nombre d'Euler-Poincaré d'une image binaire, soit le nombre de ses composantes connexes diminué du nombre de trous (voir Annexe A).

Pour calculer le nombre de composantes connexes d'une image, il suffit de boucher ses trous. On va alors faire la *reconstruction géodésique* d'un

$p_0 = X^c$;
$p_1 = M$;
$(p_0, p_1, p_2) = (p_0, recons_K(p_1, p_0), \tilde{p_2})$;
$p_2 = p_1 \overrightarrow{\rightarrow} p_0$;

TAB. 4.2 – *Le bouchage de trous pour compter les K-composantes connexes.*

ensemble inclus dans la composante infinie du complémentaire. Les parties du complémentaire non reconstruites sont les trous. Le tableau 4.2 explicite l'algorithme de comptage des K-composantes connexes de X (où la reconstruction géodésique du plan p_1 dans le plan p_0 est utilisée comme une macro-instruction qui utilise trois plans binaires, sans modifier le plan p_0). Le plan p_1 est initialisé à une valeur de marqueur M obtenu en mettant les pixels à 1 sur le bord de l'image.

Remarquons que l'intérieur des trous est conservé dans le plan p_2, pour tester s'il n'y a pas de composante connexe contenue dans le trou d'une autre, et éventuellement continuer le calcul. La complexité du calcul est pratiquement la même que celle d'une reconstruction géodésique, proportionnelle à Δ', diamètre géodésique de X^c.

4.3.3 Seuillage par hystérésis

Une application simple et utile de la reconstruction géodésique est le seuillage par hystérésis. Soit $I : \mathbb{Z}^2 \longrightarrow \mathbb{N}$ une image en niveaux de gris. Soient deux entiers a et b tels que $0 < a < b$. Le seuillage de niveau $n \in \mathbb{N}$ est l'image binaire :

$$I_n = \{z \in \mathbb{Z}^2; I(z) \geq n\}.$$

Le seuillage par hystérésis de I sur $[a, b]$ est l'image binaire suivante :

$$I_a^b = R_{I_a}(I_b).$$

La figure 4.2 illustre ce procédé de seuillage.

4.4 Maxima régionaux. R-h maxima

4.4.1 Définitions

Les opérateurs géodésiques sont liés à la notion de maxima régionaux. On a vu dans le premier chapitre la définition des maxima locaux, qui pouvaient être calculés à partir d'une simple ouverture. Le calcul des maxima régionaux nécessite en revanche la connaissance d'un voisinage non défini *a priori* et

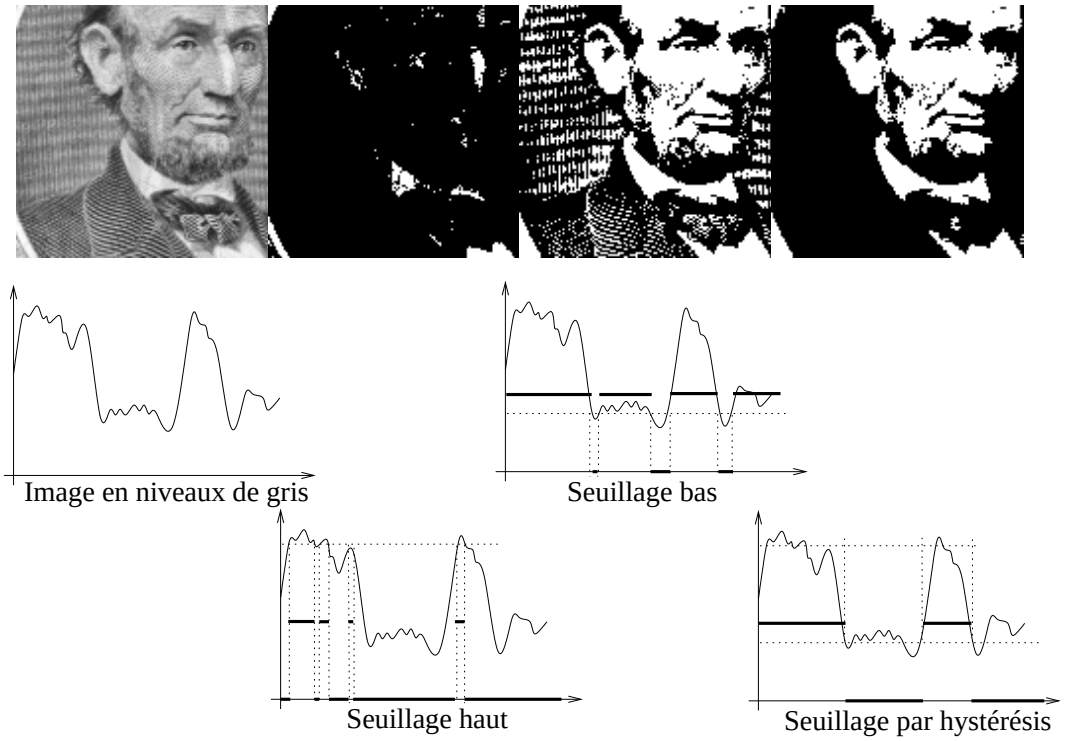


FIG. 4.2 – *Seuillage par hystérésis.*

entraîne une certaine propagation dans l'image, qui sera effectuée par la reconstruction géodésique.

Définition 36 (Maxima régionaux) *Soit $X \subset \mathbb{Z}^2$. Soit $\Phi : X \longrightarrow \mathbb{N}$ une fonction numérique sur X. Si $n \in \mathbb{N}$, on note X_n l'image binaire définie par :*

$$X_n = \{x \in X; \Phi(X) \geq n\}.$$

L'ensemble des Maxima régionaux *de la fonction Φ sur X est alors défini par:*

$$MR_\Phi(X) = \bigcup_{n\in\mathbb{N}} X_n \setminus R_{X_n}(X_{n+1}).$$

Dire que x est un maximum régional pour x signifie qu'on ne peut pas atteindre un point strictement plus grand à partir de x sans passer par un point strictement plus petit (pour l'ordre induit par Φ). Cette notion a été généralisée dans [105] par les *r-h maxima* : On dira que x est un r-h maximum pour Φ s'il domine tous les points se trouvant à une distance inférieure à r de lui, d'une hauteur au moins égale à h, soit en reprenant les notations de la définition 36 :

Définition 37 (r-h maxima)

$$Max_\Phi^{(r,h)}(X) = \bigcup_{n\in\mathbb{N}} X_n \setminus d_{X_n}^r(X_{n+h}).$$

Notons que $MR_\Phi(X) = Max_\Phi^{(\infty,1)}(X)$.

Formes binaires/numériques : Les images de maxima régionaux et de r-h maxima introduites dans les définitions précédentes sont des images binaires. On peut leur associer une forme en niveau de gris en utilisant comme définition :

$$Max_\Phi^{(r,h)}(X) = \sum_{n\in\mathbb{N}} n \times (X_n \setminus d_{X_n}^r(X_{n+h}))$$

Exemples fondamentaux : La fonction Φ peut représenter le niveau de gris d'une image numérique $I : \mathbb{Z}^2 \longrightarrow \mathbb{N}$. Dans ce cas l'image I_n de la définition 36 est simplement la coupe $I^{-1}([n, +\infty[)$ (voir figure 4.3). Dans le cas d'une image binaire, la fonction Φ peut représenter une fonction distance $x \mapsto d(x, I^c)$. Dans ce cas, si B est la boule unité de la distance d, $I_n = I \ominus nB$, et donc :

$$MR_d(I) = \bigcup_{n\in\mathbb{N}} ((I \ominus nB) \setminus R_{(I\ominus nB)}(I \ominus (n+1)B)).$$

L'ensemble des maxima régionaux de la fonction distance coïncide avec l'ensemble des composantes connexes qui disparaissent au cours d'érosions réitérées par la boule unité pour cette distance, ensemble appelé les *érodés ultimes*.

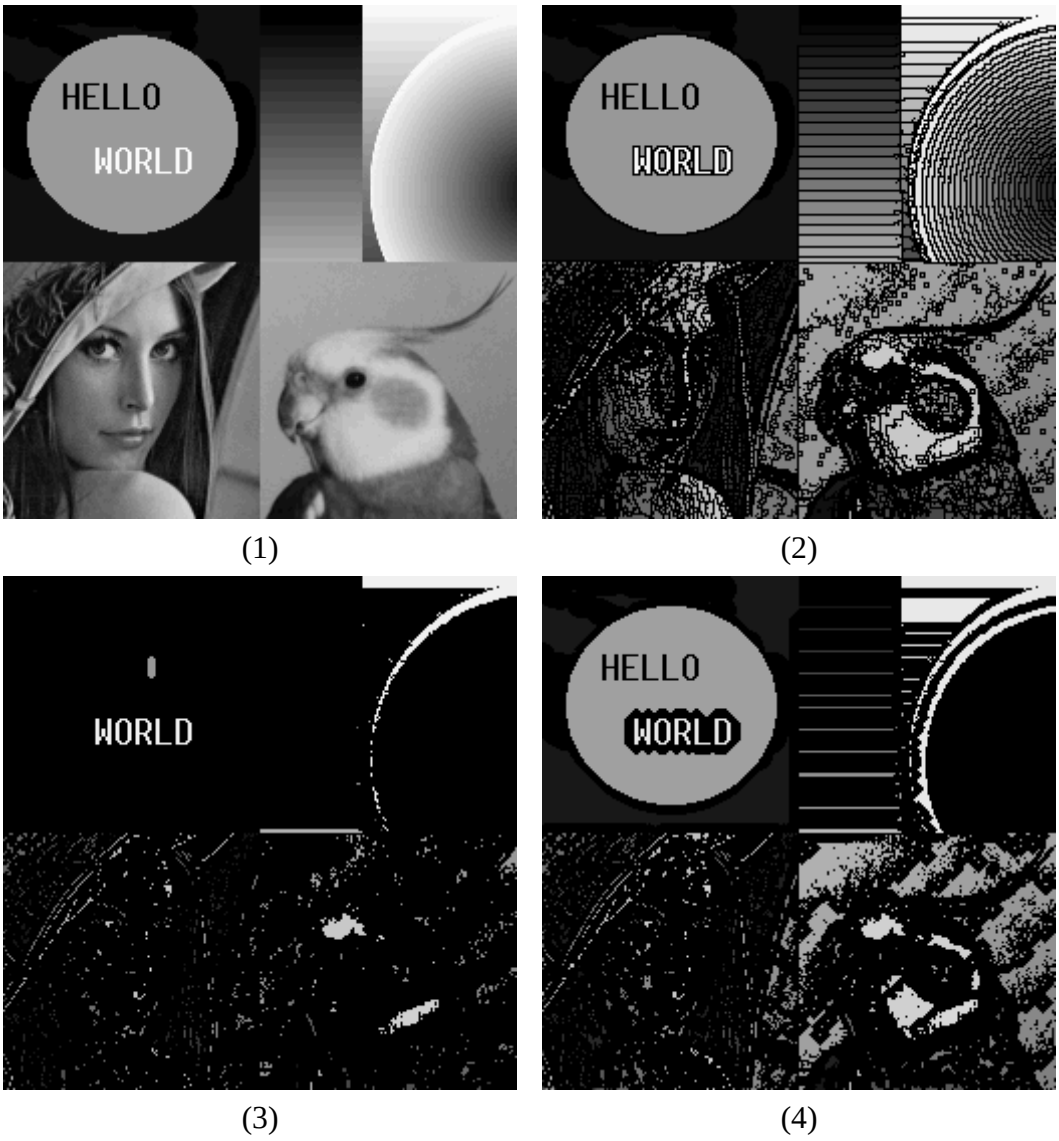


FIG. 4.3 – *Les images de maxima pour une image en niveaux de gris. (1) Image originale. (2) Maxima locaux. (3) Maxima régionaux. (4) r-h maxima, avec r=5 et h=3.*

4.4.2 Séparation de particules par érosion ultime

Une application classique de l'érosion ultime est la séparation de particules plus ou moins interpénétrées par singularisation de composantes connexes. En maille carrée, cependant, l'érosion réitérée par la boule unité de la 4-connexité ou de la 8-connexité ne sépare pas correctement en général les particules; en effet le caractère non euclidien de la distance choisie se ressent de plus en plus lorsque le nombre d'itérations augmente !

On obtient néanmoins facilement une distance plus proche de la distance euclidienne grâce à la propriété d'associativité de la dilatation. En alternant dilatation (resp. érosion) par la boule unité de la 4-connexité et celle de la 8-connexité, on obtient une dilatation (resp. érosion) par une boule octogonale de taille croissante.

Précisons un peu l'importance du choix de la distance discrète et donc de l'élément structurant à utiliser. Les distances canoniques de la trame carrée induisent une anisotropie très importante dans les algorithmes à base de fonction distance, ce que nous cherchons à éviter.

Soit B_4 (resp. B_8) la boule unité de la 4-connexité (resp. 8-connexité).

$d_{conex}(x,y) = \min\{n, y \in \{x\} \oplus nB_{conex}\}$, avec $conex = 4$ ou 8.

Ces deux distances présentent un écart par rapport à la distance euclidienne qui est maximun pour les directions $(2p+1)\frac{\pi}{4}, p \in \mathbb{Z}$ (voir figure 4.4). Si d désigne la distance euclidienne, x et y deux éléments de $\mathbb{Z}^2$, on a les inégalités suivantes :

$$d(x,y) \leq d_4(x,y) \leq \sqrt{2}d(x,y),$$
$$d_8(x,y) \leq d(x,y) \leq \sqrt{2}d_8(x,y).$$

La distance octogonale est définie par :

$$d_{oct}(x,y) = \min\{n, y \in \{x\} \oplus \lceil n/2 \rceil B_8 \oplus \lfloor n/2 \rfloor B_4\}.$$

Cette fois on a un écart maximum pour les directions $\{(3p+1)\frac{\pi}{6}, (3p+2)\frac{\pi}{6}\}, p \in \mathbb{Z}$ (figure 4.4), et on a l'inégalité :

$$d_{oct}(x,y) \leq d(x,y) \leq \frac{\sqrt{5}}{2} d_{oct}(x,y).$$

La figure 4.5 illustre les différences de comportement d'un algorithme d'érosion ultime en fonction de la distance discrète. Dans cet exemple, c'est seulement dans le cas de la distance octogonale qu'on obtient une composante connexe par particule.

Calcul sur la rétine

L'algorithme est effectué sur 4 plans-mémoire dans la rétine, par la procédure du tableau 4.3. Le résultat est calculé dans le plan p_3. Le nombre d'opérations maximum vaut $5d(\Delta + 2)$.

avec : d : diamètre de la boule maximale incluse dans l'image,
Δ : diamètre géodésique maximal de l'image.

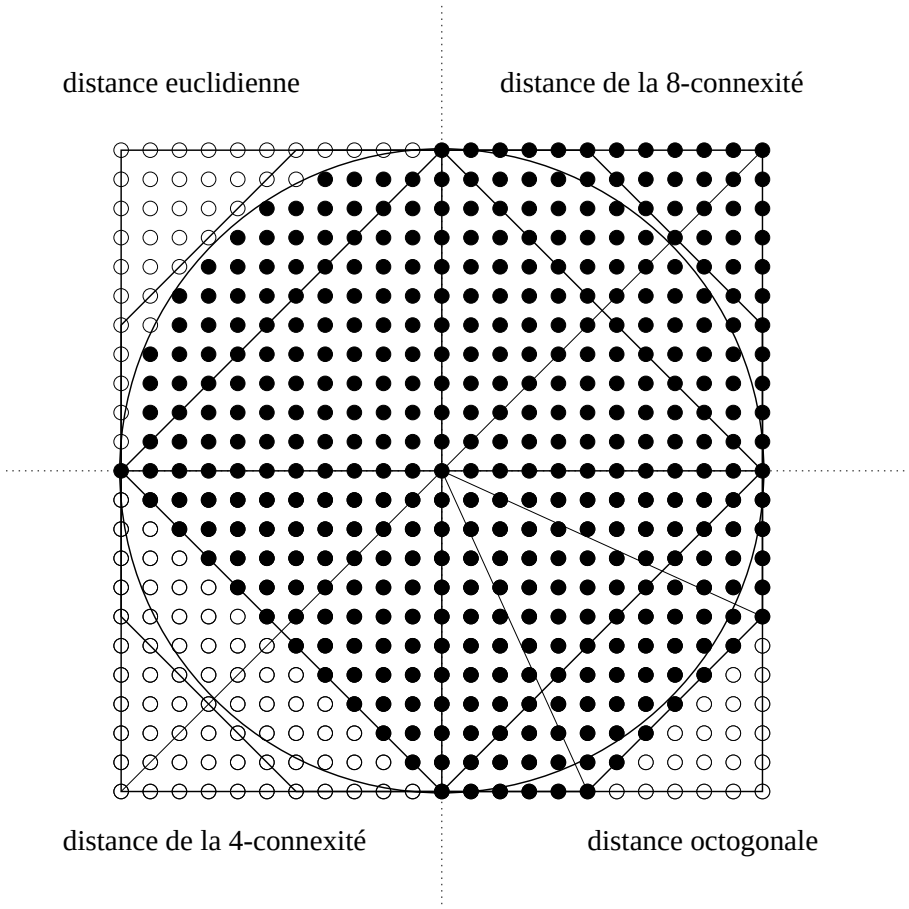


FIG. 4.4 – *Différentes boules discrètes.*

4.5 Filtrage non linéaire

4.5.1 Généralités

Un *filtre morphologique* binaire Ψ est défini comme un opérateur à la fois *croissant* et *idempotent*, i.e. pour toute image X et Y :

$$X \subset Y \Rightarrow \Psi(X) \subset \Psi(Y),$$
$$\Psi(\Psi(X)) = \Psi(X).$$

De même si I et J sont deux fonctions $\mathbb{Z}^2 \longrightarrow \mathbb{N}$, un filtre morphologique numérique est défini en remplaçant la première équation par :

$$I \leq J \Rightarrow \Psi(I) \leq \Psi(J).$$

Propriété 4 *Si I est une image numérique définie par ses n coupes avec*

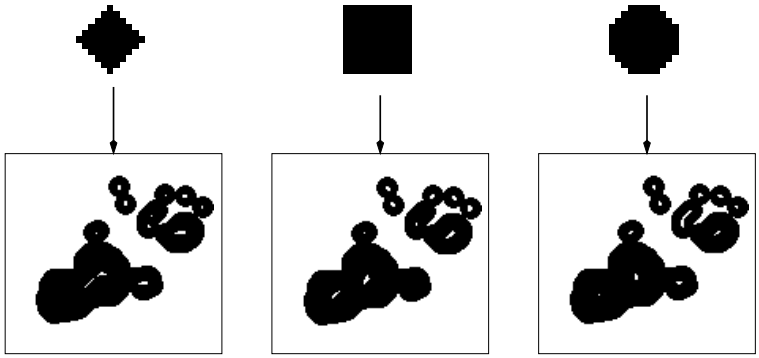


FIG. 4.5 – *Séparation de particules par érosion ultime : sensibilité par rapport au choix de la distance discrète. On a représenté en haut des boules de même rayon pour les différentes distances utilisées : de gauche à droite, 4-distance, 8-distance, distance octogonale. L'image des érodés ultimes est superposée (en blanc) sur l'image originale.*

$p_0 = image$;
répéter {
$p_1 = p_0$;
$(p_0, p_2) = (erode_K(p_0), \tilde{p_2})$;
$(p_0, p_1, p_2) = (reconst_K(p_0, p_1), p_1, \tilde{p_2})$;
$p_2 = p_1 \bar{\rightarrow} p_0$;
$p_3 = p_3 \vee p_2$;
} tant que $(p_0 \neq \emptyset)$;

TAB. 4.3 – *Procédure de calcul des maxima régionaux de la fonction distance (érodés ultimes).*

$I = \sum_{i=1}^{n} I_i$, *si* Ψ *est un filtre morphologique binaire, alors l'opérateur* $\hat{\Psi}$ *défini par* $\hat{\Psi}(I) = \sum_{i=1}^{n} \Psi(I_i)$ *est un filtre morphologique numérique.*

Cette propriété montre l'adéquation du filtrage morphologique pour la rétine programmable. Grâce à elle, il suffit de définir les différents filtres présentés pour des images binaires. Ces opérateurs sont très utilisés dans les applications que nous avons développées, comme illustré au chapitre suivant.

Dans la littérature, la propriété 4 correspond à la définition des *stack filters* [67], notion plus générale, Ψ étant alors une forme disjonctive positive (i.e. sans négation).

Les filtres morphologiques les plus simples sont les ouvertures et fermetures, déja rencontrées dans les chapitres précédents, définies respectivement par $\gamma_B(I) = (I \ominus \check{B}) \oplus B$ et $\phi_B(I) = (I \oplus \check{B}) \ominus B$. Elles sont respectivement anti-extensive ($\gamma_B(I) \subset I$) et extensive ($I \subset \phi_B(I)$).

Notons que nous réserverons le terme ouverture (resp. fermeture) à l'opérateur défini plus haut, alors qu'en Morphologie, ce terme qualifie en général tout filtre anti-extensif (resp. extensif), couvrant ainsi une large variété de filtres, par exemple les opérateurs «rectangles/octogones englobants» présentés dans l'annexe A, qui sont des filtres morphologiques par relaxation.

4.5.2 Ouvertures par reconstruction

Comme on le voit sur la figure 4.6 (images (1a) et (1c), (2a) et (2c)) l'effet d'une ouverture (resp. fermeture) est une simplification de l'image par élimination des composantes connexes (resp. des trous) qui ne contiennent pas l'élément structurant B. Les contours de l'image sont également modifiés, en fonction de la forme de l'élément structurant. Celle-ci pervertit donc l'aspect de l'image, ce qui fait des ouvertures/fermetures des outils peu adaptés par exemple à la restauration d'images bruitées.

La reconstruction géodésique permet pourtant d'utiliser les ouvertures/fermetures pour un filtrage plus élaboré, grâce aux opérateurs de la définition 38 [108].

Définition 38 *Soient* I *et* B *deux parties de* $\mathbb{Z}^2$.
L'ouverture (resp. fermeture) par reconstruction de I *par* B *est l'ensemble :*

$$R_I(\gamma_B(I)) \text{ (resp. } [R_{I^c}((\phi_B(I))^c)]^c).$$

Cette fois, l'élément structurant ne fait que discriminer les composantes connexes (ou trous) par leur forme et taille, mais sans modifier les contours des composantes connexes (ou trous) qui sont conservés (voir figure 4.6, images (1b) et (1d), (2b) et (2d)).

4.5.3 Nivellements

L'effet des opérateurs précédents pour la restauration est clairement visible (images (1d) et (2d)). On note cependant une asymétrie dans leur action, peu satisfaisante dans un tel cadre : l'ouverture par reconstruction élimine les petites composantes connexes, mais conserve les petits trous, et vice-versa pour la fermeture par reconstruction. Cela est dû à la contrainte d'ordre $\gamma_B(I) \subset I \subset \phi_B(I)$. Un comportement plus symétrique étant souhaitable, on a recours à une généralisation des opérateurs précédents par la notion de *nivellement* [77], définie à partir d'une image de «marqueur» quelconque :

Définition 39 *Soit I et M deux parties de $\mathbb{Z}^2$. Le nivellement de I à partir de M est l'image :*

$$[E_{I^c}((E_I(M))^c)]^c.$$

La définition 39 n'a d'intérêt que si l'on n'a ni $I \subset M$, ni $M \subset I$. Typiquement, M est une simplification de I, soit par un filtre morphologique mixte tel qu'un filtre alterné séquentiel [107], soit par un pré-filtre tel que l'opérateur majoritaire, c'est-à-dire la fonction seuil $T_n^{\lceil n/2 \rceil}$ (voir chapitre 2) et son pendant numérique le «filtre» médian. C'est ce dernier qui est utilisé sur la figure 4.6 (images (3a-b-c-d)) : à partir de l'image «majoritaire» (3a) qui joue ici le rôle du marqueur M, on effectue successivement la reconstruction géodésique de M dans $I \cup M$ où I est l'image originale (0) puis la reconstruction duale, c'est-à-dire la reconstruction géodésique du complémentaire de (3a) dans le complémentaire de (0), complémentée. On voit que l'image obtenue (3b) respecte les contours de (0), tout en éliminant à la fois petites composantes connexes et petits trous.

4.6 Ligne de partage des eaux

Cette partie traite du problème de la segmentation d'un signal bidimensionnel relativement à certaines variations. Ces variations peuvent être celles du signal lui-même, ou être d'un ordre différent. La *détection des contours* constitue la base de la *représentation* d'une image qui permettra d'aller vers une description évoluée de la scène [68]. Classiquement, les contours sont obtenus à partir des points de passage par zéro de la dérivée seconde (laplacien) du signal [69], ou bien en sélectionnant les maxima locaux de la norme de la dérivée première (gradient), dans la direction du gradient [24]. La segmentation est ensuite complétée par des opérations d'amincissement, d'élimination des petits contours et de fermeture de contours.

L'application brute de ces méthodes montre qu'on n'obtient pas, en général, une bonne segmentation en se limitant à des calculs locaux, et qu'il faut donc

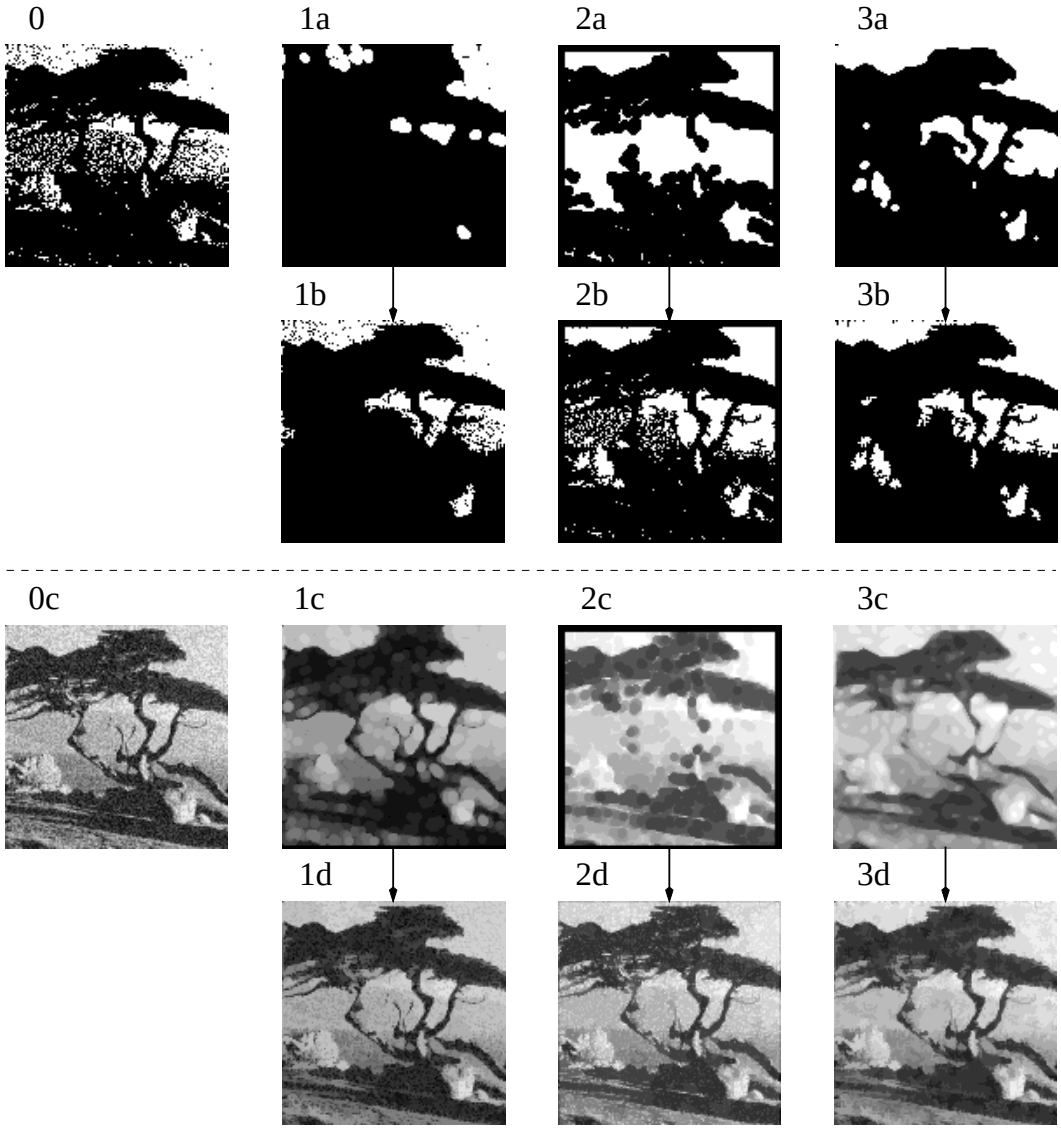


FIG. 4.6 – *Filtrage non linéaire. (i) Morphologique. En binaire : (0) Image originale. (1a) Ouverture par un octogone et (1b) sa reconstruction géodésique dans l'image originale. (2a) Fermeture par un octogone et (2b) sa reconstruction duale. En niveaux de gris : (0c) Image originale (bruitée). (1c) Ouverture. (1d) Ouverture par reconstruction. (2c) Fermeture. (2d) Fermeture par reconstruction. (ii) Non morphologique (non idempotent). En binaire : (3a) Opérateur majorité et (3b) reconstruction géodésique suivie d'une reconstruction duale. En niveaux de gris : (3c) «Filtre» (Pré-filtre morphologique) médian et (3d) nivellement du médian.*

faire intervenir des interactions globales. C'est précisément ce que fait Canny [24], lorsqu'il obtient, à partir des maxima locaux du gradient, l'image de contour en éliminant les points de faible gradient, grâce à un *seuillage par hystérésis*. Ainsi, un point de faible gradient peut-il appartenir à un contour pour peu qu'il «se trouve sur la même courbe» que des maxima de fort gradient... L'approche de la segmentation par *Ligne de partage des eaux* (LPE) [18] exploite cette idée de suivi des «lignes de crêtes» de la fonction gradient. De plus, l'algorithme de LPE, de par sa définition, conduit à une segmentation complète, c'est-à-dire à un ensemble de courbes fermées d'épaisseur unité.

Nous décrivons ci-dessous le principe de la LPE et ses applications, puis nous proposons un algorithme original permettant l'implantation de la LPE sur la rétine numérique.

4.6.1 Principe

La notion de Ligne de partage des eaux [18] trouve, comme son nom l'indique, son origine en topographie, où elle désigne la frontière des différentes zones d'attraction des eaux de ruissellement sur un relief. On peut l'appliquer à toute fonction numérique $\Phi : \mathbb{Z}^2 \to \mathbb{N}$ que l'on considère alors comme une surface topographique. La figure 4.7 illustre le principe de la LPE sur un signal monodimensionnel, vu comme un profil topographique : en imaginant que les minima régionaux du signal sont percés, on immerge le profil progressivement dans l'eau (images (1) à (6)). A chaque fois que le niveau d'eau atteint un nouveau minimum régional (images (1) à (4)), un nouveau bassin versant est formé, qui grossit au fur et à mesure que l'eau monte. On interdit la fusion de différents bassins versants en créant une «digue» à chaque fois que deux bassins se rencontrent (images (4) à (6)). L'ensemble des points digues forme la LPE (3 points dans l'exemple de la figure 4.7).

Certes, l'illustration dans le cas monodimensionnel - à laquelle nous aurons plusieurs fois recours - peut conduire à s'interroger sur l'utilité du concept puisque, en 1D, la LPE coïncide avec les maxima régionaux (voir figure 4.7). Néanmoins, il suffit d'examiner la figure 4.8 pour constater que ce n'est plus le cas en 2D.

Pour le traitement d'images, la LPE constitue un outil générique de segmentation. On a parlé précédemment de la détection de contours, pour laquelle la fonction Φ représente l'intensité du gradient spatial. On citera aussi la séparation de particules dans les images binaires, en prenant pour Φ la fonction distance [117].

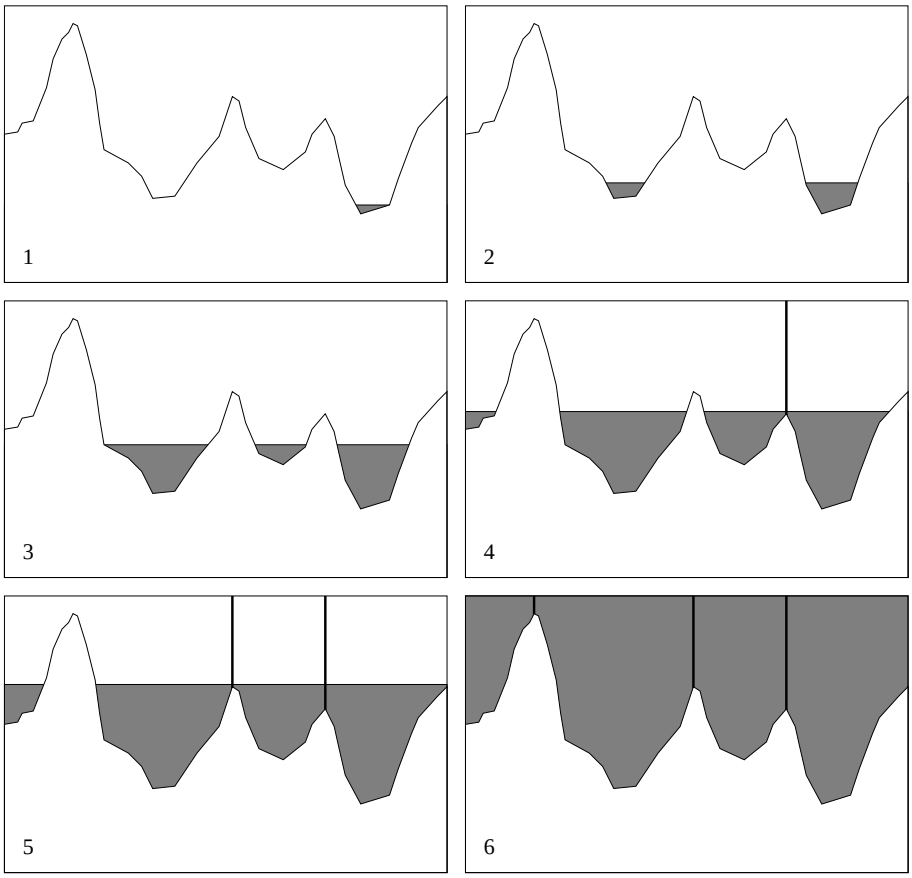


FIG. 4.7 – *Ligne de Partage des Eaux morphologique par immersion d'un signal monodimensionnel.*

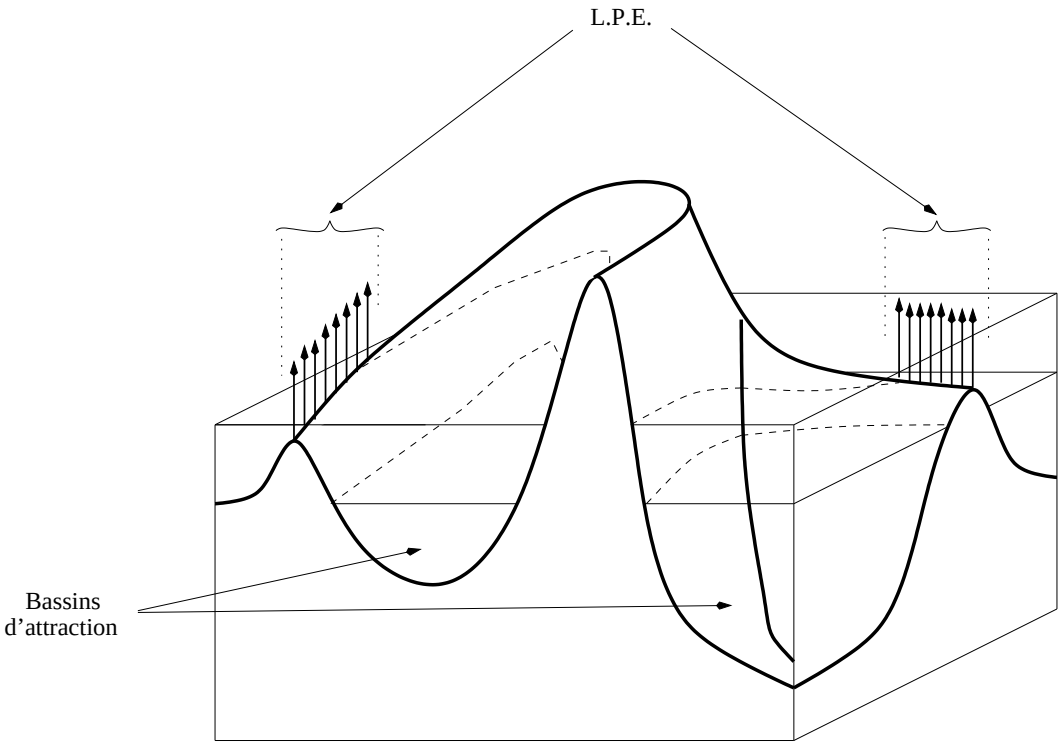


FIG. 4.8 – *LPE par immersion d'un signal bidimensionnel.*

4.6.2 Calcul sur la rétine

La figure 4.9 donne une représentation synoptique de l'algorithme de la LPE en 2D. La colonne (1) représente l'ensemble des coupes binaires de l'image originale (0). La colonne (2) est l'ensemble des maxima régionaux correspondant à chaque coupe (composantes connexes n'apparaissant pas à la coupe précédente). La colonne (3) montre la formation de la LPE : on dilate géodésiquement l'ensemble des maxima régionaux de niveau n dans la coupe de niveau $(n-1)$, en ajoutant à chaque itération l'ensemble des maxima régionaux correspondants; on produit ainsi un ensemble appelé *image des bassins versants*. Cet ensemble est constitué d'un certain nombre de composantes connexes, et au cours de leur croissance, certaines de ces composantes vont fusionner. On conserve alors tous les points de *collision* des composantes connexes pour former l'image de *LPE*.

Notons que l'algorithme ainsi présenté examine les niveaux de gris dans l'ordre décroissant, qui est l'ordre naturel de phototransduction (voir chapitre 1). Par conséquent, d'un point de vue topographique, ce sont plutôt les fonds de vallée que la ligne de partage des eaux qui sont détectés par

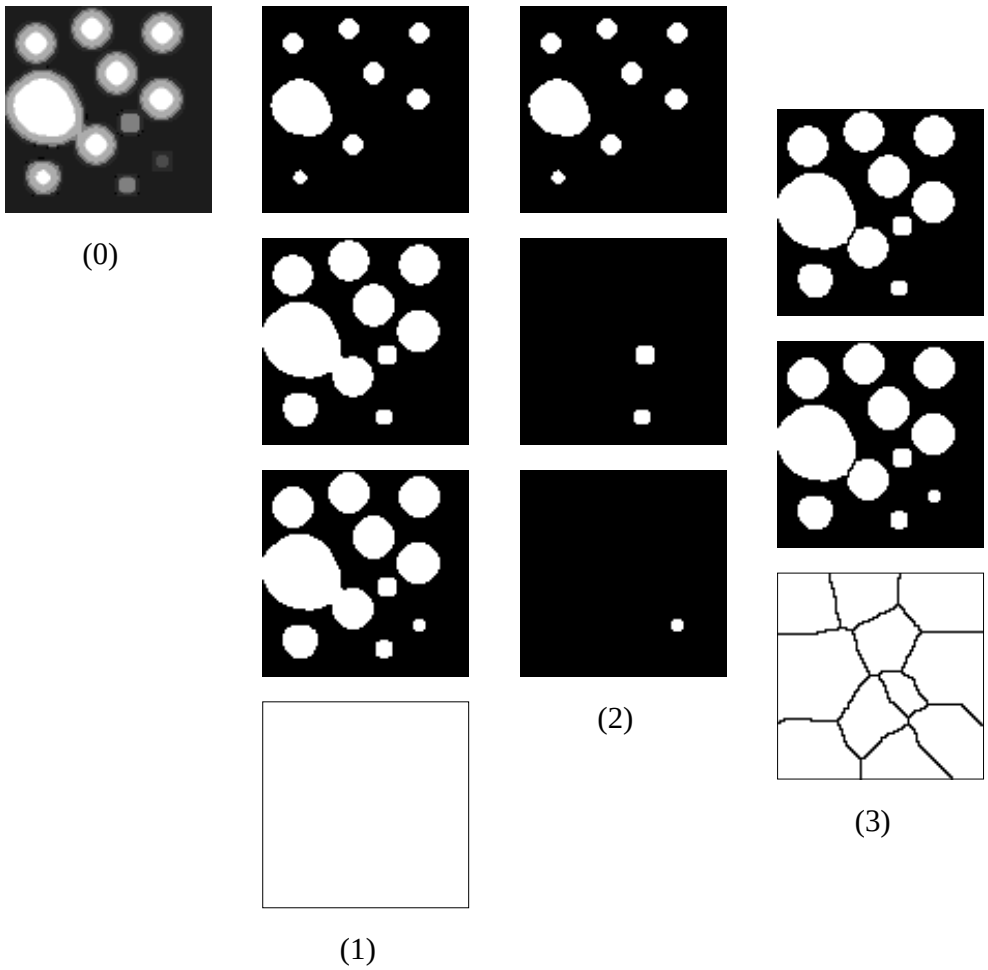


FIG. 4.9 – *Schéma de principe de la ligne de partage des eaux.*

l'algorithme. Mais cela n'est qu'un détail d'implantation lié au prototype de rétine (*Pvlsar2.2*) dont nous disposons pour l'expérimentation. En installant un comparateur analogique dans chaque pixel, ce qui se fait plus ou moins simplement [33], [121], on peut obtenir des coupes dans n'importe quel ordre, dont l'ordre croissant.

L'opérateur de base de la LPE consiste donc en une reconstruction géodésique où l'on interdit la fusion de composantes connexes. Cela revient à un *SKIZ géodésique*. Nous avons vu l'implantation du SKIZ dans le chapitre précédent, défini par une union de quatre TTR appliquées successivement et correspondant aux différentes directions de dilatation. La seule différence réside dans le fait de contraindre l'image à demeurer dans le plan de référence. Rappelons que les TTR utilisées se calculent sur 3 plans binaires. L'algo-

FIG. 4.10 – *Le SKIZ géodésique, opérateur de base de la LPE. (1) Image originale I. (2) Image de référence J. (3) SKIZ géodésique de I dans J. (4) Superposition symbolique.*

$p_0 = I[0]$;
pour $i = [1..n]$ {
$p_1 = I[i]$;
$(p_0, p_1, p_2) = (p_0, p_1, reconst_K(p_0, p_1))$;
$p_2 = p_1 \bar{\rightarrow} p_2$;
// Le plan p_2 contient les minima régionaux de niveaux i.
$p_0 = p_0 \vee p_2$;
$(p_0, p_1, p_2, p_3) = (SKIZGEO(p_0, p_1), p_1, \tilde{p_2}, \tilde{p_3})$;
}

TAB. 4.4 – *Procédure de calcul de la ligne de partage des eaux.*

rithme du SKIZ géodésique, avec le plan de référence, est donc effectué sur quatre plans binaires.

Il faut noter que nous ne calculons pas la «LPE vraie», mais une approximation, de la même façon que nous ne calculions pas le SKIZ vrai (voir chapitre précédent). Cette approximation est très classique et la différence est en général négligée. De plus, les courbes supplémentaires qui sont obtenues lorsqu'une composante connexe est contenue dans la concavité d'une autre sont très difficile à discriminer sur notre architecture. Il faudrait en effet procéder à une différenciation de toutes les composantes connexes, soit par étiquetage, ce qui est coûteux en espace, soit séquentiellement, ce qui est coûteux en temps.

Le tableau 4.4 présente la procédure de calcul tel qu'elle pourrait être implantée sur la rétine. Cet algorithme n'utilise que 4 registres binaires. Il est cependant inutilisable tel quel, comme le montre les résultats de la figure 4.11 : La LPE brute sur-segmente fortement les images. La puissance de cet outil de segmentation est rendue possible par la souplesse que lui confère le *paramétrage*. Nous allons voir par la suite le principe de ce paramétrage,

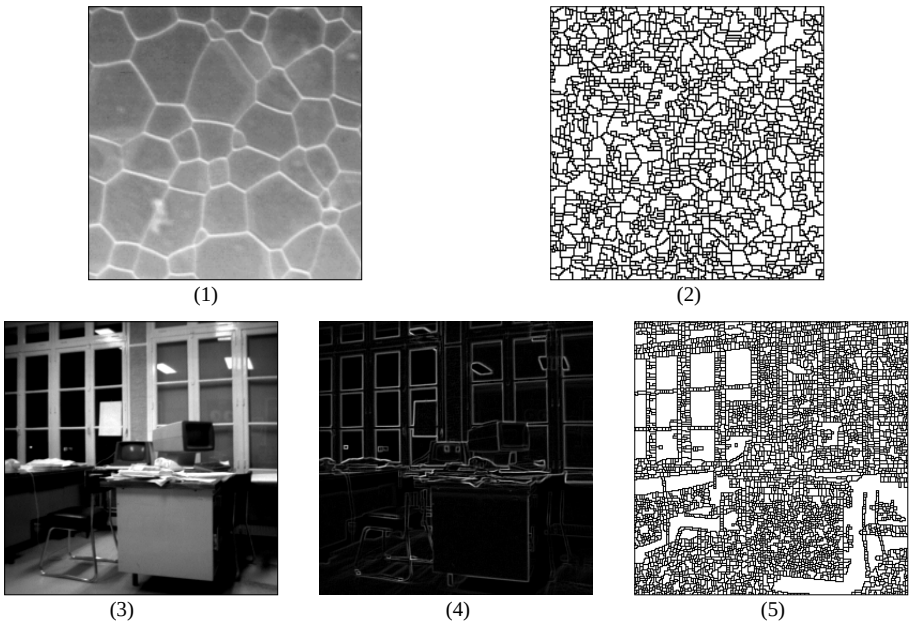


FIG. 4.11 – *L'application brute de la LPE conduit à une importante sur-segmentation. (1) Image microscopique de minerai d'uranium. (2) LPE de l'image 1. (3) Image du bureau. (4) Intensité du gradient morphologique de l'image 3. (5) LPE de l'image 4.*

et comment il est possible de le mettre en œuvre sur la rétine sans utiliser beaucoup de mémoire.

4.6.3 Le problème de la sur-segmentation

Nous analysons ici les causes de la sur-segmentation de la LPE. Si l'on considère un signal monodimensionnel (figures 4.12 et 4.13), deux facteurs différents peuvent être à l'origine de points «non significatifs» de la LPE :

- La présence de «pics» étroits sur le signal (figure 4.12(1)), mais de hauteur quelconque,
- La présence de petites oscillations sur le signal créant des bassins de faible profondeur (figure 4.13(1)), mais de largeur quelconque.

Le premier facteur est éliminé grâce au filtrage non linéaire (voir section 4.5). On parlera ici de *filtrage horizontal*. Il est clair qu'éliminer les petits pics du signal bidimensionnel I revient à éliminer les petites composantes connexes de chaque coupe binaire $I(i)$.

Soit ρ notre paramètre de filtrage horizontal, correspondant à la taille minimal des pics sur l'image I. On note B_ρ la boule de rayon ρ. Le filtrage

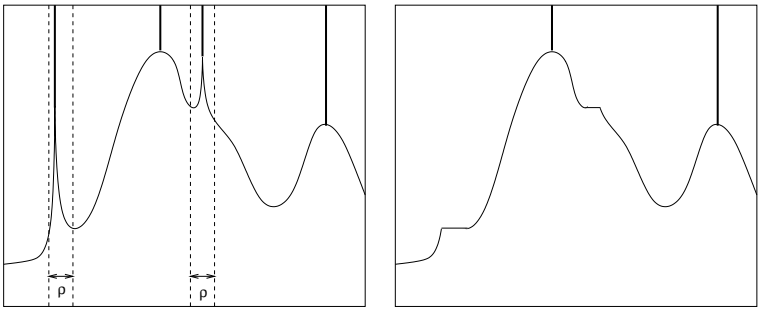


FIG. 4.12 – *Le premier facteur de sur-segmentation de la LPE : le bruit sur le signal (1). La solution consiste à imposer une taille minimale ρ aux pics du signal grâce à un filtrage non linéaire (2).*

horizontal consiste simplement à calculer l'ouverture par reconstruction de chaque coupe binaire $I(i)$, soit $R_{I(i)}(\gamma_{B_\rho}(I(i)))$.

L'élimination du deuxième facteur de sur-segmentation correspond à ce que nous appellerons le *filtrage vertical*. Sa mise en œuvre peut être explicitée en utilisant la notion de *dynamique* [104] ou de *coût de connexion* [92].

Dans notre topologie de référence (celle qui est définie implicitement par l'élément structurant utilisé pour les reconstructions géodésiques), on note, pour tout couple de points $x \in \mathbb{Z}^2$ et $y \in \mathbb{Z}^2$, $\Gamma(x, y)$ l'ensemble des arcs reliant x et y.

Définition 40 *Soient $x \in \mathbb{Z}^2$ et $y \in \mathbb{Z}^2$. Soit $I : \mathbb{Z}^2 \to \{0, ..., N\}$ une image numérique. On appelle* coût de connexion *de x à y dans I la quantité $C_I(x, y) \in \{0, ..., N\}$, définie par :*

$$C_I(x, y) = \min_{\gamma \in \Gamma(x,y)} \max_{z \in \gamma} I(z).$$

Le coût de connexion est une notion familière des cyclistes : d'un point de vue topographique, en considérant l'image I comme un relief, $C_I(x, y)$ correspond à l'altitude minimale qu'il faut atteindre pour relier les points x et y (figure 4.13(1)).

Si x est un minimum régional, sa *dynamique* est définie comme la hauteur (relativement à x) minimale qu'il faut atteindre pour accéder, à partir de x, à un point d'altitude strictement inférieur.

Le principe du filtrage vertical est d'ignorer les minima régionaux de faible dynamique. On se donne un paramètre h, correspondant à la hauteur minimum d'une digue par rapport aux fonds des bassins versants qu'elle sépare. On considère deux bassins versants adjacents, correspondant à des minima régionaux atteints pour x et y respectivement. Le principe est de ne

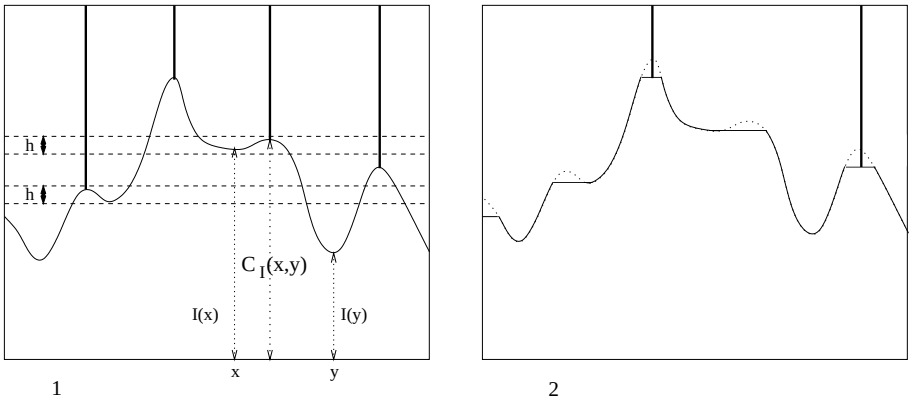


FIG. 4.13 – *Le deuxième facteur de sur-segmentation de la LPE: les faibles oscillations du signal créent des bassins de faible profondeur (1). Ces bassins correspondent à des minima régionaux de faible dynamique, qu'on peut supprimer en calculant la recontruction du signal I dans le signal $I + h$, où h est la dynamique minimale souhaitée (2).*

séparer effectivement ces deux bassins que si $(C_I(x,y) - Max(I(x), I(y))) > h$.

4.6.4 La LPE paramétrée

Venons-en maintenant au problème du calcul de la LPE paramétrée sur la rétine. Le filtrage horizontal ne pose pas de difficultés, il suffit d'effectuer une ouverture par reconstruction pour chaque niveau $I(i)$; cela n'augmente pas le nombre de registres binaires nécessaires par rapport à l'algorithme du tableau 4.4.

Le filtrage vertical est plus délicat. On peut l'exprimer simplement en utilisant la notion de *noyau homotopique inférieur* de Bertrand et Couprie [14], [29]. C'est l'image obtenue en propageant lors de l'expansion des bassins versants, la valeur des minima régionaux correspondants, de sorte que chaque bassin se trouve étiqueté par l'altitude de son point le plus profond. Dans ce cadre, le filtrage vertical s'exprime simplement : lors de la rencontre de deux bassins versants, on compare le maximum des étiquettes avec l'altitude courante (qui correspond alors au coût de connexion), et on ne crée une digue que si la différence est supérieure au seuil h. Sinon, les étiquettes sont fusionnées en prenant la valeur minimale.

Cet algorithme n'est pas adapté à un circuit SIMD rétinien car il faut coder en chaque pixel la valeur de l'étiquette propagée, ce qui est coûteux en

espace, et rend la fusion des étiquettes particulièrement difficile.

Une solution envisageable est de supprimer sur le signal I les minima régionaux de dynamique inférieure à h, en calculant la reconstruction de I dans $I+h$ (figure 4.13(2)). Cette reconstruction peut être calculée sur la rétine en remplaçant chaque seuil de niveau n par sa reconstruction dans le seuil de niveau $n+h$. Il est nécessaire dans ce cas de disposer en permanence de h seuils consécutifs en mémoire, l'implantation de ce filtre coûte par conséquent $h-1$ registres supplémentaires par rapport à la LPE brute.

Nous proposons une autre technique, adaptée à notre architecture, à la fois sur le plan du calcul SIMD, et de la pénurie de mémoire. Le principe de cette technique est présenté figure 4.14 : si l'on souhaite éliminer toutes les digues engendrées par des bassins de profondeur inférieure à h, on va calculer deux LPE du signal, mais en sous-échantillonnant sa *dynamique* (quantification des niveaux de gris), en prenant un pas égal à $2h$. Si les deux sous-échantillonnages sont en opposition de phase, alors aucune digue appartenant aux deux LPE ne peut être engendrée par un bassin de profondeur inférieure à h. Le résultat correspond donc à une «intersection» des deux LPE.

Plus précisément, le comportement de cet algorithme est lié à une version monodimensionnelle de l'expérience de l'épingle de Buffon : imaginons que l'on positionne de manière aléatoire un segment de longueur l sur une droite échantillonnée régulièrement avec un pas h. Il est facile de voir que la probabilité que le segment contiennent deux échantillons est égale à $H(\frac{l-h}{h})$, avec $H(x) = 0$ si $x < 0$, $H(x) = x$ si $0 \leq x \leq 1$ et $H(x) = 1$ si $x > 1$. Par conséquent, le comportement réel de cet algorithme vis-à-vis de la profondeur des bassins ne correspond pas à un «filtrage» déterministe, mais on a les propriétés suivantes :

- aucun bassin de profondeur inférieure à h n'apparaît dans les deux LPE à la fois.
- un bassin de profondeur $h \leq l \leq 2h$ apparaît dans les deux LPE à la fois avec une probabilité de $\frac{l-h}{h}$.
- tous les bassins de profondeurs supérieures à $2h$ apparaissent dans les deux LPE à la fois.

Ces propriétés sont conformes à notre objectif d'éliminer la sur-segmentation. Discutons à présent du calcul de l'«intersection» des deux LPE.

En réalité, si toutes les digues que l'on considère comme significatives appartiennent aux deux LPE, elles n'apparaissent pas forcément exactement aux mêmes endroits sur les deux LPE. En effet, le décalage vertical de h qui existe entre les deux LPE peut être à l'origine de petites déviations sur les digues «principales». Pour h fixé, cette déviation sera d'autant plus faible que le gradient est élevé dans le voisinage de la digue. Si l'on fait l'hypothèse qu'il

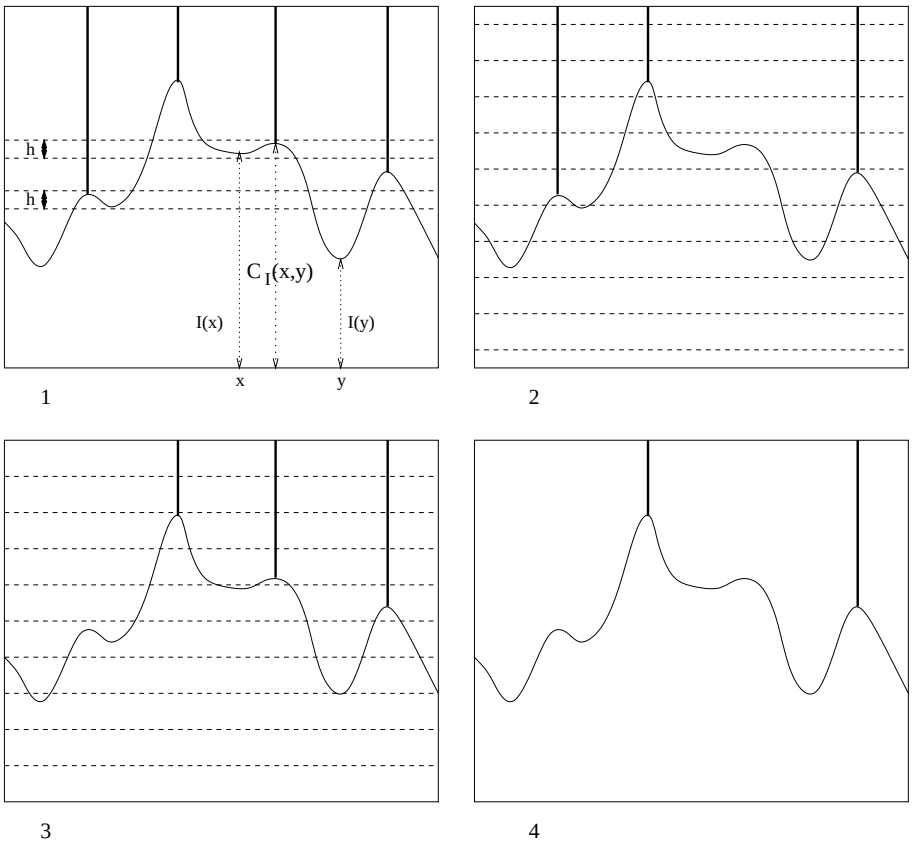


FIG. 4.14 – *Implantation du filtrage vertical : h est la dynamique minimale souhaitée (1). On sous-échantillonne «verticalement» le signal, en fixant un espace inter-coupes égal à 2h. Deux LPE sont calculées de telle sorte que leurs échantillonnages soient en opposition de phase (2 et 3). Seules les frontières des bassins significatifs appartiennent aux deux LPE, le résultat (4) correspond donc à un ET logique des deux calculs précédents.*

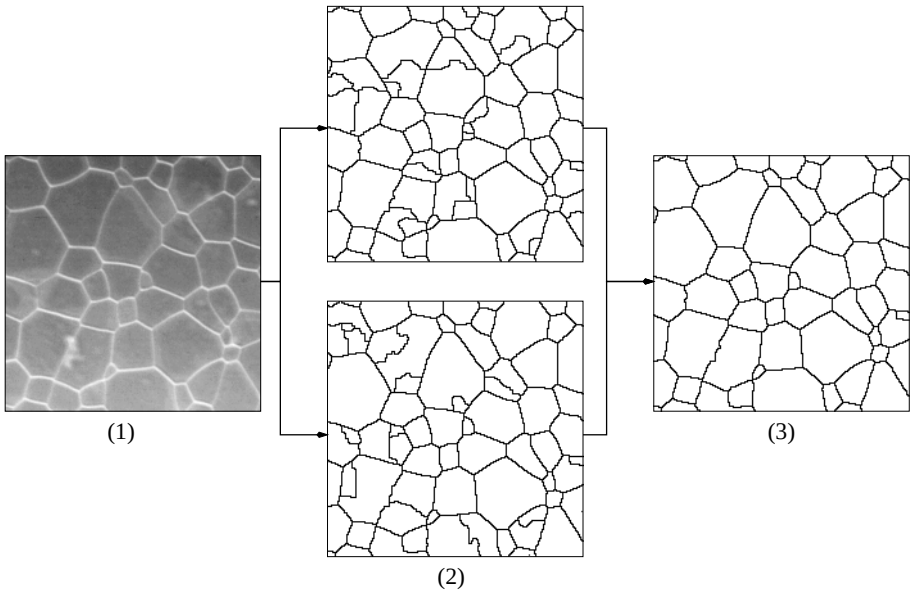


FIG. 4.15 – *Calcul de la LPE sur la rétine en éliminant la sur-segmentation. (1) Image originale. (2) Les deux LPE sont calculées en utilisant des quantifications en opposition de phase, chaque LPE faisant intervenir du filtrage horizontal. (3) Le résultat correspond aux courbes fermées qui apparaissent dans les deux LPE.*

est en moyenne supérieur à 50% au voisinage de tous les points de la LPE que l'on veut conserver, alors l'intersection correspond au calcul suivant : Si L_1 et L_2 sont les deux LPE calculées, alors le résultat sera $L = (L_1 \oplus B_h) \cap L_2$.

Enfin, il faut préciser qu'en dimension 2, ce n'est pas seulement l'ensemble des points communs qu'il faut calculer, mais l'ensemble des *courbes fermées communes*. En 2D, la bonne formule est donc $L = SKIZ[(L_1 \oplus B_h) \cap L_2]$. Un exemple de ce calcul est présenté figure 4.15.

Le tableau 4.5 décrit la procédure de calcul de la LPE paramétrée sur la rétine. Cet algorithme n'utilise finalement qu'un seul registre binaire supplémentaire par rapport à la procédure brute (voir tableau 4.4), indépendamment de la valeur de h.

4.6.5 Résultats

Nous présentons ici quelques résultats de la LPE paramétrée. La figure 4.16 montre les effets séparés des filtrages horizontal et vertical, ce qui illustre la nécessité d'introduire ces deux paramètres.

La figure 4.17 montre l'application de la LPE paramétrée dans le cadre de

$p_0 = I[0];$
$(p_0, p_1, p_2) = (OUVRECO_\rho(p_0), \tilde{p_1}, \tilde{p_2});$
$p_1 = I[1];$
$(p_1, p_2, p_3) = (OUVRECO_\rho(p_1), \tilde{p_2}, \tilde{p_3});$
pour $i = [1..\lceil \frac{n}{h} \rceil]$ {
    $p_2 = I[i \times h];$
    $(p_2, p_3, p_4) = (OUVRECO_\rho(p_2), \tilde{p_3}, \tilde{p_4});$
    si i est pair : {
        $(p_0, p_2, p_3) = (p_0, p_2, reconst_K(p_0, p_2));$
        $p_3 = p_2 \vec{\to} p_3;$
        $p_0 = p_0 \vee p_3;$
        $(p_0, p_2, p_3, p_4) = (SKIZGEO(p_0, p_2), p_2, \tilde{p_3}, \tilde{p_4});$
        }
    sinon : {
        $(p_1, p_2, p_3) = (p_1, p_2, reconst_K(p_1, p_2));$
        $p_3 = p_2 \vec{\to} p_3;$
        $p_1 = p_1 \vee p_3;$
        $(p_1, p_2, p_3, p_4) = (SKIZGEO(p_1, p_2), p_2, \tilde{p_3}, \tilde{p_4});$
        }
    }
pour $i = [1..h]$ {
    $(p_0, p_2) = (dilate_K(p_0), \tilde{p2});$
    }
$p_0 = p_0 \wedge p_1;$
$(p_0, p_1, p_2) = (SKIZ(p_0), \tilde{p_1}, \tilde{p_2});$

TAB. 4.5 – *Procédure de calcul de la LPE paramétrée sur la rétine.*

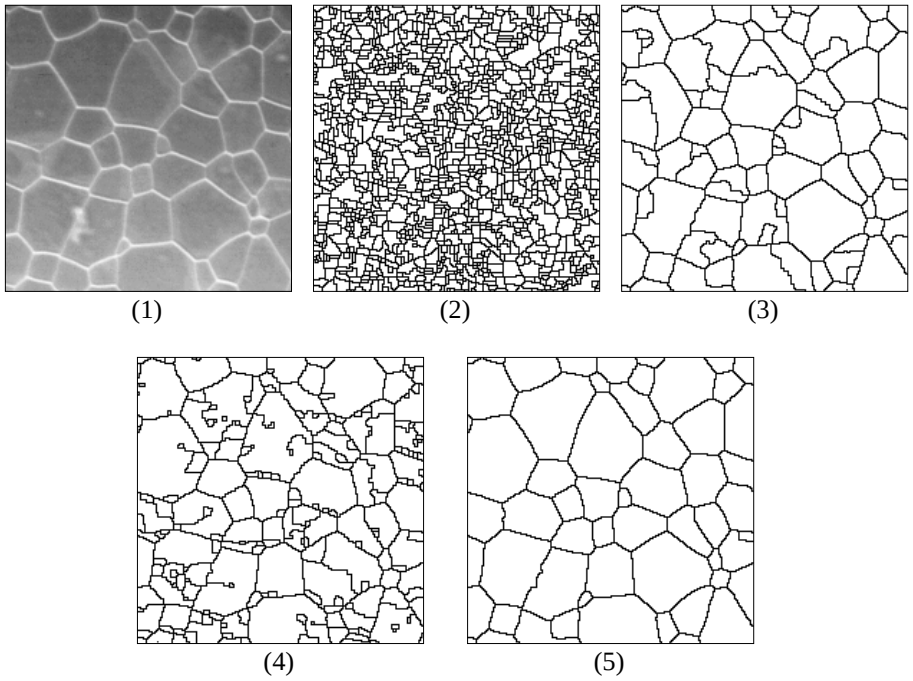


FIG. 4.16 – *Résultat de la LPE. (1) Image originale. (2) LPE brute. (3) Avec filtrage horizontal ($\rho = 2, h = 0$). (4) Avec filtrage vertical ($\rho = 0, h = 4$). (5) Avec filtrages horizontal et vertical ($\rho = 2, h = 4$).*

```
Soit m le plus petit k tel que a_k = 1.
    x = x_m
    pour i = m + 1 à n − 1 {
                si a_i = 1    x = x ∨ x_i
                sinon         x = x ∧ x_i
                }
```

TAB. 4.6 – *Comparaison d'un nombre X codé sur n bits $\{x_0, ...x_{n-1}\}$ dans la rétine avec une constante $A = \sum_{k=0}^{n-1} a_k 2^k$.*

la détection de contours. Il faut noter que dans ce contexte, l'algorithme s'applique au gradient de l'image et non à l'image elle-même. Pour l'implantation sur rétine artificielle, cela implique le choix entre deux options :

- L'implantation matérielle du gradient, en tant qu'opérateur d'une couche analogique,
- Le calcul préalable du gradient morphologique (voir premier chapitre), et son stockage, ce qui ajoute n bits pour 2^n niveaux de gradient. Il faut aussi tenir compte du calcul des $2^n - 1$ «seuillages» du gradient. Chaque seuillage peut être effectué par l'opérateur de comparaison qui apparaît dans le tableau 4.6. Cet opérateur, dû à Bernard, n'utilise pas de bit supplémentaire pour le calcul. La complexité en espace totale est donc de $n+5$. La complexité en temps du calcul de $\{X \geq A\}$ est égale à $n - m$, avec m le plus petit k tel que $a_k = 1$ Le coût total du calcul des seuillages est donc $\sum_{k=0}^{n-1}(k+1)2^k = 2^n(n-1)+1$.

La figure 4.18 illustre le problème du paramétrage de la méthode. On a en effet sur ces images d'angiographie un «profil» très différent de celui qu'on peut trouver sur les images microscopiques de minerai d'uranium ou sur les images de gradients. En particulier, l'hypothèse du gradient supérieur à 50% au voisinage des points significatifs de la LPE n'est pas vérifiée. On observe alors une légère sous-segmentation, ce qui montre que dans le cas général, la méthode doit avoir un troisième paramètre, qui est le rayon de la boule B_r qui définit la «tolérance» de l'intersection : $L = SKIZ[(L_1 \oplus B_r) \cap L_2]$.

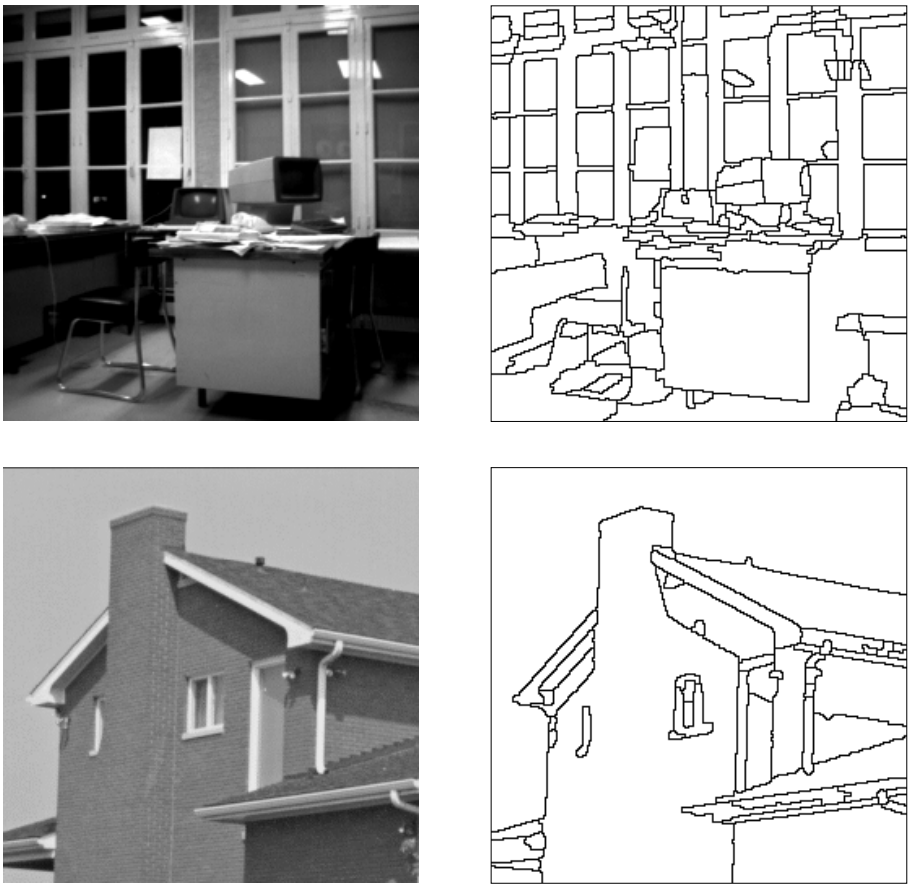


FIG. 4.17 – *Détection de contours par application de la LPE à l'image de l'intensité du gradient morphologique (paramètres utilisés : $\rho = 2$ et $h = 2$).*

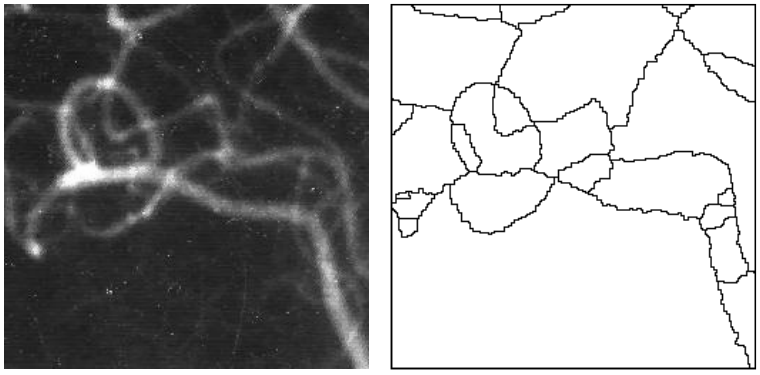


FIG. 4.18 – *La LPE paramétrée sur une image d'angiographie, avec* $\rho = 2$ *et* $h = 4$. *Cet exemple suggère l'utilisation d'un troisième paramètre qui tiendrait compte du profil du signal.*

4.7 Conclusion et discussion

Dans cette partie consacrée aux opérateurs géodésiques, nous avons mis en évidence l'intérêt de la reconstruction géodésique comme opérateur fondamental d'une algorithmique cellulaire telle que celle de la rétine. La reconstruction géodésique apparaît comme un opérateur «local vers global» par excellence. C'est en effet un traitement de base qui agit à un niveau global et qui est obtenu par la relaxation d'un opérateur local.

Examinons à présent le comportement d'un algorithme géodésique sur notre architecture entièrement synchrone. Nous avons rencontré au cours de notre travail plusieurs algorithmes fondés sur la relaxation (le squelette est un exemple remarquable). Mais dans le cas des opérateurs géodésiques, l'opérateur de base est lui-même un opérateur fondé sur la relaxation. Or sur une architecture uniquement synchrone, le coût d'une reconstruction géodésique est égal au coût d'une interaction locale de base (typiquement dilatation + intersection), multiplié par le nombre d'itérations. Or le nombre d'itérations maximum est proportionnel au diamètre géodésique des images. Dans le pire des cas, il est facile de voir que le diamètre géodésique d'une image peut dépendre linéairement de la surface de la rétine (composantes connexes enroulées). Dans ce cas, l'effet du calcul parallèle synchrone est que chaque itération va agir sur une toute petite partie de l'image, l'effet n'étant dû finalement qu'à un grand nombre d'itérations.

Si l'on considère l'*efficacité* du calcul, qu'on peut définir comme la proportion de processeurs qui font un calcul «utile», on s'aperçoit que, dans le cas d'un opérateur géodésique, cette proportion diminue d'un ordre de grandeur

correspondant à une dimension, lorsque l'on passe d'une image épaisse à une image filaire. Il en résulte une certaine inefficacité des opérateurs géodésiques sur une architecture cellulaire synchrone.

Dans le cadre des opérateurs géodésiques, le recours à des fonctions asynchrones pourrait donc se révéler très intéressant. Sur le circuit *Pulsar 2.2*, une fonction originale a été découverte et testée : à partir d'un seul pixel à 1 sur l'un des plans binaires, on peut mettre à 1 l'ensemble du plan par une propagation asynchrone, d'une part, de manière sensiblement plus rapide qu'on ne le ferait par une utilisation classique (synchrone) de la rétine (dilatations itérées), mais surtout en consommant beaucoup moins d'énergie. En effet, dans le cas d'une propagation asynchrone, seuls les processeurs actifs (ceux dont la mémoire change d'état) consomment de l'énergie. Sous sa forme actuelle, cette fonction n'a qu'un intérêt très limité (tout au plus permet-elle de savoir s'il y a au moins un point à 1 dans l'image). En revanche, elle se révèlerait très intéressante si l'on disposait de *connexions programmables*, c'est-à-dire de la possibilité d'ouvrir ou fermer une connexion entre un pixel et son voisin en fonction d'un calcul sur un voisinage de ce pixel.

On disposerait ainsi d'une machine à topologie programmable. Regardons la figure 4.19 : l'image binaire (1) est représentée dans une maille carrée en allumant certains nœuds de la maille (2). Dans une topologie programmable, on peut allumer (et éteindre) à la fois les nœuds et les arêtes de la maille. Ainsi, si l'on n'ouvre que les connexions reliant deux points appartenant à l'image, on obtient la topologie (3). Dans cette maille, on peut obtenir très efficacement grâce à la propagation asynchrone une reconstruction géodésique, et ainsi gagner beaucoup sur l'ensemble des opérateurs géodésiques.

Une technique de ce type a été proposée et testée avec succès sur une rétine 32×32 à l'Université de Linköping [33]. Bien qu'il n'y ait pas eu de rétines de grande taille intégrant cette fonction, et peu de développements algorithmiques, nous soulignons tout l'intérêt qu'aurait une solution architecturale combinant opérations élémentaires synchrones et fonctions de relaxation asynchrones.

Notons de plus que l'intérêt des propagations asynchrones dans une topologie programmable va au-delà de la simple reconstruction. La figure 4.20 en fournit un autre exemple. Si l'on n'ouvre que les connexions touchant un pixel ayant deux voisins propres, il suffit ensuite de repérer les extrémités (points n'ayant qu'un voisin propre) pour obtenir efficacement l'ensemble des lacets simples ouverts. Cette fonction permet ainsi de rendre beaucoup plus efficace le calcul du SKIZ : le calcul est effectué en synchrone tant que l'objet est épais, et dès que l'objet devient filaire, le calcul est achevé grâce à une seule propagation asynchrone .

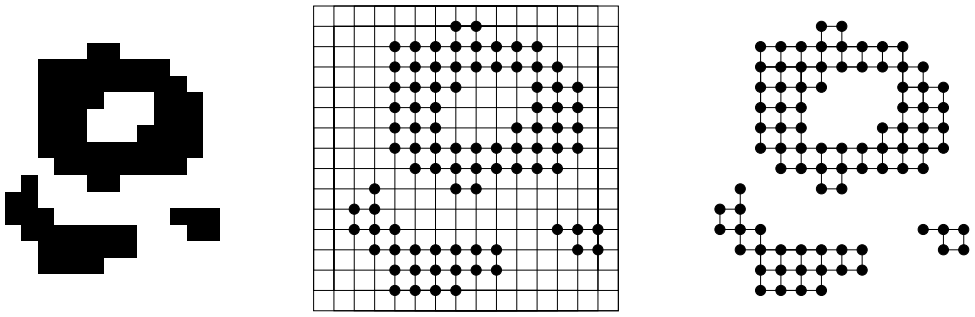


FIG. 4.19 – *Utilisation de connexions programmables pour une architecture à topologie géodésique.*

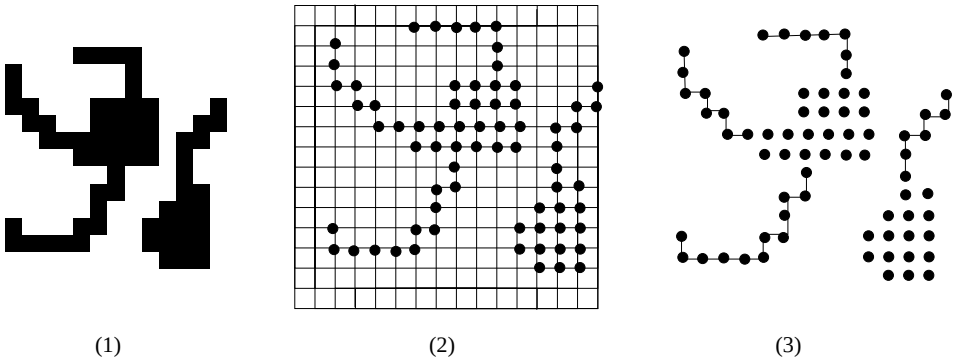


FIG. 4.20 – *Un exemple d'une autre utilisation des connexions programmables : la détection des courbes ouvertes.*

Chapitre 5

Mouvement et application

5.1 Enjeux et objectifs

Après avoir étudié l'implantation d'un certain nombre d'algorithmes sur la rétine programmable, examinons à présent leurs avantages dans un contexte de recherche et développement applicatif.

Tout d'abord, ce travail algorithmique a démontré la capacité de la rétine programmable à calculer, à des cadences allant souvent bien au delà du temps réel, un nombre important d'algorithmes classiques de la morphologie mathématique.

Ensuite, il a permis de constituer une base algorithmique qui va former le début de la «boîte à outils» du concepteur d'applications à base de rétines programmables.

Il reste enfin à faire la preuve de l'efficacité de la rétine programmable pour aborder et résoudre une application complète de vision par ordinateur. Le contexte applicatif a été spécifié par *E.A.D.S./Aérospatiale-Matra Missiles* conformément à la composante industrielle de cette thèse. L'application visée est la *balise automatique* avec capteur de vision. Ce système est destiné à fournir à distance des renseignements sur les événements qui se produisent dans une zone d'importance stratégique.

L'objectif est d'utiliser la rétine programmable en tant que capteur de *surveillance*. La problématique scientifique sous-jacente concerne la *détection de mouvement par la rétine programmable*.

Dans ce cadre, si la contrainte de temps réel est bien sûr incontournable, deux autres avantages de la rétine programmable se révèlent déterminants pour le système décrit dans la section suivante :

- la *compacité*, compatible avec les contraintes de discrétion et d'embarquement.

– la *faible puissance*, nécessaire pour disposer de systèmes à longue autonomie.

Nous avons utilisé à dessein le terme un peu vague de «détection» de mouvement, car nous ignorons pour l'instant jusqu'où la rétine pourra aller dans sa «connaissance» du mouvement, puisque l'application abordée est totalement innovante. De ce niveau de connaissance dépendront fortement l'importance du rôle que la rétine jouera dans le système de surveillance ainsi que la puissance de ce dernier. Le niveau minimum d'intelligence exigible pour l'application visée est de type *alerte* : «un objet de taille significative a bougé dans la scène». On pourra par exemple demander à la rétine qu'elle fournisse une zone d'intérêt rectangulaire, circonscrite aux objets en mouvement.

Dans une approche plus ambitieuse, on cherchera à avoir plus d'information sur les objets mobiles : qu'est-ce qui a bougé ? (niveaux de *reconnaissance et identification*). Dans ce cas, il est souhaitable, classiquement, que la rétine fournisse une *segmentation* des objets mobiles.

On peut chercher aussi à en savoir plus sur la nature du mouvement : à quelle vitesse se déplacent les objets mobiles ? (niveau *analyse*). L'information qu'on attendra de la rétine pourra alors être un champ de déplacement (flot optique).

Dans ce chapitre, nous présentons tout d'abord une description globale du système de surveillance envisagé pour l'application considérée. Nous étudions ensuite les contraintes algorithmiques liées aux traitements temporels avec la rétine programmable. Ensuite, nous présentons un algorithme de segmentation d'objets mobiles par un capteur fixe que nous avons implanté sur le prototype de rétine dont nous disposons (*Pvlsar 2.2*). Puis, nous étudions l'implantation d'un algorithme d'approche probabiliste dans le contexte de la segmentation markovienne. Enfin, nous concluons sur les contributions apportées, les problèmes soulevés et les voies d'amélioration à explorer.

5.2 Des balises intelligentes

Dans la terminologie militaire, une *balise* est un émissaire de renseignement chargé de recueillir des informations sur une scène de grande importance stratégique (passage de véhicules, de troupes, activité humaine). Le recueil de ces informations nécessite une précision inaccessible aux satellites, et une discrétion interdisant l'usage des observations aériennes. Ces balises sont actuellement des unités humaines très spécialisées qui interviennent la plupart du temps en zone de conflit. Il va de soi qu'elles courrent des risques très importants, d'autant plus que le type même de leur mission les oblige à demeurer en général très longtemps loin de leur base.

On conçoit dès lors l'expression du besoin opérationnel pour des balises automatiques [72], qui auraient pour fonction d'acquérir et de transmettre des images lorsqu'elles perçoivent un événement significatif. Les systèmes de ce type existant à l'heure actuelle sont des capteurs sismiques : ils déclenchent une acquisition dès qu'une vibration d'une certaine amplitude est détectée. Les renseignements que le système d'alerte est susceptible de fournir sont cependant très insuffisants pour les utilisations envisagées.

Des informations plus fines sur le mouvement (localisation dans la scène, amplitude, nature des objets mobiles) s'avérant nécessaires, ou tout au moins souhaitables, le choix d'un système à base de vision s'impose aujourd'hui de plus en plus. En outre, la balise automatique ne doit pas être facilement repérable, et doit pouvoir fonctionner durant plusieurs semaines sans intervention humaine physique. Ces deux contraintes justifient le choix de la rétine programmable comme capteur de vision du système, la compacité et la faible puissance [87] étant gages de discrétion et de longue autonomie.

La figure 5.1 donne le principe de fonctionnement de la balise automatique telle que nous l'envisageons. Les images transmises, destinées à des humains à des fins de preuves d'un certain événement, doivent être de grande qualité. Cela implique une grande résolution, bien supérieure à celles des rétines programmables existantes. L'acquisition des images transmises est donc faite dans ce système par un autre capteur, par exemple un imageur CCD. En l'absence d'événement dans la scène, seul le capteur de réveil fonctionne, c'est-à-dire la rétine et le processeur qui la pilote (typiquement un petit micro-contrôleur basse puissance), le reste du système étant en veille. On devrait ainsi obtenir une puissance en mode continu de quelques dizaines de milliwatts.

Lorsqu'un événement est détecté, la rétine «réveille» le capteur d'acquisition et le reste du système. La caméra haute résolution transmet l'image à l'unité de compression qui reçoit également de la rétine des informations sur les objets mobiles utiles pour optimiser la compression. L'exemple le plus élémentaire consiste à fournir les coordonnées de zones d'intérêt rectangulaires pour faire une compression sélective.

Finalement, les données comprimées sont envoyées au module de transmission. Les images reçues doivent être de bonne qualité. Cependant, les contraintes d'autonomie et de discrétion électromagnétique impliquent une basse puissance et un débit relativement faible pour la transmission. Le rôle de la compression est donc déterminant, d'où l'importance des informations sur les objets mobiles fournies par la rétine.

Dans la suite, nous allons donc nous efforcer de calculer sur la rétine programmable l'information la plus pertinente possible relative à des objets mobiles. Nous détaillons tout d'abord le contexte algorithmique très parti-

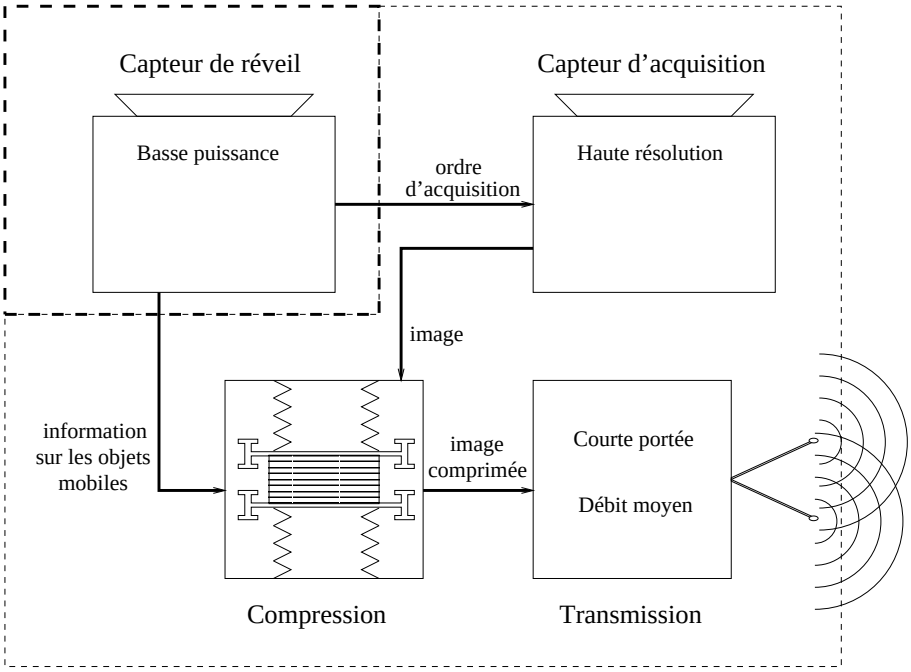


FIG. 5.1 – *Les différents constituants de la balise intelligente. La frontière en pointillés gras délimite la partie du système qui fonctionne en continu : c'est le capteur de réveil, composé de la rétine programmable et de son séquenceur. Le reste est en veille et ne fonctionne que lors de la détection d'un événement par la rétine.*

culier qu'implique l'étude du mouvement avec un tel circuit.

5.3 Les traitements temporels sur la rétine programmable

Si la contrainte liée à la taille extrêmement réduite de l'espace mémoire était déjà très présente dans les chapitres précédents, il est clair qu'elle sera plus critique encore dans le cadre des traitements temporels. En effet, il est évident que le calcul doit faire interagir des images acquises à des instants différents. Au moment de l'acquisition d'une image, la rétine doit posséder en mémoire des informations sur une ou deux images précédentes, et ce dans une certaine dynamique de niveaux de gris ! Imaginons par exemple le calcul sur une rétine possédant 5 bits de mémoire par processeur, d'une simple différence d'images sur 8 niveaux de gris. Si l'on a codé les niveaux de gris de

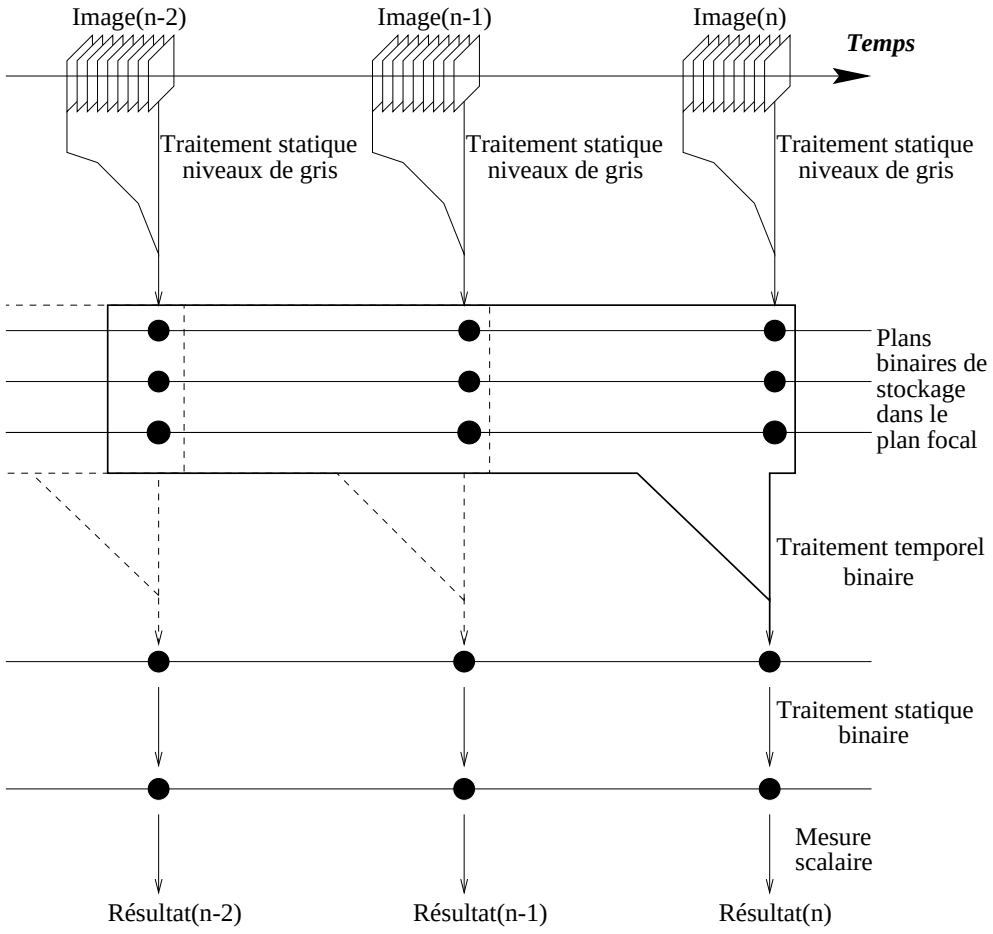


FIG. 5.2 – *Les caractéristiques générales du traitement temporel sur la rétine programmable.*

l'image précédente sur le circuit, 3 bits sont utilisés, et le codage de l'image courante est impossible. Le problème de la représentation du niveau de gris dans le circuit est donc particulièrement crucial. Nous le développons dans la section suivante.

La figure 5.2 représente schématiquement les principes du traitement temporel sur la rétine programmable. Les axes horizontaux représentent le temps. Chaque point noir correspond à un bit de mémoire par processeur. Expliquons ce schéma de bas en haut : la «réponse» de la rétine consiste en un certain nombre de mesures scalaires (coordonnées, taille, vitesse, forme). Elle est fournie à partir d'une image binaire qui est donc le résultat ultime des traitements rétiniens parallèles. Cette image est le résultat de traitements binaires effectués sur un certain nombre d'images représentant des instants

différents. L'ensemble des bits d'information nécessaires au calcul correspond au rectangle en traits pleins sur le schéma. Comme le nombre de bits de cet ensemble ne doit pas être supérieur au nombre de bits disponibles par processeur sur le circuit, il est indispensable de limiter le codage à un très petit nombre de bits. Ce codage est donc le résultat de traitements statiques en niveaux de gris afin de produire une représentation compacte de l'image et pertinente pour l'étude du mouvement.

L'algorithme que nous présentons dans la section 5.5 constitue une instance particulièrement significative de ce schéma. Il a été implanté sur le circuit *Pulsar2.2*, avec 5 bits de mémoire.

5.4 Représentation du niveau de gris

Dans cette section, nous présentons l'ensemble des techniques que nous avons utilisées pour représenter une information en niveaux de gris dans le circuit, afin de proposer des solutions au problème de codage des données dans le cadre d'un traitement temporel.

5.4.1 Codages binaires simples

La technique la plus élémentaire consiste à effectuer le codage binaire du niveau de gris sur k plans binaires du circuit. On conserve alors en mémoire une image ayant une dynamique de 2^k niveaux de gris.

Examinons la figure 5.3. Une image numérique (en haut à droite) est acquise en n niveaux de gris, grâce à la lecture multiple du photocapteur, qui fournit $(n-1)$ coupes binaires (en haut). On effectue alors, pour chaque pixel, un comptage sur $log(n)$ bits du nombre de coupes où le pixel vaut 1.

La technique de comptage la plus rapide sur le circuit consiste à utiliser un *compteur de Gray* (ligne du milieu sur la figure 5.3). En effet, le principe d'un tel compteur est que les codages de deux nombres consécutifs ne diffèrent que d'un seul bit. Si l'image est représentée par ses coupes $(c_i)_{1 \leq i \leq n-1}$, et si l'on note $Z(p)$ la plus grande puissance de 2 qui divise un naturel p, alors le code Gray du niveau de gris est défini par ses chiffres binaires $(g_j)_{0 \leq j \leq log(n)-1}$, avec :

$$g_j = \bigoplus_{Z(i)=2^j} c_i,$$

où $\bigoplus$ désigne l'opérateur d'imparité. Le comptage en code Gray utilise seulement $n - log(n)$ XOR, et pas d'autres registres binaires que ceux utilisés pour le comptage.

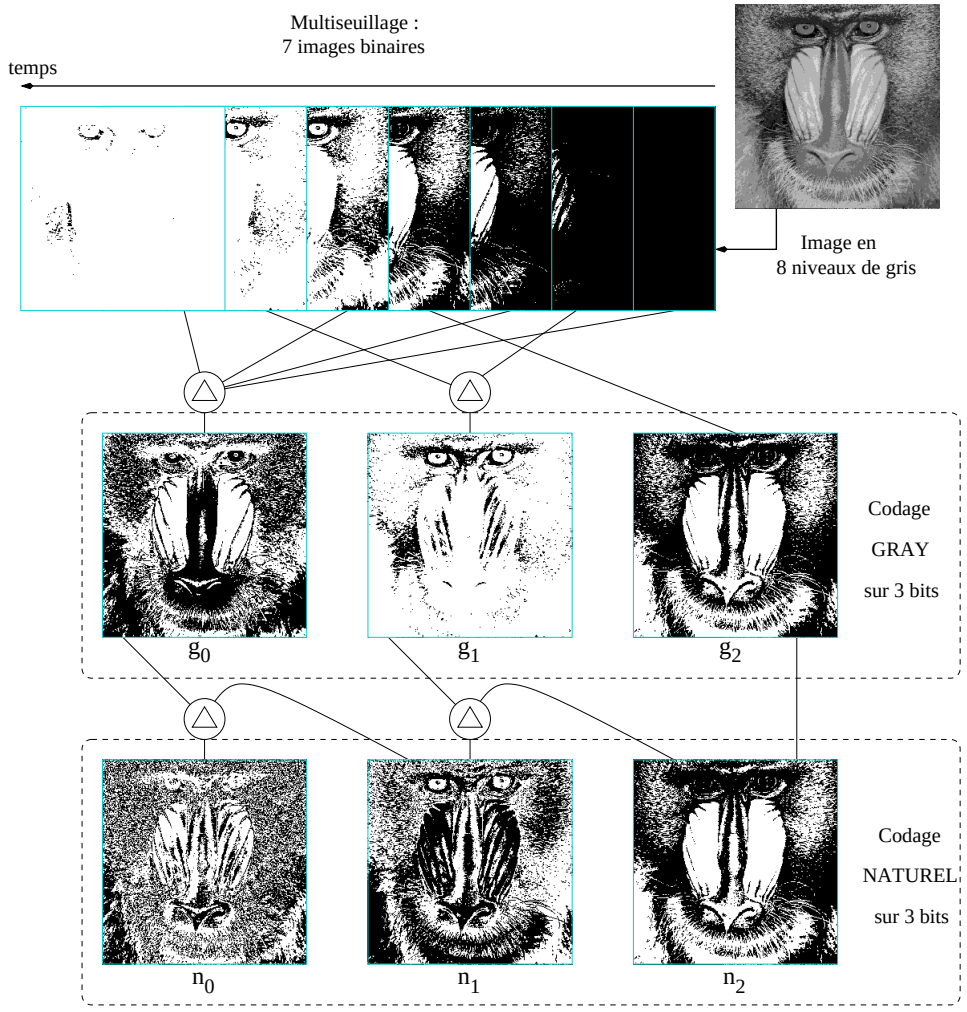


FIG. 5.3 – *Le représentation du niveau de gris par un codage binaire simple.*

On peut si nécessaire passer très facilement à un codage naturel (ligne du bas). Le code naturel est défini par ses chiffres binaires $(n_j)_{0 \leq j \leq log(n)-1}$ tels que la valeur du niveau de gris représenté est $\sum_{j=0}^{log(n)-1} 2^{n_j}$. On passe du code Gray au code naturel par $(log(n) - 1)$ XOR supplémentaires, avec la procédure suivante, en posant $h = log(n) - 1$:

$$\begin{aligned} & n_h = g_h; \\ & \text{pour} \quad i = h - 1 \text{ à } 0 \\ & \qquad n_i = g_i \triangle n_{i+1}; \end{aligned}$$

Le code naturel est ainsi obtenu sans mémoire supplémentaire.

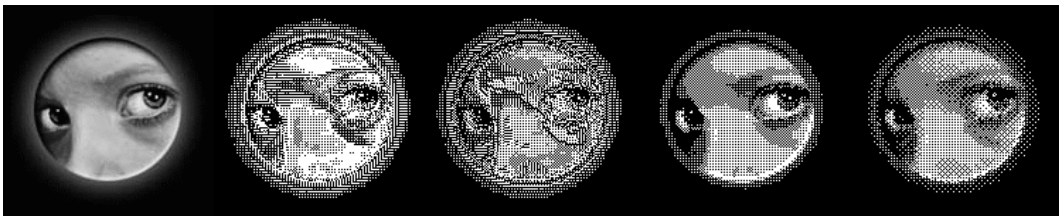
Dans le cadre du traitement temporel, sur la rétine actuelle à 5 bits de mémoire, il ne saurait être question de coder les images sur 3 ou 4 bits pour conserver une dynamique de 8 ou 16 niveaux de gris. Inversement, il n'est pas raisonnable d'espérer étudier le mouvement à partir d'acquisitions binaires. Il faut donc trouver un codage compact, c'est-à-dire à la fois économe en mémoire et représentatif d'une dynamique correcte. Dans la suite, nous étudions les possibilités offertes dans ce sens par la multirésolution.

5.4.2 Codages de Bayer

Les images présentées ici sont des codages sur 1 bit, d'images sous-résolues. Grâce à la multigranularité, on peut regrouper les pixels par groupes (clusters), à l'intérieur desquels chaque pixel est commandé différemment. Le principe est d'associer à tous les pixels d'un cluster une unique valeur de niveau de gris et de coder cette valeur sur les différents bits du cluster.

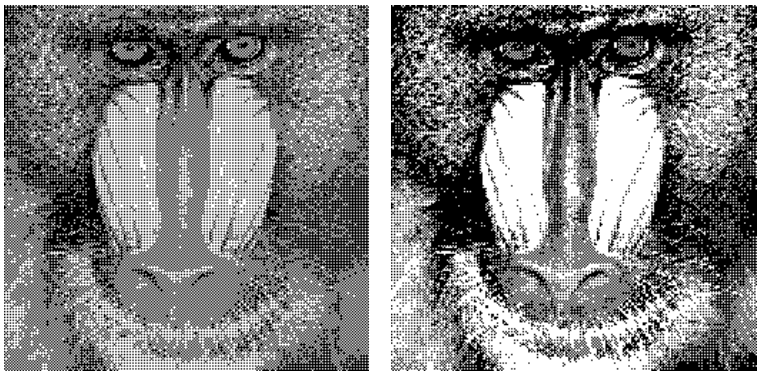
Examinons par exemple la figure 5.4. L'image en niveau de gris (0) est codée à mi-résolution sur l'image (1), en codant le niveau de gris de l'image sous-résolue sur les pavés 2×2 du cluster en code Gray. On peut, si nécessaire, passer facilement au code naturel (image (2)). Dans les deux cas, la représentation fournit une dynamique de 16 niveaux de gris à mi-résolution, sur un seul bit. Le codage est effectué en 11 OE pour (1) et 14 OE pour (2). Il est assez compact, mais plutôt difficile à exploiter car il nécessite des opérations arithmétiques, qui devront être effectuées de manière non invariante en translation (avec la multigranularité), et avec des registres supplémentaires pour les calculs intermédiaires.

Une représentation moins compacte du même type est celle du codage *en demi-teinte* par matrices de Bayer [115]. Le niveau de gris correspond à une densité de points blancs, le codage étant effectué par des motifs binaires réguliers. On ne code donc que $n+1$ niveaux de gris sur des clusters de taille n. Le coût du codage est nul puisqu'il n'y a pas d'opérations, mais seulement



(0) (1) (2) (3) (4)

FIG. 5.4 – *Codage du niveau de gris dans le cluster en sous-résolution. (0) Image originale. (1) 16 niveaux de gris en codage Gray. (2) 16 niveaux de gris en codage naturel. (3) 5 niveaux de gris en codage en demi-teinte. (4) 9 niveaux de gris en codage en demi-teinte.*



(1) (2)

FIG. 5.5 – *Effet de l'égalisation d'histogramme sur la dynamique du codage demi-teinte par matrices de Bayer 5 niveaux: (1) sans (2) avec.*

des permutations dégénérées. Les images (3) et (4) montrent deux exemples de codage en demi-teinte, avec 5 et 9 niveaux de gris respectivement sur des clusters de tailles respectives 4 et 8.

Ce type de codage offre l'avantage de pouvoir être exploité de manière invariante en translation, par un simple XOR, pour obtenir une image de «différence» qu'il est possible de «seuiller» par les fonctions seuils binaires (cf. chapitre 2).

Mais ce codage, peu compact, offre une dynamique très faible. Il est donc essentiel d'exploiter le mieux possible cette dynamique, en utilisant par exemple l'égalisation d'histogramme (figure 5.5).

5.4.3 Transformées en ondelettes

Enfin, une approche prometteuse sur le codage de l'information dans la rétine nous semble être l'utilisation de techniques du type transformées en ondelettes. Appliqué à la rétine programmable, le principe consiste à tranformer un signal binaire par un opérateur réversible, qui corresponde à une décomposition basses fréquences/hautes fréquences.

L'exemple le plus élémentaire, si l'on considère deux bits a et b, est la transformation suivante :

$$(a, b) \longrightarrow (a, a \triangle b).$$

Cet opérateur est inversible (et égal à son inverse), sa composante gauche, la projection, est un opérateur passe-bas, et sa composante droite, le XOR, un opérateur passe-haut. L'application de cette transformation sur deux pixels voisins, horizontalement, puis verticalement est appelée *transformée de Haar morphologique séparable* (THMNS) par Heijmans et Goutsias [45].

Ces auteurs présentent également des versions non séparables, pour lesquelles le choix des opérateurs binaires est plus important, puisqu'on opère sur quatre pixels voisins. La transformée de Haar morphologique non séparable (THMNS) peut être définie simplement par :

$$(a, b, c, d) \longrightarrow (a, a \triangle b, a \triangle c, a \triangle d).$$

Cet opérateur est aussi auto-inverse. La transformée suivante est également proposée dans [45] :

$$(a, b, c, d) \longrightarrow (T_5^3(a, a, b, c, d), a \triangle b, a \triangle c, a \triangle d).$$

qui a pour transformée inverse :

$(a, b, c, d) \longrightarrow (a \triangle \xi, b \triangle a \triangle \xi, c \triangle a \triangle \xi, d \triangle a \triangle \xi)$, avec $\xi = (b \wedge c \wedge d)$.

Grâce à la multigranularité, on peut calculer ces différents opérateurs sur la rétine en quelques opérations (figure 5.6(1)-(3)). En appliquant la transformée sur tous les seuils d'une image en niveaux de gris (figure 5.6(4)(5)), l'image obtenue présente bien les caractéristiques d'une transformée en ondelettes, (1 composante BF, 3 composantes HF dans les directions horizontale, verticale et diagonale). Toutefois, elle n'est plus réversible, la différence représentée dans les composantes HF n'étant pas signée.

Si l'on dispose de plusieurs niveaux de multigranularité, on peut bien entendu appliquer de telles transformations de manière récursive (figure 5.6(6)).

La non-réversibilité de la transformée en niveaux de gris pose le problème de la transformée inverse (synthèse) qui n'est pas exacte. Pour l'opérateur de synthèse, nous avons étudié deux possibilités (figure 5.7), soit par interpolation (1), ce qui consiste à décider du signe en examinant la valeur des pixels plus loin dans la même direction et en faisant une hypothèse de monotonie,

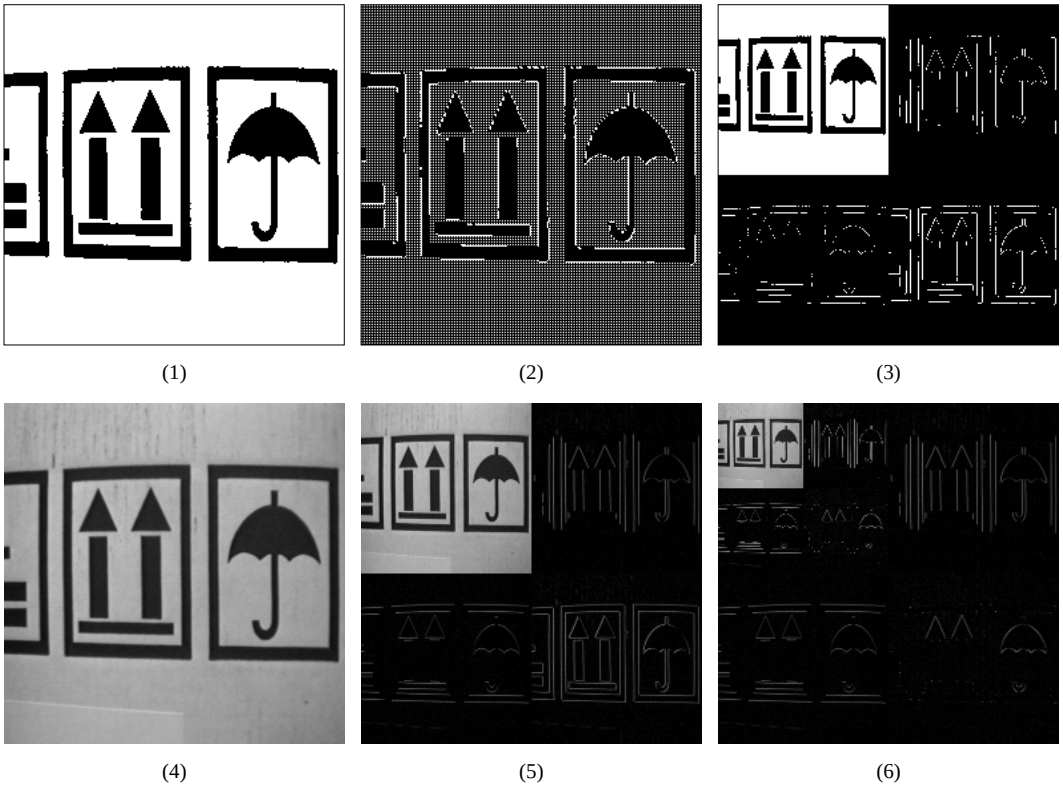


FIG. 5.6 – *Transformées en ondelettes : (1) Image binaire. (2) THMNS binaire (réversible) : information présente dans la rétine. (3) THMNS binaire : représentation symbolique équivalente. (4) Image en niveaux de gris (6 bits). (5) THMNS (non réversible). (6) Application récursive de la THMS.*

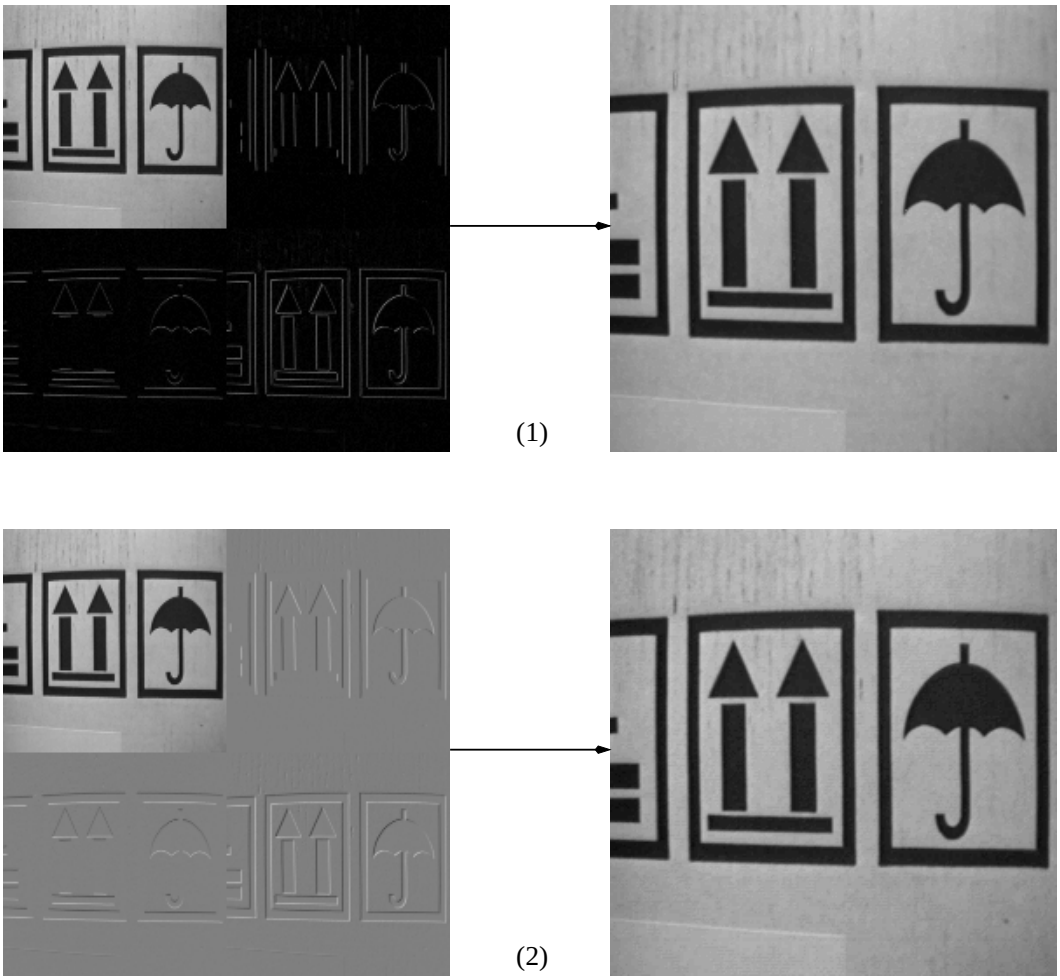


FIG. 5.7 – *Transformées inverses des ondelettes en niveaux de gris (sur 6 bits). (1) THMNS non signée, reconstruite par interpolation. (2) THMNS signée, HF codées sur 5bits + 1 bit de signe (la valeur 0 correspond au gris moyen).*

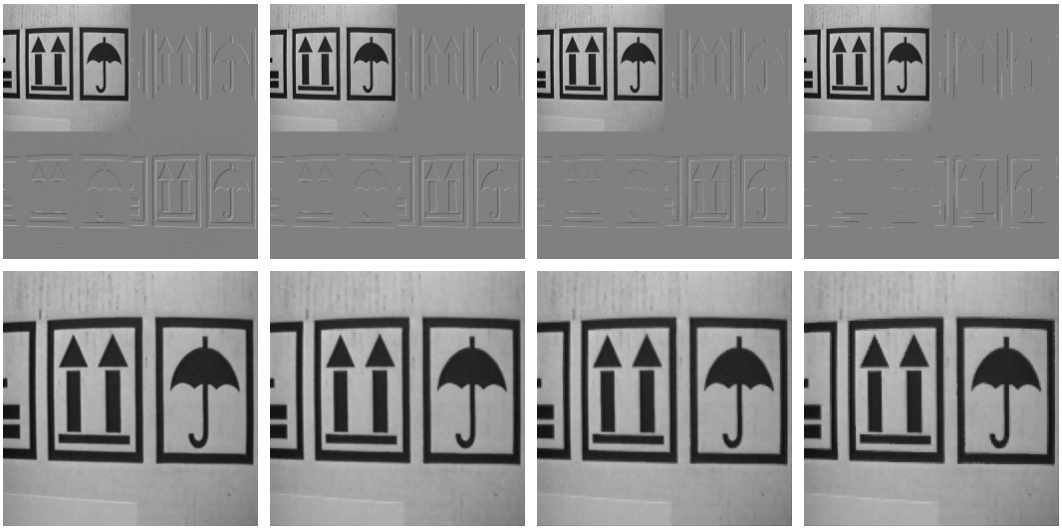


FIG. 5.8 – *Transformées en ondelette à codage différencié (en haut), et images reconstruites (en bas). (1) BF codées sur 6 bits, HF codées sur 6 bits (5 bits + 1 bit de signe). (2) BF codées sur 8 bits, HF codées sur 5 bits (256 niveaux de gris sur 6 bits). (3) BF codées sur 8 bits, HF codées sur 4 bits (256 niveaux de gris sur 5 bits). (4) BF codées sur 7 bits, HF codées sur 3 bits (128 niveaux de gris sur 4 bits).*

soit en quantifiant les niveaux de gris des composantes HF et en utilisant un bit pour coder le signe (2). La première solution a tendance à lisser l'image, tandis que la seconde réduit la dynamique des HF. En ce qui concerne l'écart quadratique moyen, et en l'absence d'hypothèses de régularité, la seconde solution est bien sûr meilleure.

Cela amène à considérer une manière intéressante d'exploiter ce type de transformation pour obtenir un codage compact sur la rétine programmable, en combinant les deux principes de synthèse que nous venons de présenter. On utilise ainsi une dynamique plus importante pour coder la partie BF. Cela est illustré sur la figure 5.8, en haut : les trois sous-images codant les HF peuvent «céder» chacune un bit à la sous-image codant les BF, qui «récupère» donc trois bits supplémentaires.

De plus, pour les opérateurs de synthèse (figure 5.8, en bas), on compense la perte de dynamique des HF par une interpolation : dans l'intervalle défini par la quantification, on choisit le niveau de gris conduisant à la plus grande régularité.

Ce codage représente finalement une forme primitive de compression, qui

semble attrayante dans le cadre d'un traitement temporel. Son comportement reste néanmoins à étudier plus en détail.

5.5 Segmentation d'objets mobiles

Nous présentons dans cette section un premier algorithme de détection de mouvement implanté sur *Pvlsar 2.2*. Le but recherché est le suivant : observant une scène avec un capteur fixe, nous souhaitons attribuer à chaque pixel une étiquette, selon qu'il appartient ou non à un objet mobile. Le résultat attendu est donc une image binaire correspondant à une *segmentation* en objets mobiles.

5.5.1 Contraintes et implantation

Nous détaillons à présent les principes de l'algorithme proposé, en expliquant les choix algorithmiques effectués en regard des contraintes mémoire. La description suivante peut paraître un peu triviale, elle est cependant représentative de la sobriété à laquelle doit se plier le concepteur d'algorithmes pour rétine programmable, dans la mesure où nous nous sommes imposés de pouvoir faire tourner ce premier algorithme sur le prototype actuel, ce qui limite considérablement les choix possibles. Heureusement, ces contraintes vont s'adoucir avec les prochaines générations de rétine.

Nous nous appuyons sur la figure 5.9 pour illustrer notre propos. Le premier choix correspond au nombre d'images intervenant dans le calcul d'une image de segmentation. Nous avons opté pour le minimum : deux images (figure 5.9 (0a) et (0b)).

Mais que faut-il conserver en mémoire de l'image (n-1) au moment de l'acquisition de l'image n ? Combien de bits sont-ils disponibles pour cela ? Avec 5 bits, le maximum est vraisemblablement 2 bits. Ce chiffre étant choisi, que coder d'une image sur 2 bits ? Un codage binaire simple offrant une dynamique de 4 niveaux de gris, cette solution n'est pas acceptable. Nous proposons donc une représentation plus complexe mieux adaptée à notre problème. En effet, sur une scène dynamique, les régions dans lesquelles le mouvement est détectable entre deux images correspondent aux régions de contraste, c'est-à-dire aux contours. Nous utiliserons donc un bit pour coder les contours de l'image courante. Les contours sont calculés par l'algorithme du gradient seuillé présenté précédemment, effectué sur deux bits (figure 5.9 (1a) et (1b), ces images binaires correspondent à un traitement sur 16 niveaux de gris).

Ces images de contours sont utilisées pour la phase de *détection* : la

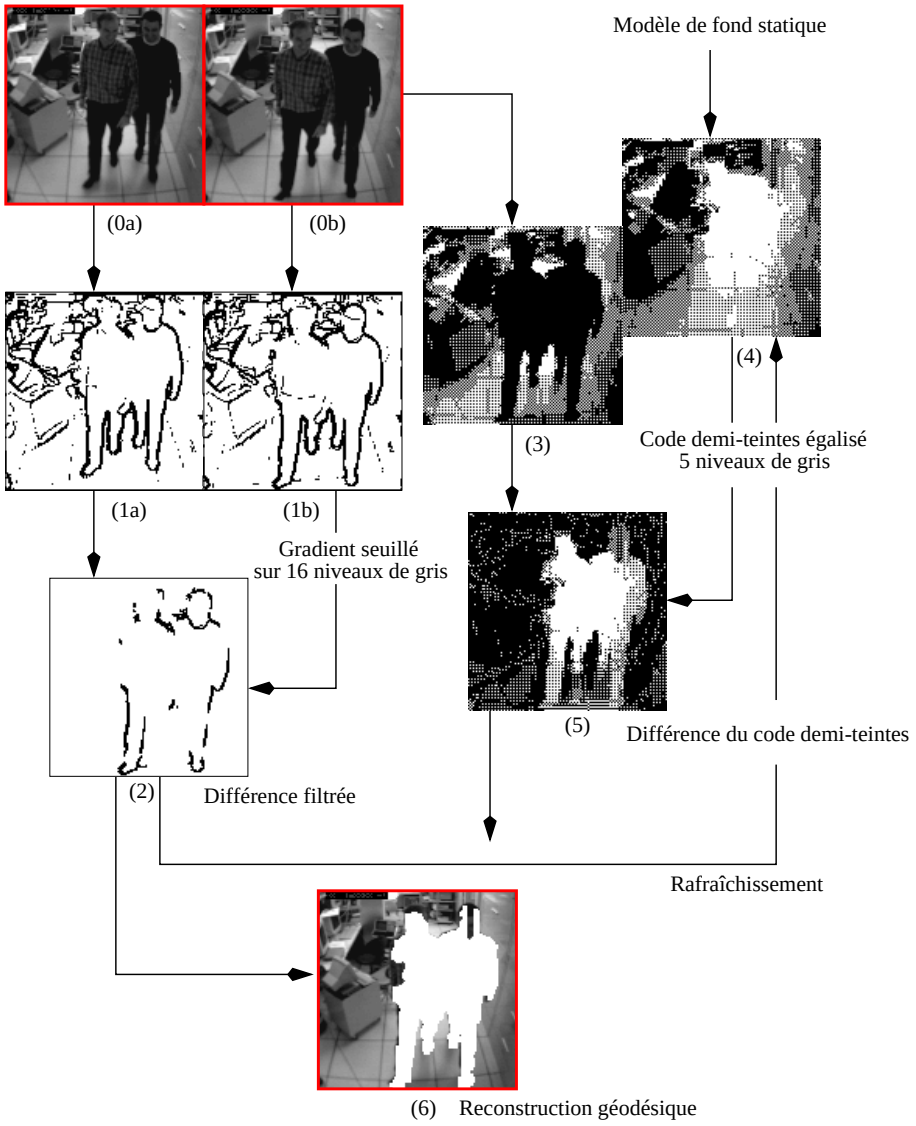


FIG. 5.9 – *L'algorithme de segmentation d'objets mobiles implanté sur la rétine programmable. On utilise deux bits pour coder les images : une image de contours sur 16 niveaux (1) et une image codée en demi-teinte sur 5 niveaux (3). La détection correspond à une différence de contours (2), la segmentation (6) est la reconstruction géodésique de cette différence dans l'image seuillée de la différence des codages demi-teinte (5).*

différence asymétrique «(1b) et non(1a)» n'utilise aucun bit supplémentaire pour son calcul. Après filtrage, on obtient le résultat de la détection (figure 5.9 (2)), les «contours en mouvement», c'est-à-dire soit des contours d'objets mobiles, soit des contours d'objets du fond dévoilés par un objet mobile.

A ce stade, on peut considérer disposer d'une détection de mouvement «binaire» (un objet a bougé dans la scène). Pour disposer d'une information plus pertinente, on doit résoudre deux problèmes : (1) repérer l'intégralité de l'objet mobile, au lieu d'une partie de ses contours; (2) éliminer les contours d'objets du fond détectés en mouvement.

Que représenter sur le deuxième bit de codage de l'image courante? Puisqu'un bit donne une information de contours, il manque une information sur le niveau de gris des régions délimitées par ces contours pour effectuer une segmentation. Sur un bit, nous effectuons un codage en demi-teintes du niveau de gris par technique de Bayer.

Se présente à nouveau un choix important : comment utiliser le bit libre restant pour effectuer la segmentation? Nous avons décidé d'exploiter une caractéristique très forte de l'application, le fait que le capteur soit fixe. Le bit supplémentaire est donc utilisé pour garder en mémoire une image du fond statique, c'est-à-dire de la scène en l'absence d'objet mobile.

Nous utilisons le même type de codage pour l'image courante (figure 5.9 (3)) et pour le fond statique (figure 5.9 (4)) : un codage de Bayer. Dans l'exemple présenté, on code 5 niveaux de gris sur un pavé 2×2. A chaque image, la différence avec le fond statique est calculée (figure 5.9 (5)). En «seuillant» cette image de différence (par des calculs locaux simples sur deux plans mémoires), on obtient une approximation des objets n'appartenant pas au fond statique.

Il reste ensuite à fusionner les deux phases de l'algorithme, ce qui est fait par une reconstruction géodésique de l'image de détection dans l'image de différence seuillée. On obtient donc les objets mobiles *et* n'appartenant pas au fond statique (figure 5.9 (6)). La phase de détection reste essentielle, car elle seule permet de savoir s'il n'y a pas de mouvement, de manière à *remettre à jour* régulièrement le modèle de fond statique, qui va évoluer selon les conditions d'éclairement.

Cet algorithme a été implanté sur le prototype *Pulsar 2.2*, la figure 5.10 montre l'écran du banc de test, et la détection de mouvement effectuée en temps réel. Bien que rudimentaire, une telle implantation montre qu'on peut effectuer des traitements temporels pertinents même avec une mémoire de capacité ridicule.

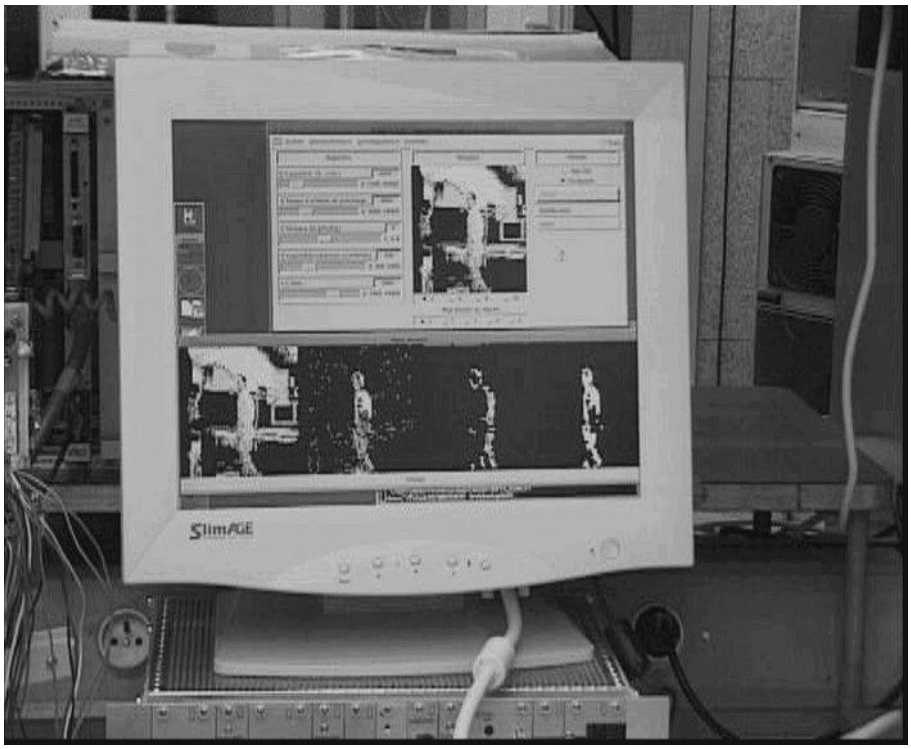


FIG. 5.10 – *Implantation de l'algorithme de segmentation d'objets mobiles sur le banc de test de* Pvlsar 2.2. *En bas de l'écran, on affiche quatre plans binaires présents simultanément dans la rétine. De gauche à droite : (1) Image courante en codage demi-teintes. (2) Différence filtrée de l'image précédente avec l'image de fond statique. (3) Image des contours en mouvement. (5) Résultat : reconstruction géodésique de (3) dans (2). Le cinquième plan, qui n'apparaît pas sur l'écran, contient l'image de fond statique.*

5.5.2 Conclusion

L'algorithme présenté dans cette section, et implanté sur les 5 bits de mémoire de *Pvlsar 2.2*, produit en temps réel une segmentation grossière d'objets mobiles dans le plan focal. Sous sa forme actuelle, il permet de détecter de manière relativement correcte :

- la *présence* d'un objet mobile dans le champ du capteur.
- la *taille* qu'il occupe dans l'image.
- le *lieu* du mouvement dans l'image.

Ces propriétés correspondent à peine aux exigences minimales que nous avions fixées : fournir les coordonnées des rectangles englobant les objets mobiles. Afin de rendre le système de balise autonome plus robuste et plus efficace, il est nécessaire de mettre en œuvre des approches méthodologiques plus élaborées.

Examinons tout d'abord comment améliorer le traitement élémentaire développé précédemment. Pour obtenir une meilleure segmentation, une possibilité serait de faire coopérer notre segmentation différentielle avec une segmentation statique du type détection de zones homogènes. Toutefois, ces techniques échouent dès que l'objet est texturé.

Ensuite, il serait opportun de relâcher la contrainte de connaissance d'un fond statique. En effet, dans notre expérimentation de laboratoire, le type de rafraîchissement utilisé a conduit à un comportement satisfaisant, mais qu'en sera-t-il se produire dans un environnement peu contrôlé ? Un rafraîchissement permanent comme celui implanté peut conduire à des oscillations. A l'opposé, un rafraîchissement trop peu dynamique sera inadapté à des conditions naturelles où l'éclairement change tout le temps.

En l'absence de connaissance relative au fond statique, on ne pourra segmenter l'intégralité de l'objet mobile qu'en faisant certaines hypothèses de régularité des images et de cohérence du déplacement.

Cette analyse nous conduit à adopter une approche d'inspiration markovienne : l'attribution de l'étiquette fixe/mobile à chaque pixel doit être attribuée sur des critères de «confiance» plus fins que la simple connexité, et l'absence de fond statique nécessite de modéliser une certaine connaissance *a priori*.

Une option à considérer concerne la caractérisation du mouvement en chaque pixel, qui peut fournir de précieux renseignements : vitesse des objets, possibilité de distinguer plusieurs objets (rigides) mobiles sur des critères de cohérence du champ de déplacement. Sur le plan méthodologique, il est naturel de s'orienter vers des méthodes de type flot optique. Ces méthodes s'accompagnent toujours d'une régularisation, qui peut être réalisée par une approche markovienne.

Nous allons dans la section suivante étudier l'implantation d'une segmentation markovienne sur la rétine numérique. Précisons immédiatement que nous allons, dans une certaine mesure, relâcher un peu la contrainte sur la quantité de mémoire disponible par pixel. Cette contrainte, vertueuse dans le cadre de la squelettisation (cf. chapitre 3), apparaît ici comme un carcan. Nous continuerons en revanche à tenter de minimiser la complexité en espace, pour que puisse découler de ce travail le dimensionnement d'un nouveau circuit, que l'on souhaite plus adapté à des applications temporelles.

5.6 Segmentation markovienne

L'approche markovienne de la segmentation entre dans le cadre de la résolution probabiliste de problèmes inverses [70]. En analyse d'images, ces problèmes sont en général *mal posés* au sens de Hadamard, et leur résolution implique une régularisation, sous la forme d'une injection de connaissances *a priori* [71]. Classiquement cette régularisation correspond à une restriction de l'espace des solutions par ajout de contraintes. Dans l'approche probabiliste, en revanche, la solution est vue comme la réalisation la plus probable d'un phénomène aléatoire, et la régularisation consiste donc à modéliser le problème en terme de probabilité.

Le cadre markovien est particulièrement adapté à l'analyse d'images, pour les raisons suivantes :

- Les propriétés spatiales s'expriment de manière locale par la donnée d'une topologie et de relations de dépendance : c'est la notion de *champ de Markov*.
- Grâce au théorème de Hammersley-Clifford, ces propriétés peuvent être spécifiées de manière simple et souple, par le paramétrage d'un *champ de Gibbs*.
- Il est possible de générer des échantillons des champs spécifiés grâce aux simulations par *chaînes de Markov*.

La segmentation markovienne en imagerie a été principalement utilisée dans le cadre de la restauration d'images bruitées [40], [17], [94] et de la modélisation de textures [30]. Elle a été utilisée plus récemment dans le contexte de la détection de mouvement [21], [73]. La littérature consacrée depuis à ce thème a traité des aspects multi-échelles [120], [26], ou de la mise en œuvre sur des architectures temps réel [25], [54]. Sur le principe cependant, la méthode utilisée dans le cadre de la segmentation d'objets mobiles est toujours la même : le *champ de Markov caché* (celui dont on recherche la réalisation la plus probable) est un champ binaire correspondant

```
// (a) Calcul de (I_t − I_{t+1}), en complément à 2
Pour chaque bit incident b, faire :
        pour i = 0 à n − 1, faire :
                b_i = b_i △ b;
                b = b ∧ b_i;
        s = s ∨ b;
           // (b) Calcul de |I_{t+1} − I_t|
b = s→b_0;
pour i = 1 à n − 2, faire :
        b_i = b_i △ b △ s;
        b = b→b_i;
b_{n−1} = b_{n−1} △ b △ s;
```

TAB. 5.1 – *Différence d'images destructive codée sur n bits. Le temps de calcul est équivalent à* $2n \times 2^n$. *L'algorithme utilise* $n+2$ *registres binaires.*

à la segmentation fixe/mobile. La donnée dont on dispose en entrée, et qui sert d'initialisation au champ estimé est une différence d'images.

Nous allons d'abord nous intéresser au calcul de la différence d'images, qui n'est pas un problème tout à fait trivial pour la rétine programmable, puis nous présenterons le contexte mathématique de la segmentation markovienne, avant d'aborder le problème de son implantation.

5.6.1 Différence d'images

Dans une séquence d'images $(I_t)_t$, pour calculer la valeur absolue de la différence $|I_{t+1} - I_t|$, il faut effectuer une opération *arithmétique* élémentaire (la soustraction), et calculer une valeur absolue à l'aide d'une suite d'opérations booléennes. Il convient bien entendu de minimiser le nombre de registres binaires nécessaire au calcul. L'algorithme qui apparaît dans le tableau 5.1 est dû à Basset [4]. Il calcule la valeur absolue de la différence entre une image I_t codée sur les n bits $\{b_0, ..., b_{n-1}\}$ dans la rétine et une image I_{t+1} en cours d'acquisition. Les bits incidents b correspondent aux $2^n - 1$ coupes binaires de I_{t+1}.

Dans la partie (a) de la procédure, pour chaque bit b, on décrémente de b la valeur codée sur les bits $\{b_0, ..., b_{n-1}\}$. On voit que b représente la retenue propagée. Si elle se propage au delà du bit b_{n-1}, alors le $(n+1)$ème bit noté s vaut 1. Cela signifie que le résultat est négatif. Le résultat obtenu sur les bits $\{b_0, ..., b_{n-1}, s\}$ correspond finalement au codage de $(I_t - I_{t+1})$ *en complément*

à 2, i.e. $(I_t - I_{t+1}) = \sum_{i=0}^{n-1} b_i 2^i - s2^n$.

Si la différence est négative, la valeur absolue vaut $2^n - \sum_{i=0}^{n-1} b_i 2^i$. Ce calcul est effectué par la partie (b) : pour tous les bits b_i, i variant de 0 à $n-1$, on laisse b_i inchangé tant que $b_i = 0$, puis on inverse tous les suivants si $s = 1$. Notons que le bit b sert ici de «bascule».

Remarquons que cette différence est destructive : la valeur de I_t est perdue. En revanche, elle n'utilise que 2 registres en plus de ceux qui codent les images.

5.6.2 Champs de Markov

Définition 41 *Une* suite d'images aléatoire X *est une fonction de* $\Omega \times \mathbb{Z}^2 \times \mathbb{N}$ *dans* $\{0, ..., N\}$, *où :*

- Ω *est l'univers probabiliste,*
- $\mathbb{Z}^2$ *est l'espace discret,*
- $\mathbb{N}$ *est le temps discret,*
- $N \in \mathbb{N}$ *est le nombre de niveaux de gris.*

Pour $s \in \mathbb{Z}^2 \times \mathbb{N}$, on notera $X(s) : \Omega \to \{0, ..., N\}$ la *variable aléatoire au site s.*

Pour $\omega \in \Omega$, on notera $X(\omega) : \mathbb{Z}^2 \times \mathbb{N} \to \{0, ..., N\}$ la *suite d'images correspondant à l'événement* ω.

On notera $\mathbb{S} = \mathbb{Z}^2 \times \mathbb{N}$ l'ensemble des sites, $\mathbb{V} = \{0, ..., N\}$ l'ensemble des niveaux de gris, et $E = \mathbb{V}^{\mathbb{S}}$ l'ensemble des suites d'images.

Nous nous intéressons à un *modèle probabiliste* de séquence d'images, qui contient en outre les notions suivantes :

- une *tribu* $\mathcal{T}$ sur Ω, et une *probabilité* P sur $\mathcal{T}$.
- une *topologie* sur $\mathbb{S}$, qui détermine les *relations de dépendance* entre les variables aléatoires $X(s)$.

Pour $s \in \mathbb{S}$ et $x_s \in \mathbb{V}$, on notera $X(s) = x_s$ l'événement $X(s)^{-1}(x_s) \in \mathcal{T}$.

Pour A et B appartenant à $\mathcal{T}$, la *probabilité conditionnelle de A sachant B* est la grandeur notée $P(A/B)$, et définie par :

$$P(A \cap B) = P(B) \times P(A/B).$$

De cette définition est déduite immédiatement la *loi de Bayes* :

$$P(A/B) = \frac{P(B/A) \times P(A)}{P(B)}.$$

La topologie sur $\mathbb{S}$ est caractérisée par la donnée d'un *système de voisinages* $\mathcal{V}$, défini par une fonction de $\mathbb{S}$ dans $\mathcal{P}(\mathbb{S})$ telle que, pour tout couple (s,r) dans $\mathbb{S}^2$, $s \notin \mathcal{V}(s)$ et $s \in \mathcal{V}(r) \Rightarrow r \in \mathcal{V}(s)$.

Le *principe de connaissance locale* inhérent à la rétine numérique que nous avons déjà évoqué dans le premier chapitre implique que dans l'implantation d'un modèle probabiliste sur la rétine, les probabilités conditionnelles doivent être calculables dans un voisinage restreint, ce qui nous amène à la notion de champ de Markov :

Définition 42 *La suite d'images aléatoire X est un* champ de Markov *relativement au système de voisinages $\mathcal{V}$ si et seulement si, pour tout $s \in \mathbb{S}$, et pour tout $x_r \in \mathbb{V}$, on a :*

$$P(X(s) = x_s / X(r) = x_r; r \neq s) = P(X(s) = x_s / X(r) = x_r; r \in \mathcal{V}(s)).$$

De plus, le *principe d'invariance en translation* implique une certaine régularité de la topologie (au moins dans l'espace), ce qui ajoute des contraintes sur le système de voisinages $\mathcal{V}$. Cependant, il reste difficile à ce stade de spécifier un modèle directement à partir des probabilités conditionnelles. Si l'on y parvient, le comportement du champ aléatoire ne peut pas être prévu intuitivement à partir de ces probabilités conditionnelles [16]. Ce problème a été résolu par Hammersley et Clifford qui ont montré l'équivalence entre un champ de Markov et un champ de Gibbs (ou champ de Boltzmann).

Définition 43 *Soit U une fonction de $\mathbb{E}$ dans $\mathbb{R}$.*
La probabilité définie sur $\mathcal{T}$ telle que :

$$P(X = x) = \frac{e^{-U(x)}}{Z},$$

avec $Z = \sum_{x \in E} e^{-U(x)}$ (constante de normalisation), est appelée mesure de Gibbs *(ou* mesure de Boltzmann*) d'*énergie *U.*

Définition 44 *Soit $C \subset \mathbb{S}$. C est une* clique *pour le système de voisinages $\mathcal{V}$ si et seulement si : s est un singleton, ou pour tout couple $(s,r) \in C^2$, $r \in \mathcal{V}(s)$.*

Par exemple (figure 5.11) dans $\mathbb{Z}^2$ muni de la 4-connexité, chaque site appartient à 1 clique d'ordre 1 et 4 cliques d'ordre 2. Pour la 8-connexité, chaque site appartient en outre à 4 autres cliques d'ordre 2, ainsi qu'à 12 cliques d'ordre 3 et 4 cliques d'ordre 4.

Définition 45 *Une mesure de Gibbs est dite* associée *au système de voisinages $\mathcal{V}$ si et seulement si son énergie U s'écrit, pour tout $x \in \mathbb{E}$:*

$$U(x) = \sum_{C \in \mathcal{C}} V_C(x),$$

où $\mathcal{C}$ est l'ensemble des cliques de $\mathcal{V}$, et $V_C(x)$ est une fonction dite potentiel *qui dépend des valeurs $\{x(s)\}_{s \in C}$.*

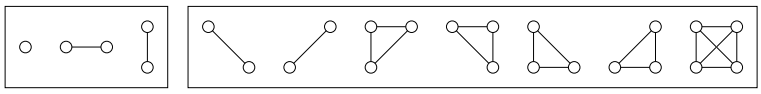


FIG. 5.11 – *La forme des différentes cliques sur $\mathbb{Z}^2$ en 4-connexité (1) et en 8-connexité (1 et 2).*

Le théorème suivant exprime l'équivalence entre champs de Markov et champs de Gibbs. Une preuve est donnée par Besag dans [16].

Théorème 9 *Hammersley-Clifford*
Soit X un champ aléatoire à valeur dans $\mathbb{E}$ tel que $\forall x \in \mathbb{E}, P(X = x) > 0$. X est un champ de Markov relativement au système de voisinages $\mathcal{V}$ si et seulement si *sa distribution $P(X = x)$ est une mesure de Gibbs associée à $\mathcal{V}$.*

Ce résultat signifie qu'on peut décrire un modèle de champ de Markov dans une topologie donnée en spécifiant les fonctions potentiels attachées à chaque clique. De plus, la donnée de la fonctionnelle d'énergie permet de prévoir plus intuitivement le comportement du champ aléatoire, puisqu'une réalisation est d'autant plus probable que son énergie est faible.

Nous allons à présent définir un modèle adapté à notre problème par la spécification d'une fonctionnelle d'énergie.

5.6.3 Définition du modèle

Le modèle d'énergie que nous utilisons est celui des *potentiels à niveaux*, qui n'est autre qu'un modèle d'Ising pondéré. L'énergie se calcule en chaque site par une somme sur l'ensemble des cliques contenant le site, de grandeurs qui valent plus ou moins une certaine constante (qui ne dépend que de la nature de la clique) selon que les valeurs prises par les sites de la clique sont différentes ou non.

Si on se limite par exemple aux cliques d'ordre 2, la fonction d'énergie U associée à notre champ de Gibbs est donnée, pour tout $x \in \mathbb{E}$, par :

$$U(x) = \sum_{s \in \mathbb{S}} U_s(x),$$

où $U_s(x)$, l'*énergie locale* au site s (qui ne dépend que des valeurs de x prises dans les cliques contenant s), est donnée par :

$$U_s(x) = \sum_{r \in \mathcal{V}(s)} V_x(s, r),$$

$$\begin{aligned} \text{avec } V_x(s,r) \quad &= -\beta_{sr} \text{ si } x(s) = x(r), \\ &= +\beta_{sr} \text{ si } x(s) \neq x(r). \end{aligned}$$

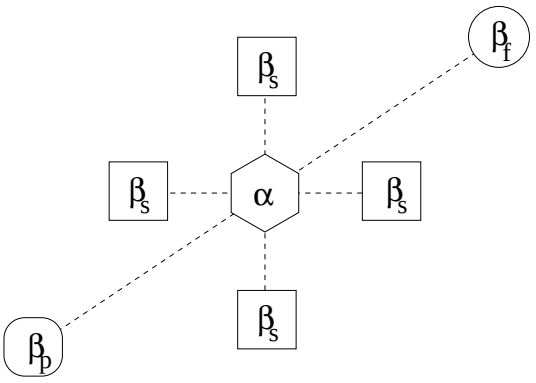


FIG. 5.12 – *Les paramètres intervenant dans le calcul de la fonctionnelle d'énergie dans le cas du 6-voisinage spatio-temporel.*

Les termes β_{sr} sont des constantes qui ne dépendent que de la nature du voisinage. Il reste donc à définir la topologie, puis à déterminer une pondération des liens de voisinage.

La topologie que nous utilisons fait intervenir deux types de lien :

- Une topologie «standard» sur $\mathbb{Z}^2 \times \mathbb{N}$, qui permet de modéliser une certaine régularité spatio-temporelle.
- Un lien à l'initialisation (qui correspond dans notre cas à une différence d'images seuillée), pour autoriser dans une certaine mesure l'existence de «discontinuités».

La figure 5.12 donne un exemple de topologie pondérée dans le cas où l'on considère un 6-voisinage spatio-temporel. On a ici 7 voisins par site, et 4 niveaux de potentiels différents : le poids accordé à chaque voisin spatial β_s, au voisin passé β_p, au voisin futur β_f, et enfin le poids accordé à l'initialisation α.

Le fait de fonder un modèle sur des compromis entre régularisation et discontinuités est extrêmement classique en modélisation markovienne (Geman et Geman, par exemple, considèrent deux topologies dans [40]), mais aussi dans d'autres représentations (Energie interne *vs* énergie externe dans l'évolution d'un contour actif [114]). Nous pouvons en outre interpréter le modèle que nous venons de définir dans le cadre de l'estimation bayesienne dite du *maximum a posteriori* (MAP) :

Interprétation en terme d'estimation du MAP

L'énergie que nous avons définie peut s'écrire comme la somme de deux termes :

$$U_s(x) = U_s^1(x) + U_s^2(x, o).$$

Le terme U^1 est appelé *énergie du modèle*. Il exprime une hypothèse de *régularité*, exprimée par des potentiels qui mesurent la *disparité*. Si l'on note $\mathcal{V}'(s)$ le sous-ensemble de $\mathcal{V}(s)$ qui ne contient que les voisins strictement spatio-temporels, on a :

$$U_s^1(x) = \sum_{r \in \mathcal{V}'(s)} V_x(s, r).$$

Le terme U^2 est appelé *énergie d'adéquation*. Il assure un lien significatif entre le résultat de la segmentation et les données du problème. Son rôle consiste à éviter que le terme de régularisation représenté par l'énergie du modèle n'éloigne trop le résultat de l'initialisation. Or, l'initialisation est obtenue à partir d'un *champ d'observation* O représentant la différence d'images, par seuillage. On a précisément :

$$U_s^2(x, o) = \alpha((o(s)_\tau \triangle x(s)) - (o(s)_\tau \bar{\triangle} x(s))),$$

où α est le niveau de potentiel correspondant à l'initialisation, et si x est une valeur numérique telle que $0 \leq x \leq N$, alors pour τ tel que $0 \leq \tau \leq N$, x_τ est la variable binaire définie par $x_\tau = 1 \Leftrightarrow x \geq \tau$.

Ainsi, le champ des différences $O = o$ étant connu, et moyennant une hypothèse sur le caractère markovien des champs considérés, la réalisation x qui *minimise l'énergie* $U(x)$ est aussi celle qui *maximise* le produit :

$$\max_x P(X = x) P(O = o / X = x),$$

avec :

$$P(X = x) = \frac{e^{-U^1(x)}}{Z_1} \text{ et } P(O = o / X = x) = \frac{e^{-U^2(x,o)}}{Z_2},$$

où Z_1 et Z_2 sont les constantes de normalisations appropriées.

La loi de probabilité sur les observations O étant supposée uniforme, la loi de Bayes implique que la réalisation x minimisant l'énergie $U(x)$ maximise la probabilité conditionnelle, soit :

$$\max_e P(X = x / O = o).$$

Cela correspond au critère d'estimation du *maximum a posteriori* (MAP), qui consiste à trouver à partir de la donnée observée $O = o$, l'étiquette x la plus probable.

Le calcul de l'énergie est donc déterminé par une pondération des liens spatio-temporels et de la configuration initiale correspondant à un seuil de la différence d'images. Les paramètres à définir sont donc les suivants :

- $\{\beta_i\}$ les poids des voisins spatio-temporels, dont la valeur dépend du type de lien,
- α le poids attaché à l'initialisation,
- τ la valeur de seuillage de la différence.

La détermination de ces paramètres est effectuée dans une large mesure par tâtonnement, encore que l'étude du comportement de l'algorithme sur des

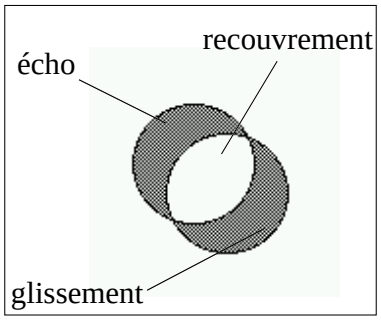


FIG. 5.13 – *Les différentes régions dans une image de différence.*

cas simples permettent de fournir un certain nombre de contraintes. Les poids spatiaux correspondent à une hypothèse de régularité des images segmentées. Le champ est isotrope dans l'espace, il n'y a donc qu'un paramètre spatial β_s. Les poids temporels expriment une hypothèse de cohérence du mouvement. On attribue des valeurs différentes aux poids temporels passé et futur (resp. β_p et β_f), pour la raison suivante : lorsqu'un objet uniforme tel qu'un disque se déplace, la différence d'image ressemble à la figure 5.13. Les zones de forte intensité correspondent à la différence symétrique des positions de l'objet aux instants t et $t+1$. Dans le cadre de la segmentation, on souhaite éliminer la zone d'écho, qui correspond aux régions qui ne sont plus occupées par l'objet au temps $t+1$, et récupérer la zone de recouvrement, qui correspond à l'intersection des positions de l'objet aux deux instants. On introduit donc un déséquilibre dans la direction temporelle pour faciliter l'effacement de l'écho. De plus, on cherche à obtenir un bon équilibre entre les poids spatiaux d'une part et les poids temporels et le poids d'adéquation d'autre part, cela afin de permettre une régularisation spatiale correcte sans pour autant demeurer dans le cadre du filtrage spatial non-linéaire. Enfin, nous allons voir qu'il est intéressant, pour des raisons techniques, de considérer que la somme de toutes les pondérations s'exprime sous la forme $2^n - 1$.

5.6.4 Chaînes de Markov et simulation

La fonctionnelle d'énergie étant définie, nous en venons au problème de la minimisation. On recherche la réalisation la plus probable du champ de Markov que nous venons de définir en minimisant l'énergie totale du champ de Gibbs correspondant. Bien entendu, cela suppose qu'on soit capable d'obtenir une réalisation du champ aléatoire qui correspond au modèle défini.

Le principe employé est de construire une suite de champs aléatoires

$(X_n)_n$ telle que $\lim_{n\to+\infty} P(X_n = x) = \Pi(x)$, où $\Pi(x)$ est la mesure de Gibbs de notre modèle. Une telle suite est réalisée par la construction d'une *chaîne de Markov* qui posséde Π comme *mesure invariante.*

Définition 46 *Une suite $(X_n)_n$ de variables aléatoires à valeurs dans $\mathbb{E}$ est une* chaîne de Markov *(d'ordre 1) si et seulement si, pour tout $n \in \mathbb{N}$, pour tous $x_0, ..., x_n$ de $\mathbb{E}$ tels que $P(X_0 = x_0, ..., X_n = x_n) > 0$, et pour tout x_{n+1} de $\mathbb{E}$:*

$$P(X_{n+1} = x_{n+1}/X_n = x_n, ..., X_0 = x_0) = P(X_{n+1} = x_{n+1}/X_n = x_n).$$

Définition 47 *Une mesure Π est invariante pour la chaîne $(X_n)_n$, si et seulement si, pour tout $n \in \mathbb{N}$ et pour tout $x \in \mathbb{E}$:*

$$P(X_n = x) = \Pi(x) \Rightarrow P(X_{n+1} = x) = \Pi(x).$$

Nous présentons à présent deux algorithmes de *simulation* classiques : l'échantillonneur de Gibbs [40] et l'algorithme de Metropolis [76], qui permettent de réaliser de telles chaînes.

Echantillonneur de Gibbs [40]

Considérons un état initial $X_0 = x_0$ quelconque. Pour $n \geq 0$, on applique la procédure suivante :

1. Tirer un site s au hasard (Loi uniforme sur $\mathbb{S}$).
2. Calculer les probabilités conditionnelles :
   $P(X_{n+1}(s) = x_s/X_n(r) = x_r, r \neq s) = P(X_{n+1}(s) = x_s/X_n(r) = x_r, r \in \mathcal{V}(s))$.
3. Effectuer un tirage de X_{n+1} en fonction de cette loi.

Précisons que les probabilités conditionnelles se calculent de manière entièrement locale. En effet, on a, par définition :

$$P(X(s) = x_s/X(r) = x_r, r \neq s) = \frac{P(X=x)}{P(X(r)=x_r, r\neq s)}.$$

Or, $P(X = x) = \dfrac{\exp(-\sum_{C\in\mathcal{C}} V_C(x))}{Z}$.

Si l'on note $\mathcal{C}_s$ (resp. $\mathcal{C}'_s$) l'ensemble des cliques qui contiennent (resp. ne contiennent pas) s, on a :

$$P(X = x) = \frac{\exp(-\sum_{C\in\mathcal{C}_s} V_C(x) - \sum_{C\in\mathcal{C}'_s} V_C(x))}{Z}.$$

Et $P(X(r) = x_r, r \neq s) = \sum_{v\in\mathbb{V}} P(X(r) = x_r, r \neq s, X(s) = v)$.

Si l'on note $\tilde{x} \in \mathbb{E}$ tel que $\tilde{x}(s) = v$ et $\tilde{x}(r) = x(r)$ pour $r \neq s$, alors on a :

$$P(X(r) = x_r, r \neq s) = \sum_{v\in\mathbb{V}} \frac{\exp(-\sum_{C\in\mathcal{C}_s} V_C(\tilde{x}) - \sum_{C\in\mathcal{C}'_s} V_C(\tilde{x}))}{Z}.$$

Or, clairement : $\sum_{C \in \mathcal{C}'_s} V_C(\tilde{x}) = \sum_{C \in \mathcal{C}'_s} V_C(x)$.
On a donc finalement :

$$P(X(s) = x_s / X(r) = x_r, r \neq s) = \frac{\exp(-\sum_{C \in \mathcal{C}_s} V_C(x))}{\sum_{v \in \mathbb{V}} \exp(-\sum_{C \in \mathcal{C}_s} V_C(\tilde{x}))}.$$

Algorithme de Metropolis [76]

Considérons un état initial $X_0 = x_0$ quelconque. Pour $n \geq 0$, on applique la procédure suivante :

1. Tirer un site s au hasard (Loi uniforme sur $\mathbb{S}$).
2. Tirer une étiquette e au hasard (Loi uniforme sur $\mathbb{V}$).
   On note $\tilde{x} \in \mathbb{E}$ tel que $\tilde{x}(r) = x_n(r)$ si $r \neq s$ et $\tilde{x}(s) = e$.
3. Calculer la différence $\Delta U = U(\tilde{x}) - U(x_n)$.
   Si $\Delta U < 0$, alors : $x_{n+1} = \tilde{x}$.
   sinon : $x_{n+1} = \tilde{x}$ avec une probabilité $e^{-\Delta U}$
   $= x_n$ avec une probabilité $1 - e^{-\Delta U}$

Ces deux algorithmes produisent des chaînes de Markov qui admettent Π comme mesure invariante. Cela implique que la suite des distributions converge vers Π, quel que soit l'état initial de la chaîne. Nous allons étudier la mise en œuvre de ces simulations par des algorithmes de relaxation qui consistent à remettre à jour itérativement l'ensemble des étiquettes selon un certain principe de décision, jusqu'à convergence.

Nous allons voir par la suite que le point clef réside dans la *représentation* de l'énergie. Celle que nous proposons permet de calculer simplement l'énergie associée au champ de Gibbs de notre modèle, ce qui mènera à une implantation très simple de la segmentation dans le cadre déterministe des I.C.M. (section 5.6.5). De plus, elle conduit à une mise en œuvre élégante de la relaxation stochastique par l'algorithme de Metropolis (section 5.6.6).

5.6.5 Relaxation déterministe

Le champ de Gibbs correspondant à notre modèle est défini par les poids β_i attribués à chaque type de voisinage dans la topologie choisie. Pour tout $x \in \mathbb{E}$, et pour chaque site $s \in \mathbb{S}$, rappelons que l'énergie locale est définie par $U_s(x) = \sum_{r \in \mathcal{V}(s)} V_x(s, r)$, les potentiels $V_x(s, r)$ étant définis par :

$$\begin{aligned} V_x(s,r) &= -\beta_{sr} \text{ si } x(s) = x(r). \\ &= +\beta_{sr} \text{ si } x(s) \neq x(r). \end{aligned}$$

Pour $e \in \mathbb{V}$, on notera $u_s(e)$ l'énergie locale *associée à l'étiquette* e telle que $u_s(e) = U_s(x) \Leftrightarrow x(s) = e$.

Nous sommes dans le cadre d'une segmentation binaire, donc $\mathbb{V} = \{0, 1\}$. Notre représentation de l'énergie consiste à coder, dans chaque site s, la grandeur W_s, définie comme suit :

$$W_s = \sum_{r \in \mathcal{V}(s)} w(s, r), \text{ avec :}$$
$$w(s, r) = \beta_{sr} \quad \text{si } e(r) = 1.$$
$$= 0 \quad \text{si } e(r) = 0.$$

Si la somme des poids (qui est constante sur $\mathbb{S}$) vaut $S = \sum_{r \in \mathcal{V}(s)} \beta_{sr}$, alors l'énergie locale associée à chaque étiquette se calcule par :

$$\begin{cases} u_s(0) = 2W_s - S \\ u_s(1) = S - 2W_s \end{cases}$$

Si l'on impose $S = 2^n - 1$, on a alors :

$$u_s(1) < u_s(0) \Leftrightarrow W_s > 2^{n-1}.$$

Autrement dit, en calculant W_s pour chaque site et en le codant sur n bits ($0 < W_s < 2^n$), le bit de poids fort du codage indique la valeur de l'étiquette correspondant à l'énergie locale minimale.

Le calcul de W_s est effectué par l'opérateur de rétine représenté sur la figure 5.14. Dans cet exemple, on a un 6-voisinage spatio-temporel, avec les coefficients suivants : $\beta_s = 2$, $\beta_p = 1$, $\beta_f = 3$ et $\alpha = 3$. La somme varie donc entre 0 et $4\beta_s + \beta_p + \beta_f + \alpha = 15$, elle est codée sur les 4 bits $\{p_0, ..., p_3\}$, et le résultat du vote majoritaire est simplement le bit de poids fort, p_3. Ce bit devient la nouvelle valeur estimée, sur laquelle le calcul est réitéré.

L'opérateur de rétine de la figure 5.14 fournit en fait l'implantation de l'algorithme d'optimisation entièrement déterministe dit *iterated conditional mode* (ICM) [17]. Il consiste à choisir à chaque itération, pour chaque site, l'étiquette entraînant la plus petite augmentation locale d'énergie. En utilisant les notations de la figure 5.14, l'algorithme peut être complétement décrit comme suit :

O_t est la donnée observée de coordonnée temporelle t.

$\hat{E}_t^{\,n}$ est l'étiquette estimée pour les sites de coordonnée temporelle t, après n itérations de relaxation.

$E_t = \hat{E}_t^{\,\infty}$ est l'étiquette attribuée aux sites de coordonnée temporelle t.

Pour $X \in \{0, ..., N\}^{\mathbb{Z}^2}$, et $\tau \in \{0, ..., N\}$ on note $X_\tau \subset \mathbb{Z}^2$ telle que $X_\tau = \{x \in \mathbb{Z}^2; X(s) \geq \tau\}$.

Pour une séquence d'images $\{I_t\}_{t \in \mathbb{N}}$, on a :

$O_t = (|I_{2t} - I_{2t-1}|)_{\tau_1}$

$\hat{E}_t^{\,0} = O_t$

Et pour $n > 0$:

$\hat{E}_t^{\,n} = (\beta_s(\hat{E}_t^{\,n-1}.Nord + \hat{E}_t^{\,n-1}.Est + \hat{E}_t^{\,n-1}.Ouest + \hat{E}_t^{\,n-1}.Sud) + \beta_p E_{t-1} +$

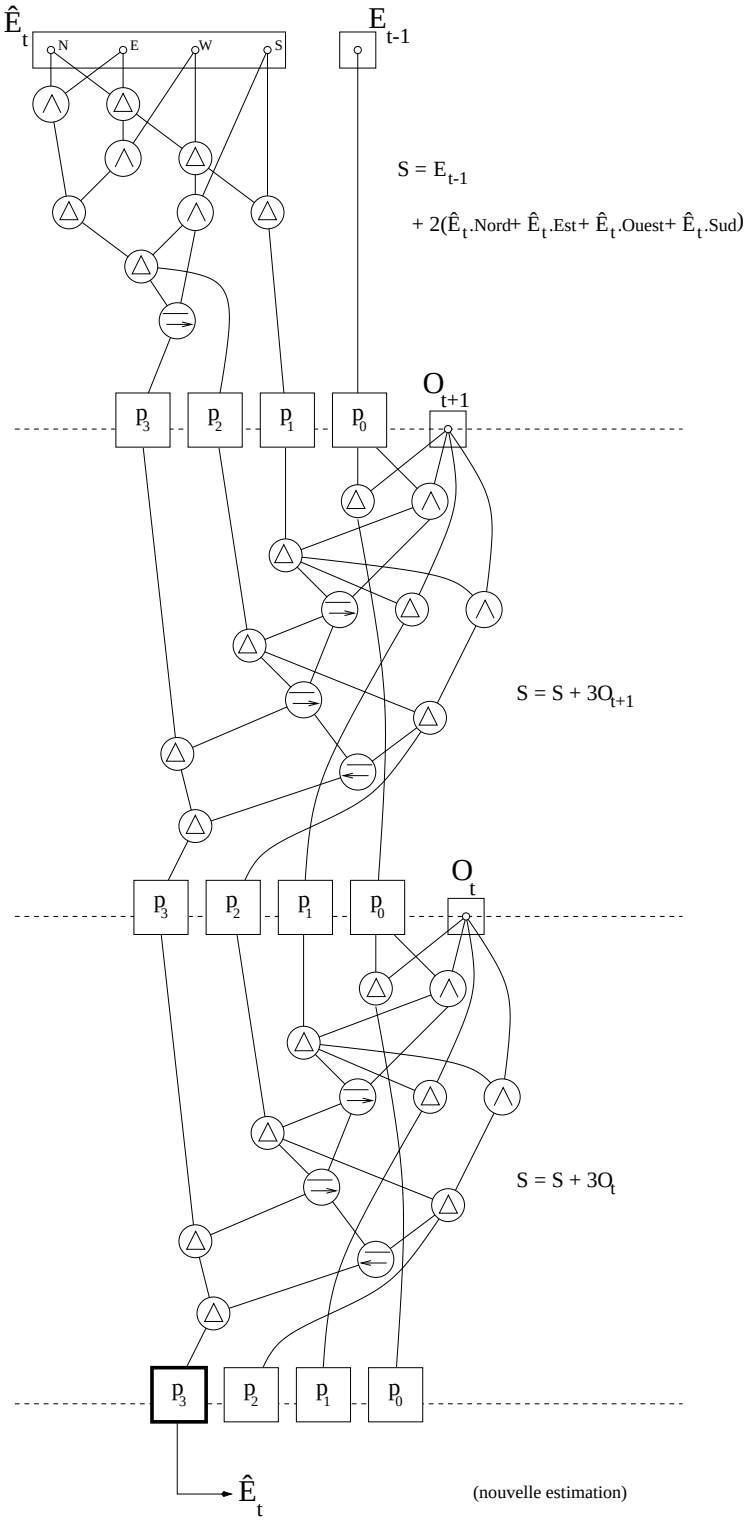


FIG. 5.14 – *Calcul de l'énergie pour la relaxation markovienne.*

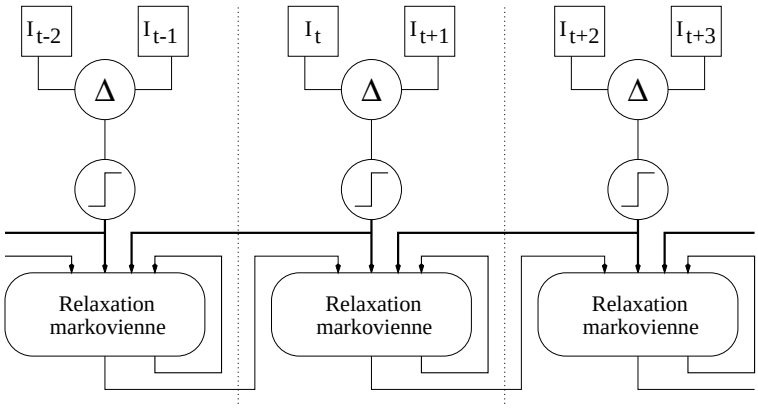


FIG. 5.15 – *Représentation synoptique de l'algorithme de segmentation d'objets mobiles.*

$\beta_f O_{t+1} + \alpha O_t)_{\tau_2}$.

L'algorithme de relaxation est décrit schématiquement figure 5.15. On calcule une différence d'images (destructive) qui est seuillée, puis on modifie itérativement la valeur de l'étiquette en fonction d'un vote majoritaire qui fait intervenir l'image elle-même, par les relations spatiales, l'image correspondant à la segmentation précédente, l'image correspondant à une estimation de la segmentation prochaine (qui n'est autre qu'une différence seuillée), et enfin l'image d'initialisation. La différence d'épaisseur des flèches correspond aux différentes pondérations intervenant dans le vote majoritaire.

La relaxation déterministe par I.C.M. est la méthode d'optimisation la plus répandue dans la littérature consacrée à la segmentation markovienne d'objets mobiles [21], [120],[25], [54]. Néanmoins, il est important de souligner les hypothèses fortes qui sont implicitement prises en compte par les méthodes déterministes. Tout d'abord, si l'on se place dans le cadre de la simulation d'une chaîne de Markov (section 5.6.4), on considère ici que *l'état initial de la chaîne constitue déja une réalisation crédible du champ de Gibbs correspondant à notre modèle.* Mais plus encore, il est nécessaire que l'initialisation ne soit pas trop éloignée de la solution optimale. En effet, lorsque les sites sont explorés de manière séquentielle, cet algorithme converge vers le premier minimum local de la fonction d'énergie.

Dans le cas d'une mise à jour parallèle des sites, comment peut-on garantir la convergence? Sur notre architecture SIMD, un calcul entièrement parallèle ne converge pas en général. L'exemple (mais cet exemple a-t-il un sens?) de l'image de l'échiquier tridimensionnel fournit pour notre problème l'oscillation maximale.

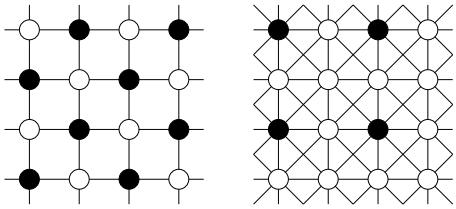


FIG. 5.16 – *Processeurs actifs (en noir) lors d'une mise à jour parallèle des sites. L'efficacité du calcul est égale à l'inverse du nombre chromatique de la topologie, soit* $1/2$ *pour la 4-connexité (à gauche) et* $1/4$ *pour la 8-connexité (à droite).*

Quoi qu'il en soit, la convergence ne peut être garantie que si deux sites voisins (au sens de $\mathcal{V}$) ne sont pas remis à jour simultanément. Pour une machine cellulaire, cela implique soit un fonctionnement asynchrone des processeurs [40], soit une dégénérescence périodique du parallélisme synchrone [70], [82]. Dans ce dernier cas, comme on ne met à jour que des sites ayant la même coordonnée temporelle, l'efficacité du parallélisme dépend de la topologie induite par $\mathcal{V}$ sur $\mathbb{Z}^2$, elle vaut $1/\chi$, χ étant le nombre chromatique du graphe de voisinage (figure 5.16), c'est-à-dire le nombre de couleurs minimum pour colorier les sites de telle sorte que deux sites voisins aient des couleurs différentes.

5.6.6 Relaxation stochastique

La minimisation par ICM se comporte comme une simple descente de gradient : on modifie une certaine configuration de telle sorte que la modification entraîne la plus grande diminution d'une fonction de coût. Un tel algorithme ne converge vers un minimum global que si la fonction de coût est *convexe*.

Les problèmes d'optimisation non convexe peuvent être résolus par la méthode du recuit simulé [50], qui trouve son origine en physique statistique, où l'état d'équilibre d'un système de particules peut être décrit par le minimum d'une fonctionnelle d'énergie. On peut prendre comme analogie le refroidissement d'un matériau en fusion possédant des propriétés de cristallisation. On constate que lors d'un refroidissement soudain, le matériau solide ne présente pas de structure microscopique régulière (verre), et au contraire que les molécules présentent une organisation régulière (cristal) dans le cas d'un refroidissement progressif.

La justification mathématique à la base du recuit simulé est donnée par

le théorème 10. Nous discutons ensuite sa mise en œuvre dans le cadre de la simulation d'une chaîne de Markov.

Définition 48 *La mesure de Gibbs d'énergie U et de* température T *est la probabilité :*

$$P(X = x) = \frac{e^{-\frac{U(x)}{T}}}{Z_T},$$

avec $Z_T = \sum_{x \in E} e^{-\frac{U(x)}{T}}$.

Théorème 10 *Recuit simulé*

1. *Lorsque T tend vers $+\infty$, la mesure de Gibbs d'énergie U et de température T tend vers la probabilité uniforme sur E.*
2. *Lorsque T tend vers 0, la mesure de Gibbs d'énergie U et de température T tend vers la probabilité uniforme sur l'ensemble $\{\mu_1, ..., \mu_n\}$ des minima de la fonction U.*

Ainsi, si l'on utilise la mesure de Gibbs d'énergie U et de *température* T dans des algorithmes de simulation tels que l'échantillonneur de Gibbs ou l'algorithme de Metropolis, on produit une chaîne de Markov dont la distribution converge vers la mesure $\Pi_T(x) = \frac{e^{-\frac{U(x)}{T}}}{Z_T}$. Si l'on fait tendre T vers 0, la distribution va tendre vers la mesure de Dirac sur le minimum μ (ou la probabilité uniforme sur l'ensemble des minima) de la fonction d'énergie U.

Examinons à présent les possibilités d'implantation d'une telle relaxation stochastique. La première condition est d'être capable de générer un nombre aléatoire. La figure 5.17 montre comment on peut, sur la rétine programmable, générer un bit aléatoire de loi uniforme sur $\{0, 1\}$ grâce à une seule instruction élémentaire.

Nous allons voir que ce dispositif est suffisant pour mettre en œuvre une simulation dans le cadre de l'algorithme de Metropolis.

Notons d'abord que l'implantation de l'échantillonneur de Gibbs est assez difficile. En effet, d'après la section 5.6.4, on doit faire en chaque site un tirage selon la loi :

$$P(X_{n+1}(s) = x_s / X_n(r) = x_r, r \in \mathcal{V}(s)) = \frac{e^{-u_s(x_s)}}{e^{-u_s(0)} + e^{-u_s(1)}},$$

où $u_s(x_s)$ est l'énergie locale en s lorsque $X(s) = x_s$.

La donnée de W_s permet bien sûr de calculer cette grandeur, mais son calcul semble *a priori* assez complexe.

En revanche, pour l'implantation de l'algorithme de Metropolis, l'implantation devient très simple si l'on sait effectuer un tirage sur $\{0, 1\}$ avec une probabilité exponentielle en fonction d'un nombre codé dans la rétine (figure 5.18). Pour tous les bits x_k pour k de 0 à $(n-1)$ de l'écriture binaire

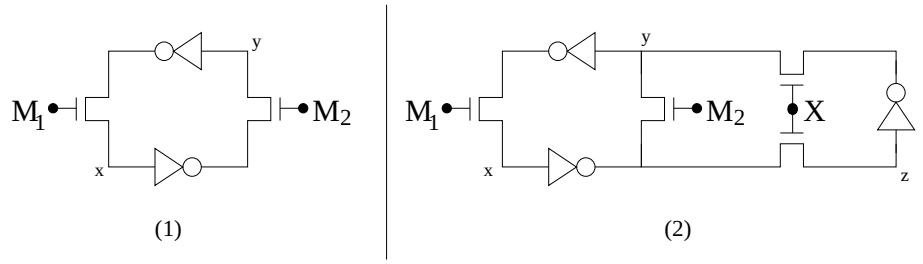


FIG. 5.17 – *Génération d'un bit aléatoire. (1) Le registre binaire semi-statique utilisé dans la rétine programmable est un double inverseur. Dans l'état statique on a $x = \bar{y}$; cet état est obtenu en actionnant simultanément les signaux de contrôle M_1 et M_2, instruction qui «valide» la valeur du registre. (2) Le générateur aléatoire : l'action simultanée des signaux M_1 et X crée un triple inverseur qui oscille à des fréquences se comptant en GHz, donc supérieures de plusieurs ordres de grandeur à la fréquence de contrôle. Par conséquent, quand on revient dans un état statique, x vaut 1 avec une probabilité $1/2$.*

de X, on effectue 2^k tirages aléatoires uniformes sur $\{0,1\}$, dont on calcule le OU avec la négation du bit x_k. Le ET de tous les bits de sortie des OU est égal à 1 avec une probabilité égale à 2^{-X}.

Le calcul de W_s fournit comme nous l'avons dit plus haut l'étiquette favorisée sur son bit de poids fort, noté $\hat{e}_d$, qui correspond au choix déterministe.

De plus, on a $W_s = \frac{u_s(0)+2^n-1}{2} = \frac{-u_s(1)+2^n-1}{2}$.

Donc, si $W_s \leq 2^{n-1}, 2^{n-1} - W_s = \frac{u_s(1)-u_s(0)}{4} + \frac{1}{2}$,

et si $W_s \geq 2^{n-1}, W_s - 2^{n-1} = \frac{u_s(0)-u_s(1)}{4} - \frac{1}{2}$.

Finalement, on a $|W_s - 2^{n-1}| \simeq \frac{|\Delta U|}{4}$.

Or, pour calculer la différence $|X - 2^{n-1}|$ pour un nombre X codé sur n bits dans la rétine, il suffit de calculer le XOR entre la négation du bit de poids fort et chacun des autres bits, le résultat est dans les $(n-1)$ premiers bits (à 1 près si le bit de poids fort vaut 0).

L'algorithme de Metropolis est finalement décrit par la figure 5.19. On sait comment calculer $|\Delta U|$, on sait effectuer un tirage sur $\{0,1\}$ selon une loi de Bernoulli exponentielle (haut et milieu de la figure). Ecartons pour l'instant la température et imaginons qu'on tire un nombre aléatoire K qui vaut 1 avec la probabilité $2^{-|\Delta U|}$. Le bas de la figure correspond à la fin de l'algorithme.

On tire un bit B au hasard avec une loi uniforme sur $\{0,1\}$ (c'est le tirage numéro 8 sur la figure 5.19) : Si $B = \hat{e}_d$, alors c'est que le bit tiré au sort

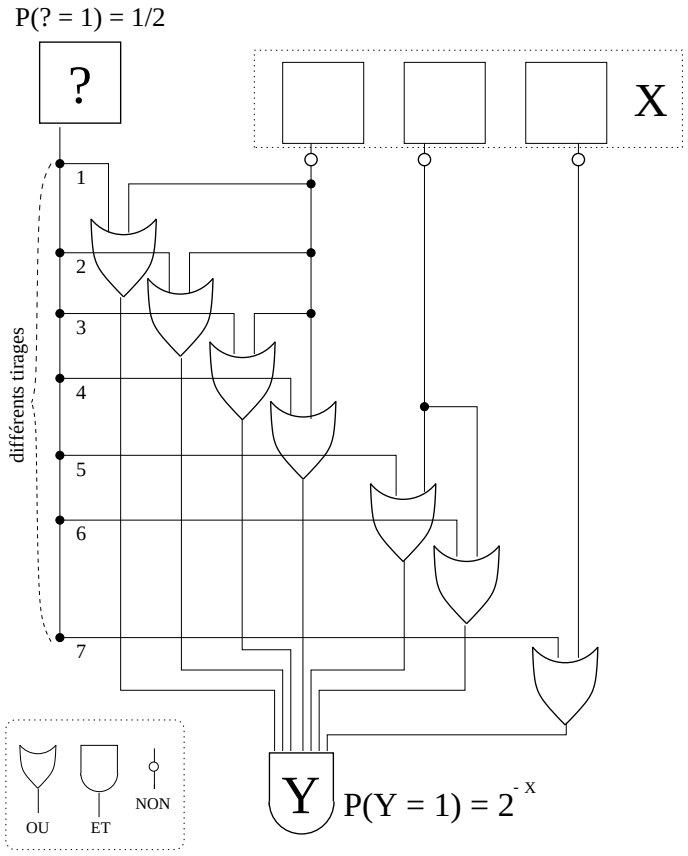


FIG. 5.18 – *Tirage d'une variable de Bernoulli avec* $p = 2^{-X}$.

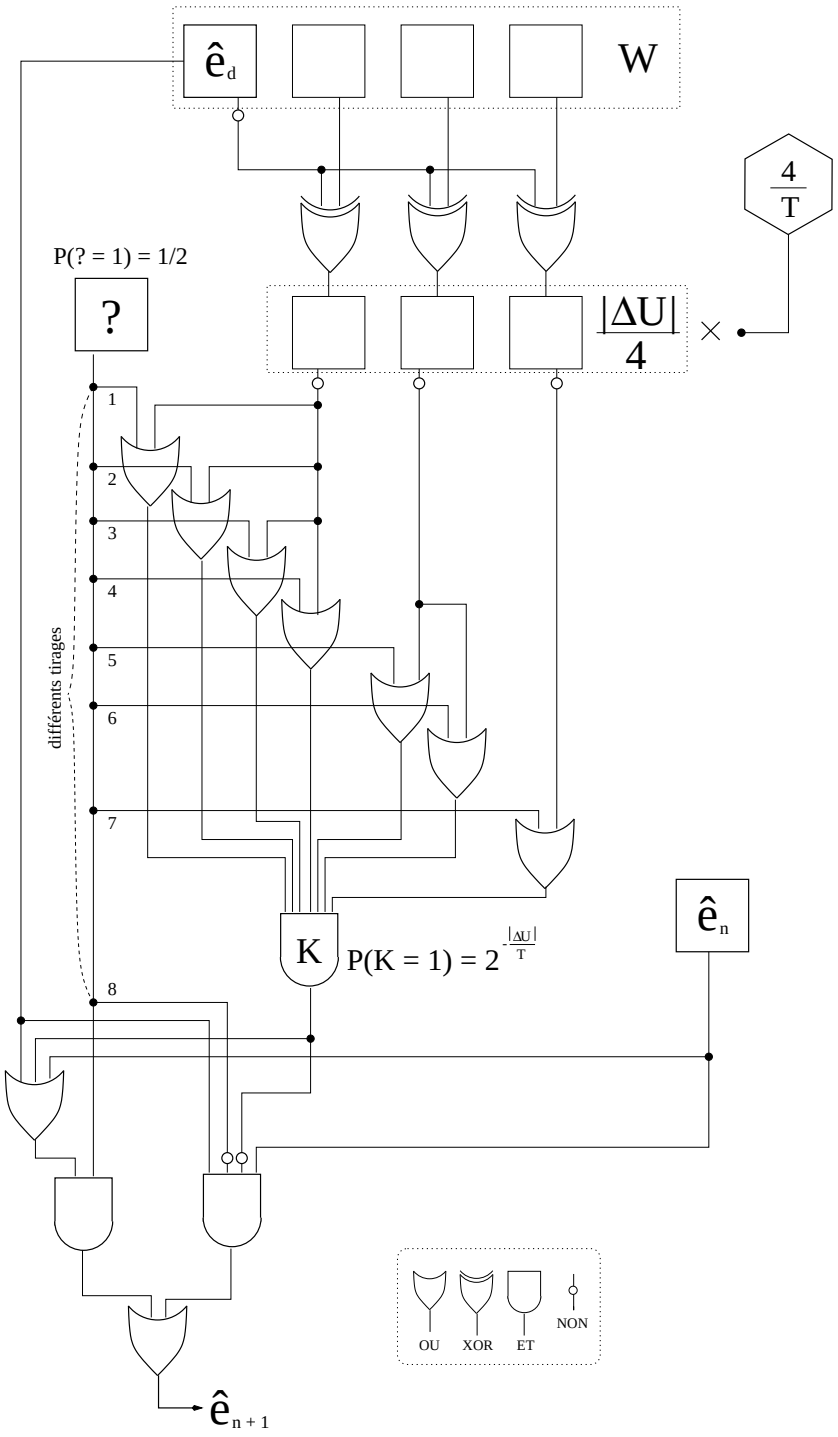


FIG. 5.19 – *Mise en œuvre de l'algorithme de Metropolis.*

entraîne un ΔU négatif, et donc $\hat{e}_{n+1} = B = \hat{e}_d$. Sinon, on s'en remet au résultat du tirage de K : si K vaut 1, alors $\hat{e}_{n+1} = B$, sinon $\hat{e}_{n+1} = \hat{e}_n$.

Après réduction logique, on obtient :

$$\hat{e}_{n+1} = ((\hat{e}_d \vee K \vee \hat{e}_n) \wedge B) \vee (\hat{e}_d \bar{\to} B \bar{\to} K \wedge \hat{e}_n).$$

C'est ce que calcule le circuit de la figure 5.19 (en bas).

On sait donc mettre en œuvre l'algorithme de Metropolis qui a pour mesure invariante $\Pi(x) = \frac{e^{-U(x)}}{Z}$. Pour converger vers un minimum global, il faut à présent tenir compte du paramètre T qui est la température. Supposons d'abord T constant. L'algorithme de Metropolis doit converger vers la distribution $\Pi_T(x) = \frac{\exp(\frac{-U(x)}{T})}{Z_T} = \frac{e^{-U_T(x)}}{Z_T}$. On voit bien que :

$$\Delta U_T = U_T(x_1) - U_T(x_2) = \frac{U(x_1)}{T} - \frac{U(x_2)}{T} = \frac{\Delta U}{T}.$$

Pour que l'algorithme de Metropolis décrit dans la section 5.6.4 converge vers la distribution de Gibbs de température T, il suffit donc de diviser la différence ΔU par T.

Il faut enfin faire décroître T, et le problème de la vitesse de décroissance est intimement lié à la facilité de calcul de la division. En effet, imaginons que T puisse décroître de manière exponentielle. Si l'on choisit $T_n = 2^{c-n}$, où c est une constante entière donnée, dans ce cas, la division correspond à un simple décalage «à droite» des bits codant ΔU dans la rétine. Malheureusement, un résultat dû à Geman et Geman [40] montre que la convergence vers un minimum global ne peut avoir lieu que si la décroissance de T est telle que $T_n \geq \frac{c^2}{\log(n+1)}$, pour une certaine constante c.

Dans notre algorithme, le nombre codé dans la rétine représente $\frac{|\Delta U|}{4}$. Nous devons donc multiplier ce nombre par la constante extérieure $\frac{4}{T}$. On peut supposer que pour tout n, $0 \leq \frac{4}{T_n} < \Delta_{max}$, où Δ_{max} est la valeur maximum prise par $\frac{|\Delta U|}{4}$, mais le schéma de décroissance implique que, dans la précision choisie, on puisse prendre toutes les valeurs entre 0 et Δ_{max}.

Ainsi, on considère Y une constante extérieure telle que $0 \leq Y < 2^n$. On note $Y = \sum_{|i|<n} y_i 2^i$. Pour un nombre X codé dans la rétine, on a $XY = \sum_{|i|<n} y_i X 2^i$. Si on veut coder XY sur autant de bits que X, on aura :

$$XY = \sum_{i=-n+1}^{n-1} y_i (X \ll i).$$

Où $(X \ll i)$ est le décalage de X de i bits vers la gauche si $i > 0$, vers la droite si $i < 0$. C'est ce que calcule la procédure du tableau 5.2.

Cette procédure, qui achève l'implantation du recuit simulé par l'algorithme de Metropolis, nécessite au maximum $2n - 1$ registres binaires et $O(n^3)$ opérations élémentaires, où n est le nombre de bits de l'opérande codé

$d = 0$;
Si $(m_p > 0)$ {
  Pour $i = (n-1)$ à $(n-1-m_p)$ {
    $d = d \vee x_i$;
    }
  Pour $i = (n-1)$ à 0 {
    Si $(i - m_p \leq 0)$ $x_i = x_{i-m_p}$; sinon $x_i = 0$;
    }
  }
Si $(m_p < 0)$ {
  Pour $i = 0$ à $(n-1)$ {
    Si $(i - m_p \leq n-1)$ $x_i = x_{i-m_p}$; sinon $x_i = 0$;
    }
  }
Pour $k = (p-1)$ à 0 {
  Pour $i = \max(m_k, 0)$ à $\min(n-1, n-1+m_k)$ {
    Si $(k = p-1)$ $b_i = x_{i+m_p-m_{p-1}}$; sinon $b_i = b_{i+m_k-m_{k-1}}$;
    $b_0 = b_i$;
    Pour $j = i$ à $(n-1)$ {
      $x_j = x_j \triangle b_0$;
      $b_0 = b_0 \bar{\rightarrow} x_j$;
      }
    $d = d \vee b_0$;
    }
  }
Pour $i = 0$ à $(n-1)$ {
  $x_i = x_i \vee d$;
  }

TAB. 5.2 – *Calcul de la multiplication d'un nombre X codé dans la rétine sur les n bits $\{x_0, ..., x_{n-1}\}$, par une constante extérieure $Y = \sum_{i=d}^{f} y_i 2^{-i}$, avec $-n+1 \leq d \leq f \leq n-1$, et $f - d < n$. On note $\{m_0, ..., m_p\}$ l'ensemble des indices tels que $y_j = 1 \Leftrightarrow j \in \{m_0, ..., m_p\}$, et $-n+1 < m_0 < ... < m_p < n$. Le résultat XY est codé sur les bits $\{x_0, ..., x_{n-1}\}$.*

dans la rétine. Il serait possible d'améliorer cette vitesse d'un facteur 2 en utilisant la classique technique de recodage de Booth (voir par exemple [81]).

5.6.7 Occupation mémoire et dimensionnement

Nous nous intéressons à présent au coût en mémoire global de l'algorithme, qui dépend essentiellement du nombre de bits utilisés pour coder les deux grandeurs suivantes :

- la différence d'images (n bits),
- la représentation de l'énergie (m bits).

La figure 5.20 montre comment évolue l'occupation de la mémoire pendant le déroulement de l'algorithme. La conclusion est que dans le cas de l'algorithme I.C.M., l'espace mémoire total occupé est $\max(m, n) + 4$, et dans le cas du recuit simulé, $\max(n + 4, 2m + 1)$.

Ce calcul détermine le dimensionnement d'une rétine numérique en nombre de registres pour calculer la segmentation sur une dynamique raisonnable. Expérimentalement, l'ordre de grandeur requis est une dizaine de bits par pixel. Pour les résultats que nous présentons dans la section suivante, 8 bits ont été employés pour le codage de la différence, et 4 ou 5 bits pour le codage de l'énergie. Ils correspondent donc à une rétine ayant 12 bits de mémoire par pixel.

5.6.8 Résultats et discussion

La figure 5.21 montre le résultat de la segmentation markovienne sur une séquence d'images. Ce résultat est comparé à la segmentation obtenue par la méthode présentée dans la section 5.5.

Il ressort que la qualité de la segmentation est en général sensiblement améliorée, et sans utiliser de modèle de fond statique.

Les figures 5.22 et 5.23 montrent différents résultats sur une autre séquence. Deux petites voitures se déplacent sur un tapis et se croisent. En haut à gauche, une main qui manipule les fils de nylon transparents se déplace. Tous ces algorithmes sont implantables sur 12 bits. Les résultats 5.22(2) et 5.23(1) correspondent à la relaxation déterministe I.C.M. Dans le premier cas, on se limite à un 4-voisinage spatial. La somme des poids vaut 15, et l'énergie est codée sur 4 bits. Dans le deuxième cas, on utilise un 8-voisinage spatial, limité aux cliques d'ordre 2. La somme des poids vaut 31, et l'énergie est codée sur 5 bits. Dans les deux cas, la convergence est extrêmement rapide : les résultats présentés sont obtenus avec 5 itérations complètes. (soit 10 passes dans le premier cas, et 20 passes dans le second.

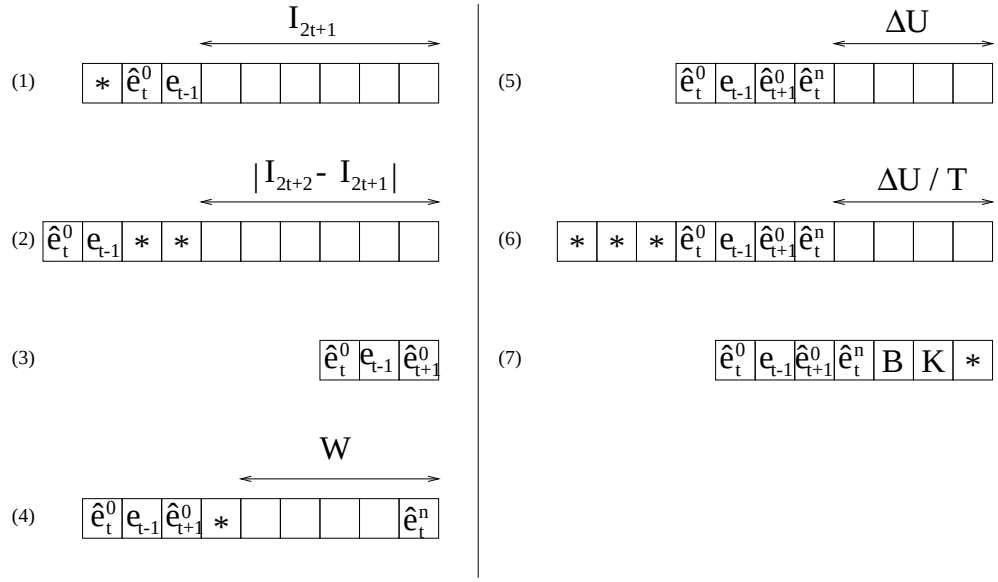


FIG. 5.20 – *Occupation des registres de mémoire pendant le déroulement de l'algorithme. Une case représente un registre, qui peut être utilisé pour garder une valeur booléenne en mémoire (les notations sont celles des sections précédentes), ou pour le calcul courant (les cases sont alors représentées par une étoile). La colonne de gauche correspond à la phase déterministe : (1) Codage de l'image courante. (2) Calcul de la différence d'images. (3) Seuillage de la différence. (4) Calcul de l'énergie. La colonne de droite correspond à la phase stochastique : (5) Calcul de ΔU. (6) Division par la température. (7) Tirages aléatoire et application de la règle de Metropolis.*

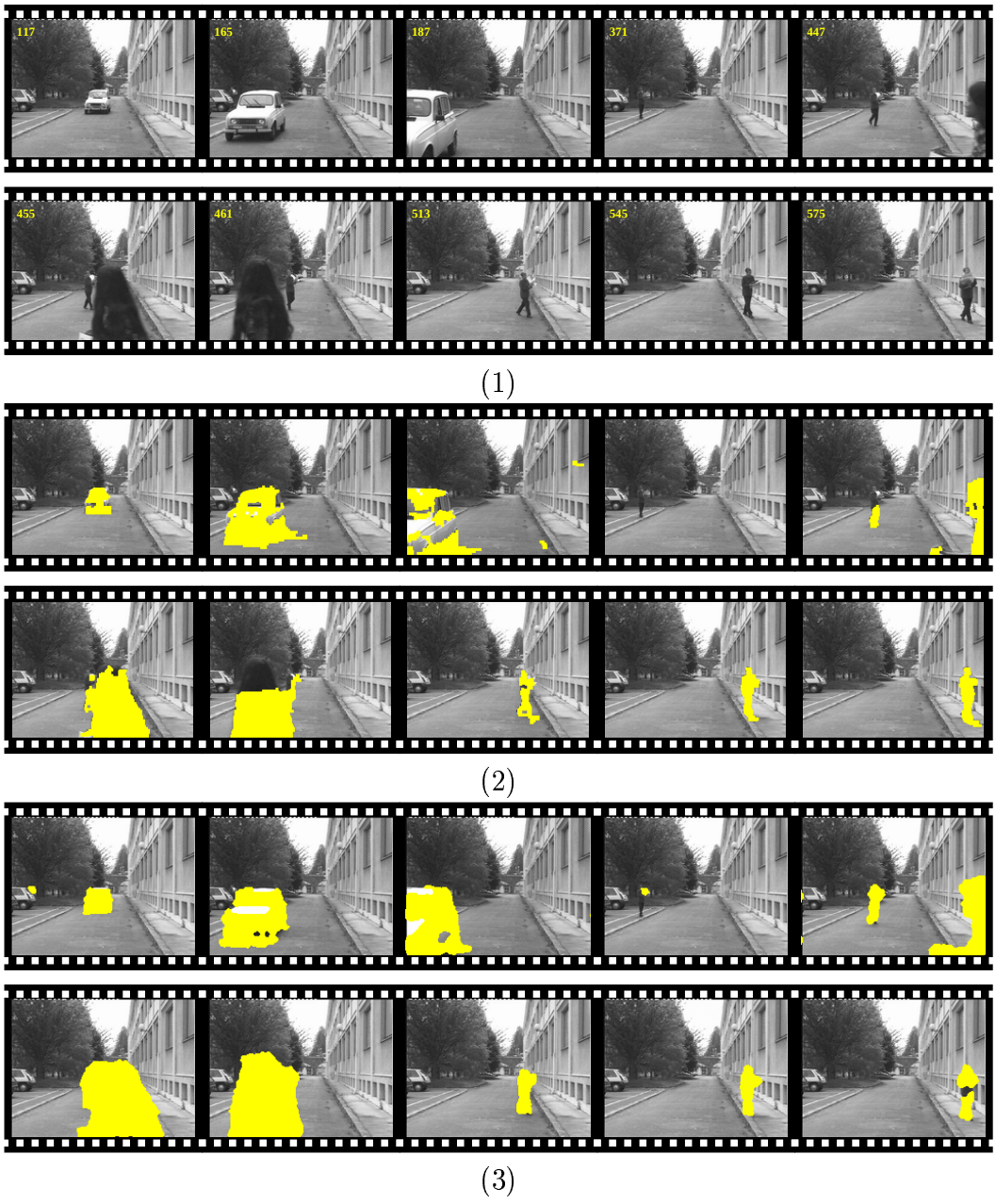
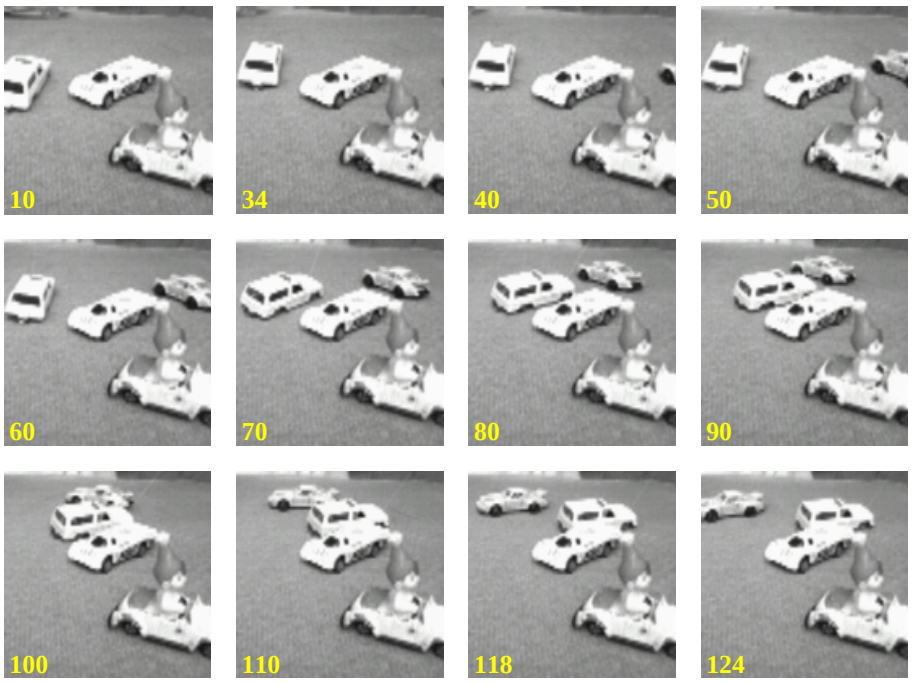
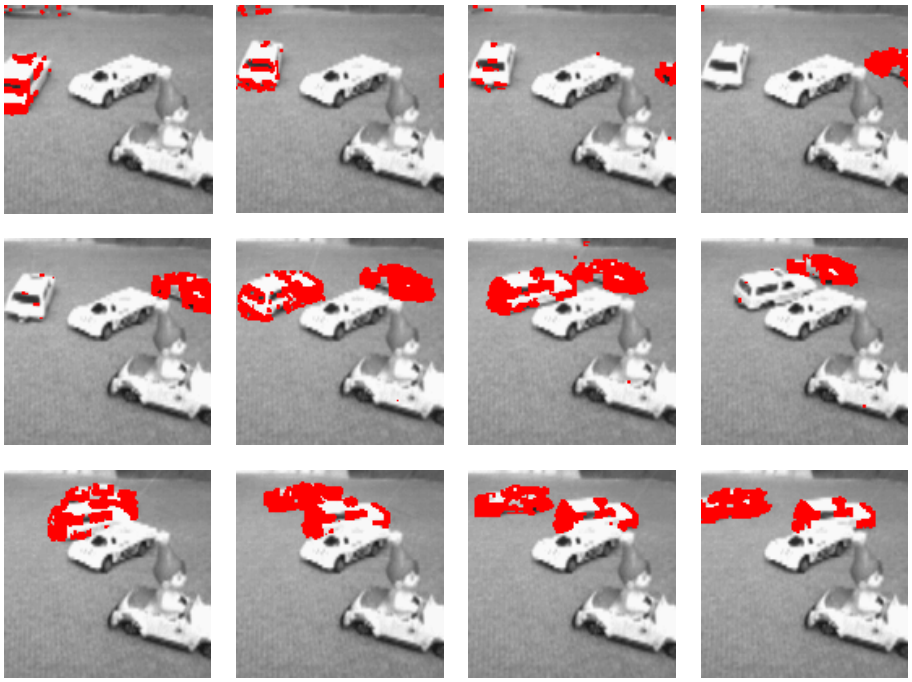


FIG. 5.21 – *Segmentation d'objets mobiles. (1) Séquence originale (On a indiqué le numéro de trame). (2) Segmentation par la méthode implantée sur* Pvlsar 2.2. *(3) Segmentation markovienne (Recuit simulé).*



(1)



(2)

FIG. 5.22 – *(1) Séquence originale. (2) ICM avec un voisinage spatial 4-connexe. L'algorithme utilise 12 bits pour le calcul. La différence d'images est codée sur 8 bits, l'énergie sur 4 bits. Le nombre d'itérations est 5.*

	Algo 1	**Algo 2**	**Algo 3**
Temps de calcul d'une itération à 10 MHz.	60 μs	110 μs	400 μs
Nombre d'itérations possibles à la cadence video, à 10 MHz.	1 250	650	180
Cadence d'acquisition possible à 10 MHz.	360 Hz	360 Hz	80 Hz
Fréquence de contrôle nécessaire à la cadence video.	700 kHz	700 kHz	4 MHz

TAB. 5.3 – *Performances des trois algorithmes présentés dans cette section. Algo 1 : ICM en 4-connexité. Algo 2 : ICM en 8-connexité. Algo 3 : Recuit simulé.*

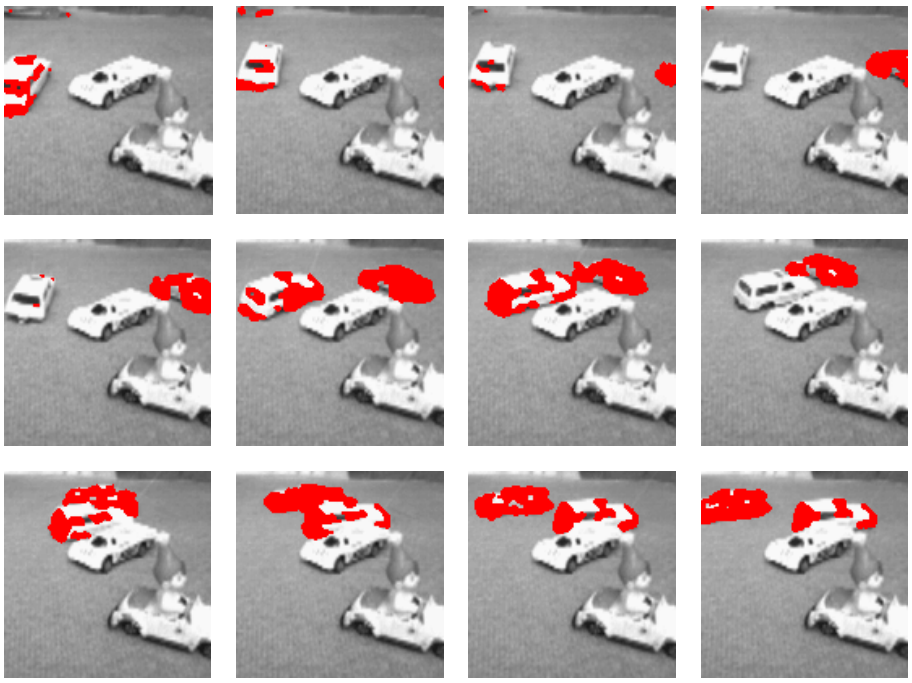
Le résultat 5.23(2) correspond à l'application du recuit simulé avec la règle de Metropolis, en utilisant un 8-voisinage spatial, et un schéma de décroissance de température linéaire. Les masques obtenus sont un peu plus réguliers, mais le nombre d'itérations est 10 fois plus important.

Il est important de préciser que nous avons également expérimenté l'application de ces 3 algorithmes de manière *entièrement parallèle*, et que dans tous les cas, les différences qu'on obtient dans la segmentation ne sont pas perceptibles.

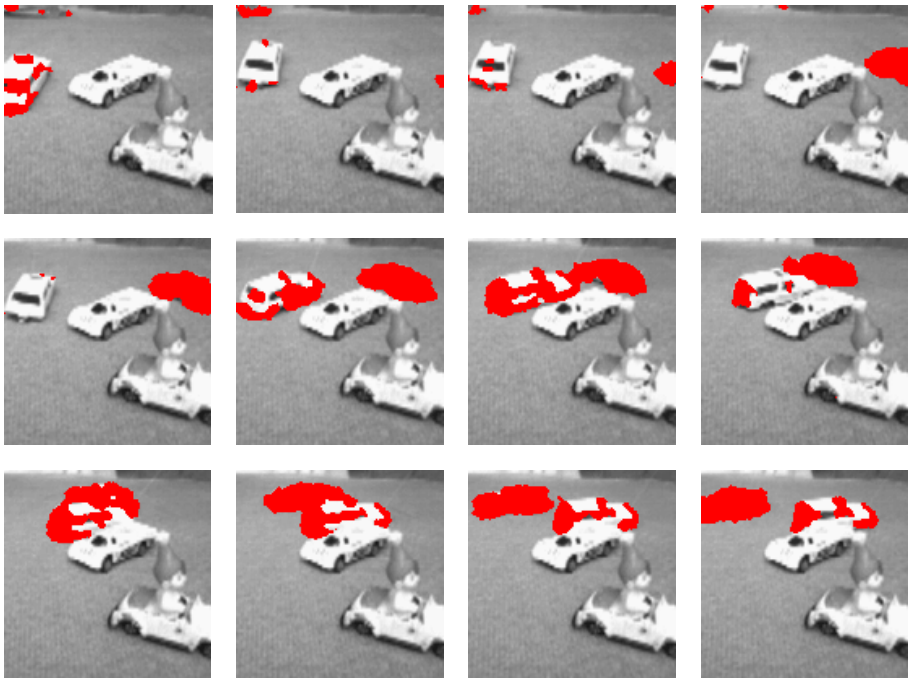
Finalement, nous précisons dans le tableau 5.3 les performances attendues des algorithmes correspondant aux résultats présentés. La fréquence de 10 MHz que nous prenons comme référence correspond à celle du circuit actuel, elle est par conséquent sensiblement plus faible que celle de la génération prochaine de rétine. Nous tenons compte dans ces calculs du temps nécessaire au codage de l'image n et à la différence avec l'image $(n+1)$. A 10 MHz, ce temps est inférieur à 5 ms. Les calculs des lignes (3) et (4) sont basés sur la convergence observée expérimentalement. Précisons que les cadences d'acquisition maximales admissibles sont indiquées ligne (3) en supposant que la sensibilité du capteur permette de telles cadences. Enfin, la ligne (4) est intéressante du point de vue énergétique, puisque la puissance consommée par le circuit est proportionnelle à sa fréquence de contrôle.

5.7 Conclusion

Pour conclure ce chapitre, nous allons rappeler les principaux enseignements que nous a apporté cette étude des traitements temporels dans la



(1)



(2)

FIG. 5.23 – *Les deux algorithmes utilisent 12 bits pour le calcul, la différence d'images est codée sur 8 bits, l'énergie sur 5 bits. (1) ICM avec un voisinage spatial 8-connexe. Le nombre d'itérations est 5. (2) Recuit simulé en utilisant la même fonctionnelle d'énergie, et la règle de Metropolis, avec un schéma de décroissance de température linéaire. Le nombre d'itérations est 50.*

rétine programmable, et présenter en détail les nombreux aspects qui restent à développer dans ce contexte, et qui constitueront la suite de ce travail.

Nous avons mis l'accent sur l'importance du *codage* à l'intérieur du circuit rétinien. Ajoutons qu'il paraît particulièrement important dans le cadre de l'algorithmique rétinienne en général, et dans celui des traitements temporels en particulier, d'exploiter de la manière la plus intelligente possible la *dynamique* du signal à traiter. A cet effet, une mise en œuvre fine d'un *contrôle de l'acquisition* semble souhaitable. On a évoqué dans ce chapitre l'égalisation d'histogramme, qui est l'exemple le plus élémentaire d'un tel contrôle. Elle peut être mise en œuvre de manière complètement matérielle [83], mais aussi par contrôle logiciel grâce à l'histogramme binaire obtenu par mesure de courant consommé (voir Annexe). On peut généraliser ce principe en *étudiant les variations* de l'histogramme binaire et en codant l'image dans la rétine en fonction de ces variations. Un tel mécanisme permettrait par exemple d'exploiter des images présentant des dynamiques de luminosité très importantes, et comportant des détails qu'on ne peut discerner qu'avec des quantifications très fines (scènes nocturnes comportant une forte source de lumière non diffusive typiquement), ou encore d'effectuer des pré-segmentations sur des scènes présentants des histogrammes multimodaux.

Plusieurs types de codage ont été présentés. Il serait intéressant de chercher à exploiter la transformée en ondelettes, d'autant plus que la représentation en composantes BF et HF est adaptée au calcul du mouvement.

L'implantation de l'algorithme de segmentation d'objets mobiles que nous avons réalisée sur le prototype a montré que la faible mémoire ne constitue pas un obstacle insurmontable pour les traitements temporels. Le résultat de la segmentation est assez rudimentaire, mais il dépasse les espérances que nous avions en entreprenant le portage sur *Pvlsar 2.2*. Evaluer ce résultat par rapport à la segmentation obtenue par un circuit analogique dédié à la détection de mouvement reste à faire.

Enfin, l'étude sur des techniques de segmentation markovienne est encourageante; elle montre la possibilité d'une implantation sur une rétine comportant une dizaine de registres binaires par pixels, ce qui est largement à la portée de la prochaine génération. A ce propos, il nous semble inutile, tant qu'on demeure en dessous de 5 ou 6 octets de mémoire par pixel, de chercher à câbler des opérateurs arithmétiques. Ceci étant, il faudra mener une étude approfondie sur le sujet pour les générations futures.

Dans le cadre de la segmentation markovienne, plusieurs problèmes restent à étudier, ainsi que des possibilités d'amélioration. Nous les examinons à présent en détail.

Convergence de la relaxation parallèle

Nous avons évoqué la parallélisation de la mise à jour des sites dans le cas déterministe, mais pas dans le cas stochastique. Or les algorithmes de simulation présentés sont fondés sur des méthodes de Monte-Carlo, et les preuves de convergence basées sur le fait que deux éléments consécutifs de la chaîne de Markov ne diffèrent que d'un site au plus. Bien entendu, de telles méthodes se révèlent inapplicables si l'on n'envisage pas un certain degré de parallélisme, mais alors que deviennent les fondements théoriques ? La littérature fournit certains éléments de réponse, mais les conclusions qu'on peut en tirer sont en partie contradictoires, et il reste à positionner notre approche dans ce contexte.

Ainsi, Marroquín [70] indique qu'une condition nécessaire de convergence est que deux variables dépendantes du champ de Markov ne soit pas mises à jour simultanément. Est-ce aussi une condition suffisante ? Oui, dans une certaine mesure, mais dans le cas de l'algorithme de Metropolis, il montre qu'on ne peut pas montrer que la mesure de Gibbs correspondant à notre modèle est la mesure invariante de la chaîne construite. Autrement dit, l'algorithme de Metropolis converge, mais vers quoi ? Cependant son raisonnement, aussi bien pour la nécessité de la condition dans le cas général que pour sa non-suffisance dans le cas de l'algorithme de Metropolis est basé sur le même contre-exemple : une réalisation du champ en échiquier binaire.

Or Azencott [2] a montré qu'un algorithme complètement parallèle convergeait toujours, sauf si tous les sites changent d'état simultanément, ce qui, dans un cadre binaire ne se produit justement que dans le cas d'un échiquier. Certains auteurs, comme [112] par exemple, exploitent ce résultat pour proposer des implantations complètement parallèles. Dans nos propres expérimentations, nous avons indiqué que la différence entre la segmentation obtenue en semi-parallèle et celle obtenue de manière complètement parallèle n'était pas perceptible. Néanmoins, le raccourci nous semble discutable, dans la mesure où, si l'on gagne un facteur χ dans l'efficacité du calcul (en terme de taux de processeurs actifs), rien ne prouve que la convergence ne sera pas plus de χ fois plus lente, puisqu'on peut imaginer que des oscillations locales puissent avoir une durée de vie beaucoup plus longue qu'en semi-parallélisme. Il est intéressant de mentionner à ce propos l'expérience de Bernard qui, pour la convergence d'un processus stochastique bidimensionnel dans le cadre du codage en demi-teintes [6], indique qu'il obtient les meilleurs résultats de convergence en inhibant aléatoirement 90% des sites.

Finalement, il faut étudier le point de vue de Younes [123], qui propose un cadre formel de champs aléatoires synchrones, pour lequel certains résultats de convergence peuvent être obtenus dans tous les cas, si l'on considère des

chaînes de Markov d'ordres supérieurs dans les algorithmes de simulation.

Décroissance de la température

Le problème de la vitesse de décroissance de la température dans l'algorithme de recuit simulé doit être examiné. D'après les expérimentations relatées dans la littérature, le schéma de décroissance de Geman et Geman n'est jamais appliqué, parce que trop lent. Dans les résultats que nous avons présentés, avec une décroissance linéaire, nous demeurons largement au delà du temps réel. Cependant, pour un modèle plus sophistiqué, le coût pourrait devenir plus critique. Or, dans la partie stochastique de l'algorithme, la partie la plus coûteuse reste la multiplication par la constante extérieure pour la prise en compte de la température. Peut-on appliquer un autre schéma de décroissance qui rendrait moins coûteux ces calculs? On peut également étudier l'implantation de méthodes plus astucieuses pour faire intervenir la température.

Champs multivalués

Au delà de la détection de mouvement, nous pensons que l'intérêt de l'étude menée sur la segmentation markovienne concerne la possibilité d'implanter des schémas de relaxation markoviens pour des champs de Markov binaires sur la rétine numérique. Il sera important de voir dans quelle mesure ces schémas peuvent s'appliquer dans le cadre de champs multivalués.

Aspects multirésolution

Enfin, les aspects multi-échelle ont été largement débattus dans la littérature. Une relaxation faisant intervenir une topologie à différentes échelles est souvent pertinente, particulièrement dans le cadre de la détection de mouvement, où le système se doit de réagir indépendamment de l'amplitude des différents mouvements observés dans la scène. La multirésolution étant une caractéristique forte du circuit rétinien, l'implantation d'une relaxation multi-grille doit être étudiée, sur les plans théorique et pratique.

Annexe A

Extraire l'information de la rétine

A.1 Pourquoi ?

Les algorithmes que nous avons présentés dans ce rapport concernent les traitements bas niveau effectués en parallélisme SIMD par le circuit rétinien. Ils ont pour objet de réduire la quantité d'information tout en fournissant des données pertinentes au calculateur chargé des tâches de haut niveau. Il est par conséquent très important d'extraire efficacement ces informations de la rétine.

Dans cette annexe, nous traitons du problème de l'extraction d'information du circuit, qui repose sur les deux questions suivantes :

- Quelle est la nature des informations fournies par la rétine au processeur hôte ?
- Comment sortir efficacement ces informations ?

Ces deux questions étant intimement corrélées, il s'agit d'un problème d'adéquation algorithme-architecture, et à ce titre, l'ordre dans lequel on les présente a son importance. Nous commencerons par la deuxième question, qui est de nature électronique, et pour laquelle nous nous contenterons de résumer les méthodes qui ont été proposées pour extraire de la rétine un ensemble de coordonnées. Ensuite, nous examinerons la deuxième question, qui est de nature algorithmique, et nous présenterons quelques mesures, dont la plupart sont très classiques, et qui peuvent fournir des représentations intéressantes.

A.2 Comment ?

Différentes méthodes pour extraire efficacement du circuit rétinien les coordonnées de pixels dits «actifs» ont été proposées [11], [10], [86]. Nous les résumons dans cette section.

A.2.1 Histogramme binaire

Considérons l'un des registres binaires du processeur élémentaire attaché à chaque pixel. On appelera pixels *actifs* les pixels pour lesquels ce registre contient la valeur binaire 1. Le nombre de points actifs dans une image binaire constitue un opérateur de réduction fondamental, et son intérêt en tant que structure de contrôle est multiple :

- Il indique la convergence d'un opérateur par relaxation. Notons qu'un OU global peut d'ailleurs suffire dans ce cas.
- Il rend possible le contrôle automatique de gain dans un système d'acquisition par multiseuillage tel que le nôtre.
- Il fournit des mesures de corrélation entre images binaires, qui constituent les données de base du flux sensoriel d'un robot mobile à base de rétine [28].

Une telle mesure dans notre architecture paraît *a priori* dispendieuse car foncièrement anti-cellulaire. Il a cependant été montré dans [75] comment cette mesure pouvait être estimée très précisément par une mesure du courant consommé par la rétine, et donc en temps constant.

La figure A.1 montre comment obtenir les coordonnées des pixels actifs sur une image grâce à l'histogramme binaire. La propagation par des dilatations jusqu'à convergence dans trois directions Est, Nord et Sud (respectivement Ouest, Est et Sud) permet par une mesure du nombre de points de déterminer l'abscisse (respectivement l'ordonnée) du pixel noir le plus en haut à gauche, puis de l'éliminer.

La donnée de l'histogramme binaire peut par conséquent suffire à elle-seule pour toutes les extractions d'information. Son efficacité est cependant limitée. Si le nombre de points à extraire est grand, il devient vite moins coûteux de sortir toute l'image binaire.

D'autres implantations dédiées à l'extraction d'information ont donc été envisagées. Nous les présentons ci-dessous.

A.2.2 OU global

Cette méthode utilise un opérateur analogique effectuant en temps constant un OU global sur chaque ligne et sur chaque colonne. On repère dans un pre-

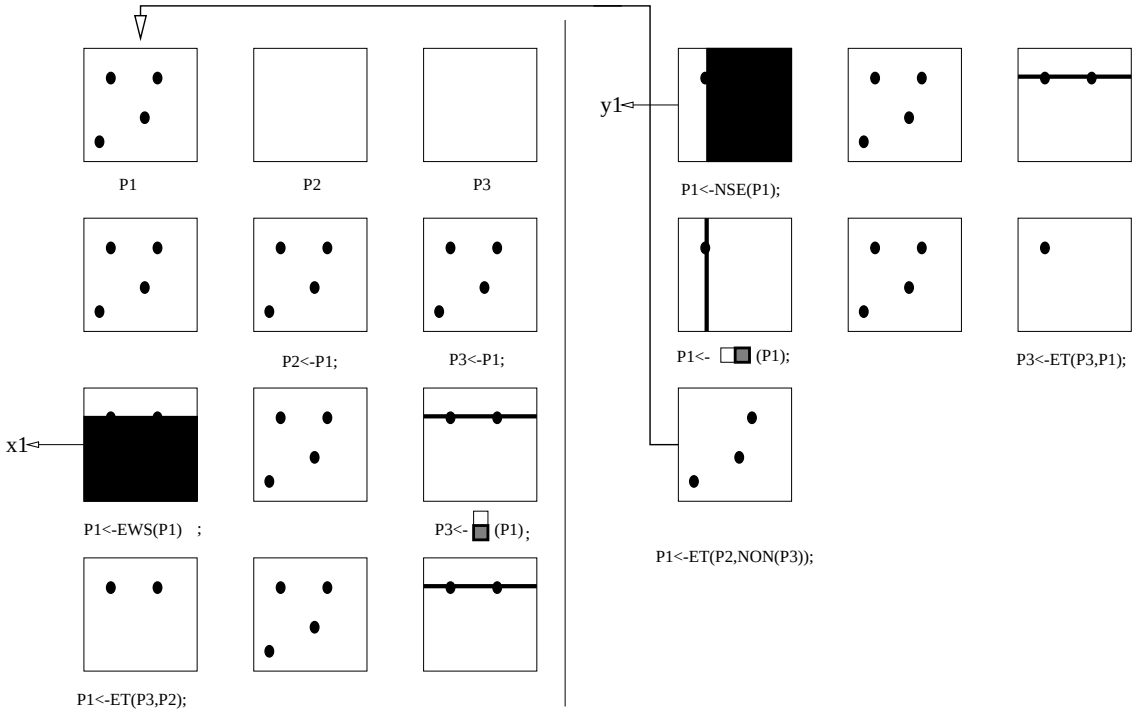


FIG. A.1 – *Trouver les coordonnées d'un ensemble de points grâce a l'histogramme binaire.*

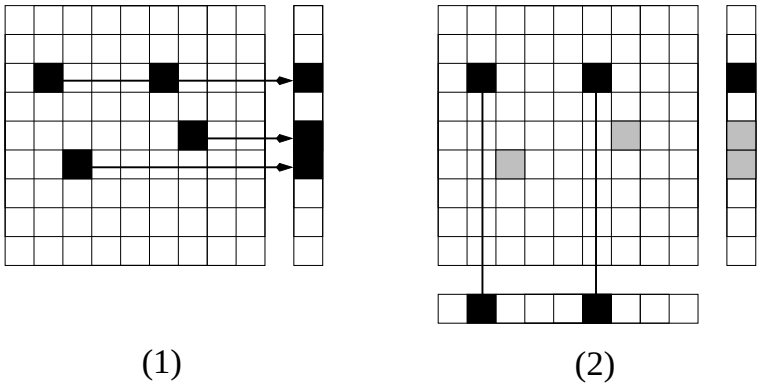


FIG. A.2 – *Méthode du OU global par ligne/colonne.*

mier temps les lignes contenant des pixels actifs (figure A.2(1)), puis pour chacune de ces lignes, on effectue un OU global par colonne en «éteignant» toutes les autres lignes (figure A.2(2)).

Notons que le OU global par lignes/colonnes permet d'obtenir très rapidement les coordonnées du point le plus en haut à gauche de l'image. C'est une donnée qui est, comme nous le verrons dans la section suivante, intéressante pour extraire une description de moyen niveau de l'image binaire. Mais il faudrait que l'on puisse de la même façon adresser un pixel donné en mettant à 1, en temps constant, la ligne et la colonne correspondantes.

A.2.3 Tableau de De Bruijn

Une implantation envisagée dans [10] propose de nantir la rétine d'un «arrière-plan» binaire en mémorisant dans chaque pixel un bit constant de telle sorte que deux positions différentes d'un curseur de taille N fournissent deux «coloriages» différents de ce curseur. Bien sûr, il faut que 2^N soit supérieur au nombre de pixels. Dans l'exemple de la figure A.3, on peut vérifier une propriété plus forte de bijectivité : 2^N est égal au nombre de pixels. Un tel coloriage est appelé *tableau de De Bruijn*.

Cette solution permet de résoudre le problème de l'adressage de pixels par le calcul d'une transformée en tout-ou-rien.

A.3 Quoi ?

Idéalement, dans un système à base de rétine, tout le traitement d'images est effectué par la matrice de processeurs rétiniens, et l'on extrait du circuit

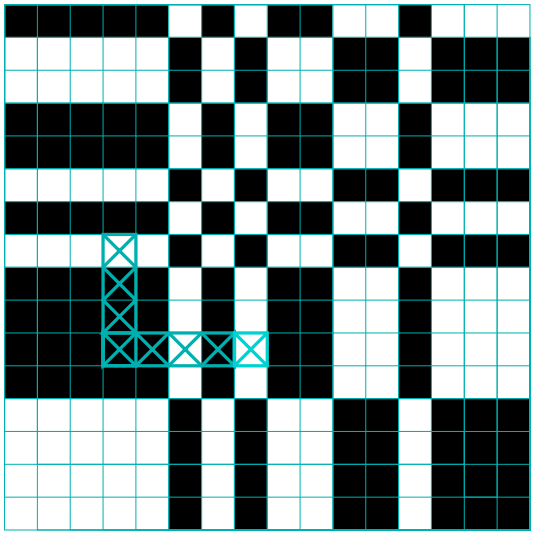


FIG. A.3 – *Adresse des pixels par leur voisinage dans un tableau de De Bruijn.*

des données *structurées*, et non pas des images complètes.

En pratique, on peut avoir à effectuer des traitements d'images qui ne sont pas réalisables par la machine SIMD rétinienne, et il faut sortir des images. Précisons que cela n'enlève pas tout l'intérêt de la rétine, dans la mesure où celle-ci réalise au minimum une *segmentation* qui réduit le flux sortant (une image segmentée est typiquement codée sur 1 ou 2 bits), et décharge le processeur externe de manière importante.

Néanmoins, dans la mise en œuvre d'un algorithme de vision sur un système comportant une rétine programmable, on cherchera systématiquement à changer de *niveau de représentation* lorsqu'on sort de la rétine. La phase d'extraction d'information va structurer les données images, en les amenant à un niveau *descriptif*, où l'on spécifie «des propriétés globales de l'image ou des propriétés de parties de l'image et les relations entre ces parties» [100]. La nature de ces propriétés a été largement étudiée dans la littérature [91], [100]. Nous en présentons quelques-unes dans cette section, en décrivant la manière dont elles sont extraites.

A.3.1 Nombre d'Euler

Les «différentes parties» qui sont considérées dans la description d'une image sont souvent les composantes connexes, d'où l'importance d'extraire des mesures topologiques globales d'une image. Le *nombre d'Euler-Poincaré*

d'une image binaire correspond au nombre des composantes connexes, diminué du nombre de trous.

Ce nombre se calcule très rapidement [106] grâce au théorème d'Euler. En effet, la topologie induisant une structure de graphe sur l'image, le théorème d'Euler donne : $f + s - a = c + 1$, où f est le nombre de faces, s le nombre de sommets, a le nombre d'arêtes, c le nombre de composantes connexes. Par exemple, pour la 8-connexité, on a les égalités suivantes (C étant une configuration, $N(C)$ désigne le nombre de fois où cette configuration est présente dans l'image) :

$f = N\begin{pmatrix} 1 & \\ 1 & 1 \end{pmatrix} + N\begin{pmatrix} & 1 \\ 1 & 1 \end{pmatrix} + N\begin{pmatrix} 1 & 1 \\ & 1 \end{pmatrix} + N\begin{pmatrix} 1 & 1 \\ 1 & \end{pmatrix} + t + 1,$

(t est le nombre de trous, $+1$ pour la face externe),

$s = N(1),$

$a = N\begin{pmatrix} 1 & 1 \end{pmatrix} + N\begin{pmatrix} 1 & \\ & 1 \end{pmatrix} + N\begin{pmatrix} 1 \\ 1 \end{pmatrix} + N\begin{pmatrix} & 1 \\ 1 & \end{pmatrix}.$

Après simplification, on obtient :

$c - t = N\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} - N\begin{pmatrix} 1 & \\ 0 & 1 \end{pmatrix}.$

Sur la rétine, on applique deux transformées en tout-ou-rien, chacune étant suivie d'un calcul de somme. La différence donne le nombre d'Euler-Poincaré, ou nombre spécifique de connexité (figure A.4).

Cette mesure permet de connaître le nombre de composantes connexes, à condition de boucher préalablement les trous (voir section 4.3).

A.3.2 Attributs géométriques

On s'intéresse ici à des mesures géométriques simples sur des composantes connexes.

L'*aire* A d'une composante connexe X est simplement égale au nombre de points de X. La notion de *périmètre* est plus délicate, car différentes définitions acceptables mènent à des mesures très différentes. La mesure la plus simple à obtenir sur la rétine consiste à choisir comme périmètre P le nombre de points du contour de X. On obtiendra une mesure plus précise en différenciant les contours horizontaux et verticaux d'une part des contours diagonaux d'autre part, et en donnant un poids de $\sqrt{2}$ à ces derniers.

L'aire A et le périmètre P étant connus, on peut obtenir la mesure élémentaire de *compacité* définie par :

$$\kappa = \frac{P^2}{4\pi A}$$

La position du *centre* d'un objet est bien sûr une donnée essentielle. Plusieurs centres différents peuvent être définis pour une image binaire. Le *centre de gravité*, qui a pour coordonnées les moyennes arithmétiques des

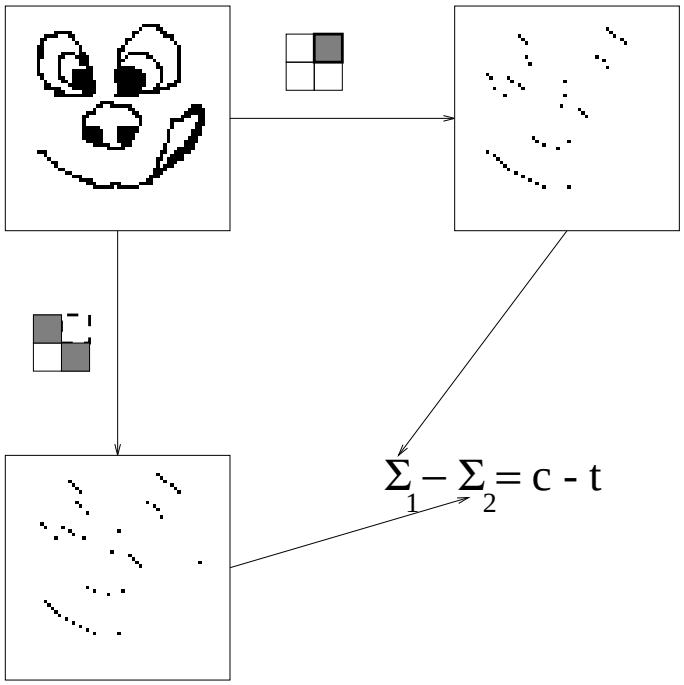


FIG. A.4 – *Calcul du nombre d'Euler-Poincaré sur la rétine.*

coordonnées de tous les points de l'objet, joue un rôle important en reconnaissance de formes. Cependant son calcul, comme celui des *moments* en général, nécessite l'extraction de tous les points de chaque composante connexe, à moins de disposer d'une mesure de projections (histogramme 1D) sur les bords de l'image.

En revanche le calcul du noyau homotopique (cf Chap. III) se calcule de manière cellulaire, et lorsque l'objet est une composante connexe sans trou, son noyau est réduit à un point, qu'on appelle *centre géodésique* (voir figure A.5(2)). Dans le cas d'objets convexes, le centre géodésique coïncide avec le centre de gravité. Il présente de plus l'avantage de toujours appartenir à l'objet (N'oublions pas que dans notre contexte l'objet est souvent une cible...)

Le centre géodésique a pour propriété de minimiser la fonction de propagation (cf Chap IV). Par recontruction de l'objet, le nombre d'itérations fournit le *diamètre géodésique* Δ_X (voir figure A.5(4)). Le rapport entre le diamètre géodésique et l'*épaisseur* de l'objet ρ_X, qui est le nombre d'érosions nécessaires pour faire disparaître l'objet (figure A.5(5)) donne une mesure d'*élongation* de l'objet (figure A.5(6)).

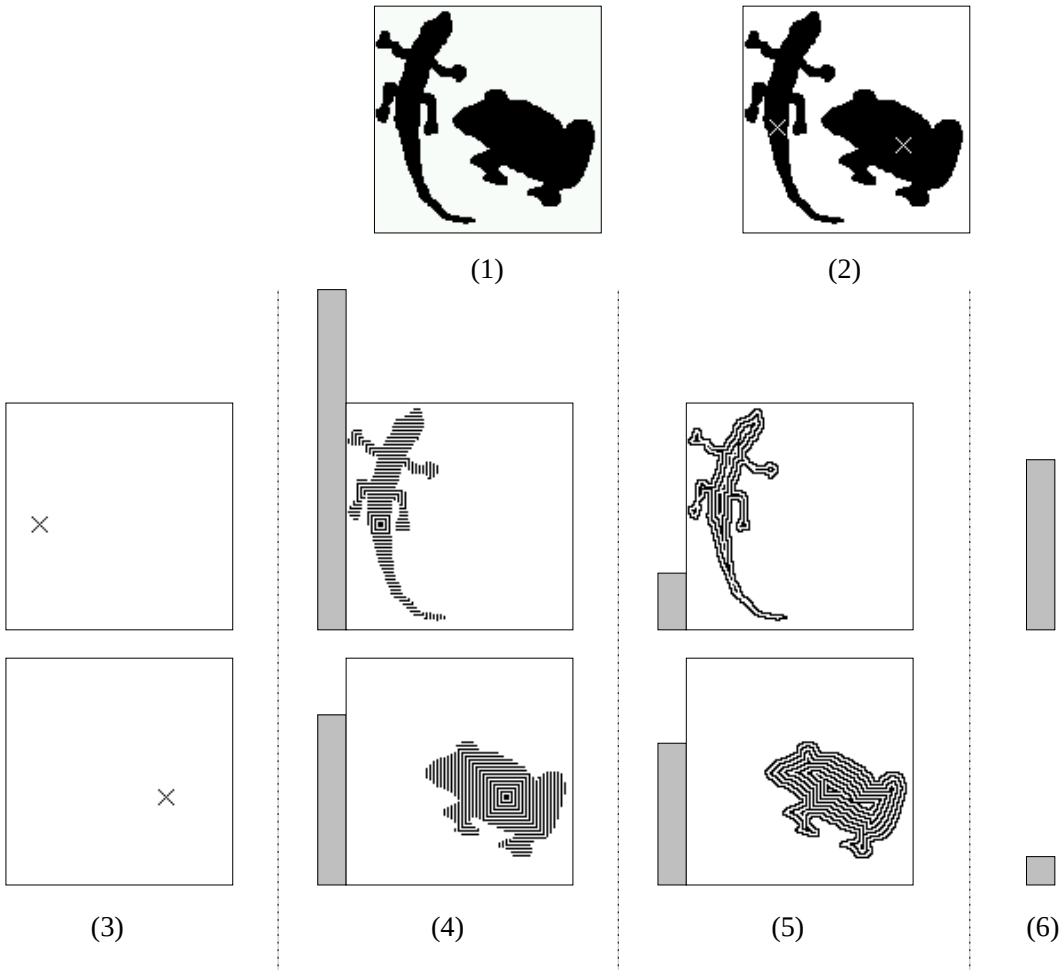


FIG. A.5 – *Extraction d'attributs géométriques. (1) Image originale, sans trou. (2) Calcul du centre géodésique (noyau homotopique dilaté par une croix et superposé à l'image originale). (3) Extraction des centres géodésiques. (4) Recontruction de (3) dans (1) avec mesure de vitesse de convergence. (5) Erosion itérative de (4) avec mesure de vitesse de convergence. (6) Rapport des deux mesures précédentes fournissant une mesure d'élongation.*

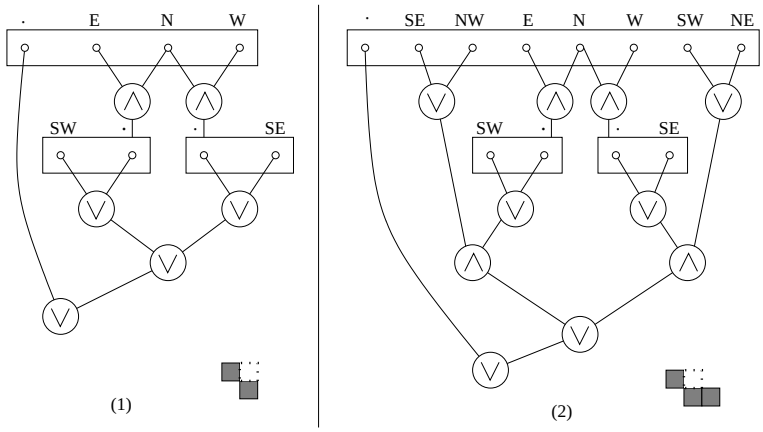


FIG. A.6 – *Calcul des rectangles (1) et des octogones (2) englobants.*

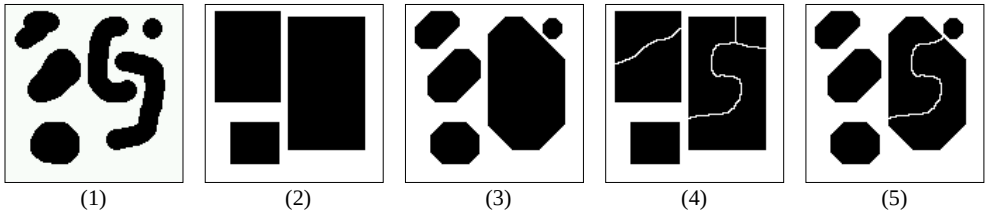


FIG. A.7 – *Utilisation des boîtes englobantes. (1) Image originale. (2) Rectangles englobants. (3) Octogones englobants. (4) SKIZ géodésique de (1) dans (2). (5) SKIZ géodésique de (1) dans (3).*

A.3.3 Polygones englobants

L'utilisation des polygones englobants permet de connaître de manière approximative la taille et la localisation des objets, et sous une forme facile à extraire. Le principe est de circonscrire un objet par un polygone donné.

Si l'on impose au polygone d'être minimal, il s'agit de l'*enveloppe convexe* de l'objet. L'enveloppe convexe ne peut pas être calculée localement, en revanche les *rectangles* et les *octogones* englobants se calculent très bien de manière cellulaire [95]. Les opérateurs de rétines et les configurations correspondantes sont présentés sur la figure A.6.

Les polygones englobants fusionnent en général plusieurs composantes connexes. On peut obtenir une description simplifiée qui respecte la topologie en faisant un SKIZ géodésique de l'image initiale dans l'image des polygones englobants (voir figure A.7).

$S \leftarrow$ squelette mince;
$O \leftarrow$ points triples;
$X \leftarrow$ points extrémités;
$V \leftarrow \emptyset$;
$E \leftarrow \emptyset$;
Répéter :
    $\{p\} \leftarrow$ point le plus en haut à gauche de $X \cup O$;
    $V \leftarrow V \cup \{p\}$;
    $X \leftarrow X \setminus \{p\}$;
    $O \leftarrow O \setminus \{p\}$;
    $I \leftarrow R_{(S \setminus O)}(\{p\})$;
    $J \leftarrow (I \oplus B_8) \cap (X \cup O)$;
    Pour tout point $q \in J$, faire :
        $E \leftarrow E \cup \{(p, q)\}$;
    $S \leftarrow S \setminus I$;
Jusqu'à : $S = \emptyset$.

TAB. A.1 – *Pseudo-code d'extraction de graphe localisé.*

A.3.4 Graphes localisés

Un graphe localisé est un couple (V, E) tel que V est un sous-ensemble de $\mathbb{Z}^2$ et E un sous-ensemble de $V \times V$. Un tel graphe peut fournir une description riche et structurée d'une scène, susceptible d'être utilisée dans des traitements de haut niveau tels qu'une analyse syntaxique.

Dans l'exemple présenté figure A.8, on s'intéresse à la structure de graphe du squelette de l'image binaire. A partir du squelette d'épaisseur unité, on calcule deux ensembles :

- l'ensemble des points *extrémités* du squelette. Ces points correspondent aux barbules (voir Chap. I).
- l'ensemble des points *triples* du squelette. Ce sont les points dont le nombre de connexité est strictement supérieur à 2 (voir Chap. III).

La donnée de ces deux ensembles ainsi que du squelette permet d'extraire le graphe localisé correspondant. Le tableau A.1 présente la procédure sous-forme de pseudo-code; la figure A.8 illustre sur un exemple le résultat obtenu.

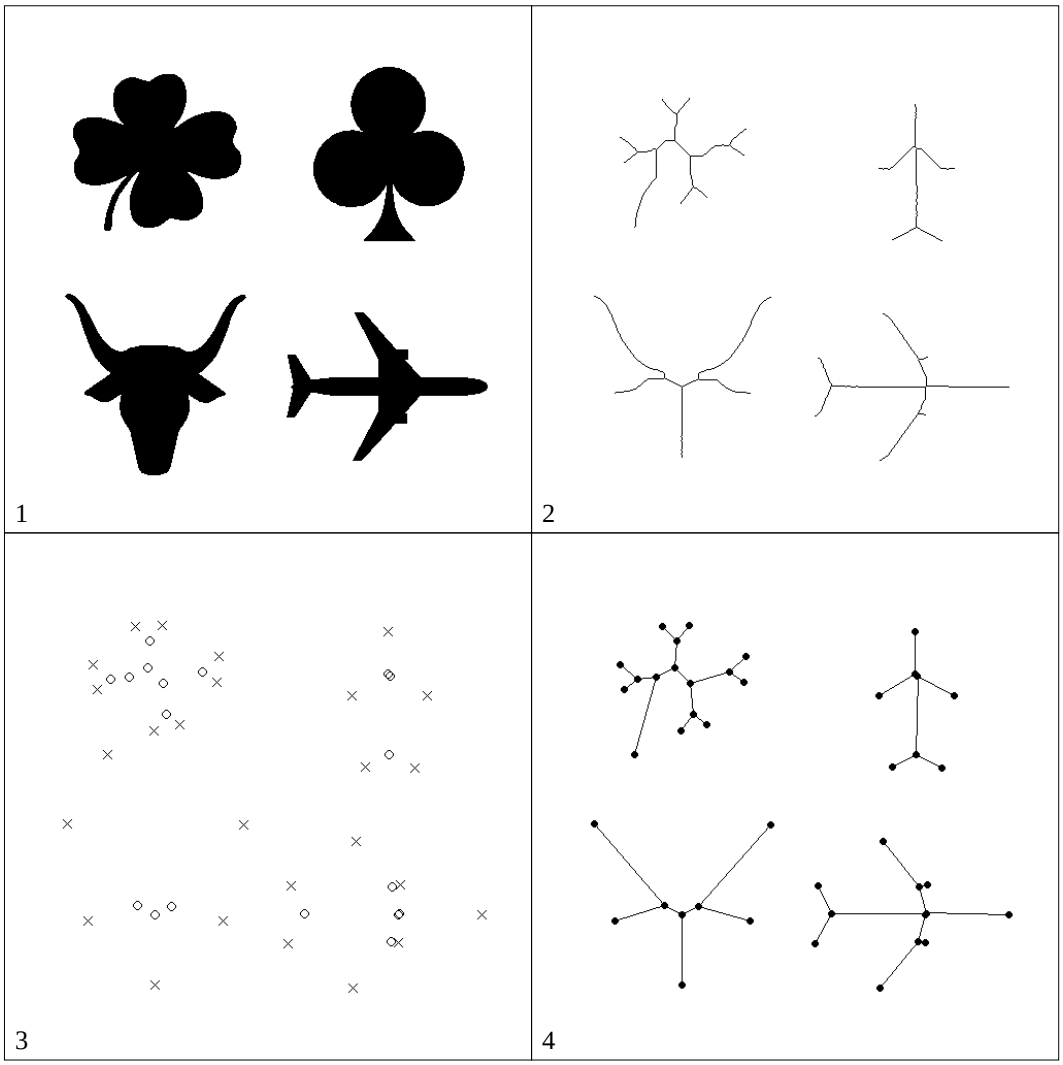


FIG. A.8 – *Extraction d'un graphe localisé de la rétine. (1) Image originale. (2) Squelette mince (Algorithme MB2 suivi de quatre itérations de l'algorithme de Jang et Chin). (3) Calcul des points extrémités et des points triples (les deux ensembles ont été dilatés avec des éléments structurants de formes différentes, puis réunis pour des raisons de clarté). (4) Information extraite de la rétine restituée symboliquement.*

A.3.5 Utilisation des projections

Le terme de projection concerne ici la transformation itérative qui conserve le nombre de pixels d'une image binaire tout en les déplaçant dans une direction donnée. Les techniques de projection et leur utilisation à des fins d'extraction d'informations globales ont été présentées dans [95]. Nous présentons ici un exemple d'utilisation originale d'une image projetée.

L'exemple concerne le comptage de personnes se présentant à l'entrée d'une porte automatique, le capteur étant situé au dessus et au milieu de la porte. Etant donné la position de la rétine et l'attitude des personnes, on suppose que le nombre de personnes correspond au nombre de pics significatifs de la projection verticale de l'image binaire de segmentation.

La figure A.9 montre alors l'algorithme de calcul. On réalise une ouverture de l'image de projection (2) par un segment horizontal, pour éliminer les pics non significatifs (3). Le résultat correspond alors aux nombres de maxima régionaux de l'image (3), vue comme un signal 1D. Les maxima régionaux du signal S sont obtenus par la différence $S \setminus R_{S-1}(S)$. Le signal $S-1$ est obtenu par translation d'un pixel vers le bas (4). Puis on effectue une reconstruction dans S en utilisant un élément structurant horizontal (5). La différence (6) fournit le résultat.

La robustesse de cet algorithme pour le comptage de personnes est loin d'être satisfaisante, nous ne nous attarderons donc pas sur les performances de cette application mais plutôt sur l'esprit «rétinien» de la démarche. En effet, dans cet exemple, l'intégralité du calcul est effectué dans la rétine, en SIMD. On peut remarquer en particulier qu'on traite sur une architecture cellulaire bidimensionnelle un signal monodimensionnel (la projection), qu'on filtre, puis dont on extrait les maxima régionaux.

A.4 Conclusion

Dans cette annexe, nous avons mis l'accent sur l'importance du problème de l'extraction d'information dans l'algorithmique de vision d'un système à base de rétine. Nous développons ici quelques conclusions et suggestions.

La mesure de l'histogramme binaire semble bien être une grandeur fondamentale. Il est par conséquent très souhaitable de l'obtenir en temps constant. Malheureusement, avec l'augmentation de la taille des rétines, obtenir l'histogramme au pixel près par une mesure analogique est de plus en plus difficile [74]. Il faut donc étudier le coût d'implantation d'opérateur de mesure au pixel près (16 bits pour une rétine 256×256, par exemple), ou voir dans quelle mesure on peut se contenter d'une mesure approximative.

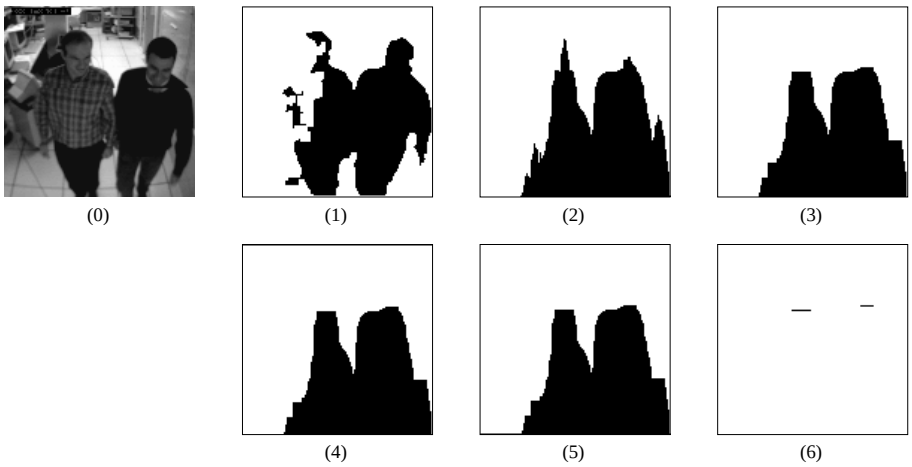


FIG. A.9 – *Comptage de personnes «complètement» effectué dans la rétine. (0) Image originale. (1) Segmentation des personnes basée sur le mouvement. (2) Projection verticale de l'image précédente. (3) Ouverture de l'image précédente par un segment horizontal. (4) Image précédente translatée d'un pixel vers le bas. (5) Reconstruction géodésique 1D de (4) dans (3). (6) Différence logique ((3) ET NON(5)), le nombre de composantes connexes correspond au résultat.*

Les opérations effectuées sur les images pour préparer l'extraction d'information utilisent intensément les propagations : reconstructions géodésiques, dilatations directionnelles réitérées, bouchage de trous... L'intérêt de la propagation asynchrone, associée à une topologie programmable se manifeste donc encore sur ce point.

Nous avons pu voir également qu'on avait recours de manière importante au mécanisme de *sélection* d'un pixel (point le plus en haut à gauche par exemple). Il serait par conséquent très intéressant de disposer, en plus du OU global par ligne/colonne, d'une possibilité de mettre à 1 en temps constant une ligne ou colonne donnée. Géométriquement, cela munirait la rétine d'un mécanisme de projection/extrusion orthogonale très efficace qui permettrait, outre la sélection de point extrême, de définir une zone d'intérêt rectangulaire, ou même «dessiner» sur un plan du circuit des motifs de corrélation...

On peut se questionner sur l'intérêt de disposer d'un mécanisme d'histogramme par ligne/colonne. On retrouve un problème assez ancien : qu'apprend-on sur une image binaire d'après ses projections en x et en y (voir [99], chap. 8) ? Cette question est un enjeu important en imagerie médicale, comme en tomographie, où l'on souhaite retrouver la forme d'un organe opacifié à partir de deux signaux 1D de densité acquis par coupe transversale. La re-

construction de l'image binaire est impossible en général, et même lorsqu'elle peut être faite, sous certaines hypothèses, le calcul est bien plus coûteux que d'extraire toute l'image (lorsqu'elle peut l'être, bien sûr). Néanmoins, il faut noter qu'un tel mécanisme peut fournir efficacement des mesures globales sur la forme telles que les moments.

En résumé, l'extraction d'information est un sujet vaste, impliquant d'importantes réflexions algorithmiques et architecturales. Nous pensons qu'il est judicieux d'y consacrer une étude beaucoup plus approfondie. Au moment où les rétines artificielles parviennent à un certain degré de maturité, c'est véritablement l'efficacité d'un système de vision à base de rétine qui est en jeu.

Conclusion et perspectives

A l'instant où s'achève ce travail de thèse, quelles contributions à la vision rétinienne sommes-nous en mesure de revendiquer ? Qu'il nous soit permis, dans ces dernières lignes, de mettre l'accent sur les principaux apports que nous défendons.

Dans le cadre de la complexité des opérateurs de rétines, nous avons développé un formalisme qui a permis de positionner notre approche dans un cadre théorique existant, pour la définition des limites universelles de complexité. L'utilité pratique de ce formalisme pour mesurer les complexités en temps et en espace des opérateurs de rétines a été exploité à plusieurs reprises pour évaluer «à la main» de petits algorithmes. Pour les prochaines générations de rétines, il nous semble intéressant d'aller plus loin en développant à partir d'un codage de ce type, des techniques d'optimisation qui pourraient déboucher sur l'écriture d'un compilateur à plusieurs niveaux (table de vérité $\rightarrow$ expression booléenne $\rightarrow$ code rétine).

Notre travail sur la squelettisation constitue certainement le thème le plus abouti de notre recherche. Nous pensons avoir apporté dans ce cadre une illustration de l'intérêt d'une minimisation logique poussée à l'extrême. Le nouvel algorithme que nous avons proposé possède plusieurs bonnes propriétés. Son intérêt dépasse par conséquent le seul cadre des rétines programmables, d'où l'importance des développements du chapitre 3. Parmi les extensions qui peuvent être envisagées, indiquons notre préférence pour l'étude de la validité n-dimensionnelle de l'algorithme.

Dans l'étude sur l'implantation des opérateurs géodésiques, nous avons montré, principalement au travers de l'exemple de la LPE, comment de puissants outils de segmentation pouvaient s'accommoder d'une aridité mémoire telle que celle de la rétine programmable. Dans les suites à donner, nous avons clairement exprimé un point de vue en faveur d'une solution architecturale pour réduire le coût énergétique des reconstructions. On envisagera alors une phase d'expérimentation des algorithmes développés sur une architecture mixte synchrone/asynchrone.

Enfin, dans l'application à la détection d'objets mobiles, nous avons présen-

té une implantation particulièrement compacte d'un modèle markovien et d'un algorithme de recuit simulé qui permet de prévoir une mise en œuvre facile sur la prochaine génération de rétine programmable d'une segmentation au sens du mouvement. Ce dernier résultat constitue un indice fort qui nous pousse à croire en l'apparition prochaine d'un système de vision complet à base de rétine, auquel - nous l'espérons - ce travail aura fourni une impulsion significative.

De façon plus générale, notre travail a contribué à prouver la pertinence du concept de rétine programmable pour la vision artificielle, en montrant que malgré les limitations propres aux contraintes rétiniennes, il était possible d'implanter des algorithmes complexes intéressants pour la vision.

De plus, conformément à nos objectifs, ce travail devrait logiquement influencer la conception des prochaines rétines du point de vue architectural. Nous songeons en particulier à l'étude sur le mouvement, qui fournit des indices sur le dimensionnement en mémoire des prochains circuits, ainsi qu'à l'algorithmique géodésique, pour la mise en évidence de l'intérêt des opérations asynchrones.

Finalement, nous avons proposé des nouveaux algorithmes ou des implantations originales (Squelettisation, LPE, Modèle de Markov), dont l'intérêt peut s'exprimer au delà du cadre rétinien. A ce titre, la mise en œuvre de ces algorithmes dans un autre contexte architectural constitue aussi une voie d'extension de ce travail que nous espérons voir prochainement se développer.

Bibliographie

[1] Carlo Arcelli. « A condition for digital points removal ». *Signal Processing*, 1-4:283–285, 1979.

[2] Robert Azencott. *Synchronous Boltzmann machines and Gibbs fields*, volume F68. F. Fogelman-Soulié et J. Hérault, eds., NATO ASI series, 1990.

[3] Gérald Banon et Junior Barrera. *Máquinas Morfológicas*. Jr Barrera ed., 1994.

[4] Etienne Basset. « Détection de mouvement par méthode markovienne : application à la rétine numérique ». Rapport de projet personnel en laboratoire, Ecole Nationale Supérieure de Techniques Avancées, 2000.

[5] Bernard Beauzamy et Gilles Darblade. « Rétine artificielle et opérateurs de rétine ». Rapport Technique, Société de Calcul Mathématique S.A., 1999.

[6] Thierry Bernard, Patrick Garda, et Bertrand Zavidovique. « A neural halftoning algorithm suiting VLSI implementation ». Dans *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Albuquerque - NM, avril 1990.

[7] Thierry M. Bernard. « *Des rétines artificielles intelligentes* ». Thèse de Doctorat, Université de Paris-sud, 1992.

[8] Thierry M. Bernard. « Rétines artificielles et robotique mobile ». Rapport Technique CTA 98 R 033, CTA/Géographie-Imagerie-Perception, 1998.

[9] Thierry M. Bernard et Antoine Manzanera. « Improved low complexity fully parallel thinning algorithm ». Dans *International Conference on Image Analysis and Processing*, pages 215–220, Venice - Italy, septembre 1999. IEEE.

[10] Thierry M. Bernard et Jean C. Meier. « Cursor-Injective Two-Valued Lattices for a Local Encoding of Pixel Position ». Dans *Proc. SPIE, Vol. 2950, Advanced Focal Plane Arrays and Electronic Cameras*, pages 230–241, Berlin, Germany, octobre 1996.

[11] Thierry M. Bernard et Fabrice Paillet. « Output methods for an associative operation of programmable artificial retinas ». Dans *International Conference on Intelligent Robots ans Systems*, pages 752–757, Grenoble - France, septembre 1997.

[12] Thierry M. Bernard, Bertrand Y. Zavidovique, et Francis Devos. « A programmable Artificial Retina ». *IEEE Journal of Solid-state Circuits*, 28-7:789–798, 1993.

[13] Gilles Bertrand. « On P-simple points ». *Compte Rendus à l'Académie des Sciences*, 321-1:1077–1084, 1995.

[14] Gilles Bertrand, Jean-Christophe Everat, et Michel Couprie. « Image segmentation through operators based on topology ». *Journal of Electronic Imaging*, 6-4:395–405, 1997.

[15] Gilles Bertrand et Grégoire Malandain. « A new characterization of three-dimensional simple points ». *Pattern Recognition Letters*, 15:169–175, 1994.

[16] Julian Besag. « Spatial interaction and the statistical analysis of lattice systems ». *Journal of Royal Statistic Society*, B-36:192–326, 1974.

[17] Julian Besag. « On the statistical analysis of dirty pictures ». *Journal of Royal Statistic Society*, B-48:259–302, 1986.

[18] Serge Beucher et Fernand Meyer. *The Morphological Approach to Segmentation : The Watershed Transformation.* Edward Dougherty ed., 1991.

[19] H. Blum. « A tranformation for extracting new descriptors of shape ». Dans *Proc. Symposium Models for the perception of speech and visual form*, pages 362–380. W.Wathen-Dunn ed. M.I.T. Press Cambridge MA, 1967.

[20] Ravi B. Boppana et Michael Sipser. *The complexity of finite functions.* Jan van Leeuwen, 1990.

[21] Patrick Bouthémy et Patrick Lalande. « Recovery of moving object in an image sequence using local spatiotemporal contextual information. ». *Optical Engineering*, 32-6:1205–1212, 1993.

[22] Randal E. Bryant. « Graph-based algorithms for Boolean function manipulation ». *IEEE Transactions on Computers*, C-35/8:677–691, 1986.

[23] Lorenzo Calabi et J.A. Riley. « The skeletons of stable plane sets ». Rapport Technique AF 19 (628-5711), Parke Mathematical Laboratories Inc., Carlisle, MA, 1967.

[24] John F. CANNY. « A computational approach to edge detection ». *IEEE Transactions on pattern analysis and machine intelligence*, 8-6:679–697, 1986.

[25] Alice CAPLIER, Christophe DUMONTIER, Franck LUTHON, et Pierre-Yves COULON. « Algorithme de détection de mouvement par modélisation markovienne. Mise en œuvre sur DSP. ». *Traitement du signal*, 13-2:175–190, 1996.

[26] Alice CAPLIER et Franck LUTHON. « Approche spatio-temporelle pour l'analyse de séquences d'images. Application en détection de mouvement. ». *Traitement du signal*, 14-2:195–208, 1997.

[27] Rafael CARDONER et Federico THOMAS. « Residuals + Directional Gaps = Skeletons ». *Pattern Recognition Letters*, 18:343–353, 1997.

[28] Louis S. CARNAPETE, Philippe E. NGUYEN, Robert G. NGUYEN, et Thierry M. BERNARD. « A miniature retina-based AGV called VAMPIRE ». Dans *Proc. IEEE Workshop on Computer Architecture for Machine Perception*, pages 29–33, Como - Italy, septembre 1995.

[29] Michel COUPRIE, Francisco Nivando BEZERRA, et Gilles BERTRAND. « Grayscale image processing using topological operators ». Dans *Vision Geometry VIII*, volume 3811, pages 261–272, Denver - CO, juillet 1999. SPIE.

[30] George C. CROSS et Anil K. JAIN. « Markov random field texture models ». *IEEE Transactions on pattern analysis and machine intelligence*, 5-1:25–39, 1983.

[31] Tobi DELBRÜCK. « Silicon retina with correlation-based, velocity tuned pixels ». *IEEE Transactions on Neural Networks*, 4-3:529–541, 1993.

[32] Ulrich ECKHARDT et Gerd MADERLECHNER. « Invariant thinning ». *International Journal of Pattern Recognition and Artificial Intelligence*, 7-5:1115–1144, 1993.

[33] Jan-Erik EKLUND, Christer SVENSSON, et Anders ÅSTRÖM. « VLSI implementation of a focal plane image processor - a realization of the near sensor image processing concept ». *IEEE Transactions on VLSI systems*, 4-3:322–335, 1996.

[34] Servando ESPEJO, Angel RODRÍGUEZ-VÁZQUEZ, Rafael DOMÍNGUEZ-CASTRO, José Luis HUERTAS, et Edgar SÁNCHEZ-SINENCIO. « Smart-pixel Cellular Neural Networks in Analog Current-mode CMOS Technology ». *IEEE Journal of Solid-state Circuits*, 29-8:895–905, 1994.

[35] Fabrizio FERRARI, Jan NIELSEN, Paolo QUESTA, et Giulio SANDINI. « Space variant imaging ». *Sensor Review*, 15-2:17–20, 1995.

[36] Robert FORCHHEIMER, Per INGELHAG, et Christer JANSSON. « MAPP2200: a second generation smart optical sensor ». Dans SPIE,

éditeur, *Image processing and interchange*, volume 1659, pages 2–11, février 1992.

[37] Patrick GARDA. « *Vers une architecture intégrée de traitement combinatoire local des images* ». Thèse de Doctorat, Université de Paris-sud, 1984.

[38] Patrick GARDA, Arnaud REICHART, Hélène RODRIGUEZ, Francis DEVOS, et Bertrand ZAVIDOVIQUE. « Une rétine électronique automate cellulaire ». *Traitement du signal*, 5-6:435–449, 1988.

[39] Patrick GARDA, Bertrand ZAVIDOVIQUE, et Francis DEVOS. « Cellular hardware for a NCP based vision ». Dans *IEEE Workshop on CAPAIDM*, Miami - FL, novembre 1985.

[40] Stuart GEMAN et Donald GEMAN. « Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images ». *IEEE Transactions on pattern analysis and machine intelligence*, 6-6:721–741, 1984.

[41] Mats GÖKSTORP et Robert FORCHHEIMER. « Smart vision sensors ». Dans IEEE, éditeur, *International Conference on Image Processing*, pages 479–482, 1998.

[42] WeiXin GONG et Gilles BERTRAND. « A simple parallel 3D thinning algorithm ». Dans *International Conference on Pattern Recognition*, pages 188–190, Atlantic City - NJ, juin 1990.

[43] Zicheng GUO et Richard W. HALL. « Fast Fully Parallel Thinning Algorithms ». *Computer Vision, Graphics and Image Processing*, 55-3:317–328, 1992.

[44] Richard W. HALL. « Optimally Small Operator Supports for Fully Parallel Thinning algorithms ». *IEEE Transactions on pattern analysis and machine intelligence*, 15-8:828–833, 1993.

[45] Henk J.A.M. HEIJMANS et John GOUTSIAS. « Multiresolution signal decomposition schemes. Part 2: Morphological Wavelets ». Rapport Technique, Centrum voor Wiskunde en Informatica, 1999.

[46] C. Judith HILDITCH. « Linear skeletons from square cupboards ». *Machine Intelligence*, 4:403–420, 1969.

[47] Masatoshi ISHIKAWA, Kazuya OGAWA, Takashi KOMURO, et Idaku ISHII. « A CMOS vision chip with SIMD processing element array for 1ms image processing ». Dans IEEE, éditeur, *International Solid-state Circuit Conference*, pages 206–207, San Francisco, CA, 1999.

[48] Ben-Kwei JANG et Roland T. CHIN. « Analysis of thinning algorithms using mathematical morphology ». *IEEE Transactions on pattern analysis and machine intelligence*, 12-6:514–551, 1990.

[49] Maurice KARNAUGH. « The map method for synthesis of combinational logic circuits ». *AIEE Transactions on Comm. Elec.*, 72:593–598, 1953.

[50] Scott KIRKPATRICK, C. Daniel Jr GELLATT, et M.P. VECCHI. « Optimization by simulated annealing ». *Science*, 220:671–690, 1983.

[51] Takashi KOMURO, Idaku ISHII, et Masatoshi ISHIKAWA. « Vision chip architecture using general-purpose processing elements for 1ms vision system ». Dans IEEE, éditeur, *Workshop on Computer Architecture for Machine Perception*, pages 276–279, Boston, MA, octobre 1997.

[52] T. Yung KONG. « On the problem of determining whether a parallel reduction operator for n-dimensional binary images always preserve topology. ». Dans *SPIE Conference on Vision Geometry*, Boston - MA, 1993.

[53] T. Yung KONG et Azriel ROSENFELD. « Digital Topology : Introduction and Survey ». *Computer Vision, Graphics and Image Processing*, 48:357–393, 1989.

[54] Lionel LACASSAGNE, Maurice MILGRAM, et Patrick GARDA. « Motion detection, labeling, data association and tracking in real-time on RISC computer ». Dans *International Conference on Image Analysis and Processing*, pages 520–525, Venice - Italy, septembre 1999. IEEE.

[55] Louisa LAM, Seong-Whan LEE, et Ching Y. SUEN. « Thinning methodologies : A Comprehensive Survey ». *IEEE Transactions on pattern analysis and machine intelligence*, 14:869–885, 1992.

[56] Christian LANTUÉJOUL. « *La squelettisation et son application aux mesures topologiques des mosaïques polycristallines.* ». Thèse de Doctorat, Ecole Nationale supérieure des Mines de Paris, 1978.

[57] Christian LANTUÉJOUL et Serge BEUCHER. « On the use of geodesic metric in image analysis ». *Journal of Microscopy*, 121:39–49, 1981.

[58] Longin Jan LATECKI, Ulrich ECKHARDT, et Azriel ROSENFELD. « Well-composed sets ». *Computer Vision and Image Understanding*, 61-1:70–83, 1995.

[59] Oleg B. LUPANOV. « A method of circuit synthesis (in Russian) ». *Izvestia VUZ Radiofizica*, 1:120–140, 1958.

[60] Cherng Min MA. « On Topology Preservation in 3D Thinning ». *CVGIP: Image Understanding*, 59-3:328–339, 1994.

[61] Cherng Min MA. « A 3D fully parallel thinning algorithm for generating medial faces ». *Pattern Recognition Letters*, 16:83–87, 1995.

[62] Cherng Min MA et Milan SONKA. « A Fully parallel 3D Thinning Algorithm and its Applications ». *Computer Vision and Image Understanding*, 64-3:420–433, 1996.

[63] Rémy MALGOUYRES. « Homotopy in 2-dimensional digital images ». Dans *7th Discrete Geometry for Computer Imagery*, volume 1347, pages

213–222, Montpellier - France, décembre 1997. Lecture Notes in Computer Science - Springer Verlag.

[64] Antoine MANZANERA, Thierry M. BERNARD, Françoise PRÊTEUX, et Bernard LONGUET. « Medial faces from a concise 3D thinning algorithm ». Dans *International Conference on Computer Vision*, pages 337–343, Kerkyra - Greece, septembre 1999. IEEE.

[65] Antoine MANZANERA, Thierry M. BERNARD, Françoise PRÊTEUX, et Bernard LONGUET. « Ultra fast skeleton based on an isotropic fully parallel algorithm ». Dans *8th Discrete Geometry for Computer Imagery*, volume 1568, pages 313–324, Marne-La-Vallée - France, mars 1999. Lecture Notes in Computer Science - Springer Verlag.

[66] Antoine MANZANERA, Thierry M. BERNARD, Françoise PRÊTEUX, et Bernard LONGUET. « A unified mathematical framework for a compact and fully parallel n-D skeletonization procedure ». Dans *Vision Geometry VIII*, volume 3811, pages 57–68, Denver - CO, juillet 1999. SPIE.

[67] Petros MARAGOS et Ronald W. SCHAFER. « Morphological filters. Part II: Their relations to median, order-statistic, and stack filters ». *IEEE Transactions on Acoustics, Speech, and Signal*, 35:1170–1184, 1987.

[68] David MARR. *Vision*. W.H. Freeman Co., San Francisco, 1982.

[69] David MARR et Ellen C. HILDRETH. « Theory of edge detection ». *Proceedings of the Royal Society of London*, B-207:187–217, 1980.

[70] José Luis MARROQUÍN. « *Probabilistic solution of inverse problems.* ». Thèse de Doctorat, Massachusetts Institute of Technology, septembre 1985.

[71] José Luis MARROQUÍN, Sanjoy K. MITTER, et Tomaso POGGIO. « Probabilistic solution of ill-posed problems in computational vision. ». A.I. Memo 897, Massachusetts Institute of Technology, mars 1987.

[72] Eric MAURY. « Système CALADIOM ». CD ESR 17-72, Aérospatiale-Matra, 1999.

[73] Etienne MEMIN, Fabrice HEITZ, et François CHAROT. « Efficient parallel non-linear multigrid relaxation algorithms for low-level vision applications ». Rapport Technique, Institut de Recherche en Informatique et Systèmes Aléatoires, 1994.

[74] Damien S. MERCIER. « Rétines artificielles pour le guidage terminal ». Rapport Technique CTA 98 R 058, CTA/Géographie-Imagerie-Perception, 1998.

[75] Damien S. MERCIER, Philippe E. NGUYEN, et Thierry M. BERNARD. « Looking for Histograms on the Power Supply of an Artificial Re-

tina ». Dans *Proc. SPIE, Vol. 2950, Advanced Focal Plane Arrays and Electronic Cameras*, pages 265–272, Berlin, Germany, octobre 1996.

[76] Nicholas C. METROPOLIS, Arianna W. ROSENBLUTH, Marshall N. ROSENBLUTH, Augusta H. TELLER, et Teller EDWARD. « Equations of state calculations by fast computing machine ». *Journal of Chemical Physics*, 21:1087–1091, 1953.

[77] Fernand MEYER. *The Levelings*. Heijmans, H. and Roerdink, J. eds., 1998.

[78] Alireza MOINI. « Vision Chips or Seeing Silicon ». Rapport Technique, The University of Adelaide, 1997.

[79] Alireza MOINI, Abdessalam BOUZERDOUM, Kamram ESHRAGHIAN, Andre YAKOVLEFF, Xuan Thong NGUYEN, Andrew BLANKSBY, Richard BEARE, Dereck ABBOTT, et Robert E. BOGNER. « An insect vision-based motion detection chip ». *IEEE Journal of Solid-state Circuits*, 32-2:279–284, 1997.

[80] Tonia G. MORRIS, Erica FLETCHER, Cyrus AFGHAHI, Sami ISSA, Kevin CONNOLLY, et Jean-Charles KORTA. « A column-based processing array for high-speed digital image processing ». Dans *Conference on Advanced Research in VLSI*, pages 42–56, Atlanta, GA, 1999.

[81] Jean-Michel MULLER. *Arithmétique des ordinateurs*. Masson, 1989.

[82] David W. MURRAY, Alex KASHKO, et Hillary BUXTON. « A parallel approach to the picture restoration algorithm of Geman and Geman on an SIMD machine ». Rapport Technique, GEC Research Ltd, 1986.

[83] Yang NI, Francis DEVOS, M. BOUJRAD, et J.H. GUAN. « Histogram-equalization-based adaptive image sensor for real-time vision ». *IEEE Journal of Solid-state Circuits*, 32-7:1027–1036, 1997.

[84] Jan OLSZEWSKI. « A flexible thinning algorithm allowing parallel, sequential and distributed application ». *ACM Transactions on Mathematical Software*, 18-1:35–45, 1992.

[85] Fabrice PAILLET, Damien S. MERCIER, et Thierry M. BERNARD. « Making the most of $15k\lambda^2$ silicon area for a digital retina PE ». Dans *Proc. SPIE, Vol. 3410, Advanced Focal Plane Arrays and Electronic Cameras*, pages 158–167, Zürich, Switzerland, mai 1998.

[86] Fabrice PAILLET, Damien S. MERCIER, et Thierry M. BERNARD. « Efficient Data Output From The Inner Of Large Size Cellular Array ». Dans *Int. Symp. on Circuits And System*, Geneva, Switzerland, mai 2000.

[87] Fabrice PAILLET, Damien S. MERCIER, Thierry M. BERNARD, et Eric SENN. « Low power issues in a digital Programmable Artificial Retina ».

Dans *Proc. IEEE Workshop on Low Power Design*, pages 153–161, Como, Italy, avril 1999.

[88] Kálmán PALÁGYI et Attila KUBA. « A 3D 6-subiteration thinning algorithm for extracting medial lines ». *Pattern Recognition Letters*, 19:613–627, 1998.

[89] Michael S. PATERSON, éditeur. *Boolean Function Complexity.* Cambridge University Press, 1992.

[90] Michael S. PATERSON et Uri ZWICK. « Boolean function Complexity ». Dans *12th Conference on Computational Complexity*, pages 247–259. IEEE, 1997.

[91] Theo PAVLIDIS. *Structural pattern recognition.* Springer, New York, 1977.

[92] Françoise PRÊTEUX. *On a distance function approach for gray-level mathematical morphology.* Edward Dougherty ed., 1991.

[93] Françoise PRÊTEUX, Catalin FETITA, André CAPDEROU, et Philippe GRENIER. « Modeling, segmentation and caliber estimation of bronchi in high-resolution computerized tomography ». *Journal of Electronic Imaging*, 8-1:36–45, 1999.

[94] Françoise PRÊTEUX, Michel MOUBARAK, et Philippe GRENIER. « Pattern recognition in pulmonary computerized tomography images using Markovian modeling ». Dans *Biomedical Image Processing II*, volume 1450, pages 72–83, San Diego - CA, février 1991. SPIE.

[95] Arnaud REICHART. « *Aspects algorithmiques d'une vision fruste d'un robot embarquable* ». Thèse de Doctorat, Université de Paris-sud, 1988.

[96] Luc ROBERT et Grégoire MALANDAIN. « Fast binary image processing using binary decision diagrams ». Rapport de recherche 3001, Institut National de Recherche en Informatique et en Automatique, 1996.

[97] Christian RONSE. « A topological characterization of thinning ». *Theoretical Computer Science*, 43:31–41, 1986.

[98] Christian RONSE. « Minimal test patterns for connectivity preservation in parallel thinning algorithms for binary digital images ». *Discrete Applied Mathematics*, 21:67–79, 1988.

[99] Azriel ROSENFELD et Avinash C. KAK. *Digital Picture Processing, Second Edition, Vol.1.* Academic Press, 1982.

[100] Azriel ROSENFELD et Avinash C. KAK. *Digital Picture Processing, Second Edition, Vol.2.* Academic Press, 1982.

[101] Denis RUTOVITZ. « Pattern Recognition ». *J.R. Statist. Soc.*, 129:504–530, 1966.

[102] Punam K. Saha, B.B. Chaudhuri, et D. Dutta Majumber. « A new shape preserving parallel thinning algorithm for 3D digital images ». *Pattern Recognition*, 30-12:1939–1955, 1997.

[103] Michel Schmitt. « *Des algorithmes morphologiques à l'intelligence artificielle.* ». Thèse de Doctorat, Ecole Nationale supérieure des Mines de Paris, février 1989.

[104] Michel Schmitt et Juliette Mattioli. *Morphologie Mathématique.* Masson, 1992.

[105] Michel Schmitt et Françoise Prêteux. « Un nouvel algorithme en morphologie mathématique : les r-h maxima et les r-h minima ». Dans *2ème Semaine Internationale de l'Image Electronique*, volume 2, pages 469–475, Nice - France, avril 1986.

[106] Jean Serra. *Image analysis and mathematical morphology.* Academic Press, Londres, 1982.

[107] Jean Serra. « Morphologie Mathématique ». Support de cours, Ecole Nationale Supérieure des Mines de Paris, 1987.

[108] Jean Serra. *Image analysis and mathematical morphology, part II.* Academic Press, Londres, 1988.

[109] Claude E. Shannon. *The synthesis of two-terminal switching circuits.* Bell Systems Tech. J., 1949.

[110] R. Stefanelli et A. Rosenfeld. « Some parallel thinning algorithms for digital pictures ». *Journal of the A.C.M.*, 18:255–264, 1971.

[111] Alan Stewart. « A one-pass thinning algorithm with interference guards ». *Pattern Recognition Letters*, 15:825–832, 1994.

[112] Tamás Szirányi, Josiane Zerubia, László Czúni, David Geldreich, et Zoltán Kato. « Image segmentation using Markov random field model in fully parallel cellular network architectures ». *Real time imaging*, 6:195–211, 2000.

[113] Hideyuki Tamura. « A comparison of line thinning algorithms from digital viewpoint ». Dans *International Conference on Pattern Recognition*, pages 715–719. IAPR, 1978.

[114] Demetri Terzopoulos. « Regularization of inverse visual problems involving discontinuities ». *IEEE Transactions on pattern analysis and machine intelligence*, 8-4:413–424, 1986.

[115] Robert Ulichney. *Digital Halftoning.* MIT Press, 1987.

[116] Julie Vandenbusche. « Etude sur l'implantation d'algorithmes de squelettisation à base de fonction distance sur la rétine programmable ». Rapport de projet personnel en laboratoire, Ecole Nationale Supérieure de Techniques Avancées, 1999.

[117] Luc VINCENT. « *Algorithmes morphologiques à base de file d'attente et de lacets. Extension aux graphes.* ». Thèse de Doctorat, Ecole Nationale supérieure des Mines de Paris, mai 1990.

[118] Ingo WEGENER, éditeur. *The complexity of Boolean functions.* Wiley-Teubner, 1987.

[119] Rei-Yao WU et Wen-Hsiang TSAI. « A new One-Pass Parallel Thinning Algorithm for binary images ». *Pattern Recognition Letters*, 13:715–723, 1992.

[120] Wei XIONG et Christine GRAFFIGNE. « A hierarchical method for the detection of moving objects in a sequence of images ». Dans *Neural, morphological and stochastic methods in image and signal processing*, volume 2568, pages 54–65, San Diego - CA, juillet 1995. SPIE.

[121] David X. D. YANG, Boyd FOWLER, et Abbas EL GAMAL. « A Nyquist-rate pixel-level ADC for CMOS image sensors ». *IEEE Journal of Solid-state Circuits*, 34-3:348–356, 1999.

[122] Shigeki YOKOI, Jun Ichiro TORIWAKI, et Teruo FUKUMURA. « Topological properties in digitized binary pictures ». *Systems, Computers, Controls*, 4-6:32–39, 1973.

[123] Laurent YOUNES. « Synchronous random fields and image restoration ». *IEEE Transactions on pattern analysis and machine intelligence*, 20-4:380–390, 1998.

[124] Uri ZWICK. « Concrete Complexity ». Notes de cours, The University of Trier, 1996.