Université de Montréal

MÉTHODES À PROGRESSION RAPIDE ET ANALYSE MULTIRÉSOLUTION POUR LA DÉTECTION DE CONTOURS EN TRAITEMENT D'IMAGES

par

Marc-André Désautels

Département de mathématiques et de statistique
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en Mathématiques appliquées

août 2002

Université de Montréal

Faculté des études supérieures

Ce mémoire intitulé

MÉTHODES À PROGRESSION RAPIDE ET ANALYSE MULTIRÉSOLUTION POUR LA DÉTECTION DE CONTOURS EN TRAITEMENT D'IMAGES

présenté par

Marc-André Désautels

a été évalué par un jury composé des personnes suivantes :

Véronique Hussin
(président-rapporteur)

Anne Bourlioux
(directrice de recherche)

Jean-Marc Lina
(co-directeur)

Fahima Nekha
(membre du jury)

Mémoire accepté le:
17 octobre 2002

RÉSUMÉ

Nous présentons une technique de détection de contours en deux dimensions avec comme application spécifique l'étude de formes anatomiques dans des images médicales. Cet algorithme modélise très bien les structures complexes comme le cerveau et le nerf optique, il est en mesure de s'adapter topologiquement, il peut être utilisé à de nombreux niveaux de résolutions, et a un temps de calcul de $O(N \log N)$, où N est le nombre de points dans le domaine. Notre technique est basée sur la méthode iso-niveaux introduite en [2], la méthode à progression rapide introduite en [1] et l'analyse multirésolution introduite en [9].

Nous utilisons cet algorithme pour détecter des contours dans des images. Plutôt que d'utiliser uniquement la méthode à progression rapide, nous allons décimer l'image à l'aide de l'analyse multirésolution pour diminuer le bruit présent. Nous allons ensuite extraire les frontières de l'image et réobtenir notre image de départ avec le contour retrouvé. À l'aide de cette technique, nous pouvons diminuer le bruit présent dans le contour cherché, sans avoir à utiliser de filtres sur l'image, ainsi faisant nous éliminons le risque de perdre de l'information potentiellement utile.

Mots clés : iso-niveaux, progression rapide, multirésolution, ondelettes, contours, détection.

ABSTRACT

We present a shape recovery technique in two dimensions with specific application to examining anatomical shapes from medical images. This algorithm is designed to deal in a robust fashion with extremely corrugated structures such as the brain and the optic nerve; it is topologically adaptable, can be utilized with multiples resolutions, and runs in $O(N \log N)$ time, where N is the number of points in the domain. Our technique combines the level-set shape recovery scheme introduced in [**2**], the fast marching method in [**1**] and the multi-resolution analysis based on wavelet representation as introduced in [**9**].

We use this algorithm to detect boundaries in images. Instead of using only the fast marching method, we first decimate our image with the multi-resolution decomposition to process out excessive noise. We then proceed to extract the boundaries in the coarsened image and we then go back to our original image with the coarsened shape detected at the previous stage. With this technique, we can diminish the noise present in the recovered shape without having to resort to any form of ad-hoc filtering of the image, therefore avoiding the loss of relevant information.

Key words : level set, fast marching, multi-resolution, wavelet, boundaries, detection.

SOMMAIRE

Depuis ces dernières années, nous assistons à un regain d'intérêt pour le traitement d'image et un certain nombre de sujets qui lui sont liés. La part croissante des images vidéo dans notre vie quotidienne et de leurs applications n'y est certainement pas étrangère : robotique mobile, imagerie satellite et médicale, réalité virtuelle, effets spéciaux, télé-surveillance et autres. Il est vrai aussi que le traitement des images et des séquences vidéo, longtemps synonymes de temps de calcul et de coût de stockage prohibitifs, est aujourd'hui à la portée de la puissance de nos ordinateurs personnels.

En tout état de cause, l'utilisation des équations aux dérivées partielles (EDP) de la théorie des évolutions de courbes et de surfaces est devenue, dans le cadre du traitement d'images, un sujet de recherche majeur au cours des dernières années. Beaucoup de travaux se font présentement sur l'application des EDP à la restauration d'images ainsi que l'utilisation des méthodes iso-niveaux, méthodes initialement utilisées pour la simulation numérique des évolutions de courbes par EDP, pour l'extraction de contours et le suivi d'objets.

Les techniques de multirésolution, par exemple celles basées sur la représentation en ondelettes, sont de plus en plus appliquées à des problèmes de traitement d'images. La caractéristique la plus intéressante des ondelettes est leur simplicité. Une seule fonction est translatée et dilatée pour générer une base pour l'analyse de toute une famille de fonctions d'une certaine classe.

Dans ce mémoire, nous utiliserons les méthodes iso-niveaux, et plus particulièrement une variante rapide de celles-ci, dite à progression rapide, pour l'extraction de contours en utilisant conjointement l'analyse multirésolution. Une représentation multiéchelle des données (l'image) est adaptée à l'extraction de l'information

recherchée à l'aide de l'analyse multirésolution. La méthode à progression rapide permet de retrouver les courbes segmentant les données en utilisant la représentation trouvée grâce à l'analyse par ondelettes.

REMERCIEMENTS

Je tiens à remercier en premier lieu mon bon ami Nick Virgilio, l'italien québécois (ou bien le québécois italien...), qui est toujours là lorsque j'en ai besoin. J'aurais beaucoup à te dire, mais en ces quelques lignes, je veux juste te signifier que j'ai bien de la chance d'être ton ami. Je te salue bien bas !

Je n'oublie pas non plus ma directrice Anne Bourlioux et mon co-directeur Jean-Marc Lina qui m'ont offert un sujet qui foisonnait de possibilités, ce qui m'a permis d'apprécier à sa juste valeur le temps que j'ai passé à faire ce mémoire. Ils ont de plus, toujours été présents pour me donner leurs commentaires judicieux, me permettant de continuer dans la bonne direction sans trop m'écarter du droit chemin.

Un gros merci à ma famille, Roger, Lucie et bien sûr ma sœur Émilie. Ils sont pour quelque chose dans la plupart (sinon la totalité) des quelques succès que j'ai pu avoir dans ma courte vie.

Il reste Pascal Charland et Francis Voyer, mes amis qui ne comprennent presque rien à ce que je fais (sans rancune !) mais avec qui on peut s'amuser sans parler de mathématiques et de physique.

En dernier lieu, je remercie aussi deux grandes amies (j'espère du moins qu'elles me considèrent également comme tel) que j'ai eu la chance de rencontrer durant mes années à l'Université, Joëlle Piché et Julie Picard, pour des raisons très différentes mais aussi tellement semblables...

P.S. : Un gros merci à ma marraine préférée, Carmen. C'est incroyable de se sentir autant aimé comme toi seule sait le faire !

Table des matières

Table des figures

Chapitre 1

INTRODUCTION

1.1. Contexte

L'objet de cette maîtrise est d'appliquer les idées des méthodes à progression rapide et de l'analyse multirésolution à la détection de contours en imagerie. Considérons une image possédant une forme complexe et des protubérances, celles-ci pouvant être cachées par du bruit, des éléments de cette image, des artefacts aléatoires ou non provenant de la prise de données, etc. Nous voulons retrouver un contour dans cette image, y compris ses protubérances "cachées" par différents facteurs en dehors de notre contrôle. Un exemple est donné à la Figure 1.1, nous voulons trouver le meilleur contour entre la zone blanche et la zone noire, sans oublier les protubérances. Notre stratégie consiste en la combinaison de la méthode à progression rapide et de l'analyse multirésolution.

Cette combinaison de méthodes nous permet de surmonter certains obstacles et donc de retrouver des formes dans des images qui n'auraient pas été trouvées en n'utilisant que les méthodes à progression rapide. La Figure 1.2 donne un exemple de l'utilisation de la méthode à progression rapide couplée avec l'analyse multirésolution. La Figure 1.2(a) représente le contour que l'on veut retrouver, la Figure 1.2(b) est le contour retrouvé avec l'aide de la méthode à progression rapide et la Figure 1.2(c) est le contour retrouvé grâce à la combinaison de la méthode à progression rapide et de l'analyse multirésolution. Les Figures 1.2(d)-(f) sont les agrandissements respectifs de deux parties de l'image particulièrement intéressantes. Comme on peut le constater dans la Figure 1.2(b), la méthode à

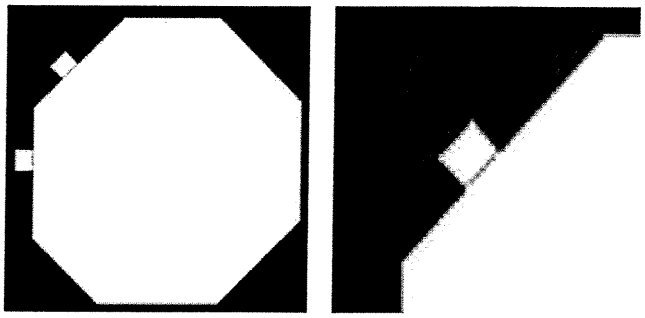


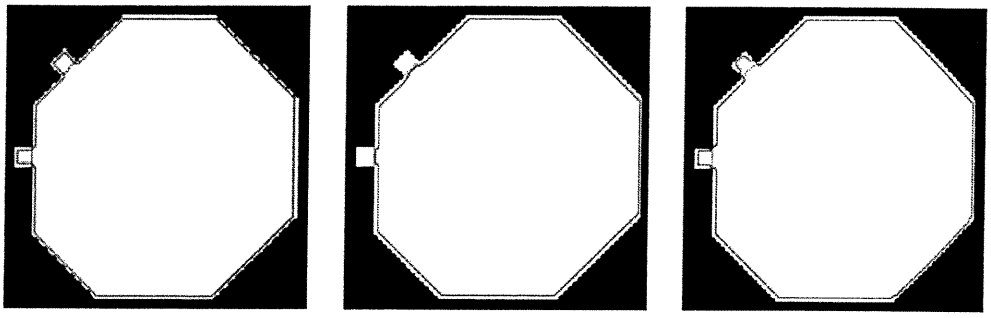
FIG. 1.1. *Exemple typique de la classe de problèmes que nous voulons résoudre à l'aide de la méthode à progression rapide et de l'analyse multirésolution. L'image contient des protubérances "cachées" qui doivent être retrouvées. On remarque que l'image n'a pas de directions privilégiées, car les protubérances sont aux extrêmes, à 90 degrés et à 45 degrés.*

progression rapide seule ne parvient pas à retrouver le contour désiré tandis que la méthode combinée, permet de retrouver une approximation du contour voulu de bien meilleure qualité. Il est donc intéressant d'étudier plus en détail les méthodes à progression rapide, l'analyse multirésolution et leur utilisation combinée pour en tirer une méthode plus générale, qui donne de meilleurs résultats dans des problèmes du type montré plus haut.

1.2. Méthodes à Progression Rapide

Les méthodes à progression rapide appartiennent à la catégorie des méthodes iso-niveaux ("level-set" en anglais). Ces méthodes sont utilisées dans une quantité impressionnante de sujets divers, elles permettent de simplifier énormément la résolution de certains problèmes en formulant ceux-ci dans une dimension supérieure. Nous nous intéresserons plus particulièrement à l'application de la méthode à progression rapide à la propagation de surfaces dans le but de détecter des contours dans une image. La référence [4] contient une bonne introduction à d'autres applications.

Ces méthodes sont basées sur les idées développées par Osher et Sethian [2], qui les ont appliqué notamment à la modélisation d'interfaces solides/liquides en mouvement. En deux dimensions, l'interface (le front) est une courbe fermée qui



(a) *Contour à retrouver.*

(b) *Contour retrouvé après avoir utilisé la méthode à progression rapide seule.*

(c) *Contour retrouvé après avoir utilisé la méthode à progression rapide conjointement avec l'analyse multirésolution.*

(d) *Agrandissement du contour que nous voulons retrouver.*

(e) *Agrandissement du contour retrouvé après avoir utilisé la méthode à progression rapide seule.*

(f) *Agrandissement du contour retrouvé après avoir utilisé la méthode à progression rapide conjointement avec l'analyse multirésolution.*

FIG. 1.2. *Récupération du contour à l'aide de différentes méthodes.*

ne s'intersecte pas et qui se propage normalement à elle-même avec une vitesse qui peut être constante ou bien varier selon la position de l'interface, sa courbure, etc.

L'idée centrale de l'approche iso-niveaux est de représenter le front $\Gamma(t)$ (une courbe dans le plan), comme le niveau zéro (u=0) d'une fonction u définie en 3

dimensions. Par exemple, à la Figure 1.3, nous ne faisons pas évoluer le cercle se trouvant dans le plan mais plutôt la surface dont le niveau zéro est le cercle dont on veut connaître l'évolution. Ainsi, ayant une surface fermée qui se déplace, c'est-à-dire, $\Gamma(t) : [0, \infty) \to \mathbb{R}^2$, l'approche iso-niveaux résulte en une formulation eulérienne pour le mouvement d'une hypersurface se propageant selon sa direction normale avec une vitesse F, où F peut être une fonction de divers arguments, incluant la courbure, la direction normale, etc. Mathématiquement, cette évolution

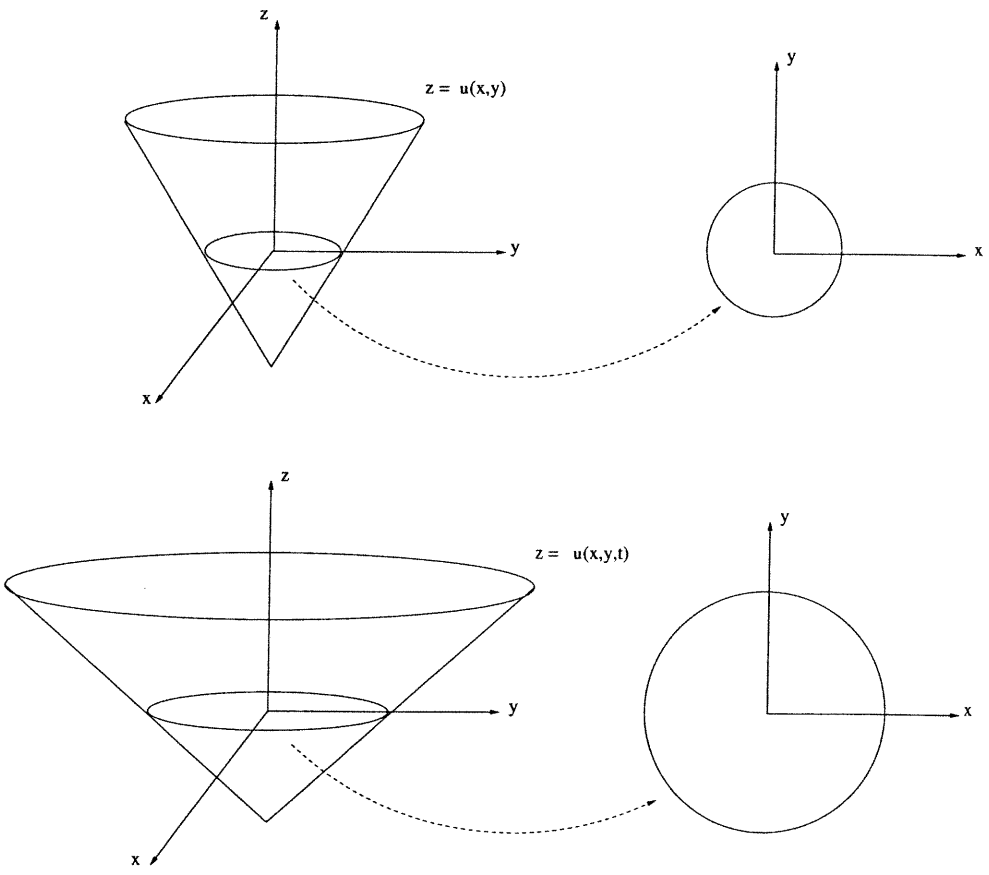


FIG. 1.3. *Cas des courbes. La fonction* $u : \mathbb{R}^2 \times \mathbb{R}_+ \to \mathbb{R}$ *évolue de telle sorte que le niveau zéro de la surface* $z = u(x, y, t)$ *se propage suivant l'équation désirée* $u_t = -F|\nabla u|$.

est décrite par l'équation suivante, de type Hamilton-Jacobi :

$$u_t + F|\nabla u| = 0, \quad u = 0 \quad \text{sur} \quad \Gamma. \tag{1.1}$$

Cette formulation eulérienne a de nombreux avantages mais demeure beaucoup trop demandante du côté des temps de calcul. Il convient donc de trouver

une meilleure façon de résoudre le problème de la propagation de courbes. Nous résoudrons plutôt une équation du type

$$F(x,y)|\nabla u(x,y)| = 1 \qquad u = 0 \quad \text{sur} \quad \Gamma. \tag{1.2}$$

Γ est la courbe que nous voulons faire évoluer et la fonction F est donnée. Ainsi, le mouvement du front est caractérisé par la solution d'un problème de valeurs frontières. Les détails de cette formulation seront donnés au chapitre 2.

Lorsque nous avons une image quelconque et que nous désirons retrouver un contour dans celle-ci, l'utilisation des méthodes à progression rapide implique la construction d'une fonction vitesse F adaptée à cette image. Nous démarrons la méthode avec un front quelconque. Celui-ci se propage dans l'image, il faut donc lui donner une fonction vitesse qui deviendra zéro près des contours si l'on veut que notre front s'arrête près de ceux-ci. On désire donc une fonction vitesse qui tend vers zéro rapidement dans les régions où se trouvent les grands gradients. En effet, les contours peuvent être identifiés comme correspondant à de grandes dérivées de l'intensité de la couleur. Plus ces dérivées sont élevées, plus le saut d'intensité est grand et plus nous sommes certains que nous sommes près d'un contour. La définition suivante de la fonction vitesse peut donc être utilisée :

$$F(x,y) = e^{-\alpha|\Delta G_\sigma * I(x,y)|}. \tag{1.3}$$

Ici, α est un paramètre choisi par l'utilisateur, paramètre qui changera selon l'image étudiée. G_σ est la gaussienne avec écart-type σ avec laquelle on fait la convolution avec notre image, pour lisser celle-ci. Enfin, $I(x,y)$ sont les données représentant l'image dans laquelle nous voulons détecter des contours et Δ est l'opérateur Laplacien. Avec ce modèle, la surface ne s'arrête que si les valeurs de la vitesse soient très près de zéro. La définition de l'équation (1.3) nous assure que la vitesse F tend vers zéro très rapidement. Puisque la vitesse atteint zéro rapidement, la valeur de notre surface deviendra très grande aux points où le Laplacien est grand, c'est-à-dire que la valeur de la surface sera grande au voisinage des contours, ce qui nous permettra de savoir que nous sommes tout près de ce que nous voulons découvrir. Nous parlerons plus en détail de ceci à la section 2.1.2. Nous utilisons dans ce cas le Laplacien de notre image, car comme nous verrons

plus tard, il est obtenu facilement grâce à l'analyse multirésolution. En général, dans la littérature, on utilise le gradient, plutôt que le Laplacien. La Figure 1.4 donne un exemple de la propagation d'une courbe dans une image en utilisant la méthode à progression rapide et un terme de vitesse construit à partir de l'image.

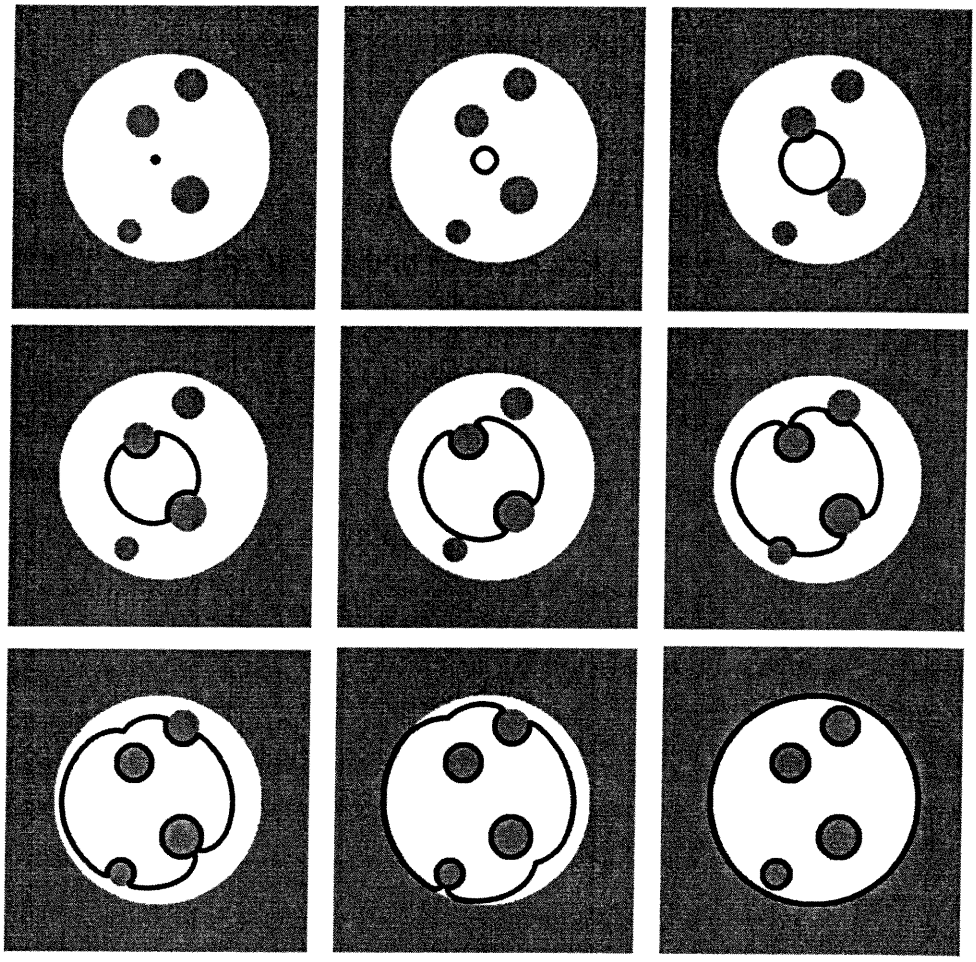


FIG. 1.4. *Application de la méthode à progression rapide à la propagation d'une courbe. La courbe initiale consiste en un petit cercle au centre de l'image et le front se propage, jusqu'à retrouver le contour voulu. Remarquons que le front se sépare en 5 parties pour retrouver tous les cercles intérieurs.*

1.3. Analyse Multirésolution

Comme nous l'avons vu au début du chapitre, à la Figure 1.2, les méthodes à progression rapide seules ne parviennent pas à retrouver les formes cachées par des artefacts (tout dépend bien sûr de la taille de ces artefacts, de leur intensité, etc.). Il convient donc de trouver une nouvelle façon de détecter ces protubérances, en permettant aux méthodes à progression rapide de surmonter les obstacles. La technique proposée ici repose sur l'analyse multirésolution des données. L'image initiale est projetée dans un espace d'approximation dont la résolution correspond à l'échelle d'acquisition, le niveau V_0. L'analyse multirésolution consiste à faire la projection de cette image dans un espace d'approximation deux fois plus grossier, le niveau V_1. Le complément d'information utile pour retrouver l'image initiale sont des fluctuations projetées dans les trois autres "secteurs" : ce sont les coefficients en ondelettes. Cette décomposition peut être répétée sur la projection dans l'espace d'approximation de résolution inférieure : on obtient ainsi une représentation pyramidale de l'image initiale, comme schématisé à la Figure 1.5. On peut ensuite appliquer l'analyse multirésolution au niveau V_1 pour obtenir un nouveau niveau V_2 de résolution inférieure et ainsi de suite. La Figure 1.6 donne un exemple de cette analyse à plusieurs résolutions, appliquée à l'image "mandrill" et aussi à l'image d'un carré. L'analyse multirésolution, comme son

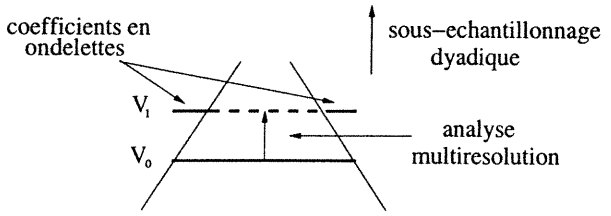


FIG. 1.5. *Schéma de l'analyse multirésolution. Le niveau V_0 est l'image initiale et le niveau V_1 est l'image lissé à un niveau de résolution plus grossier. V_0 peut donc être une image à une résolution quelconque, pas nécessairement la résolution de départ.*

nom l'indique, permet d'étudier l'image à plusieurs résolutions tout en gardant toute l'information nécessaire pour recréer l'image originale en sauvegardant les

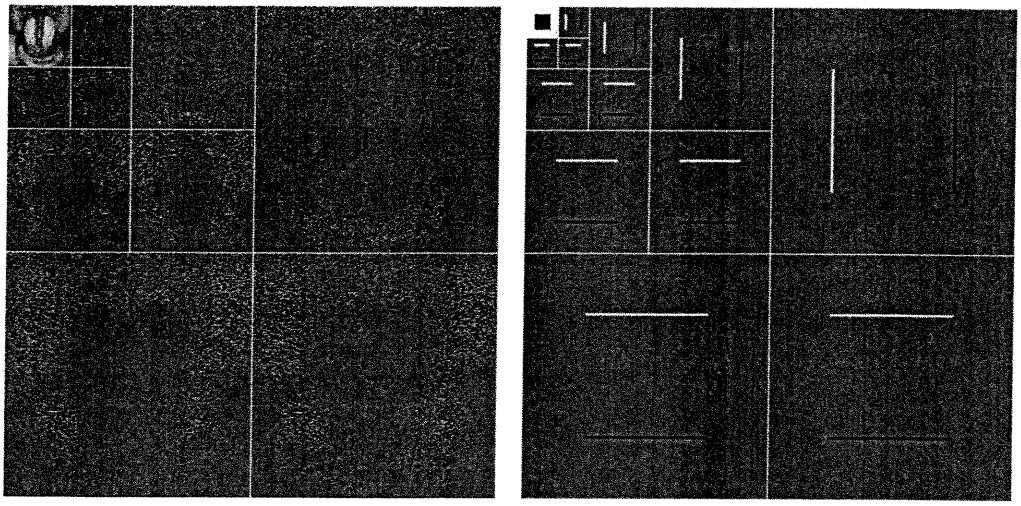


FIG. 1.6. *Analyse multirésolution du singe et d'un carré noir sur un fond blanc, décomposés respectivement sur 3 et 4 résolutions. Les pixels noirs, gris et blancs correspondent respectivement aux coefficients en ondelettes positifs, nuls et négatifs.*

coefficients en ondelettes calculés durant l'analyse. La Figure 1.7 montre exactement ce qui se passe à chaque utilisation de l'analyse multirésolution, la précision diminue comparée à l'image de départ, mais l'information perdue est conservée sous la forme de coefficients en ondelettes.

L'analyse par ondelettes ne permet pas seulement de faire l'analyse multirésolution d'un signal. Il nous est possible de faire la synthèse de celui-ci en utilisant l'analyse par ondelettes. La Figure 1.8 schématise la synthèse multirésolution. L'image à un niveau de résolution deux fois plus grossier, le niveau V_1, peut être transformée en l'image originale du niveau V_0. L'information qui manque à V_1 pour reconstruire V_0 se trouve dans les coefficients en ondelettes. Nous verrons au chapitre 3 que nous pouvons faire la synthèse du signal à n'importe quel niveau de résolution.

Nous avons donc expliqué dans les grandes lignes les deux approches que nous combinerons pour créer un nouvel algorithme de détection de contours dans une image. Cette combinaison permettra de récupérer des détails qui seraient autrement difficiles à retrouver. La Figure 1.9 nous permet de résumer rapidement comment utiliser conjointement les deux méthodes. Lorsque nous utilisons la mé-

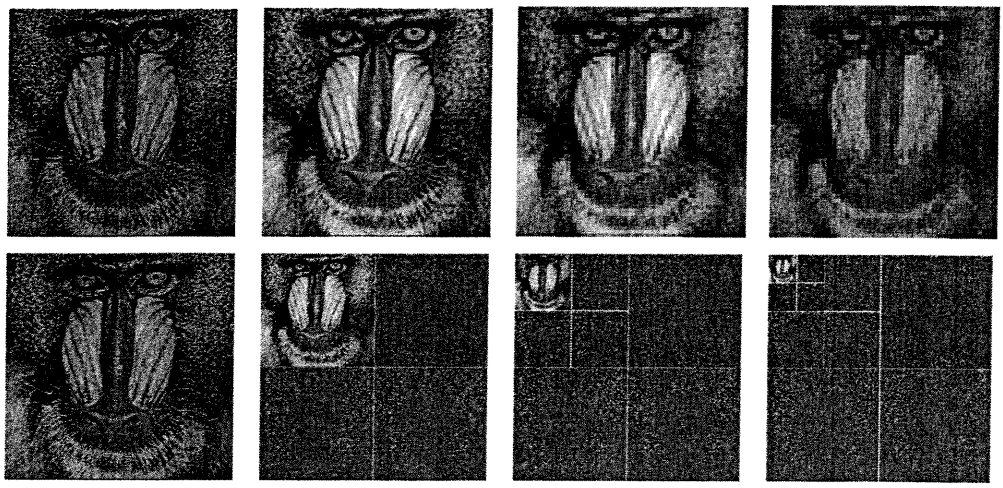


FIG. 1.7. *Image du singe à différentes résolutions de l'analyse par ondelettes. Les images du haut sont les images à résolution décroissante. Les images du bas sont ce que nous obtenons à l'aide de l'analyse multirésolution, l'image à une résolution plus faible et les coefficients en ondelettes qui forment le complément d'information nécessaire à la reconstruction de l'image initiale.*

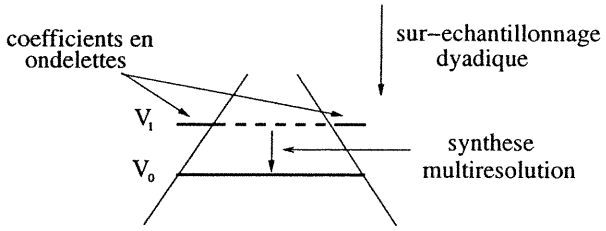


FIG. 1.8. *Schéma de la synthèse multirésolution. Les coefficients en ondelettes (au niveau V_1) contiennent l'information nécessaire à la reconstruction de l'image initiale.*

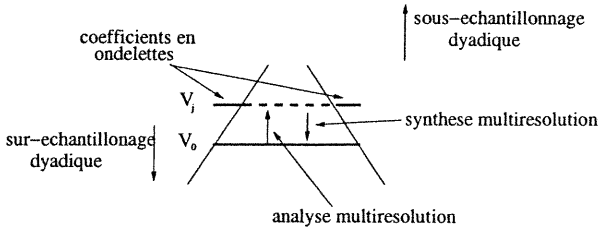


FIG. 1.9. *Schéma de l'algorithme multirésolution utilisé.*

thode à progression rapide seule, nous avons une image à une certaine résolution, le niveau V_0, et nous retrouvons le contour. Rien ne nous empêche pourtant de changer la résolution de notre image en utilisant l'analyse multirésolution. Nous voulons donc étudier une image réelle (qui a une certaine résolution initiale), à plusieurs résolutions plus faibles, par exemple le niveau V_j de multirésolution. En diminuant la résolution de l'image et en gardant les coefficients en ondelettes, nous sommes en mesure de redescendre (dans le schéma) vers l'image initiale et de la retrouver sans erreur. Lorsque nous utilisons les méthodes à progression rapide, nous ne les utilisons qu'à une seule hauteur sur le schéma, c'est-à-dire à la résolution initiale. L'analyse multirésolution va nous permettre de diminuer la résolution de notre image (aller vers le haut dans le schéma, vers le niveau V_j), d'utiliser les méthodes à progression rapide à cette nouvelle résolution (et ainsi d'enlever des détails dans notre image), et de redescendre vers l'image initiale, le niveau V_0, avec un contour moins bruité que celui obtenu de manière traditionnelle.

En plus de l'avantage d'étudier une image à plusieurs résolutions, l'analyse multirésolution, en utilisant les ondelettes complexes, permet d'obtenir en "bonus", le Laplacien de l'image étudiée. Ce Laplacien pourra donc être utilisé pour construire la fonction vitesse nécessaire à l'évolution de contours, comme suggéré dans la section précédente. La Figure 1.10 montre le Laplacien obtenu grâce à la multirésolution. Cette figure illustre bien la corrélation entre les contours et la magnitude du Laplacien. Nous expliquerons comment obtenir ce Laplacien à l'aide de la multirésolution à la section 3.3.1.

1.4. Plan du mémoire

Notre exposé suivra le plan suivant :

- **Chapitre 2** Le chapitre 2 présente les méthodes à progression rapide. Il comprend un résumé de la dérivation de cette approche ainsi qu'une description de la méthode pour sa résolution numérique.
- **Chapitre 3** Nous présentons plus en détail l'analyse par ondelettes en mettant l'accent sur l'analyse multirésolution. Nous nous concentrons sur les

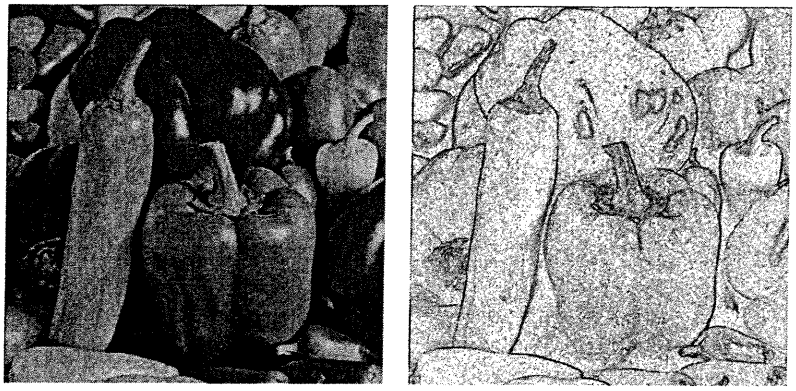


FIG. 1.10. *Le Laplacien associé à l'image piments. Les points où le Laplacien est élevé , donc les points où devraient se retrouver les contours, sont en noir et les points blanc sont des points où le Laplacien est faible.*

aspects qui seront utiles pour la mise en œuvre de notre méthode combinée. Nous discutons en particulier de la façon de trouver le Laplacien de l'image à chaque échelle et de son utilisation pour construire une fonction vitesse adaptée à chaque résolution.

- **Chapitre 4** Ce chapitre donne les détails des différents aspects de la mise en œuvre de la méthode combinée. Nous évaluerons aussi son efficacité en la testant en premier lieu sur des images synthétiques (c'est-à-dire construites pour les besoins de la cause). Nous testerons aussi la performance de la méthode sur des images réelles provenant de l'ophtalmologie et de résonnance magnétique nucléaire.

Chapitre 2

MÉTHODES À PROGRESSION RAPIDE

Nous présentons dans ce chapitre la formulation du problème que nous voulons résoudre, c'est-à-dire la détection de contours dans des images. Nous introduisons ensuite les méthodes numériques tout en motivant leur utilisation, ainsi que les algorithmes nécessaires à la détection de contours dans une image, quelle que soit sa résolution. Nous expliquons également la démarche à suivre pour construire une fonction vitesse adaptée aux différentes images que nous voulons étudier.

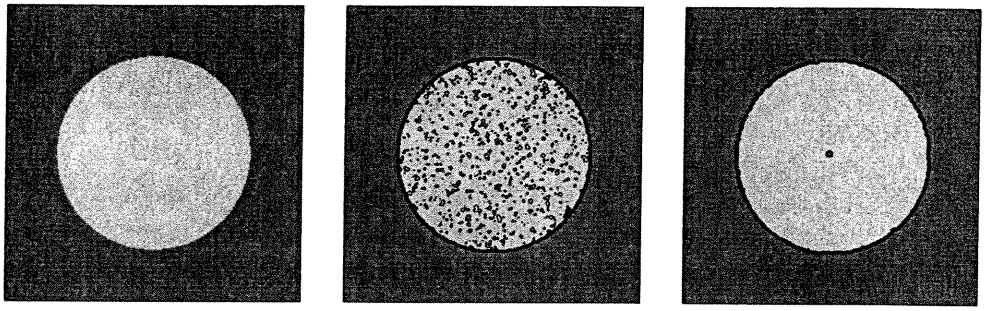
2.1. Détection de contours avec la propagation de courbes

Considérons une image médicale, dans laquelle on veut isoler une tumeur de l'image de fond. Imaginons également que cette tumeur correspond à une région où les pixels sont d'une intensité différente ; cela peut être dû à l'appareil utilisé, à l'utilisation de certains agents colorants, ... Le but est de localiser cette forme à l'intérieur de l'image le plus précisément possible.

2.1.1. Seuillage

À première vue, l'approche la plus directe à ce problème est d'utiliser un seuil arbitraire ; il suffit simplement de décider que la frontière de la forme désirée correspond aux pixels qui ont leur valeur entre celles associées respectivement aux régions intérieures et extérieures. Bien sûr, pour des images avec des régions bien délimitées, cette méthode fonctionne très bien. Dans plusieurs cas, par contre, l'image obtenue contient du bruit sous la forme de pixels dont les valeurs sont grandement différentes de celles de leurs voisins. Un simple seuillage pourrait

interpréter de petits anneaux autour de ces pixels comme faisant partie de la frontière, résultant en une représentation mauvaise et bruitée de la frontière recherchée. La Figure 2.1(a) nous montre une illustration de ce type. Nous avons



(a) *Image originale bruitée.*

(b) *Seuillage.*

(c) *Front se propageant. Le point au milieu de l'image est le front initial que nous propageons.*

FIG. 2.1. *Seuillage de l'image versus propagation d'un front.*

une région blanche par-dessus un fond gris ; des points de "bruit" gris sont représentés sur le fond blanc. Si nous tentons de retrouver la frontière entre le blanc et le gris en utilisant seulement la différence des pixels voisins, nous obtenons de multiples frontières en noir, comme à la Figure 2.1(b).

2.1.2. Un front en expansion

Comme alternative, imaginons un front qui se propage pour aller se "coller" à la frontière voulue. On part d'une forme de départ, une "graine" (*seed*), comme au centre de la Figure 2.1(c). L'idée est de faire évoluer ce front jusqu'à ce qu'il stoppe à la frontière. Cette stratégie utilise les méthodes à progression rapide pour la détection de contours, stratégie inventée par Malladi [**5**] ; son idée était d'exploiter ces techniques pour grandir le front "autour" des fausses frontières apparaissant en raison du bruit. Il y a deux composantes clé dans cette technique :

premièrement, l'habilité de nos algorithmes à exécuter des changements topologiques naturellement, et deuxièmement, un critère d'arrêt construit à partir du Laplacien de l'image.

Notre but est maintenant de définir une fonction vitesse pour les données de l'image, qui puisse être appliquée à notre courbe en mouvement et qui s'annulera effectivement aux frontières.

Il nous est possible de multiplier la fonction vitesse F par une quantité k_I. Le terme k_I est défini comme

$$k_I(x,y) = \frac{1}{1+|\Delta G_\sigma * I(x,y)|^n} \quad \text{(avec } n \text{ un exposant entier positif)}, \tag{2.1}$$

et a des valeurs proches de zéro dans des régions où le Laplacien est grand et des valeurs près de l'unité dans des régions avec une intensité relativement constante. La vitesse s'exprime donc comme,

$$k_I(x,y) \cdot F(x,y) = \frac{F(x,y)}{1+|\Delta G_\sigma * I(x,y)|^n} \quad \text{(avec } n \text{ un exposant entier positif)}, \tag{2.2}$$

L'expression $G_\sigma * I(x,y)$ dénote l'image convoluée avec un filtre gaussien de caractéristique σ. Le terme $\Delta(G_\sigma * I(x,y))$ est essentiellement nul partout sauf aux endroits où l'image change d'intensité rapidement. Ces changements correspondent aux contours de la forme désirée. En d'autres mots, la fonction filtre anticipe les chutes d'intensité dans le Laplacien de l'image, et retarde le front évoluant à l'approche de la région désirée.

Si on désire une fonction vitesse qui tende vers zéro plus rapidement que celle ci-haut, la définition suivante peut être utilisée :

$$F(x,y) = e^{-\alpha|\Delta G_\sigma * I(x,y)|}. \tag{2.3}$$

Avec ce modèle, le front ne s'arrête pas à la frontière à moins que les valeurs de la vitesse soient très près de zéro. La définition de l'équation (2.3) nous assure que la vitesse F tombe à zéro très rapidement.

Donc l'algorithme fonctionne comme suit. Un front de départ (typiquement un seul point) est initialisé à l'intérieur de la région désirée, grandit vers l'extérieur, et s'arrête à la frontière de l'image lorsque le filtre réduit la fonction vitesse F à une valeur près de zéro. Cette approche possède de nombreux aspects souhaitables.

- Le front initial peut être un ensemble de plusieurs fronts ; en raison des capacités topologiques de la méthode à progression rapide, ces fronts vont se fondre ensemble durant leur évolution vers la forme particulière recherchée.
- Le front peut suivre des formes particulièrement complexes.
- Cette technique peut être utilisée pour extraire des formes tri-dimensionnelles en initialisant une boule à l'intérieur de la région désirée.
- Les petits points de bruit où le gradient de l'image change sont ignorés ; le front se propage autour de ces points, se brise en deux, et l'anneau autour de ce point isolé se referme sur lui-même et disparaît.

Nous introduisons maintenant les méthodes nécessaires à l'étude de la propagation d'un front dans une image. Nous discuterons également des différents avantages, énumérés ci-dessus, de la méthode à progression rapide.

2.2. Premier essai dans l'implémentation d'une méthode de propagation de courbes

Avant de donner plus de détails sur la résolution numérique, nous allons répondre à la question suivante : Pourquoi envisageons-nous d'utiliser la méthode à progression rapide pour étudier l'évolution d'une courbe ?

Une implémentation naïve du calcul de l'évolution d'une courbe consisterait à échantillonner la courbe en un nombre suffisamment élevé de points et à suivre l'évolution de ces points. La stratégie est d'avancer la position des points selon la direction et l'amplitude de la vitesse de propagation en chaque point (les "flèches" à la Figure 2.2), de calculer les nouvelles "flèches", et encore avancer les points. On croit intuitivement que l'utilisation d'une plus grande quantité de points donnera une réponse plus précise. Malheureusement, une telle méthode a de nombreux inconvénients :

- Manque de précision et de stabilité. Le suivi de singularité à l'aide de cette méthode est très difficile, nous devons avoir un pas de temps très petit pour espérer avoir une précision suffisante.
- Obligation de gérer spécifiquement la topologie de la courbe. Par exemple, à la Figure 2.3, le cas de la vitesse constante négative $F = -1$ engendre

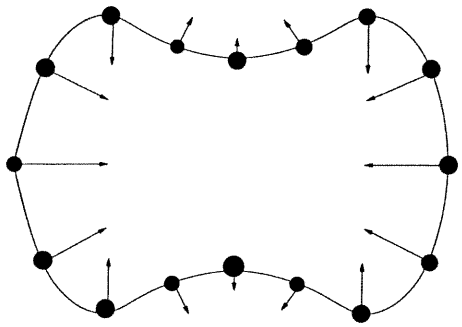


FIG. 2.2. *Premier essai pour suivre l'évolution d'une courbe en deux dimensions ($\mathbb{R}^2$).*

naturellement des changements de topologie. Il faudrait donc que l'évolution numérique soit capable de tenir compte des ruptures de la courbe en plusieurs composantes connexes, ce qui n'est pas aisé.

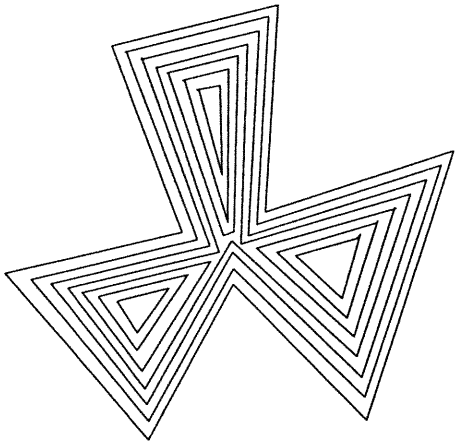


FIG. 2.3. *Évolution d'une courbe (ici $F = -1$). Des changements de topologie peuvent apparaître. La courbe initiale (à l'extérieur) se coupe en trois composantes connexes avant de disparaître.*

Comme on peut le constater, il est primordial d'avoir une méthode robuste et rapide pour l'évolution de courbe, et c'est pour ces raisons que nous utiliserons la méthode à progression rapide.

2.2.1. Topologie

On peut se demander pourquoi le changement de topologie est problématique et nous allons en discuter un peu plus en profondeur. Considérons en premier lieu deux régions séparées, par exemple la zone chaude de deux flammes, comme à

la Figure 2.4. Chacune évolue vers l'extérieur à une vitesse constante, on suit la région de combustion : l'évolution de l'interface est facilement prédite.

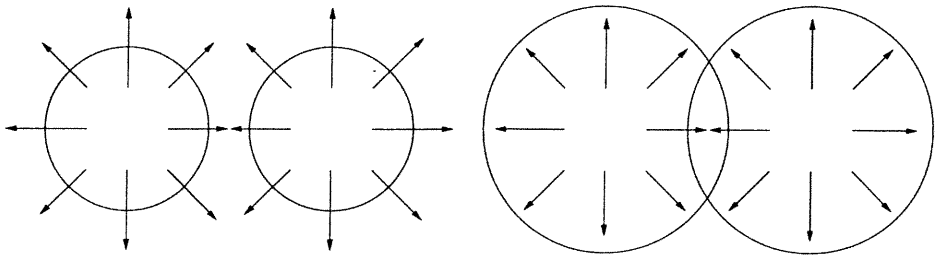


FIG. 2.4. *Nous avons la flamme initiale à droite et ensuite à gauche, nous avons la flamme plus tard dans le temps.*

Avec la progression des deux fronts, les interfaces se fondent éventuellement en un seul front, voir Figure 2.5. Un algorithme numérique basé sur une paramétrisation discrète fait face à un problème : dans la Figure 2.5, les paires de points localisées à l'intérieur de la région brûlée doivent être enlevées si nous voulons suivre l'évolution du "vrai" front de la flamme en propagation.

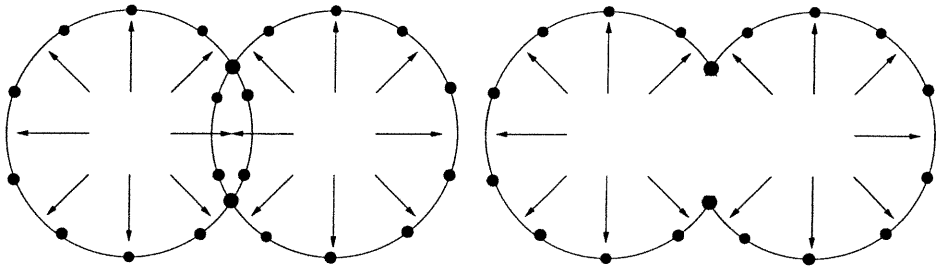


FIG. 2.5. *Seulement les points sur les bords correspondent à l'interface en mouvement.*

L'élimination systématique des points indésirables est une tâche qui prête rapidement à confusion : le faire en trois dimensions pour une surface est pratiquement impossible. C'est pourquoi la méthode à progression rapide est si versatile : elle tient compte du changement de topologie de la courbe sans traitement particulier.

2.3. Introduction aux Méthodes à Progression Rapide

La méthode à progression rapide est une technique numérique pour l'approximation de la solution de l'équation Eikonale non-linéaire de la forme

$$F(x,y)|\nabla u(x,y)| = 1 \quad \text{dans} \quad \Omega, \qquad F(x,y) > 0 \tag{2.4}$$

$$u = 0 \quad \text{sur} \quad \Gamma,$$

où Ω est un domaine dans $\mathbb{R}^2$. Les algorithmes de la théorie et de la méthode à progression rapide présentés dans ce texte sont valables dans $\mathbb{R}^n$, néanmoins pour les besoins de notre application, nous discuterons uniquement de la méthode dans $\mathbb{R}^2$. On suppose pour l'instant que $F(x,y) > 0$ est connu. Nous verrons un peu plus tard (équation 2.5) que $u(x,y)$ correspond à une surface de temps d'arrivée et c'est grâce à elle que nous pourrons retrouver les contours de l'image.

Tel que discuté dans [1], une des difficultés majeures lorsqu'on désire résoudre ces équations est que la solution n'est pas nécessairement différentiable partout, même si les données sur la frontière sont lisses. Cette non-différentiabilité est intimement liée à la notion de solutions faibles appropriées. L'objectif est de construire des techniques numériques qui vont naturellement prendre en compte cette non-différentiabilité en construisant des schémas approximatifs précis et efficaces qui admettront des solutions non-lisses physiquements correctes, correspondant aux solutions faibles admissibles. La méthode à progression rapide, introduite en [1], atteint cet objectif grâce à deux composantes clés. Premièrement, en exploitant des schémas de viscosité amont ("upwind" en anglais), ceux-ci sélectionnent automatiquement des solutions non-différentiables admissibles. Deuxièmement, le recours à des techniques de tri rapides, rend cette méthode très efficace sur le plan numérique ; la complexité de ces algorithmes est $O(N\log N)$, où N est le nombre total de points dans le domaine.

Il existe une grande collection d'applications qui requiert la solution de l'équation Eikonale et donc utilisent la méthode à progression rapide. En particulier, nous étudierons les applications de cette méthode pour retrouver des contours dans des images.

Les avantages de cette méthode sont multiples :

- Les changements de topologie sont pris en compte sans effort particulier : le niveau zéro peut se fractionner, se fusionner ou former des angles sans traitement spécifique (Figure 2.6).
- Le schéma numérique d'évolution de u peut être étudié rigoureusement et adapté à la précision requise.
- La méthode se généralise facilement à l'évolution des hypersurfaces dans des dimensions supérieures.

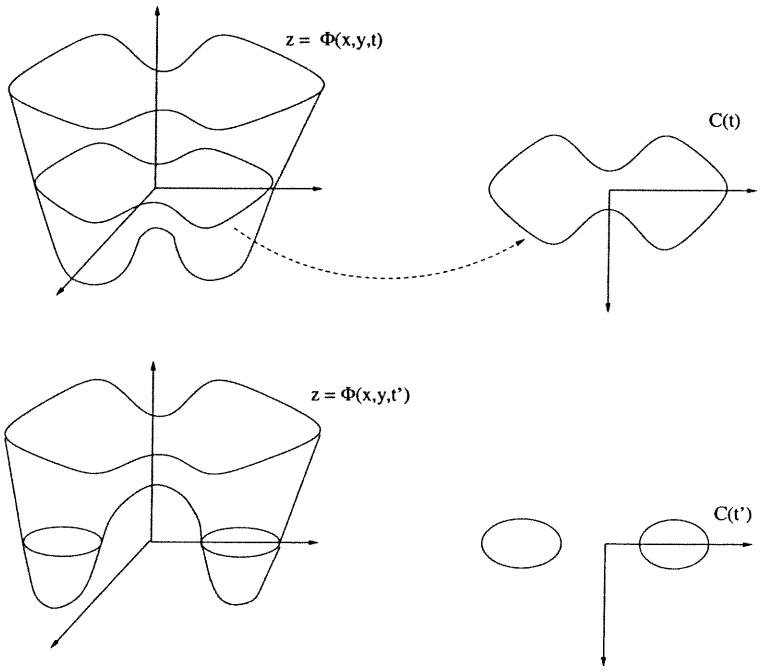


FIG. 2.6. *Changement de topologie. Le niveau zéro peut se casser, fusionner ou former des angles. Aucun traitement particulier n'est requis.*

2.4. Schémas amont et approximations numériques

Comme motivation de l'utilisation de schémas amont pour approximer l'opérateur gradient, considérons l'équation Eikonale à une dimension donnée par

$$\sqrt{u_x^2} = \frac{1}{F(x)} \qquad u(0) = 0.$$

Ici, le côté droit de l'équation, $F(x) > 0$, est donné, et le but est de construire $u(x)$ à partir de la condition frontière $u(0) = 0$. Nous remarquons immédiatement

que la solution de ce problème n'est pas unique ; si $v(x)$ est une solution du problème, alors $-v(x)$ est elle aussi solution du problème. Donc, nous nous penchons seulement sur les solutions non-négatives.

Nous pouvons imaginer construire la solution en nous déplacant vers l'extérieur en partant de la solution de départ. En résolvant également chaque problème séparément. Nous considérons les équations différentielles ordinaires suivantes :

$$\begin{aligned} \frac{du}{dx} &= \frac{1}{F(x)} \qquad u(0) = 0 \quad x \geq 0 \\ \frac{du}{dx} &= \frac{-1}{F(x)} \qquad u(0) = 0 \quad x \leq 0. \end{aligned}$$

En utilisant la notation standard des différences finies, soit $u_i = u(i\Delta x)$, et $F_i = F(i\Delta x)$, nous pouvons approximer chacune de ces deux solutions en utilisant la méthode de Euler :

$$\begin{aligned} \frac{u_{i+1}-u_i}{\Delta x} &= \frac{1}{F_i} \quad i > 0 \\ \frac{u_i-u_{i-1}}{\Delta x} &= \frac{-1}{F_i} \quad i \leq 0 \end{aligned}$$

où $u_0 = 0$. Ceci est un schéma amont ; nous calculons des dérivées en utilisant des points se trouvant *vers* la condition frontière. En d'autres mots, chaque équation différentielle ordinaire est résolue en s'éloignant de la condition frontière. Nous avons construit un schéma qui nous donnera toujours une fonction convexe, comme on peut le voir à la Figure 2.7.

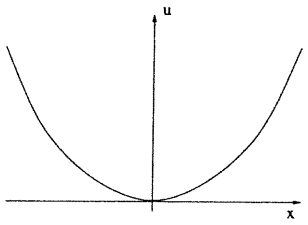


FIG. 2.7. *Le schéma donne toujours une fonction convexe.*

2.5. Formulation pour la Méthode à Progression Rapide

Plutôt que de suivre l'interface elle-même, la méthode à progression rapide utilise l'approche stationnaire au problème. À première vue, cela semble contre-intuitif ; nous allons échanger un problème de frontière en mouvement pour un problème où plus rien ne bouge ! Pour voir comment on s'y prend, imaginons une grille placée au-dessus de notre image, soit une interface en mouvement, comme

à la Figure 2.8. Supposons qu'on mesure le temps à chaque nœud du grillage.

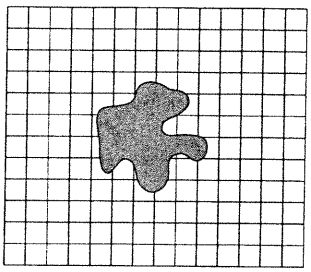


FIG. 2.8. *On note le temps d'arrivée à chaque nœud de la grille lorsque le front croise ces nœuds. À l'aide de ces valeurs, nous contruisons une surface qui donne le temps d'arrivée à chaque point de la grille.*

Lorsque le front coupe un point de la grille, on note le temps de passage u. Cette grille de temps de passage $u(x, y)$ détermine une fonction ; à chaque point de grille, $u(x, y)$ nous donne le temps auquel le front atteint le point (x, y).

Comme exemple, considérons l'évolution d'un cercle se propageant vers l'extérieur à vitesse constante. La région initiale (celle qui se trouve à gauche à la Figure 2.9) se propage vers l'extérieur, croisant chaque point du maillage. La fonction $u(x, y)$ nous donne une surface conique, qui est représentée à droite. Cette surface possède une propriété très intéressante, elle intersecte le plan xy *exactement* où se trouve la courbe initialement. La surface à droite à la Figure 2.9 est appelée la *Surface de temps d'arrivée.* Cette surface a comme propriété que n'importe quel de ses isoniveaux $u = T$ nous donne l'ensemble des points atteints au temps T.

Généralisons maintenant cette idée à une courbe fermée quelconque Γ (ici un cercle) dans le plan, se propageant dans la direction normale à elle-même avec une vitesse F. De plus, supposons que $F > 0$ et, par conséquent, que le front se propage vers l'extérieur. Une façon de caractériser la position de ce front en expansion est de calculer le temps d'arrivée $u(x, y)$ du front lorsqu'il croise chaque point (x, y). Puisque la vitesse F est inversément proportionnelle à la norme du gradient, nous devons avoir

$$F(x, y)|\nabla u(x, y)| = 1 \qquad u = 0 \quad \text{sur} \quad \Gamma \tag{2.5}$$

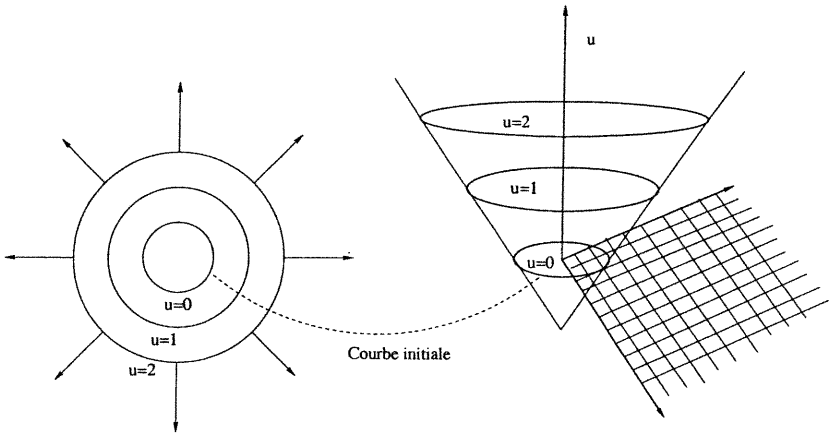


FIG. 2.9. *À chaque point de la grille se trouvant à droite, nous calculons le temps nécessaire à la courbe pour croiser ce point et nous construisons la surface de temps d'arrivée en prenant comme élévation ce temps calculé.*

Ainsi, l'évolution du front peut être caractérisée indirectement par la solution d'un problème de valeurs frontières, pour le temps de passage du front à un point donné.

2.5.1. Approximations de l'équation Eikonale

Nous allons maintenant construire les schémas numériques appropriés pour l'équation Eikonale en 2.5. Dans les équations qui suivent, le terme u représente le temps d'arrivée du front en expansion. C'est donc une fonction de deux variables dans le plan xy qui désigne une hauteur sur l'axe z. Nous allons utiliser les idées proposées dans **[3]** et **[4]** pour obtenir des schémas obéissant à la condition d'entropie. Le schéma de discrétisation amont pour le gradient est donné par Godunov **[6]** :

$$|\nabla u| \approx \left[\begin{array}{c} \max\left(D_{ij}^{-x}u, -D_{ij}^{+x}u, 0\right)^2 + \\ \max\left(D_{ij}^{-y}u, -D_{ij}^{+y}u, 0\right)^2 \end{array} \right]^{1/2} \tag{2.6}$$

où les opérateurs sont

$$D_{ij}^{-x}u = \frac{u_{ij} - u_{i-1,j}}{h} \quad D_{ij}^{+x}u = \frac{u_{i+1,j} - u_{ij}}{h} \tag{2.7}$$

Les opérateurs D^{-y} et D^{+y} dans l'autre direction se définissent de la même façon. L'équation Eikonale devient donc

$$\left[\begin{array}{c} \max\left(D_{ij}^{-x}u, -D_{ij}^{+x}u, 0\right)^2 + \\ \max\left(D_{ij}^{-y}u, -D_{ij}^{+y}u, 0\right)^2 \end{array} \right]^{1/2} = \frac{1}{F_{ij}}, \tag{2.8}$$

où F_{ij} est la vitesse au point de la grille ij.

Comment peut-on solutionner l'équation (2.8) ? Une solution, a été proposée par Rouy et Tourin en **[6]**. Considérons un ensemble composé d'un point et de ses voisins, comme montré à la Figure 2.10. Observons que l'équation (2.8) est une équation quadratique pour u_{ij}, en supposant que les valeurs voisines de u sont données. Donc, une approche est d'itérer la valeur de u à chaque point de la grille jusqu'à convergence :

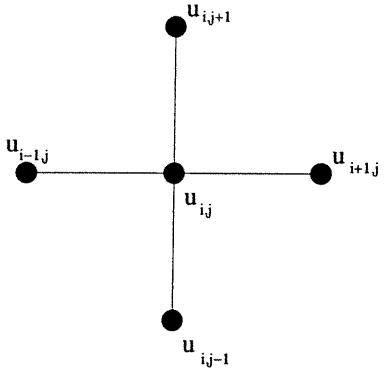


FIG. 2.10. *Voisinage d'un point de la grille.*

Pour iter=1,n

Pour i,j=1,Grille

Résoudre la quadratique pour u_{ijk}^{iter+1} , donné

$u_{i-1,j}^{iter}, u_{i+1,j}^{iter}, u_{i,j-1}^{iter}, u_{i,j+1}^{iter}$

FinPour

FinPour

Où n signifie que nous itérons jusqu'à convergence.

L'observation à la base de la méthode à progression rapide est que l'itération précédente contient une relation de causalité bien spécifique qui peut être exploitée. Dans la section suivante, nous allons décrire la méthode à progression rapide ; pour davantage de détails voir **[1, 4]**.

2.5.2. Causalité

L'idée centrale derrière la méthode à progression rapide est de construire la solution "en aval" ("downwind" en anglais) pour produire la solution u. La structure "en amont" de l'équation (2.8) signifie que l'information se propage à sens unique, en fait, des plus petites valeurs de u aux plus grandes. Donc, l'algorithme à progression rapide se base sur le fait de résoudre l'équation (2.8) en construisant la solution vers l'extérieur de la courbe fermée, en partant de la plus petite valeur de u. En fait, cette stratégie est sous-jacente à l'approche décrite plus tôt. Nous pouvons partir de la solution initiale dans une direction "en aval". Cet algorithme est rendu rapide en limitant la "zone de construction" à une mince bande ("*narrow band*") autour du front. L'idée est de se promener sur le front (Γ) en ne considérant que les points faisant partie de cette bande étroite et de faire évoluer cette bande en fixant les valeurs des points déjà trouvés. La sélection du point à visiter dans la zone de construction est réalisée de façon judicieuse. Considérons un problème comme celui de la Figure 2.11. Nous connaissons la valeur frontière à l'origine. La sphère noire à $u_{0,0}$ signifie qu'à ce point la valeur de u est connue (dans ce cas-ci, la valeur initiale), et les sphères blanches sont des points où la solution est inconnue, nous poserons donc leurs valeurs comme étant l'infini (∞).

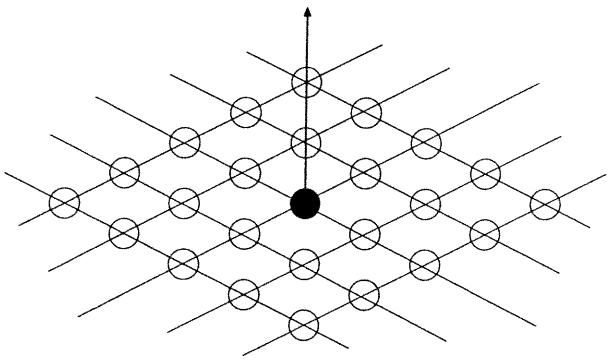


FIG. 2.11. *Début de la méthode à progression rapide.*

Nous pouvons débuter l'algorithme "en marchant" en aval à partir de cette valeur connue, calculant les nouvelles valeurs aux quatre voisins, comme montré à la Figure 2.12(a). Ceci nous donne les valeurs possibles de u à chaque point de la grille $u_{-1,0}, u_{1,0}, u_{0,-1}, u_{0,1}$, qui sont montrées comme des sphères d'un gris fonçé à la Figure 2.12(b).

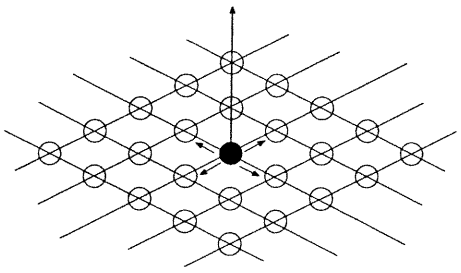
Maintenant, il faut propager en aval des valeurs données par les sphères grises, mais nous ne savons pas quels points choisir. La réponse se trouve dans l'observation que la *plus petite valeur de ces sphères grises doit être la bonne.* En raison du schéma de décentrage amont, aucun point ne peut être affecté par des points de la grille contenant de plus grandes valeurs de u. Donc, nous pouvons geler la valeur de u de la plus petite sphère grise (elle devient noire), considérer sa valeur comme connue et procéder avec l'algorithme. Cet algorithme est schématisé à la Figure 2.12.

Cet algorithme fonctionne, car le processus de calcul des valeurs de u à des points situés en aval du point étudié, ne peut pas donner une valeur plus petite que celle déjà trouvée. Donc, nous pouvons faire évoluer la solution, en sélectionnant toujours le point de la bande étroite ayant la valeur minimum pour u, et ensuite en réajustant ses voisins (voir Figure 2.13). La vitesse de cet algorithme vient d'une technique dite de "heap sort" pour localiser efficacement le plus petit élément de la bande étroite.

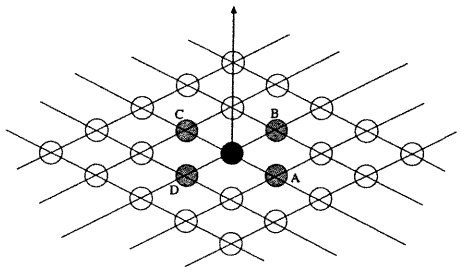
Ainsi, la méthode à progression rapide peut être résumée comme suit : Premièrement, étiqueter les points dans la condition initiale comme étant *Vivant.* Ensuite étiqueter comme étant *Près* tous les points se trouvant à un point de la grille des points *Vivant.* Finalement, étiqueter comme étant *Loin* tous les autres points de la grille.

Alors, la boucle sera la suivante :

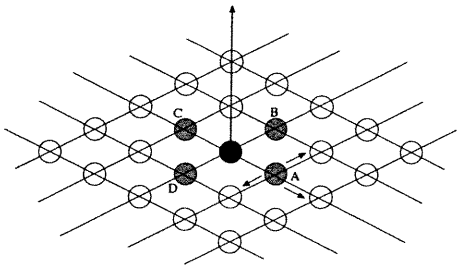
- Débuter Boucle : Identifier le point *Essai*, le point dans *Près* avec la plus petite valeur de u ;
- Étiqueter comme étant *Près* tous les voisins de *Essai* qui ne sont pas dans *Vivant.* Si le voisin est dans *Loin*, l'enlever de cette liste et l'ajouter à *Près* ;



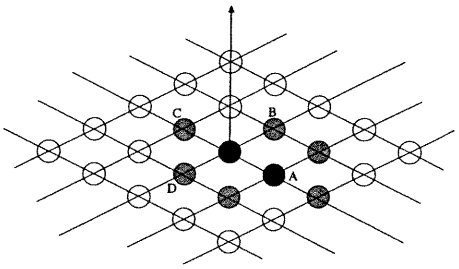
(a) *Retrouver les valeurs aval.*



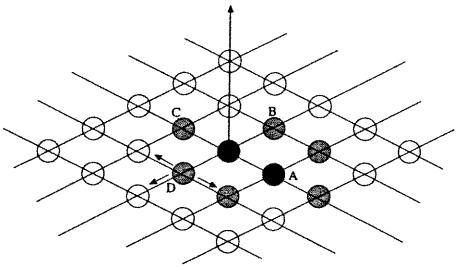
(b) *Calculer les nouvelles valeurs possibles.*



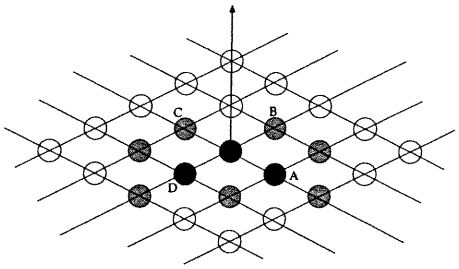
(c) *Choisir la plus petite valeur aux sphères grises (ici en A).*



(d) *Geler la valeur de A, trouver les valeurs voisines.*



(e) *Choisir la plus petite valeur aux sphères grises (ici en D).*



(f) *Geler la valeur de D, trouver les valeur des voisins.*

FIG. 2.12. *Évolution de la méthode à progression rapide.*

- Recalculer les valeurs de u pour tous les voisins de *Essai* en résolvant l'équation quadratique (2.8) ;
- Ajouter le point *Essai* à *Vivant* ; l'enlever de *Près* ;
- Retourner au début de la Boucle.

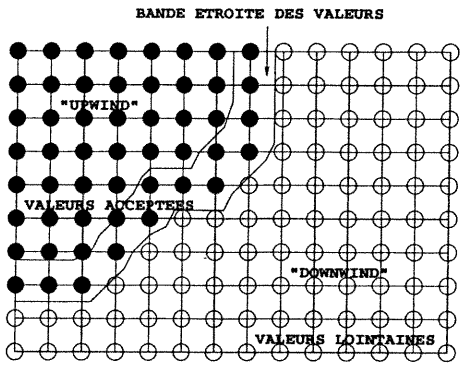


FIG. 2.13. *Construction amont des nouvelles valeurs.*

2.5.3. "Heap Sort" et efficacité calculatoire

La clé pour avoir une version efficace de la technique expliquée plus haut se trouve dans la façon de trouver rapidement le point dans la bande étroite ayant la plus petite valeur de u.

Il y a plusieurs façons d'ordonner les éléments dans *Près* pour pouvoir ensuite trouver la plus petite valeur. Dans notre cas, imaginons que nous avons une structure ordonnée des éléments dans *Près*. Lorsqu'un point est accepté, ses voisins sont renouvellés, et leur valeur de u peut changer. Donc, seulement une petite partie de la structure doit être ordonnée dans le but de réordonner l'ensemble de départ.

Cela nous mène naturellement à une variation sur un algorithme "heap" avec des pointeurs pour ranger les valeurs de u. Précisément, nous utilisons une structure dite "min-heap", voir [7]. Un "min-heap" est un arbre complet binaire avec la propriété que la valeur à n'importe quel noeud est plus petite ou égale aux valeurs de ses enfants. En pratique, on représente l'arbre par un vecteur en plaçant un noeud à la position k et ses enfants aux positions $2k$ et $2k+1$. Par cette définition, le parent d'un noeud donné k est situé à $k/2$. Donc, la racine contenant le plus petit élément est placée à la position $k = 1$ dans le vecteur.

Les valeurs de u sont gardées, ainsi que leurs indices qui donnent leur position dans la grille. L'algorithme fonctionne en cherchant en premier lieu le plus petit élément dans *BandeÉtroite*; cette opération, **FindSmallest**, est constituée de l'élimination de la racine et ensuite de l'opération **DownHeap** (Figure 2.15) pour

s'assurer que les éléments restants satisfont la propriété de l'arbre. L'algorithme procède ensuite en étiquetant les points voisins qui ne sont pas *Vivant*. Les voisins *Loin* sont ajoutés à l'arbre par une opération **Insert** (Figure 2.16) et les valeurs aux points qui restent sont trouvées grâce à l'équation (2.8). L'opération **Insert** fonctionne en augmentant la taille de l'arbre d'un nœud et en montant le nouvel élément jusqu'à sa nouvelle position à l'aide d'une opération **UpHeap** (Figure 2.14). Finalement, pour s'assurer que les nouvelles valeurs de u ne violent pas la propriété de l'arbre, nous devons faire une opération **UpHeap** partant de cette localisation et procédant jusqu'à la racine.

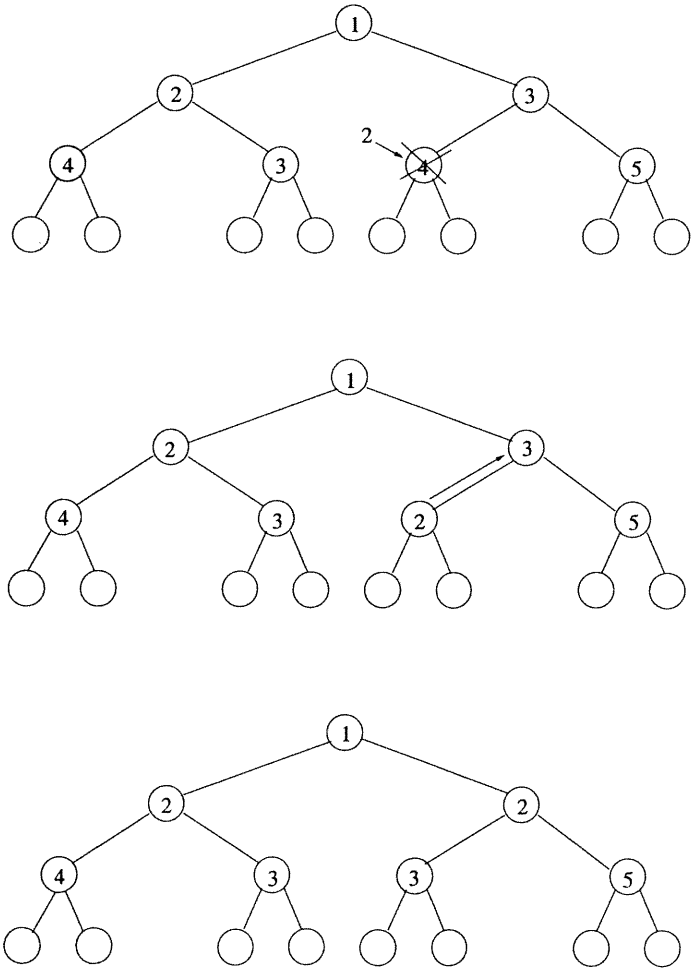


FIG. 2.14. *Structure de min-heap. L'arbre, initialement cohérent, voit la valeur d'un de ses noeuds substituée par une valeur moindre. La remontée de cette valeur en* $O(\log n)$ *suffit à rendre l'arbre de nouveau cohérent.*

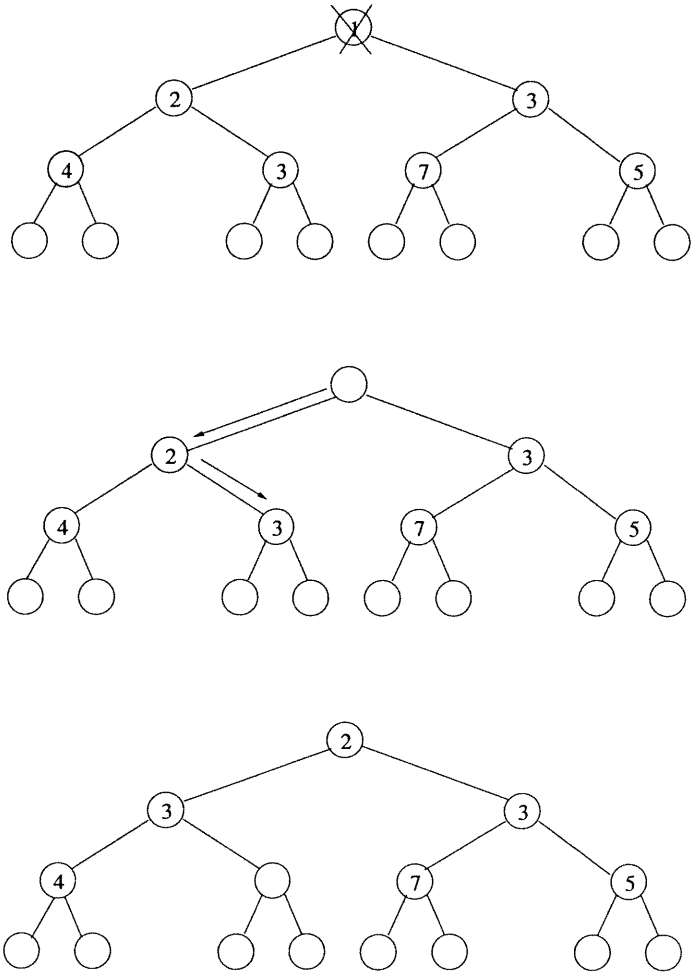


FIG. 2.15. *Structure de min-heap. L'arbre, initialement cohérent, voit sa racine détruite. Une descente avec remontée des valeurs en* $O(\log n)$ *suffit à rendre l'arbre de nouveau cohérent.*

Les opérations **DownHeap** et **UpHeap** (dans le pire des cas), prennent un élément et le transporte de la racine jusqu'à la base ou vice-versa. Donc, cela prend $O(\log N)$ en supposant qu'il y a N éléments dans l'arbre. Il est important de noter que l'arbre, qui est binaire, est toujours garanti de rester ordonné. Tout ce qui reste est l'opération de chercher pour les voisins dans *BandeÉtroite* du plus petit élément de l'arbre. La Figure 2.17 montre une structure typique d'arbre et une opération **UpHeap** après que l'élément à la position (2,7) soit renouvelé de la valeur 3.1 à la valeur 2.0.

L'algorithme est le suivant (voir Figure 2.18) :

- Les points de la grille sont de trois sortes :

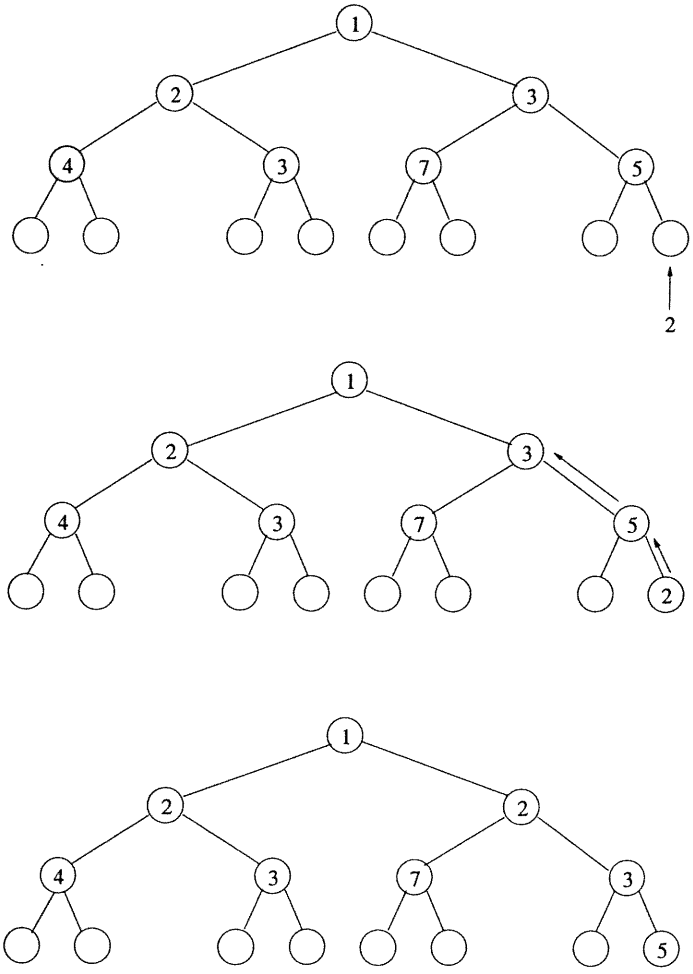


FIG. 2.16. *Structure de min-heap. L'arbre, initialement cohérent, voit une nouvelle valeur insérée au niveau d'une feuille. La remontée de cette valeur en $O(\log n)$ suffit à rendre l'arbre de nouveau cohérent.*

  - Ceux dont le temps de passage est définitivement connu : le front les a déjà dépassés.
  - Ceux dont le temps de passage a déjà été estimé lors des itérations précédentes.
  - Des points *lointains* qui n'ont pas encore été pris en compte et dont le temps de passage est inconnu et fixé arbitrairement à $+\infty$

- La position du front à $t = 0$ est initialisée par la donnée d'un nombre suffisant de temps de passage définitifs négatifs et des temps de passage estimés positifs pour leurs voisins.

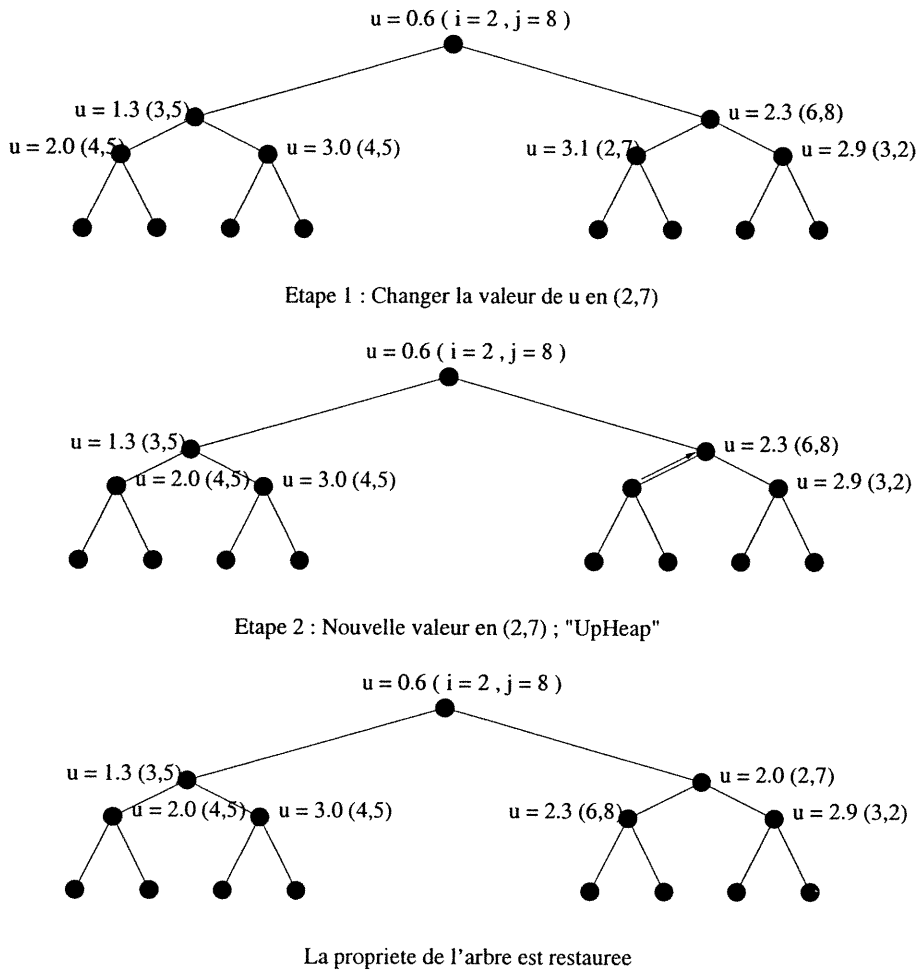


FIG. 2.17. *Structure en arbre et opération* ***UpHeap***

- Celui des points dont le temps de passage estimé est le plus petit est définitivement mis à jour avec l'équation :

$$\max\left(D_{ij}^{-x}u, -D_{ij}^{+x}u, 0\right)^2 + \max\left(D_{ij}^{-y}u, -D_{ij}^{+y}u, 0\right)^2 = \frac{1}{F_{ij}^2}. \qquad (2.9)$$

  et incorporé dans l'ensemble des points franchis par le front. Le temps de passage de ses voisins, dont certains étaient des points lointains, est lui aussi mis à jour.

- On répète l'étape 3 jusqu'à ce que tous les points soient franchis par le front.

La seule opération encore coûteuse est la détermination du point dont le temps de passage estimé est minimum. L'utilisation d'une structure de pile à minimum

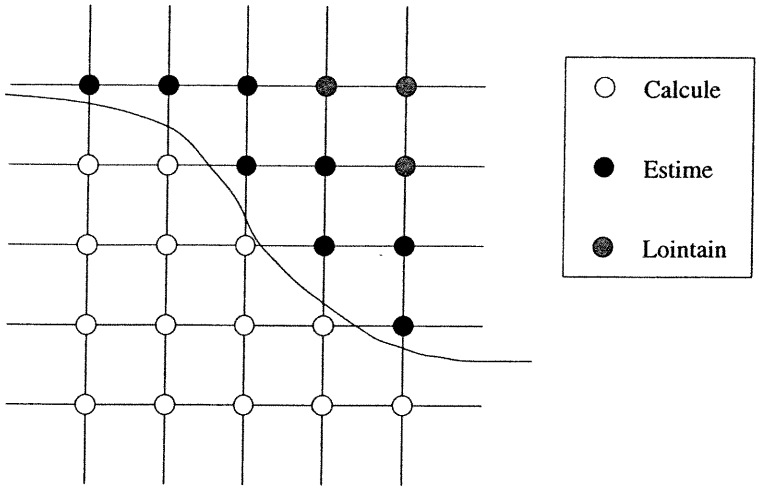


FIG. 2.18. *Méthode à progression rapide. En chaque point, on veut calculer le temps de passage du front. Certains points ont un temps de passage définitivement calculé, certains, déjà rencontrés, un temps estimé et d'autres, qui n'ont pas encore été approchés, un temps de passage inconnu.*

(voir *min-heap* dans [7]) permet une gestion rapide du problème. Il s'agit de maintenir un arbre binaire étiqueté dont les valeurs en chaque noeud sont inférieures aux valeurs des noeuds enfants. Pour n noeuds, le coût des opérations est réduit en moyenne à (pour un arbre équilibré) :

- Détermination du plus petit : $O(1)$ (simple extraction de la racine de l'arbre)
- Mise à jour d'une valeur (Figure 2.14) : $O(\log n)$ (une remontée vers la racine suffit car l'algorithme assure qu'une valeur est mise à jour par une valeur plus petite)
- Destruction d'une valeur (Figure 2.15) : $O(\log n)$ (élimination de la racine et descente)
- Insertion d'une valeur (Figure 2.16) : $O(\log n)$ (ajout d'une feuille et remontée)

2.6. Fonction distance

2.6.1. Initialisation

L'approche iso-niveaux (et celle à progression rapide) requiert une fonction initiale $u(x, y, t = 0)$ avec la propriété que le niveau zéro de cette fonction initiale correspond à la position du front initial. Une technique directe, et coûteuse, est de calculer la fonction distance signée de chacun des points de la grille au front initial. En pratique, la précision est requise seulement dans le voisinage du front initial, et une valeur discrète mesurée à l'aide de la grille peut suffire lorsque nous sommes éloignés.

2.6.2. Calcul rapide de la distance

Lorsqu'un ensemble de pixels est marqué sur une image, le calcul de la distance euclidienne de tous les autres pixels à cet ensemble peut se faire de manière rapide. L'algorithme en question peut s'adapter facilement à notre problème. Il reste ensuite à savoir si un point est à l'intérieur ou à l'extérieur du niveau zéro.

La méthode proposée par Danielsson [8] est basée sur l'idée suivante : ne pas mémoriser en chaque point la distance à l'ensemble mais la distance horizontale et verticale à l'ensemble.

Précisons les notations. Soit un ensemble I de pixels $(i, j), 0 \leq i \leq M-1, 0 \leq j \leq N-1$, et S un sous-ensemble de I, typiquement un contour. On veut en chaque point de I calculer la distance euclidienne au point de S le plus proche. On munit les couples de la norme euclidienne : $|(i, j)| = \sqrt{i^2 + j^2}$. Une façon naïve consisterait, pour chaque point de $I \backslash S$, à parcourir tout S pour trouver le point le plus proche. Cette méthode à un coût en $O(n^3)$ si M, N et $|S|$ sont de l'ordre de n. Danielsson procède en trois étapes seulement, une initialisation et deux balayages de l'image, soit un coût en $O(n^2)$ seulement :

- **Initialisation :** On mémorise les distances verticales et horizontales à S dans un tableau $L(i, j)$. Initialement :

$$L(i, j) = (0, 0) \quad \text{pour} \quad (i, j) \in S$$
$$L(i, j) = (+\infty, +\infty) \quad \text{pour} \quad (i, j) \in S$$

- **Premier balayage :**

  $(j = 1 \rightarrow N - 1)$

  $(i = 0 \rightarrow M - 1)$

$$L(i,j) = \min \begin{cases} L(i,j) \\ L(i-1,j-1) + (1,1) \quad (\text{si } i > 0)) \\ L(i,j-1) + (0,1) \\ L(i+1,j-1) + (1,1) \quad (\text{si } i < M-1)) \end{cases}$$

  $(i = 1 \rightarrow M - 1)$

$$L(i,j) = \min \begin{cases} L(i,j) \\ L(i-1,j) + (1,0) \end{cases}$$

  $(i = M - 2 \rightarrow 0)$

$$L(i,j) = \min \begin{cases} L(i,j) \\ L(i+1,j) + (1,0) \end{cases}$$

- **Deuxième balayage :**

  $(j = N - 2 \rightarrow 0)$

  $(i = 0 \rightarrow M - 1)$

$$L(i,j) = \min \begin{cases} L(i,j) \\ L(i-1,j+1) + (1,1) \quad (\text{si } i > 0)) \\ L(i,j+1) + (0,1) \\ L(i+1,j+1) + (1,1) \quad (\text{si } i < M-1)) \end{cases}$$

  $(i = 1 \rightarrow M - 1)$

$$L(i,j) = \min \begin{cases} L(i,j) \\ L(i-1,j) + (1,0) \end{cases}$$

$(i = M - 2 \rightarrow 0)$

$$L(i,j) = \min \begin{cases} L(i,j) \\ L(i+1,j) + (1,0) \end{cases}$$

(Nous désignons ici par $min(L_1, \ldots, L_n)$ celui des couples L_i dont la norme est minimale.) Après ces deux balayages, en chaque point (i,j) la distance à S est $|L(i,j)|$. La Figure 2.19 donne un exemple d'application de l'algorithme.

Dans le cas qui nous intéresse, nous avons utilisé l'algorithme de cette manière-ci :

- **Fronts initiaux arbitraires :** La technique à progression rapide ci-haut considérait que nous avions un front initial composé d'un seul ou de plusieurs points, il n'était pas difficile dans ces cas-là d'initialiser la méthode. Par contre, lorsque nous avons un front initial plus complexe, comme une courbe, il faudra utiliser la fonction distance pour initialiser notre problème. En effet, il suffit d'étiqueter tous les points comme ayant une valeur ∞. Ensuite, nous construisons la fonction distance signée à l'aide de l'algorithme expliqué plus haut. Propageons cette surface en avant et en arrière dans le temps jusqu'à ce qu'un ensemble de points ait été atteint dans chaque direction. Collectons les points avec un temps négatif comme étant des points *Vivant*, et les points avec un temps positif et se trouvant à faible distance du front comme étant des points *Près*. Finalement, on fait appel à la méthode à progression rapide.

2.7. Tout mettre ensemble

Nous avons discutés à la section 2.1.2 de la façon de procéder lorsque nous faisons évoluer un front dans une image. Nous avons créé une fonction vitesse contenant l'information relative aux contours de l'image. Nous désirons utiliser cette vitesse pour obtenir la surface de temps d'arrivée qui nous permettra d'obtenir le contour voulu. Voici rapidement la technique utilisée.

Lorsque la valeur de la fonction vitesse devient petite, alors la valeur du temps d'arrivée devient très grande. En fait, lorsque les dérivées sont grandes, la vitesse devient tellement petite que le temps pour se rendre à ces points devient très

∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0,0
0,0	∞	∞	∞	∞	∞	∞	∞	∞	∞	0,0	∞
∞	0,0	∞	∞	∞	∞	∞	∞	∞	0,0	∞	∞
∞	∞	0,0	0,0	0,0	∞	0,0	0,0	0,0	∞	∞	∞
∞	∞	∞	∞	∞	0,0	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

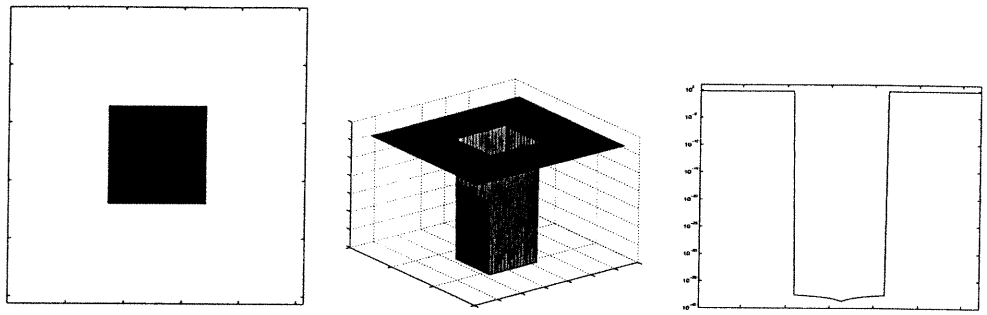
2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0	10,0	11,0	12,0	13,0
2,1	3,1	4,1	5,1	6,1	7,1	8,1	9,1	10,1	11,1	12,1	13,1
2,2	3,2	4,2	5,2	6,2	7,2	8,2	9,2	10,2	11,2	12,2	13,2
1,0	2,0	3,0	4,0	5,0	6,0	5,0	4,0	3,0	2,0	1,0	0,0
0,0	1,0	2,0	3,0	4,0	5,0	4,0	3,0	2,0	1,0	0,0	1,0
1,0	0,0	1,0	2,0	3,0	4,0	3,0	2,0	1,0	0,0	1,0	1,1
1,1	1,0	0,0	0,0	0,0	1,0	0,0	0,0	0,0	1,0	1,1	2,1
2,1	1,1	0,1	0,1	1,0	0,0	1,0	0,1	0,1	1,1	2,1	2,2
2,2	1,2	0,2	0,2	1,1	0,1	1,1	0,2	0,2	1,2	2,2	3,2
2,3	1,3	0,3	2,2	1,2	0,2	1,2	2,2	0,3	1,3	2,3	3,3

2,0	3,0	4,0	5,0	4,4	1,6	4,4	4,3	3,3	2,3	1,3	0,3
2,1	2,2	3,2	3,3	4,3	1,5	4,3	3,3	3,2	2,2	1,2	0,2
1,1	2,1	2,2	3,2	0,4	1,4	0,4	3,2	2,2	2,1	1,1	0,1
1,0	1,1	2,1	2,2	0,3	1,3	0,3	2,2	2,1	1,1	1,0	0,0
0,0	1,0	1,1	0,2	0,2	1,2	0,2	0,2	1,1	1,0	0,0	1,0
1,0	0,0	1,0	0,1	0,1	1,1	0,1	0,1	1,0	0,0	1,0	1,1
1,1	1,0	0,0	0,0	0,0	1,0	0,0	0,0	0,0	1,0	1,1	2,1
2,1	1,1	0,1	0,1	1,0	0,0	1,0	0,1	0,1	1,1	2,1	2,2
2,2	1,2	0,2	0,2	1,1	0,1	1,1	0,2	0,2	1,2	2,2	3,2
2,3	1,3	0,3	2,2	1,2	0,2	1,2	2,2	0,3	1,3	2,3	3,3

2.00	3.00	4.00	5.00	5.66	6.08	5.66	5.00	4.24	3.61	3.16	3.00
2.24	2.83	3.61	4.24	5.00	5.10	5.00	4.24	3.61	2.83	2.24	2.00
1.41	2.24	2.83	3.61	4.00	4.12	4.00	3.61	2.83	2.24	1.41	1.00
1.00	1.41	2.24	2.83	3.00	3.16	3.00	2.83	2.24	1.41	1.00	0.00
0.00	1.00	1.41	2.00	2.00	2.24	2.00	2.00	1.41	1.00	0.00	1.00
1.00	0.00	1.00	1.00	1.00	1.41	1.00	1.00	1.00	0.00	1.00	1.41
1.41	1.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	1.00	1.41	2.24
2.24	1.41	1.00	1.00	1.00	0.00	1.00	1.00	1.00	1.41	2.24	2.83
2.83	2.24	2.00	2.00	1.41	1.00	1.41	2.00	2.00	2.24	2.83	3.61
3.61	3.16	3.00	2.83	2.24	2.00	2.24	2.83	3.00	3.16	3.61	4.24

FIG. 2.19. *Les 3 étapes du calcul de distance :initialisation, première passe de haut en bas, deuxième passe de bas en haut. Le dernier tableau donne les distances effectivement calculées.*

élevé. La surface du temps d'arrivée aura donc une valeur très élevée près des contours. Prenons comme exemple la Figure 2.20, un carré noir sur fond blanc, nous sommes intéressés à retrouver le contour du carré en utilisant la méthode à progression rapide. Comme nous le remarquons, il suffit de prendre un plan et



(a) *Image d'un carré noir sur fond blanc.*

(b) *Résultat de l'algorithme à progression rapide. Les contours se trouvent où la valeur de la surface est très grande et augmente rapidement.*

(c) *Coupe du résultat de l'algorithme à progression rapide.*

FIG. 2.20. *Résultat de l'algorithme à progression rapide pour l'image d'un carré noir sur fond blanc. Lorsque l'algorithme rencontre un contour, la valeur de la surface augmente rapidement et devient très grande. Cela nous permet de savoir où se trouvent les contours une fois l'algorithme terminé.*

de couper notre surface à une certaine hauteur pour obtenir le contour voulu. La Figure 2.20(c) permet de mieux visualiser le résultat de l'algorithme, en montrant une coupe de la surface retrouvée. Cette coupe est représentée dans un graphique logarithmique, et nous remarquons que si le contraste est faible, par exemple dans le carré noir, la surface augmente lentement. Par contre, lorsque nous atteignons les contours, nous remarquons qu'il y a un grand saut dans la hauteur de notre surface, nous permettant de couper celle-ci à presque n'importe quelle hauteur pour obtenir le contour voulu.

2.8. Conclusion

Ce chapitre nous a permis de construire les schémas numériques nécessaires à l'utilisation de la méthode à progression rapide pour la détection de contours. Nous sommes maintenant en mesure de prendre une image à une résolution quelconque et de faire évoluer un front à l'intérieur de celle-ci pour retrouver un contour.

Chapitre 3

ONDELETTES ET ANALYSE MULTIRÉSOLUTION

Les bases en ondelettes sont des fonctions mathématiques qui permettent de représenter les fonctions à partir de leurs fluctuations locales calculées à différentes échelles. Cette notion a été développée indépendamment dans les domaines des mathématiques, de la mécanique quantique, de l'ingénierie électrique et en géologie sismologique. Des échanges entre ces domaines durant les dix dernières années ont conduit à de nombreuses nouvelles applications des ondelettes en compression d'images, en turbulence, pour la modélisation de la vision humaine, en radar et en prédiction de tremblements de terre. Dans ce chapitre, nous voulons savoir si l'utilisation conjointe de l'analyse multirésolution et des "méthodes à progression rapide" permet d'améliorer la fonction vitesse utilisée en exprimant celle-ci à partir des coefficients en ondelettes.

On filtre le signal (on projette la fonction), selon deux types d'espaces. D'une part, un espace d'approximation contenant la projection d'un objet à une résolution donnée. D'autre part, un espace de fluctuations contenant les oscillations locales de l'objet et mesurées à une échelle donnée. Ces fluctuations sont les détails qui manquent à l'approximation pour reconstruire le signal ou la fonction initiale. En 2 dimensions, on obtient ainsi un ensemble de projections de l'image : l'image originale est décomposée à différentes échelles en utilisant un algorithme ayant une architecture pyramidale. La décomposition est tensorielle (produit des

directions horizontales et verticales) et maintient constant, en raison de l'orthogonalité des ondelettes utilisées, le nombre de degrés de liberté requis pour décrire l'image. Le choix de l'ondelette est arbitraire. Toutefois, nous utilisons des fonctions pour lesquelles les coefficients sont à valeur complexe. Ce choix se justifie par la forme de ces coefficients qui utilisent le Laplacien de l'image à différentes résolutions étudiées. Celui-ci nous servira lorsque nous construirons la fonction vitesse nécessaire à la détection de contours dans une image dans le cadre de la méthode dite "à progression rapide".

3.1. Une courte introduction à l'analyse par ondelettes

Il est bien connu de l'analyse de Fourier qu'un signal peut être exprimé comme la somme, possiblement infinie, de sinus et de cosinus. Cette somme est nommée développement en série de Fourier. Dans certains cas, nous sommes intéressés à obtenir une bonne précision en espace de notre signal, mais ce développement ne nous permet qu'une résolution en fréquence et aucune en espace. C'est-à-dire qu'on sait exactement toutes les fréquences présentes dans un certain signal mais nous n'avons aucune information sur l'endroit où elles se trouvent ! Pour obtenir une meilleure résolution en espace, plusieurs solutions ont été développées.

L'idée sous-jacente aux représentations temps-fréquence est de découper le signal en plusieurs segments et d'ensuite analyser, en terme de fréquence, chacune des parties séparement. Il est clair que cette analyse nous donnera plus d'information sur la localisation des différentes composantes fréquentielles. Mais cela nous renvoie à un problème, de quelle manière devons nous couper le signal ? Supposons que nous voulons savoir exactement toutes les composantes spatiales présentes à un certain moment dans le temps. Nous coupons seulement une fenêtre très étroite, par exemple, en utilisant une impulsion de Dirac (delta). Transformons la dans le domaine fréquentiel et voyons ce qui se passe.

Puisque la convolution dans le plan temps est identique à une multiplication dans le plan fréquence et puisque la transformée de Fourier d'un delta de Dirac contient toutes les fréquences possibles, les composantes fréquentielles du signal seront étendues sur tout le domaine fréquence. En fait, cette situation est l'opposé

de la transformée de Fourier standard car nous avons maintenant une résolution en temps mais aucune en fréquence, d'où la recherche d'un compromis !

Bien qu'en traitement de signal on travaille avec des signaux discrets, la plupart des concepts seront présentés dans le cas continu et le passage au cas discret sera précisé lorsque nécessaire, par exemple dans le développement des algorithmes. On représentera par $f(t)$ une fonction continue et par $f[n]$ une fonction discrète. De plus, on notera $f^*(t)$ la conjugaison complexe de $f(t)$.

Soit $\phi_{u,\epsilon}(t)$ une fonction normalisée avec la norme $\mathbf{L^2}$, $||\phi_{\mu,\epsilon}(t)|| = 1$, on peut donc considérer $|\phi_{\mu,\epsilon}(t)|^2$ et $|\hat{\phi}_{\mu,\epsilon}(w)|^2$ comme des densités de probabilités en temps et en fréquence respectivement. On demande que ces fonctions soient localisées autour de u dans le temps (i.e. $u = \int_{-\infty}^{+\infty} t|\phi_{\mu,\epsilon}(t)|^2 dt$) et autour de ϵ dans les fréquences (i.e $\epsilon = \int_{-\infty}^{+\infty} w|\hat{\phi}_{\mu,\epsilon}(w)|^2 dw$). Avec un étalement autour de ces positions caractérisé par les variances :

$$\sigma_t^2(u,\epsilon) = \int_{-\infty}^{+\infty} (t-u)^2 |\phi_{\mu,\epsilon}(t)|^2 dt, \tag{3.1}$$

$$\sigma_w^2(u,\epsilon) = \int_{-\infty}^{+\infty} (w-\epsilon)^2 |\hat{\phi}_{\mu,\epsilon}(w)|^2 dw. \tag{3.2}$$

En convoluant une fonction $f(t)$ avec $\phi_{\mu,\epsilon}(t)$, la représentation temporelle $g(t) = f * \phi_{\mu,\epsilon}(t) = \int_{-\infty}^{+\infty} f(t-t')\phi_{\mu,\epsilon}(t')dt'$ contiendra l'information de $f(t)$ autour de $t = u$ dans une région de taille $\sigma_t^2(u,\epsilon)$ et sa transformée de Fourier $\hat{g}(w) = \hat{f}(w)\hat{\phi}_{\mu,\epsilon}(w)$ contiendra l'information en fréquence de $f(t)$ autour de $w = \epsilon$ avec un étalement $\sigma_w^2(u,\epsilon)$. On cherche donc des fonctions $\phi_{u,\epsilon}(t)$ telles que ces deux variables soient toutes deux minimales, pour avoir la meilleure précision possible, tant en fréquence qu'en temps.

Il est toutefois impossible de mettre simultanément $\sigma_t^2(u,\epsilon)$ et $\sigma_w^2(u,\epsilon)$ égaux à zéro. En effet, le principe d'incertitude de Heisenberg impose

$$\sigma_t(u,\epsilon)\sigma_w(u,\epsilon) \geq \frac{1}{2}, \tag{3.3}$$

l'égalité étant obtenue pour les fonctions $\phi_{u,\epsilon}(t)$ de la forme $\phi_{u,\epsilon}(t) = e^{i\epsilon t} g(t-u)$, où $g(t)$ est une gaussienne centrée en zéro. Il s'agit ici du meilleur compromis entre résolution temporelle et résolution en fréquence. En d'autres termes, un

signal ne peut tout simplement pas être représenté par un point dans le plan temps-fréquence.

Étudiée de façon intensive depuis une vingtaine d'années, la *transformée en ondelettes* ou *analyse par ondelettes* est une représentation temps-fréquence d'un signal.

L'analyse par ondelettes discutée ici est connue sous le nom de *transformée en ondelettes continue*. Les ondelettes sont générées à partir d'une seule ondelette de base $\psi(t)$, *l'ondelette mère*, en dilatant et en translatant celle-ci :

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right). \tag{3.4}$$

Dans l'équation (3.4), s est le facteur d'échelle, τ est le facteur de translation et le facteur $s^{-1/2}$ normalise l'énergie (en norme $\mathbf{L^2}$) pour toutes les échelles.

L'analyse en ondelettes s'écrit comme :

$$\mathrm{W}f(s,\tau) = \int f(t)\psi^*_{s,\tau}(t)dt, \tag{3.5}$$

où $*$ dénote le conjugué complexe. Les variables s et τ sont les nouvelles dimensions, échelle et translation, après la transformée en ondelettes. Pour compléter la théorie, l'équation (3.6) donne la transformée inverse, c'est-à-dire la synthèse,

$$f(t) = \iint \mathrm{W}f(s,\tau)\psi_{s,\tau}(t)d\tau ds. \tag{3.6}$$

Il est important de noter que dans les équations (3.4), (3.5) et (3.6) les fonctions de base ne sont pas spécifiées. Par contre, elle doivent satisfaire la condition d'admissibilité :

$$\int \frac{|\hat{\psi}(w)|^2}{|w|}dw < +\infty, \tag{3.7}$$

pour permettre de reconstruire un signal sans perte d'information, tel qu'exprimé par (3.6). Les contraintes données jusqu'ici ne définissent pas ψ de façon unique. Nous allons ajouter d'autres conditions sur ψ, et ainsi définir une famille d'ondelettes dont certains membres présentent des propriétés appropriées au problème étudié.

La condition d'admissibilité implique que la transformée de Fourier de $\psi(t)$ s'annule en zéro,

$$\hat{\psi}(0) = 0. \tag{3.8}$$

Les ondelettes ont donc un spectre passe-bande, c'est à dire qu'elles sont à support borné. Ceci est une observation très importante, que nous utiliserons plus loin pour construire une transformée en ondelettes efficace.

La condition (3.8) n'est autre qu'une condition d'oscillation, et les convolutions d'un signal avec une telle fonction, ses translatées et ses dilatées, décriront les oscillations du signal, c'est-à-dire ses fluctuations.

Nous imposons également des conditions additionnelles sur les fonctions dans le but que la transformée en ondelettes décroit rapidement avec la décroissance de l'échelle s. Ces conditions sont appelées *conditions de régularité* et elles signifient que les fonctions doivent êtres suffisament lisses et à support borné en fréquence ou en temps.

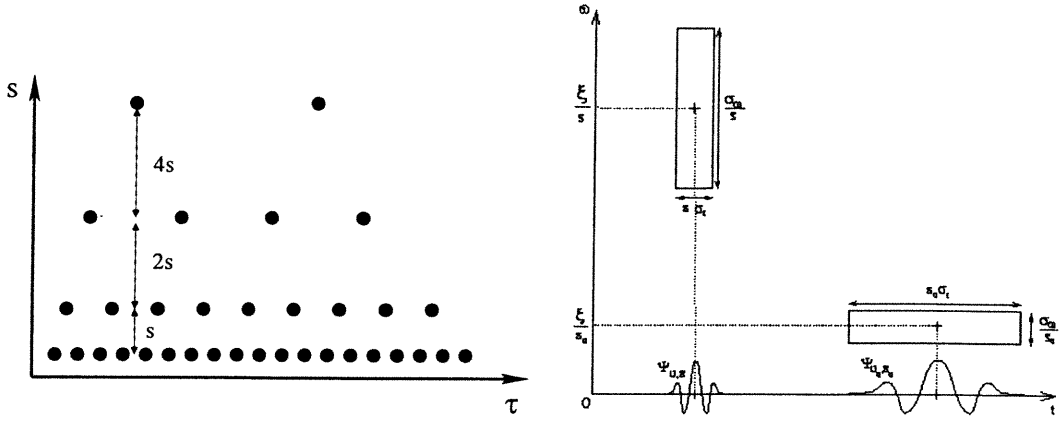
Tel que mentionné auparavant, la transformée en ondelettes discrète prend un signal en une dimension et donne un signal en deux dimensions dans la représentation temps-échelle. Ceci est fait en modifiant la représentation en ondelettes de l'équation (3.4) pour créer

$$\psi_{j,k}(t) = \frac{1}{\sqrt{s_0^j}} \psi \left(\frac{t - k\tau_0 s_0^j}{s_0^j} \right). \tag{3.9}$$

Dans l'équation (3.9), j et k sont des entiers et $s_0 > 1$ est un facteur de dilatation fixe. Dans cette définition, le facteur de translation dépend du facteur de dilatation. L'effet de la discrétisation des ondelettes est que le plan temps-échelle est maintenant discrétisé à des intervalles discrets. On choisit généralement $\tau_0 = 1$ et $s_0 = 2$ pour que l'échantillonnage de l'axe fréquence corresponde à un échantillonnage dyadique[1]. La Figure 3.1(a) illustre cette définition.

La Figure 3.1(b) quant à elle, illustre la localisation dans l'espace temps-fréquence de 2 ondelettes. Les rectangles d'incertitude donnent une idée de la résolution en temps et en fréquence. Comme le montre l'équation 3.4, on remarque qu'en réduisant l'échelle, on augmente la résolution temporelle mais qu'on diminue la résolution en fréquence. L'écart-type en temps est proportionnel à s. L'écart-type en fréquence est inversement proportionnel à s. Aux échelles plus fines,

[1]C'est un choix très naturel pour les ordinateurs, l'oreille humaine et la musique par exemple.



(a) *Localisation des ondelettes disrètes dans le plan temps-échelle sur une grille dyadique.*

(b) *Les "pavés" d'incertitude de Heisenberg pour deux localisations de l'ondelette.*

FIG. 3.1. *Localisation des ondelettes et principe d'incertitude.*

on peut "entasser" plus de boîtes de Heisenberg côte à côte car la résolution temporelle est meilleure.

3.1.1. Analyse multirésolution

D'après Daubechies **[15]**, nous savons que nous pouvons construire des ondelettes ψ telles que la famille constituée des translations et des dilatations de cette fonction,

$$\left\{\psi_{j,n}(t) = \frac{1}{\sqrt{2^j}}\psi\left(\frac{t-2^j n}{2^j}\right)\right\}_{(j,n)\in\mathbb{Z}^2}, \tag{3.10}$$

soit une base orthonormale de $\mathbf{L^2}(\mathbb{R})$. En vision par ordinateur, Burt et Adelson **[14]** ont introduit un algorithme multirésolution pyramidal qui peut être utilisé pour étudier une image à basse résolution et augmenter sélectivement la résolution lorsque nécessaire. Nous formalisons ici le principe de multirésolution.

Mallat et Meyer **[10]** en ont donné la définition suivante :

Définition 3.1 (MULTIRÉSOLUTION). *Une séquence $\{\mathbf{V}_j\}_{j\in\mathbb{Z}}$ de sous-espaces fermés de $\mathbf{L^2}(\mathbb{R})$ est une approximation en multirésolution si les 6 propriétés*

suivantes sont satisfaites :

$$\forall (j,k) \in \mathbb{Z}^2 \quad , \quad f(t) \in \mathbf{V}_j \Leftrightarrow f(t-2^j k) \in \mathbf{V}_j, \tag{3.11}$$

$$\forall j \in \mathbb{Z} \quad , \quad \mathbf{V}_{j+1} \subset \mathbf{V}_j, \tag{3.12}$$

$$\forall j \in \mathbb{Z} \quad , \quad f(t) \in \mathbf{V}_j \Leftrightarrow f\left(\frac{t}{2}\right) \in \mathbf{V}_{j+1}, \tag{3.13}$$

$$\lim_{j \to +\infty} \mathbf{V}_j = \bigcap_{j \to -\infty}^{+\infty} \mathbf{V}_j = \{0\}, \tag{3.14}$$

$$\lim_{j \to -\infty} \mathbf{V}_j = \textit{Fermeture}\left(\bigcup_{j \to -\infty}^{+\infty} \mathbf{V}_j \right) = \mathbf{L^2}(\mathbb{R}). \tag{3.15}$$

Il existe θ tel que $\{\theta(t-n)\}_{n \in \mathbb{Z}}$ est une base de Riesz de $\mathbf{V}_0$.

Donnons maintenant une explication intuitive de ces propriétés mathématiques. La propriété (3.11) signifie que $\mathbf{V}_j$ est invariant sous n'importe quelle translation proportionnelle à l'échelle 2^j. Comme nous avons vu à la Figure 3.1, cet espace peut être comparé à une grille uniforme avec intervalles 2^j, ce qui caractérise l'approximation du signal à la résolution 2^{-j}. L'inclusion (3.12) est une propriété causale qui prouve qu'une approximation à la résolution 2^{-j} contient toute l'information nécessaire pour calculer l'approximation à une résolution 2^{-j-1} plus grossière. Dilater les fonctions dans $\mathbf{V}_j$ par 2 élargit les détails par 2 et (3.13) garantit que cela définit une approximation à une résolution plus grossière 2^{-j-1}. Lorsque la résolution 2^{-j} tend vers 0, (3.14) implique que nous perdons tous les détails de f et

$$\lim_{j \to +\infty} ||P_{\mathbf{V}_j} f|| = 0. \tag{3.16}$$

Par contre, lorsque la résolution 2^{-j} tend vers $+\infty$, la propriété (3.15) exprime la convergence de l'approximation du signal vers le signal original :

$$\lim_{j \to -\infty} ||f - P_{\mathbf{V}_j} f|| = 0. \tag{3.17}$$

D'après Mallat, il est possible d'obtenir une base orthonormale de chaque espace V_j à partir des translations et des dilatations d'une seule fonction en utilisant la fonction θ de la définition 3.1. En effet, en orthonormalisant la base de Riesz de cette définition, on montre l'existence d'une fonction $\varphi(t)$, appelée fonction

d'échelle, telle que $\forall j \in \mathbb{Z}$

$$\left\{\varphi_{j,n}(t) = \frac{1}{\sqrt{2^j}}\varphi\left(\frac{t-2^j n}{2^j}\right)\right\}_{n\in\mathbb{Z}}, \tag{3.18}$$

forment une base orthonormale de V_j.

Puisque $\forall j \in \mathbb{Z}$, $V_j \subset V_{j-1}$, on note W_j le complément orthogonal de V_j dans V_{j-1} :

$$V_{j-1} = V_j \oplus W_j. \tag{3.19}$$

On peut montrer qu'il existe $\psi(t)$, une ondelette mère, telle que sa dilatation à l'échelle 2^j et ses translations

$$\left\{\psi_{j,n}(t) = \frac{1}{\sqrt{2^j}}\psi\left(\frac{t-2^j n}{2^j}\right)\right\}_{n\in\mathbb{Z}}, \tag{3.20}$$

forment une base orthonormale en ondelettes de W_j. Nous notons par V_j l'espace engendré par les $\varphi_{j,n}$; ces espaces V_j décrivent les espaces d'approximation successifs, $\cdots \quad V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \quad \cdots$, chacun avec une résolution 2^j. Pour chaque j, les $\varphi_{j,n}$ engendrent un espace W_j qui est exactement le complément orthogonal dans V_{j-1} de V_j ; les coefficients $\langle\psi_{m,n}, f\rangle$, quant à eux, décrivent l'information perdue lorsque nous allons d'une approximation de f avec résolution 2^{j-1} à l'échelle plus faible 2^j.

La Figure 3.2 permet de voir l'emboîtement des espaces d'approximation V_j, emboîtement qui permet de définir l'espace de fluctuations W_j, complément orthogonal de V_j dans V_{j-1} :

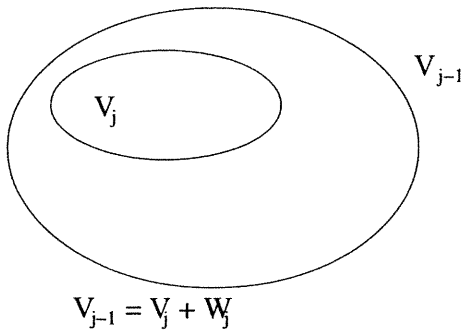


FIG. 3.2. *L'emboîtement des espaces d'approximation V_j permet de définir l'espace de fluctuations W_j, complément orthogonal de V_j dans V_{j-1}.*

Ainsi, l'ondelette engendre l'espace des détails qui complète l'espace V_0 pour décrire la fonction dans V_{-1}, i.e. à une résolution deux fois plus fine que celle dans V_0. Plus généralement, pour une valeur fixée de j, l'ensemble des fonctions

$$\varphi_{j,n}(t) = \frac{1}{\sqrt{2^j}}\varphi\left(\frac{t-2^j n}{2^j}\right) \tag{3.21}$$

où n est un entier quelconque, forme une base orthonormale de W_j. De plus, d'après (3.15), on a

$$\mathbf{L^2}(\mathbb{R}) = \lim_{j\to-\infty} V_j = \bigoplus_{j=-\infty}^{+\infty} W_j, \tag{3.22}$$

et par conséquent, l'ensemble

$$\left\{\psi_{j,n}(t) = \frac{1}{\sqrt{2^j}}\psi\left(\frac{t-2^j n}{2^j}\right)\right\}_{(j,n)\in\mathbb{Z}^2}, \tag{3.23}$$

constitue une base orthonormale de $\mathbf{L^2}(\mathbb{R})$.

3.1.2. Équation d'échelle

Étant donné que $V_1 \subset V_0$, la fonction $\varphi_{1,0}(t) = \frac{1}{\sqrt{2}}\varphi\left(\frac{t}{2}\right) \in V_1$ peut être décomposée sur la base orthonormale de V_0, $\{\varphi_{0,n}(t) = \varphi(t-n)\}_{n\in\mathbb{Z}}$. On obtient alors l'équation d'échelle pour φ :

$$\frac{1}{\sqrt{2}}\varphi\left(\frac{t}{2}\right) = \sum_{n=-\infty}^{+\infty} h[n]\varphi(t-n). \tag{3.24}$$

Il en va de même pour $\psi\left(\frac{t}{2}\right) \in W_1 \subset V_0$ et on obtient l'équation d'échelle pour ψ :

$$\frac{1}{\sqrt{2}}\psi\left(\frac{t}{2}\right) = \sum_{n=-\infty}^{+\infty} g[n]\varphi(t-n), \tag{3.25}$$

avec $h[n] = \left\langle \frac{1}{\sqrt{2}}\varphi\left(\frac{t}{2}\right), \varphi(t-n)\right\rangle$ et $g[n] = \left\langle \frac{1}{\sqrt{2}}\psi\left(\frac{t}{2}\right), \varphi(t-n)\right\rangle$, puisque les $\varphi(t-n)$ sont orthogonaux.

La fonction d'échelle φ et l'ondelette ψ ne sont pas indépendants. On a la relation suivante,

$$\hat{g}(w) = e^{-iw}\hat{h}^*(w+\pi) \Rightarrow g[n] = (-1)^{1-n}h[1-n]. \tag{3.26}$$

Bien que $h[n]$ soit défini à partir de la fonction $\varphi(t)$, on peut montrer que sous certaines conditions, la connaissance de $h[n]$ définit de façon unique $\varphi(t)$. En pratique, c'est le filtre $h[n]$ qu'on utilisera, la connaissance exacte de $\varphi(t)$ et de

$\psi(t)$ n'étant même pas nécessaire, et la plupart du temps pas même possible. En effet, comme nous le verrons à la section 3.1.4, la décomposition en ondelettes d'un signal discret ne fait appel qu'aux filtres $h[n]$ et $g[n]$.

3.1.3. Caractéristiques des ondelettes

Il est possible de construire une multitude d'ondelettes, dont les différentes carctéristiques sont plus ou moins adaptées à tel ou tel problème. Dans le choix d'une ondelette, on portera en général une attention particulière aux trois critères suivants : le nombre de moments nuls, la régularité et la taille du support. Une ondelette ψ possède p moments nuls si

$$\int_{-\infty}^{+\infty} t^k \psi(t) dt = 0, \forall k \in \{0, 1, \ldots, p\}. \tag{3.27}$$

On cherche à ce que le support soit compact pour ψ, afin de diminuer le nombre de calculs, et qu'il soit le plus local possible. On peut montrer que si une ondelette orthogonale possède p moments nuls, alors la taille de son support doit être au moins égale à $2p-1$. Les ondelettes de Daubechies sont optimales en ce sens : la taille de leur support vaut $2p-1$. Toutefois, les ondelettes de Daubechies réelles ne sont pas symétriques par rapport au centre de leur support. Par contre, il existe des ondelettes de Daubechies complexes qui le sont. De plus, en utilisant des ondelettes complexes, nous sommes en mesure d'obtenir le Laplacien de l'image aux diverses résolutions étudiées, comme discuté à la section 3.2. C'est en raison de ces propriétés que nous utilisons les ondelettes complexes dans ce mémoire. La Figure 3.3 montre la partie réelle et la partie complexe des fonctions $\varphi(t)$ et $\psi(t)$ associées à une ondelette complexe de Daubechies ayant 6 moments nuls. La partie imaginaire de l'équation d'échelle correspond bien au Laplacien qu'on obtient de la partie réelle de l'équation d'échelle. Cela nous sera très utile lors de l'utilisation conjointe de l'analyse multirésolution et des méthodes à progression rapide.

3.1.4. Décomposition d'un signal discret

En pratique, on est intéressé à effectuer la décomposition en ondelettes sur un signal discret provenant de l'acquisition à une certaine résolution d'une fonction

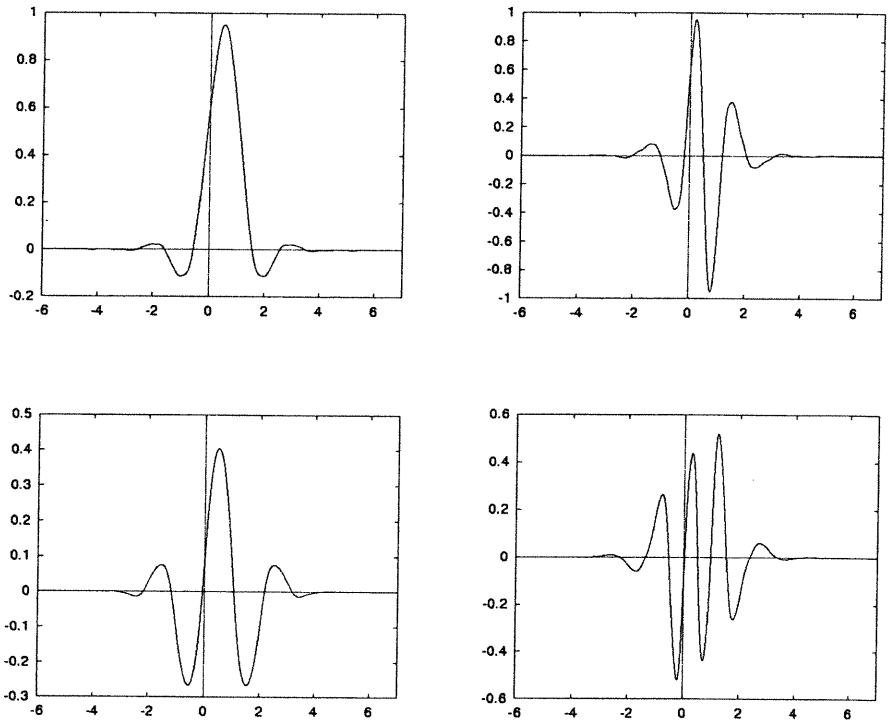


FIG. 3.3. *Une des ondelettes complexe de Daubechies pour $J = 6$ moments nuls. Sont représentées, les parties réelles (haut) et imaginaires (bas) de la fonction d'échelle φ (gauche) et de l'ondelette ψ (droite),*

$f(t)$. Ceci correspond à la projection de la fonction $f(t)$ dans un espace d'approximation que l'on notera V_0. Soit $f_0(t)$ cette projection et soit $\varphi(t-n)_{n\in\mathbb{Z}}$ une base orthogonale de V_0. On a alors

$$f_0(t) = \sum_{n=-\infty}^{+\infty} a_0[n]\varphi(t-n) \in V_0, \tag{3.28}$$

où $a_0[n] = \langle f(t), \varphi(t-n)\rangle$ ne correspond pas nécessairement à un échantillonage de la fonction f (i.e $a_0[n] \neq f(n)$, pour $n \in \mathbb{Z}$). La décomposition multi-échelle du signal consiste à l'exprimer sous la forme

$$f_0(t) = \sum_{n=-\infty}^{+\infty} a_J[n]\varphi_{J,n}(t) + \sum_{j=1}^{J}\sum_{n=-\infty}^{+\infty} d_j[n]\psi_{j,n}(t), \tag{3.29}$$

et au calcul des coefficients

$$d_j[n] = \langle f, \psi_{j,n} \rangle , \quad a_j[n] = \langle f, \varphi_{j,n} \rangle . \tag{3.30}$$

Un algorithme de calcul rapide **[10]** permet d'obtenir rapidement ces coefficients à l'aide des équations récursives suivantes pour la décomposition

$$a_{j+1}[p] = \sum_{n=-\infty}^{+\infty} h[n-2p]a_j[n], \tag{3.31}$$

$$d_{j+1}[p] = \sum_{n=-\infty}^{+\infty} g[n-2p]a_j[n], \tag{3.32}$$

et pour la reconstruction

$$a_j[p] = \sum_{n=-\infty}^{+\infty} h[p-2n]a_{j+1}[n] \tag{3.33}$$

$$+ \sum_{n=-\infty}^{+\infty} g[p-2n]d_{j+1}[n]. \tag{3.34}$$

Il est à noter que seul la connaissance de $h[n]$ et de $g[n]$ et non celle de $\varphi(t)$ et $\psi(t)$ est nécessaire pour effectuer ces calculs.

La Figure 3.4 nous montre la manière de faire de l'algorithme multirésolution. En utilisant la fonction d'échelle, l'algorithme décime le signal étudié et donne

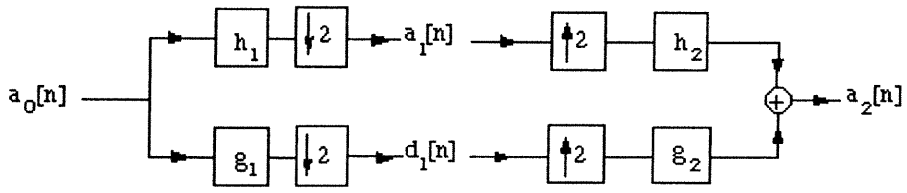
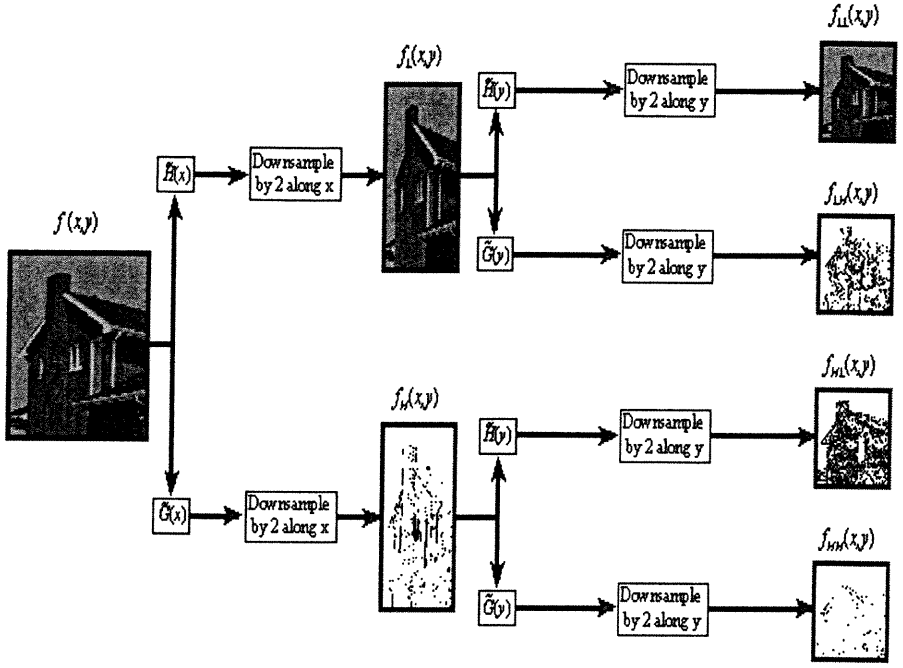


FIG. 3.4. *Algorithme multirésolution.*

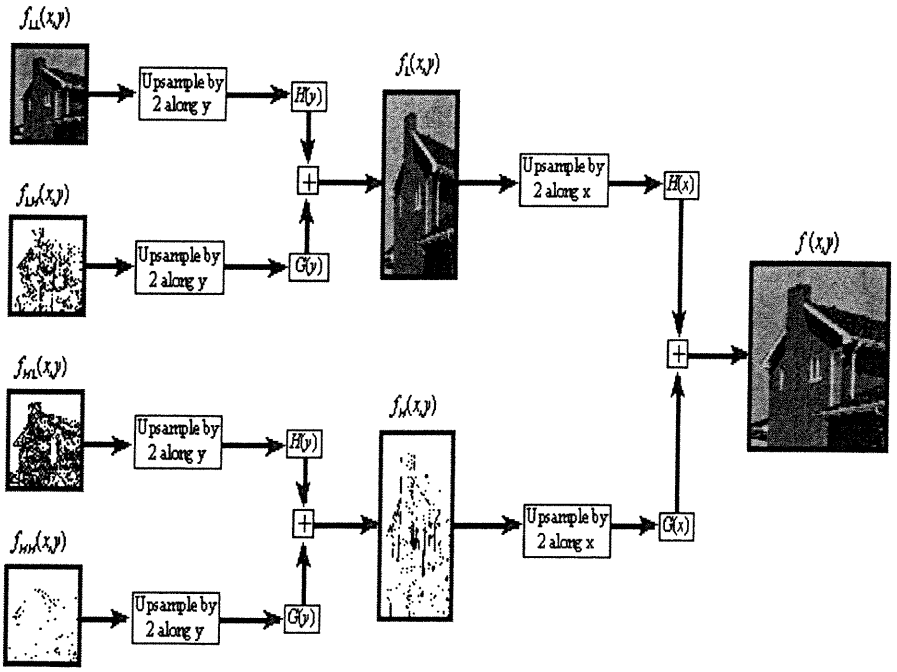
un nouveau signal à une résolution plus faible. Le complément d'information nécessaire à la reconstruction parfaite du signal est codé par les coefficients en ondelettes. La Figure 3.5 montre un exemple plus concret de l'algorithme.

3.1.5. Ondelettes en 2 dimensions

La construction d'une base d'ondelettes pour l'espace $\mathbf{L^2}(\mathbb{R})$ peut être généralisée à un espace $\mathbf{L^2}(\mathbb{R}^d)$. Nous allons illustrer le principe pour le cas en deux



(a) *Analyse multirésolution.*



(b) *Synthèse multirésolution.*

FIG. 3.5. *Analyse et synthèse avec l'algorithme multirésolution.*

dimensions. Soit une suite $\{V_j\}$ qui définit une multirésolution de $\mathbf{L^2}(\mathbb{R})$. On a alors que les espaces V_j^2, formés par le produit tensoriel des espaces V_j

$$V_j^2 = V_j \otimes V_j$$

forment une suite $\{V_j^2\}$ d'approximations de $\mathbf{L^2}(\mathbb{R}^2)$. Une base séparable de V_j^2 est obtenue à partir de la fonction d'échelle en une dimension φ. Cette base est donnée par

$$\left\{\varphi_{j,n,m}(x,y) = \frac{1}{2^j}\varphi\left(\frac{x-2^jn}{2^j}\right)\varphi\left(\frac{y-2^jm}{2^j}\right)\right\}_{(n,m)\in\mathbb{Z}^2}$$

où les fonctions $\varphi_{j,n,m}(x,y)$ sont obtenues par la dilatation et les translations de la fonction d'échelle en deux dimensions $\varphi(x,y) = \varphi(x)\varphi(y)$.

De façon analogue au cas unidimensionnel, nous cherchons à caractériser l'espace contenant les détails permettant le passage d'un niveau d'approximation au niveau le plus fin suivant. Soit W_j^2 le complémentaire orthogonal de V_j^2 dans V_{j-1}^2

$$V_{j-1}^2 = V_j^2 \oplus W_j^2$$

on obtient en distribuant $\oplus$ par rapport à $\otimes$ que

$$\begin{aligned} V_{j-1}^2 = V_j^2 \oplus W_j^2 &= (V_j \oplus W_j) \otimes (V_j \oplus W_j) \\ &= (V_j \otimes V_j) \oplus (V_j \otimes W_j) \oplus (W_j \otimes V_j) \oplus (W_j \otimes W_j) \\ &\Rightarrow \\ W_j^2 &= (V_j \otimes W_j) \oplus (W_j \otimes V_j) \oplus (W_j \otimes W_j). \end{aligned}$$

Soient les trois ondelettes

$$\psi^1(x,y) = \varphi(x)\psi(y), \psi^2(x,y) = \psi(x)\varphi(y), \psi^3(x,y) = \psi(x)\psi(y)$$

où φ et ψ sont la fonction d'échelle et l'ondelette définies précédemment en une dimension. On obtient alors la base orthonormale suivante pour W_j^2

$$\left\{\psi^1_{j,n,m}(x,y), \psi^2_{j,n,m}(x,y), \psi^3_{j,n,m}(x,y)\right\}_{(n,m)\in\mathbb{Z}^2}$$

où les

$$\psi^k_{j,n,m}(x,y) = \frac{1}{2^j}\psi^k\left(\frac{x-2^jn}{2^j}, \frac{y-2^jm}{2^j}\right), k = 1,2,3$$

correspondent à la dilatation et aux translations des trois ondelettes mères. De plus, comme $\bigoplus_{j=-\infty}^{+\infty} W_j^2 = \mathbf{L^2}(\mathbb{R}^2)$, on a alors que

$$\left\{\psi^1_{j,n,m}(x,y), \psi^2_{j,n,m}(x,y), \psi^3_{j,n,m}(x,y)\right\}_{(n,m)\in\mathbb{Z}^3}$$

constitue une base orthonormale en ondelettes de $\mathbf{L^2}(\mathbb{R}^2)$.

3.2. Ondelettes complexes

Revenons rapidement sur la notion de transformée en ondelettes. Nous considérons des bases orthonormales de $\mathbf{L}^2(\mathbb{R})$ de la forme

$$\psi_{j,n}(x) = 2^{\frac{-j}{2}}\psi(2^{-j}x - n) \tag{3.35}$$

où j et k sont des entiers. Toutes les ondelettes de Daubechies symétriques ont les propriétés usuelles des bases de Daubechies *réelles* habituelles. En écrivant $\varphi(x) = h(x) + ig(x)$, nous étudions la relation entre les fonctions réelles h et g.

Comme montré dans **[13]**, lorsque $J = 2, 4, 6$ ou 8, la fonction d'échelle complexe est bien approchée par l'expression

$$\varphi(x) \approx (1 + i\alpha\partial_x^2)h(x). \tag{3.36}$$

Cette approximation est vérifiée précisément dans le domaine fréquence défini par le taux d'échantillonnage du signal analysé. Pour $J > 10$, les termes dérivatifs d'ordre supérieur deviennent non-négligeables. C'est l'une des raisons pour lesquelles il est intéressant d'utiliser des ondelettes de Daubechies complexes avec un moment nul inférieur à 10, puisqu'elles permettent d'obtenir le Laplacien de notre image en effectuant l'analyse multirésolution. Lorsque nous utilisons une fonction d'échelle à deux dimensions :

$$\begin{aligned}
\varphi(x,y) &= \varphi(x)\varphi(y) \\
&\approx (1 + i\alpha\partial_x^2)h(x)(1 + i\alpha\partial_y^2)h(y) \\
&= [h(x)h(y) - \alpha^2 h"(x)h"(y)] + i\alpha[h"(x) + h"(y)] \\
&\approx h(x)h(y) + i\alpha\Delta h(x)h(y).
\end{aligned}$$

Nous obtenons ainsi un noyau lissant $G(x,y) = h(x)h(y)$ dans la partie réelle et le Laplacien dans la partie imaginaire. La convolution de ce noyau complexe

avec une image réelle, conduit donc à obtenir le Laplacien de cette image. Nous utiliserons ce Laplacien pour construire la fonction vitesse utilisée avec la méthode à progression rapide, comme discuté à la section 2.1.2.

3.3. Paramètres à choisir pour l'utilisation de la méthode hybride

Lorsque nous voulons implémenter notre méthode combinée, certains paramètres peuvent être choisis arbitrairement. Par exemple le nombre d'échelles à utiliser pour décimer l'image étudiée. En effet, plus le nombre d'échelles choisi sera grand, plus l'image va avoir une résolution faible et moins il restera de détails dans cette image. Nous aurons par contre perdu beaucoup d'information et il faudra faire un compromis pour ne pas avoir finalement une image trop difficile à reconnaître, celle-ci étant à une résolution trop grossière.

Une des choses qu'il est possible d'améliorer est certainement la fonction vitesse utilisée dans la propagation du front. Il est possible d'améliorer cette fonction vitesse en utilisant les coefficients en ondelettes calculés durant l'analyse multirésolution. Par exemple, comme vu à la section 3.2, dans la partie imaginaire de notre analyse multirésolution, nous avons une partie proportionnelle au Laplacien de l'image étudiée à une échelle donnée. Nous devrions donc utiliser ces coefficients déjà calculés plutôt que de faire du travail supplémentaire et calculer le gradient de notre image à l'aide des différences finies. De la même manière, les coefficients en ondelettes ont une valeur maximale dans les régions où l'intensité change de manière brusque, c'est-à-dire les contours. Nous voulons en fait prendre le gradient (ou le Laplacien) de notre image comme étant la plus forte contribution à la fonction vitesse, mais aussi utiliser les coefficients en ondelettes pour venir améliorer celle-ci.

3.3.1. Laplacien à chaque échelle

Comme nous en avons discuté à la section 2.1.2, la fonction vitesse à utiliser est de la forme $F(x,y) = e^{-\alpha|\Delta G_\sigma * I(x,y)|}, \quad \alpha > 0$. Dans la plupart des cas, on utilise le gradient pour reconnaître les endroits où se trouvent les contours, par contre

cette décision ne repose que sur la facilité du calcul du gradient. Il est également possible d'utiliser le Laplacien pour retrouver les endroits de fort contraste. En effet, nous voulons utiliser une fonction qui aura de grandes valeurs lorsque le changement d'intensité est brusque et qui aura de petites valeurs lorsque qu'il n'y a pratiquement pas de changement d'intensité. Nous savons déjà que lorsque nous calculons la transformée en ondelettes complexe, nous disposons du Laplacien de l'image à chaque résolution. Il serait donc plus rentable du côté des temps de calcul de ne pas calculer le gradient mais d'utiliser le Laplacien qui nous est donné.

Le lecteur se dira peut-être que nous avons accès, avec la transformée en ondelettes complexe, seulement qu'au Laplacien aux résolutions plus grossières que la résolution de l'image de départ. Lorsque nous prenons une image et que nous voulons retrouver un contour dans celle-ci, nous avons cette image à une certaine résolution qui nous est inconnue. Nous ne connaissons donc pas les coefficients en ondelettes de cette image à cette résolution. Lorsque nous diminuons notre résolution (nous faisons l'analyse), nous avons le Laplacien en prime mais lorsqu'il s'agit de l'image de départ à une résolution qui nous est maximale, nous devons trouver le Laplacien d'une manière différente.

La première idée est bien sûr de calculer ce Laplacien par les différences finies, comme on le faisait avec le gradient avant d'utiliser la transformée en ondelettes. Mais, en utilisant les différences finies, nous avons une représentation du Laplacien qui diffère selon l'échelle. Nous serions intéressés à obtenir une représentation qui soit trouvée de la même manière à n'importe quelle échelle. Il faut donc trouver le moyen de retrouver le Laplacien de l'image initiale à l'aide de la transformée en ondelettes.

Pour mieux cerner ce que nous faisons, étudions la Figure 3.6. Lorsque nous utilisons la transformée en ondelettes, nous faisons sur l'image ce que nous nommons du sous-échantillonnage, car nous diminuons le nombre de pixels de l'image. L'idée est de maintenant faire du sur-échantillonnage. Nous disons que l'image originale est à une certaine résolution inconnue, il est donc possible d'aller vers les échelles supérieures comme lorsque nous faisons la synthèse de notre signal.

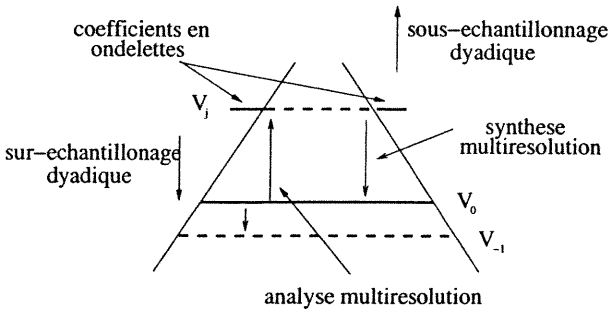


FIG. 3.6. *Schéma de l'analyse et de la synthèse multirésolution. Le niveau V_{-1} est le niveau de sur-échantillonnage dyadique nécessaire à la construction du Laplacien de l'image au niveau V_0.*

Comme expliqué à la section 1.3, l'analyse nous permet de remonter dans le schéma vers des échelles plus grossières. Par contre, il nous est possible de descendre vers des échelles plus fines en utilisant l'algorithme pyramidal. Nous avons notre image à la résolution initiale V_0, mais nous ne savons rien sur les coefficients en ondelettes de l'image à cette résolution, nous devons donc les poser tous à zéro, comme à la Figure 3.7.

Nous utilisons l'algorithme pour aller une résolution plus bas sur notre schéma, la résolution du niveau V_{-1}. Nous obtenons maintenant une image complexe dont la partie réelle correspond à un lissage de l'image à la résolution initiale et la partie complexe ne comprend que du bruit. Lorsque nous avons fait la transformée en ondelettes inverse de la Figure 3.7, nous obtenons une image à une résolution supérieure. Nous mettons toute la partie complexe à zéro et nous disons que l'image à cette résolution est notre nouvelle image à résolution initiale. Nous appliquons donc l'algorithme multirésolution et montons d'une échelle vers le haut. La partie réelle de cette nouvelle image correspond bel et bien à l'image originale mais la partie complexe, montrée à la Figure 3.8, correspond quant à elle au Laplacien de celle-ci. Nous avonc donc maintenant une méthode, utilisant l'analyse multirésolution, pour trouver le Laplacien à n'importe quelle échelle, même à l'échelle de départ.

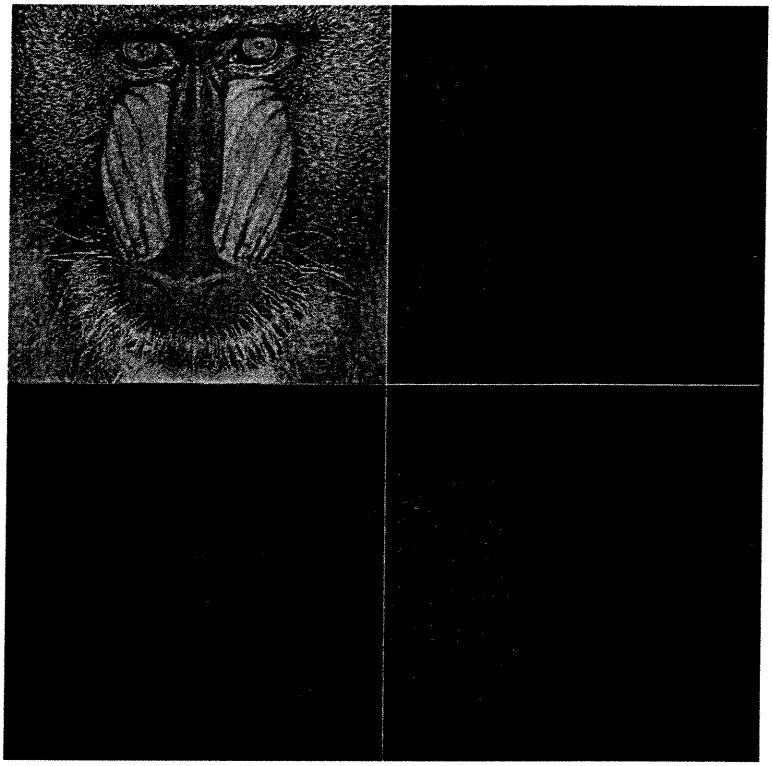


FIG. 3.7. *Partie réelle de l'image que nous allons utiliser pour faire du "sursampling" et retrouver le Laplacien. Les matrices de pixel noirs sont les coefficients en ondelettes nuls à cette résolution. En effet, à l'échelle d'acquisition, il n'y a pas d'information sur les détails.*

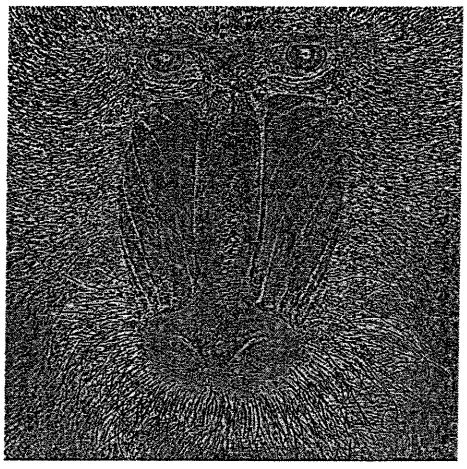


FIG. 3.8. *Laplacien retrouvé grâce au "sursampling".*

3.3.2. Comment le Laplacien change à chaque échelle

Nous allons maintenant nous pencher un peu plus sur l'équation régissant la vitesse avec laquelle se propage notre front dans notre image. Nous savons, $F(x,y) = e^{-\alpha|\Delta G_\sigma * I(x,y)|}, \quad \alpha > 0$, et nous allons discuter du α qui apparaît dans l'équation ci-haut. Lorsque nous utilisons les méthodes à progression rapide à une seule échelle comme c'est souvent le cas, ce α est choisi par l'utilisateur selon le type d'image étudiée. Il est choisit par essai et erreur jusqu'à trouver un α qui convienne. Ce sera la même chose pour nous dans le cas que nous étudions, mais il y a un autre aspect qu'il ne faudra pas perdre de vue. Lorsque nous changeons d'échelle, c'est-à-dire lorsque nous utilisons l'analyse multirésolution, est-ce que ce α change lui aussi ? La réponse est oui, la valeur moyenne de notre Laplacien change avec l'échelle. Nous voulons donc que cette valeur moyenne reste sensiblement la même à chaque échelle, en changeant α de la bonne façon.

Lorsque le temps est venu de calculer nos coefficients en ondelettes, et plus précisément la partie où se retrouve le Laplacien, nous devons faire le calcul suivant : $d_{j,n} = \langle \psi_{j,n} | f \rangle$. Nous allons expliciter la partie imaginaire de ce calcul pour savoir par quel facteur le Laplacien va changer à chaque échelle.

$$\begin{aligned}
Im(d_{j,n}) &= Im(\langle \psi_{j,n} | f \rangle) \\
&= 2^{\frac{j}{2}} * \alpha \int_{-\infty}^{\infty} h"(2^j x - n) f(x) dx \\
&= 2^{\frac{j}{2}} * \alpha \left[\frac{f(x) h'(2^j x - n)}{2^j} \Big|_{-\infty}^{\infty} - \frac{1}{2^j} \int_{-\infty}^{\infty} h'(2^j x - n) f'(x) dx \right] \\
&= \frac{-2^{\frac{j}{2}}}{2^j} * \alpha \int_{-\infty}^{\infty} h'(2^j x - n) f'(x) dx \\
&= \frac{-\alpha}{2^{\frac{j}{2}}} \left[\frac{f'(x) h(2^j x - n)}{2^j} \Big|_{-\infty}^{\infty} - \frac{1}{2^j} \int_{-\infty}^{\infty} h(2^j x - n) f"(x) dx \right] \\
&= \frac{\alpha}{2^{\frac{3j}{2}}} \int_{-\infty}^{\infty} h(2^j x - n) f"(x) dx \\
&\approx \frac{\alpha}{2^{\frac{3j}{2}}} f" \left(\frac{2n+1}{2^{j+1}} \right).
\end{aligned} \tag{3.37}$$

De plus, en se référant à [**12**], nous avons accès à plus de renseignements sur les ondelettes complexes. Nous utilisons des ondelettes complexes avec $J = 2, 4, 6$ ou

8, car comme nous l'expliquions précédemment, nous voulons pouvoir utiliser le Laplacien plutôt que le gradient dans le calcul de la fonction vitesse.

Le problème que nous venons de faire est en une seule dimension, mais nous étudions des images en deux dimensions, il convient donc d'utiliser ce résultat pour connaître celui en deux dimensions. Lorsque nous faisons la transformée en ondelettes en deux dimensions, comme vu à la section 3.1.5, nous prenons la formule en une dimension et nous l'appliquons deux fois, une fois sur l'axe des x et l'autre fois sur l'axe des y. En fait, nous convoluons deux fonctions de même type, une en x et l'autre en y. Lorsque nous convoluerons, le facteur trouvé ci-haut va être mis au carré, nous aurons donc un facteur $\left(\frac{1}{2^{3j/2}}\right)^2 = \frac{1}{2^{3j}}$. Nous aurons un facteur 8 qui apparaîtra à chaque changement d'échelle. Nous savons maintenant comment changer la valeur de notre constante α lorsque nous changeons d'échelle. Nous aurons ainsi,

$$\alpha_j = \frac{1}{2^{3j}}\alpha_0. \tag{3.38}$$

3.4. Conclusion

Ce chapitre nous a donc permis d'expliciter l'algorithme multirésolution utilisé. Nous avons également montré comment l'utilisation des ondelettes complexe peut faire en sorte d'obtenir le Laplacien de l'image à la résolution étudiée à l'aide des coefficients en ondelettes retrouvés. Ce Laplacien sera ensuite utilisé dans la construction de la fonction vitesse utilisée à la section 2.1.2.

Chapitre 4

PROGRESSION RAPIDE ET MULTIRÉSOLUTION

Ce chapitre présente les détails de l'algorithme élaboré dans ce mémoire. De plus, nous en évaluerons l'efficacité sur des images artificielles et des images réelles. Nous combinons deux méthodes, pour être en mesure de trouver des contours moins bruités, et qui se rapprochent davantage du meilleur contour possible trouvé par d'autres méthodes. De plus, cette méthode est en mesure, en diminuant le niveau de résolution de l'image, de faire "disparaître" des détails que nous ne voulons pas découvrir. Nous pouvons ainsi faire évoluer un contour dans une image lissée comportant moins de bruit que l'image à la résolution originale.

4.1. Résumé de la méthode

Pour bien comprendre comment l'analyse multirésolution peut permettre de construire une fonction vitesse plus précise, résumons rapidement la construction de l'algorithme utilisé, en se basant sur ce dont nous avons discuté à la section 3.1 et sur la Figure 4.1. Voici la procédure pour l'analyse multirésolution :

- L'image initiale est projetée dans un espace d'approximation dont la résolution correspond à l'échelle la plus fine, niveau V_0.
- L'analyse multirésolution consiste ici à faire la projection de cette image dans un espace d'approximation deux fois plus grossier (coin supérieur gauche de l'image obtenu après transformation, comme aux Figures 4.2(a) et 4.2(b)).

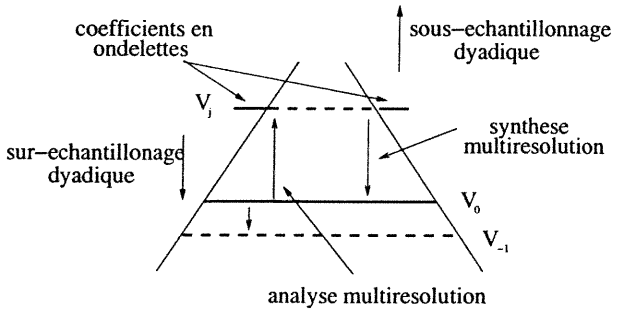
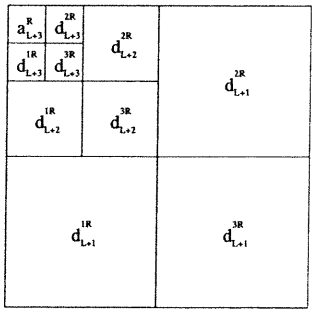


FIG. 4.1. *Schéma de l'analyse et de la synthèse multirésolution. Le niveau V_{-1} est le niveau de sur-échantillonnage dyadique nécessaire à la construction du Laplacien de l'image au niveau V_0.*

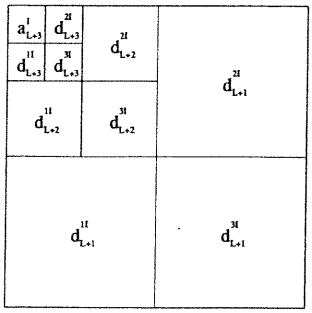
- Le complément d'information utile pour retrouver l'image initiale se trouve projeté dans les trois autres "secteurs" : ce sont les coefficients en ondelettes, comme aux Figures 4.2(a) et 4.2(b).
- Cette opération peut être répétée sur la projection dans l'espace d'approximation de résolution inférieure : on obtient ainsi une représentation pyramidale de l'image initiale et on se trouve au niveau V_j.

Une fois cette analyse terminée, nous utilisons la méthode à progression rapide décrite à la section 2.5 pour retrouver le contour à la résolution V_j. Ensuite, nous faisons synthètisons le signal tout en prenant soin de "remonter" le contour à la résolution supérieure, comme expliqué à la section 2.7. Cette démarche est réptée jusqu'à arriver à l'image de départ, à la résolution V_0, avec le contour recherché.

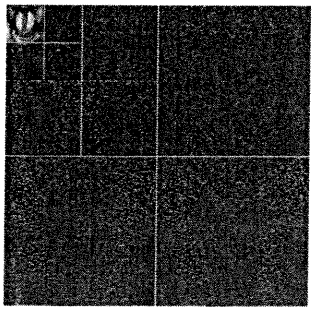
Étudions un peu plus en détail l'algorithme multirésolution. Comme on peut le voir à la Figure 4.2, l'image originale est transformée dans une échelle plus grossière et elle est placée en haut à gauche dans l'image (position a^R_{L+3}). Les 3 autres parties (positions d^{1R-I}_{L+3}, d^{2R-I}_{L+3} et d^{3R-I}_{L+3}) nous servent à reconstruire l'image à l'échelle précédente, elles contiennent en fait les coefficients en ondelettes. La partie en haut à droite (position d^{2R}_{L+3}) contient l'orientation horizontale de l'image, la partie en bas à gauche (position d^{1R-I}_{L+3}) contient l'orientation verticale et enfin la dernière partie (position d^{3R-I}_{L+3}) contient l'orientation diagonale de l'image. Toutes les autres parties imaginaires sont semblables aux parties réelles. Par contre, la partie d^{2I}_{L+3} contient le Laplacien de l'image à la position a^R_{L+3}. Le terme a^I_{L+2} est proportionnel au Laplacien du terme a^R_{L+2}. En fait, le terme



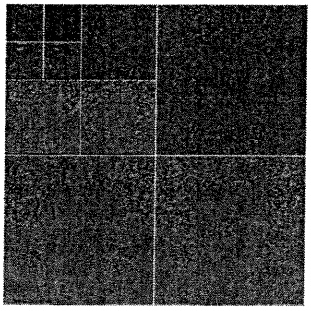
(a) *Partie réelle de l'algorithme multirésolution utilisé.*



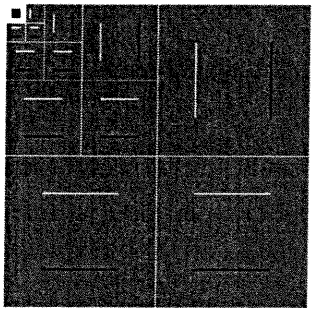
(b) *Partie imaginaire de l'algorithme multirésolution utilisé.*



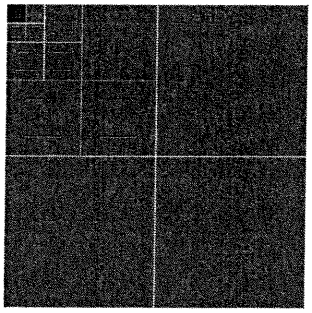
(c) *Multirésolution de l'image singe, partie réelle.*



(d) *Multirésolution de l'image singe, partie imaginaire.*



(e) *Multirésolution de l'image d'un carré, partie réelle.*



(f) *Multirésolution de l'image d'un carré, partie imaginaire.*

FIG. 4.2. *Explication de l'algorithme multirésolution utilisé. Les images du bas sont le résultat de l'algorithme multirésolution sur l'image singe et sur l'image d'un carré.*

a^I_{L+i} est proportionnel au Laplacien du terme a^R_{L+j} pour $j = 1, 2, \ldots, j_{\max}$. L'idée ensuite est de prendre nos parties réelles et imaginaires contenant les coefficients en ondelettes et d'utiliser ceux-ci pour améliorer la fonction vitesse.

Nous travaillons avec des transformées en ondelettes complexes, comme expliqué à la section 3.2, et lorsque nous analysons par la multirésolution une image réelle, nous obtenons une image complexe.

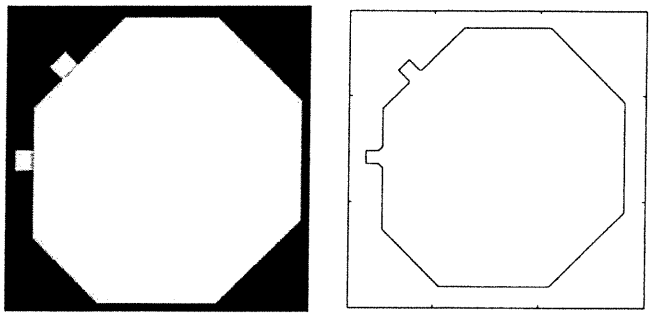
Nous utilisons donc l'algorithme multirésolution pour obtenir une image à plus faible résolution (à une échelle décidée par l'utilisateur, mais pas trop grossière car sinon nous ne distinguerons plus aucun détail !), le niveau V_j à la Figure 4.1, et nous utilisons les méthodes à progression rapide pour trouver le contour dans l'image à ce niveau. Nous avons donc une image lissée où une partie des détails est absente, nous permettant de retrouver des contours qui seraient autrement plus difficiles à retrouver dans l'image originale. En effet, dans l'image originale, toute l'information est encore présente, ce qui inclut également les détails qui peuvent nuire à la propagation du front et dans un même temps, empêcher de retrouver le contour voulu.

Une fois le contour trouvé, nous remontons vers une échelle plus fine en prenant soin de remonter aussi le contour trouvé, comme expliqué à la section 2.7. Nous répétons ceci jusqu'à l'image originale, le niveau V_0, et nous obtenons ainsi le contour voulu. En utilisant ces deux méthodes, nous nous sommes affranchis de certains détails plus fins qui ont disparus lors de l'algorithme multirésolution, ce qui a permis aux méthodes à progression rapide de retrouver un contour épuré de ces détails inutiles. L'aspect intéressant est que ces détails se trouvent toujours dans les coefficients en ondelettes, nous n'avons donc perdu aucune résolution sur l'image de départ ! La Figure 1.7 montre ce qui se passe lors de la multirésolution, lorsque nous allons des échelles fines jusqu'aux échelles plus grossières.

4.2. Validation de la méthode sur des images synthétiques

Nous allons maintenant mesurer l'efficacité de la méthode que nous venons de décrire. Pour ce faire, nous devons choisir des valeurs quantitatives permettant de calculer cette efficacité. Avant d'évaluer l'efficacité de la méthode sur des

images réelles, nous construirons une image synthétique qui nous servira à quantifier l'efficacité de notre méthode. La Figure 4.3 est l'image synthétique que nous utiliserons. Cette image artificielle représente bien le genre de problèmes que nous



(a) *Problème typique.*

(b) *Problème typique et contour à retrouver.*

FIG. 4.3. *Problème typique et le contour que notre méthode doit retrouver.*

voulons résoudre, car elle contient des protubérances que nous voulons retrouver et qui se trouvent derrière des barrières qui peuvent changer d'épaisseur et d'intensité. Pour généraliser le problème, nous avons placé deux protubérances, une à un angle de 90^o et l'autre à un angle de 45^o. Ceci nous permet d'avoir une meilleure idée de l'efficacité de la méthode, car on pouvait penser que celle-ci privilégie les directions horizontales et verticales de par sa construction. En posant une protubérance à un angle de 45^o, nous étudions le cas le plus difficile possible qu'on puisse espérer résoudre avec notre méthode. En effet, les angles les plus difficiles ne sont pas les multiples de 90^o, car ces angles sont alignés avec les directions priviliégiées, mais bien les angles de la forme $(2n+1) * 45^o$ avec $n = 0, 1, 2, \ldots$.

Pour quantifier l'efficacité de l'algorithme, nous nous servirons du rapport du nombre de points trouvés à l'intérieur du contour sur le nombre de points qui sont réellement à l'intérieur.

La Figure 4.3 nous montre le contour que l'on devrait retrouver. Avec la méthode proposée ici, nous estimerons un contour qui se rapproche le plus possible de celui-là. Plusieurs facteurs peuvent être changés dans cette image synthétique, par exemple l'épaisseur des barrières et l'intensité de celles-ci. En ce qui à trait à

la méthode elle-même, un facteur qu'on peut changer est le nombre de résolutions utilisées dans la représentation multi-échelle. On peut aussi changer le facteur α présent dans la fonction vitesse, mais celui-ci ne modifie pas le résultat, il change la vitesse de propagation du contour dans l'image.

Pour quantifier l'efficacité, nous nous concentrerons principalement sur les deux protubérances et calculerons l'efficacité seulement dans un voisinage de celles-ci. En effet, il est inutile de calculer l'efficacité sur tout le contour car la méthode à progression rapide retrouve facilement les contours intérieurs. En dépit de la présence d'obstacles, nous sommes intéressés à savoir si la méthode parvient à les dépasser. Pour quantifier cette efficacité, nous poserons un pourcentage d'efficacité seuil (dans notre cas 85%), pour un pourcentage supérieur, nous dirons que nous avons retrouvé les protubérances, pour un pourcentage plus bas, nous dirons que la méthode a échouée.

4.2.1. Images synthétiques sans bruit

La Figure 4.4 montre les résultats de la méthode, en l'absence de bruit. L'axe des abscisses du graphique nous donne le nombre de multirésolutions lors de l'utilisation de la méthode hybride. Un nombre nul de multirésolutions, signifie que nous prenons la méthode à progression rapide seule. L'axe des ordonnées est le contraste normalisé maximal que la barrière peut avoir pour que la méthode permette de retrouver un pourcentage d'efficacité de 85%. Nous remarquons immédiatement que l'utilisation de la multirésolution permet d'améliorer les résultats de la méthode à progression rapide seule, niveau 0. En effet, lorsque nous gardons l'image à sa résolution initiale (i.e que nous n'allons à aucune résolution inférieure), alors le contraste seuil est extrêmement petit. Par contre, si nous prenons notre image et l'étudions à plusieurs résolutions, nous remarquons que le contraste seuil est beaucoup plus grand, ce qui nous permet de traverser des barrières ayant un contraste beaucoup plus élevé que si nous n'utilisons pas l'algorithme multirésolution. On peut remarquer également que nous avons une certaine limite sur le nombre de résolutions que nous pouvons prendre. Si nous prenons une résolution trop grossière de l'image, nous perdons tellement de détails que l'image ne

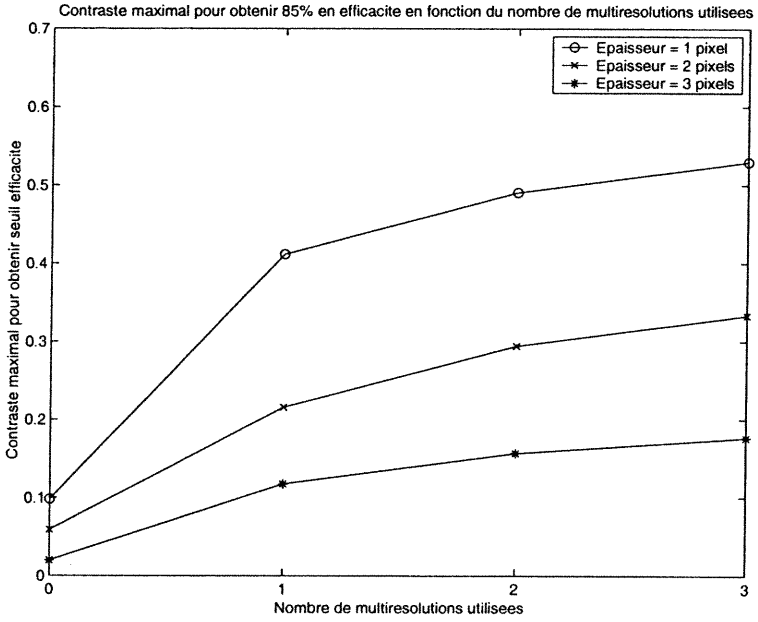


FIG. 4.4. *Résultats de la méthode combinée. Les abscisses nous donnent le nombre de résolutions utilisées et les ordonnées nous donnent le contraste seuil à partir duquel nous obtenons un pourcentage d'efficacité supérieur à 85%.*

signifie plus rien. En effet, on remarque que la courbe, qui augmente très rapidement, ralentit sa montée assez vite, on peut donc croire qu'à partir d'une certaine résolution, on ne puisse plus retrouver des barrières de plus grand contraste. À la limite, on peut penser que si nous décimons l'image et allons à une résolution trop grossière, nous allons perdre de l'efficacité car nous aurons une image à une si faible résolution que nous allons dépasser tous les détails et nous perdons le contour d'intérêt. Dépendant de la taille des artefacts que nous désirons étudier, nous ne pouvons pas aller dans une résolution trop grossière, cette résolution est directement liée à la taille des objets étudiés.

Plus l'épaisseur de la barrière augmente, plus il est difficile d'avoir un pourcentage d'efficacité de plus de 85%, ce qui est intuitivement juste. Si la barrière n'existe pas, la méthode continue à évoluer mais si l'épaisseur de la barrière augmente, il sera de plus en plus difficile de la traverser.

4.2.2. Images synthétiques bruitées

Nous voulons maintenant quantifier l'efficacité de la méthode tout en observant sa robustesse en présence de bruit. Nous allons donc étudier notre image synthétique avec du bruit gaussien non-corrélé dans le cas ou l'intensité du bruit peut être plus élevée que l'intensité présente dans l'image.

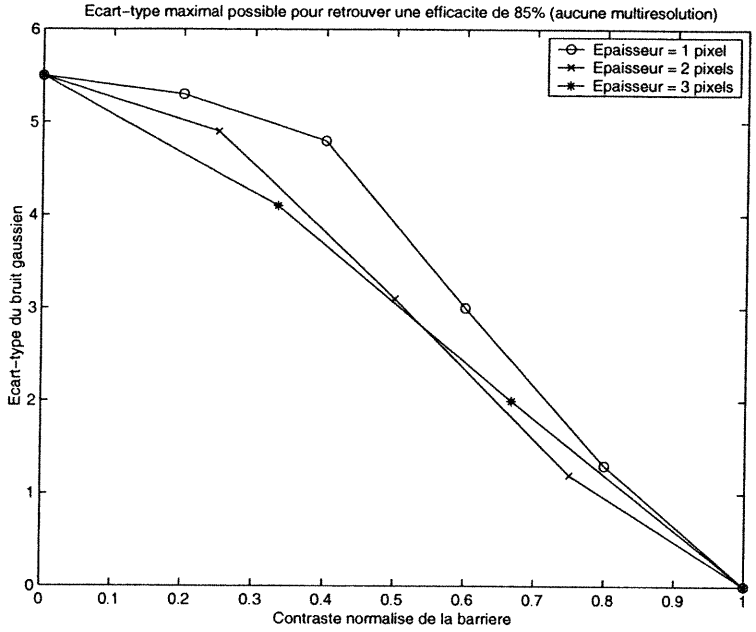
Nous voulons établir le lien entre l'écart-type du bruit et l'efficacité de notre méthode. On veut connaître, par exemple, l'écart-type maximal du bruit qu'on peut avoir dans une image pour un pourcentage d'efficacité donné.

Nous utiliserons encore l'image synthétique définie plus haut pour faire les tests de notre méthode. Nous prenons donc notre image, nous introduisons du bruit dans celle-ci ayant un certain écart-type et nous observons si nous sommes en mesure de retrouver une efficacité de 85%. La façon dont nous présenterons les résultats est la suivante. Le nombre de multirésolutions utilisé sera toujours le même et les paramètres seront l'épaisseur de notre barrière, l'écart-type du bruit et le contraste de la barrière. Pous se rappeler le problème étudié, la Figure 1.2 montre le problème et un zoom sur une région difficile à retrouver avec les méthodes traditionnelles.

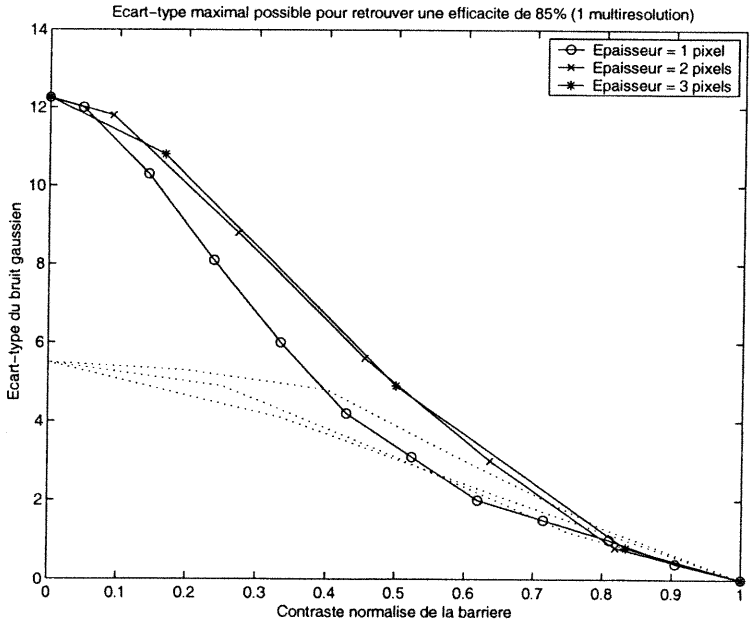
Les Figures 4.5 et 4.6 montrent les résultats obtenus en étudiant notre méthode en présence de bruit.

Le premier graphique, 4.5(a), correspond à la méthode à progression rapide sans l'utilisation de l'analyse multirésolution. Les autres correspondent à un ou plusieurs niveaux de résolutions. Chaque graphique a comme abscisse le contraste normalisé de la barrière avec le fond de l'image et en ordonnée l'écart-type maximal du bruit pour obtenir 85% d'efficacité. Le contraste normalisé est le contraste trouvé pour un certain jeu de paramètres divisé par le constraste maximal que la méthode peut retrouver pour ce même jeu de paramètres. Les graphiques signifient donc que pour tout jeu de paramètres se trouvant *sous* les courbes, nous avons une efficacité d'au moins 85%.

Plus un point se trouve à droite sur les abscisses, plus nous sommes en mesure de dépasser une barrière avec un très grand contraste et plus un point est haut sur les ordonnées, plus nous pouvons avoir du bruit dans notre image tout en ayant

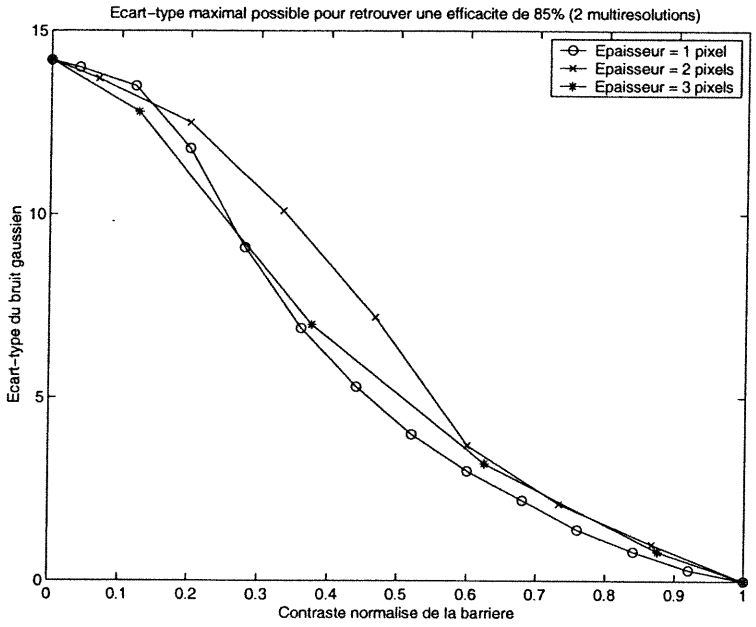


(a) *Sans utilisation de la transformée en ondelettes (aucune multirésolution). Le contraste est normalisé.*

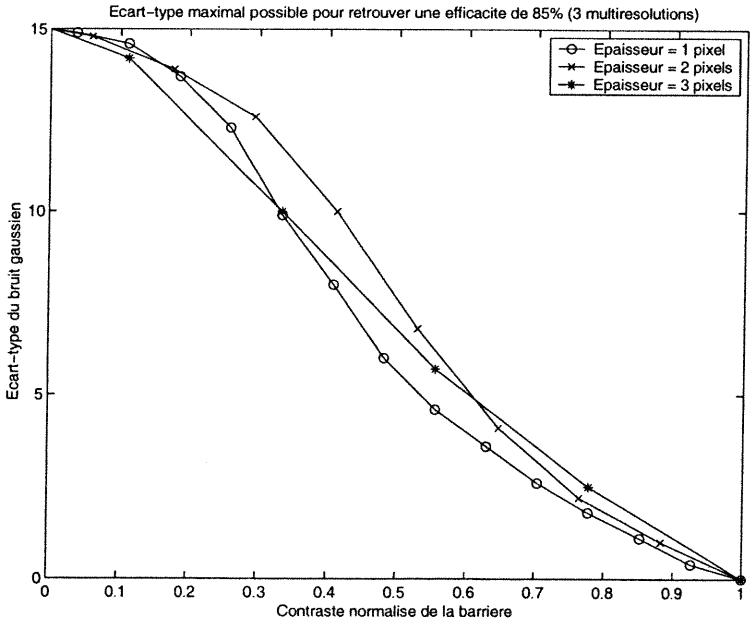


(b) *Utilisation d'une multirésolution. Le contraste est normalisé. Les courbes en pointillées sont celles obtenues à la Figure 4.5(a).*

FIG. 4.5. *Résultats de la méthode lorsque soumise à du bruit gaussien. Le contraste est normalisé.*



(a) *Utilisation de deux multirésolutions. Le contraste est normalisé.*



(b) *Utilisation de trois multirésolutions. Le contraste est normalisé.*

FIG. 4.6. *Résultats de la méthode lorsque soumise à du bruit gaussien (suite). Le contraste est normalisé.*

une efficacité de 85%. Dans le meilleur des cas, nous aurions voulu obtenir un graphique qui donne une droite placée très haut sur les ordonnées et qui descend très rapidement rendu à un certain contraste plus ou moins grand. En effet, plus nos courbes sont élevées sur les ordonnées, plus nous pouvons avoir du bruit dans une image tout en étant capable d'avoir une excellente efficacité. En fait, cela nous donnerait une méthode très robuste par rapport au bruit. Si la courbe avait eue une grande pente négative à partir d'un certain contraste, la méthode aurait été très robuste. Elle aurait donc donnée un résultat semblable pour un grand éventail de contraste, ce qui aurait permis de ne pas poser d'hypothèses a-priori sur l'image étudiée.

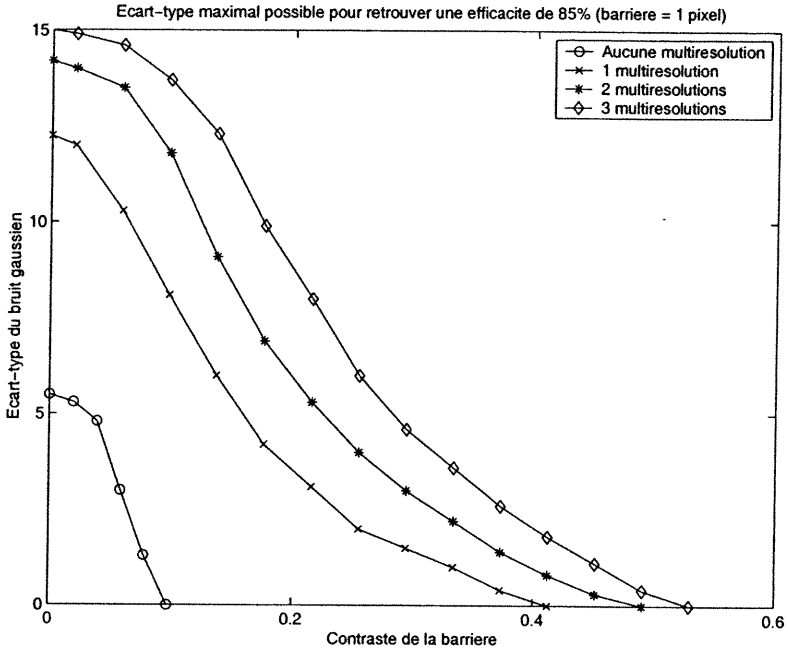
Les 4 graphiques ont des allures très semblables. Nous remarquons que plus le contraste de la barrière augmente, plus l'écart-type du bruit doit diminuer. Intuitivement, cela est juste puisque pour un fort contraste de la barrière, il devient difficile de dépasser celle-ci. Inversement, lorsque le bruit augmente, il devient aussi plus difficile de retrouver le contour voulu. Il faut donc compenser un contraste très grand par un bruit plus faible et vice-versa. Nous remarquons également que si l'épaisseur de la barrière augmente, le bruit et le contraste doivent nécessairement baisser eux aussi si on veut retrouver notre pourcentage d'efficacité. En effet, si l'épaisseur de la barrière augmente, la difficulté pour la traverser augmente, et si le bruit ou le contraste augmentaient par rapport à ceux d'une barrière moins épaisse, la difficulté serait augmentée et nous ne pourrions pas obtenir notre pourcentage d'efficacité.

Nous remarquons également que le contraste que nous pouvons retrouver grâce à la méthode n'est pas "trop" grand. Si ça s'avérait être le cas, il se pourrait que notre méthode traverse la barrière mais également traverse des frontières qui ne doivent pas être traversées. Il doit toujours y avoir une limite pour le contraste car sinon nous traverserions quelque chose qui fait bien partie du contour voulu. Il faut bien comprendre que le choix du contraste seuil entre lequel nous avons un "vrai" contour et un contour que l'on veut traverser, est une notion arbitraire, qui devra être résolue pour chaque problème étudié.

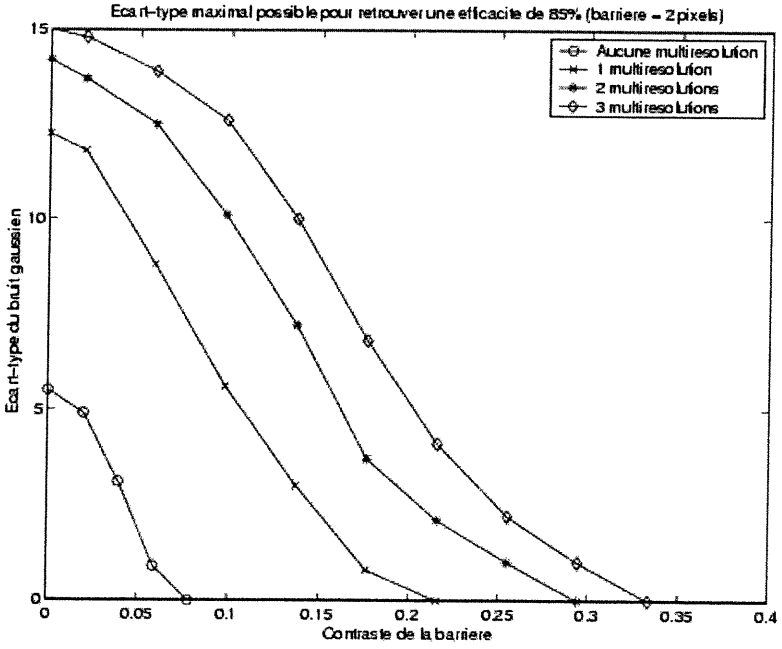
Ces graphiques nous permettent d"etablir une marche à suivre lorsque nous considérerons des images réelles. En effet, ces graphiques nous indiquent le nombre de résolutions à utiliser dépendant de l'épaisseur des barrières et du contraste de celles-ci. On peut donc estimer le nombre de résolutions à prendre dans un problème concret en estimant l'intensité et l'épaisseur des barrières à traverser.

Les Figures 4.7 et 4.8 contiennent la même information que les figures discutées plus avant, mais au lieu d'avoir pour chaque graphique le nombre de multirésolution constant, nous aurons maintenant l'épaisseur de la barrière constante. Contrairement aux graphiques plus haut, ceux-ci nous renseignent sur l'influence du nombre de multirésolution sur l'efficacité de la méthode. En effet, chaque graphique nous donne l'écart-type maximal en fonction du contraste pour avoir une efficacité de 85%. Par contre, chaque graphique nous donne les résultats à une épaisseur fixe de la barrière. Nous remarquons que si nous n'utilisons pas l'analyse multirésolution, le bruit et le contraste que nous pouvons avoir sont très faibles. Nous remarquons par contre une grande amélioration si nous utilisons un niveau de multirésolution. Si nous prenons deux ou trois niveaux, nous observons une amélioration, mais celle-ci est beaucoup plus faible que celle obtenue avec le premier niveau de multirésolution. Dans l'image étudiée, nous remarquons que ne prendre qu'un seul niveau de multirésolution nous donne un contour de bien meilleure qualité. Par contre, le nombre de niveaux à utiliser pour avoir un bon contour va changer d'une image à une autre, en fonction du bruit, de la taille des artefacts, etc. Dans le cas général, il faut donc essayer plusieurs niveaux pour savoir quel sera le meilleur jeu de paramètres pour obtenir le meilleur contour possible.

À l'aide de ces différents graphiques, nous pouvons donc décider d'une bonne marche à suivre lorsque nous aurons à traiter des images réelles. Lorsque nous avons une image réelle, la seule chose que nous pourrons choisir pour notre méthode est le nombre de multirésolutions que nous allons utiliser. Lorsque nous aurons notre image réelle, nous devrons évaluer l'écart-type du bruit qui est présent dans l'image, celui-ci nous renseignera sur le nombre de multirésolutions à utiliser pour retrouver le contour voulu.

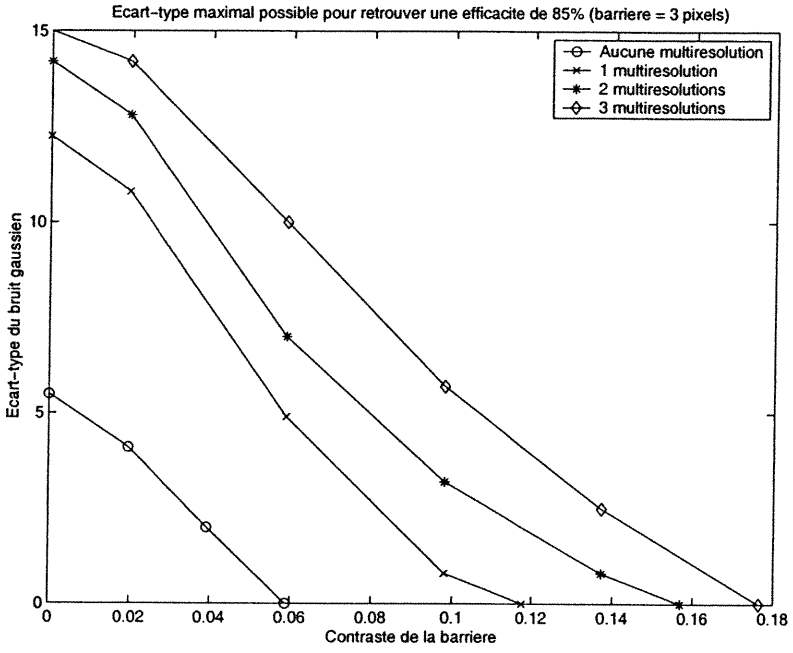


(a) *Avec épaisseur de la barrière, 1 pixel.*



(b) *Avec épaisseur de la barrière, 2 pixels.*

FIG. 4.7. *Résultats de la méthode en présence de bruit gaussien.*



(a) *Avec épaisseur de la barrière, 3 pixels.*

FIG. 4.8. *Résultats de la méthode en présence de bruit gaussien (suite).*

Nous avons déjà remarqué qu'une seule multirésolution semble le bon choix à faire dans le cas de notre image synthétique. Il faut par contre faire attention, car le nombre de multirésolutions à utiliser dépend fortement de la taille de l'objet à traverser et aussi de la taille de l'image étudiée. En effet, l'image utilisée avait une taille de 256 par 256 pixels et la taille de notre barrière n'était que de 3 pixels au maximum. Une seule multirésolution nous donnait donc une image de 128 par 128 pixels et la barrière était déjà presque totalement disparue, c'est pour cette raison que nous observons un si grand saut dans l'efficacité avec seulement une multirésolution. Si nous avions une image plus grande ou alors une barrière plus large, il aurait fallu prendre un plus grand nombre de multirésolutions pour obtenir un résultat optimal. Le choix du nombre de multirésolutions dépendra donc de la taille de l'image étudiée. Pour une image, au-delà d'un certain nombre de résolutions, les pixels composant celle-ci ne contiennent plus d'information pertinente. En effet, si nous prenons trop de résolutions, l'image est à une résolution tellement faible que les pixels ne renferment plus aucune information, ils ne représentent que du bruit.

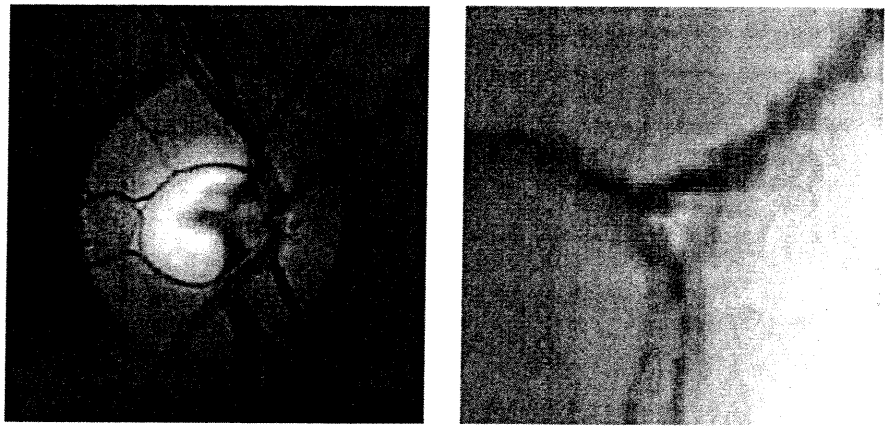
4.3. Application de la méthode aux images réelles

Nous allons maintenant appliquer l'algorithme à des images réelles. En premier lieu, nous étudierons des images ayant des caractéristiques se rapprochant de celles des images synthétiques étudiées précédemment. Nous prenons donc des images ayant des caractéristiques semblables aux barrières de nos images précédentes. Il ne faut pas perdre de vue que dans le cas d'images réelles, nous ne pouvons pas utiliser un test d'efficacité comme celui utilisé pour les images test. En effet, nous ne pouvons pas avoir un test quantitatif de la qualité du contour retrouvé car nous ne connaissons pas le contour que nous voulons obtenir. Dans ces cas-là, la seule validation possible est celle donnée par un expert. Ainsi, nous nous limiterons à évaluer *qualitativement* le résultat obtenu pour savoir si nous avons un contour suffisament près du contour voulu.

La première image que nous allons étudier avec l'aide de notre méthode combinée, est celle d'une rétine. Nous sommes intéressés à retrouver le contour de la macula à l'aide de la méthode discutée dans ce mémoire[1]. En se basant sur les différents graphiques montrés plus avant, à la section 4.2.2, nous sommes en mesure d'estimer le nombre de résolutions nécessaires à la résolution du problème. Dans une image de ce type, comme à la Figure 4.9, il y a très peu de bruit. Aussi, la barrière que nous voulons traverser ne semble être que de l'ordre d'un seul pixel en épaisseur. On peut donc prévoir qu'en utilisant une seule multirésolution, nous soyons en mesure de retrouver un contour qui nous semble le plus satisfaisant. C'est exactement ce qui se produit, comme on peut le constater sur la Figure 4.10. On remarque immédiatement que l'utilisation de la multirésolution permet de retrouver un contour qui semble se rapprocher davantage du contour réel.

Lorsque nous voulons visualiser le résultat de l'algorithme à progression rapide, nous devons utiliser un algorithme pour retrouver le contour, c'est-à-dire prendre un plan et couper la surface à une hauteur choisie. Par contre, ces algorithmes ont beaucoup de difficulté à retrouver des endroits où se forment des

[1]La surface de la macula nous donne de précieux renseignements sur l'état du diabète chez une personne. Le diabète affecte la macula en diminuant sa surface, permettant ainsi un diagnostic de l'état de la maladie.



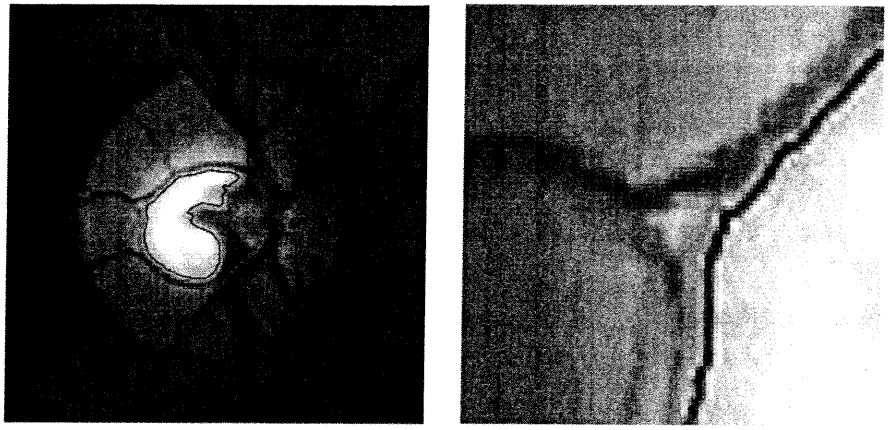
(a) *Image dont on veut retrouver le contour.*

(b) *Zoom de la région difficile à retrouver.*

FIG. 4.9. *Nous voulons récupérer la zone agrandie ci-dessus tout en retrouvant le reste du contour.*

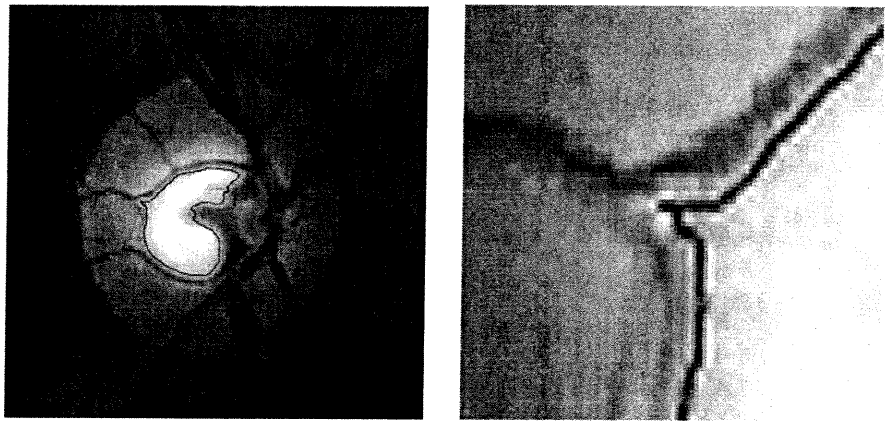
coins ou toute autre forme qui change rapidement. Nous voulons maintenant laisser de côté ces algorithmes de contour et travailler d'une manière différente. Puisque pour trouver le contour lors de l'algorithme à progression rapide nous devons "couper" notre surface à une certaine hauteur, comme expliqué à la section 2.1.2, nous allons plutôt colorier d'une certaine couleur tous les pixels plus petits qu'une certaine valeur et observer la forme qu'ils vont produire. Comme on peut le constater à la Figure 4.11, ceux-ci nous donnent la forme de la macula que nous avons trouvée à la Figure 4.10. De plus, nous avons un avantage à regarder notre contour sous cette forme, car cela nous permet de calculer rapidement l'aire de la macula à l'aide du nombre de pixels que nous identifions comme faisant partie de cette région. Dans l'exemple ci-haut, nous obtenons 2451 pixels lorsque nous n'utilisons pas l'analyse par ondelettes et 2460 pixels lorsque nous utilisons l'analyse multirésolution. Les pixels retrouvés supplémentaires correspondent à la protubérance retrouvée grâce à la méthode combinée.

Étudions maintenant ce que l'on obtient avec la méthode combinée lorsqu'on l'utilise sur une image du cerveau, prise par imagerie à résonance magnétique nucléaire. Tel que nous l'avons fait dans le cas précédent, nous allons discuter des différences dans les contours obtenus par la méthode à progression rapide seule



(a) *Contour retrouvé après avoir utilisé la méthode à progression rapide seule.*

(b) *Zoom du contour retrouvé après avoir utilisé la méthode à progression rapide seule.*

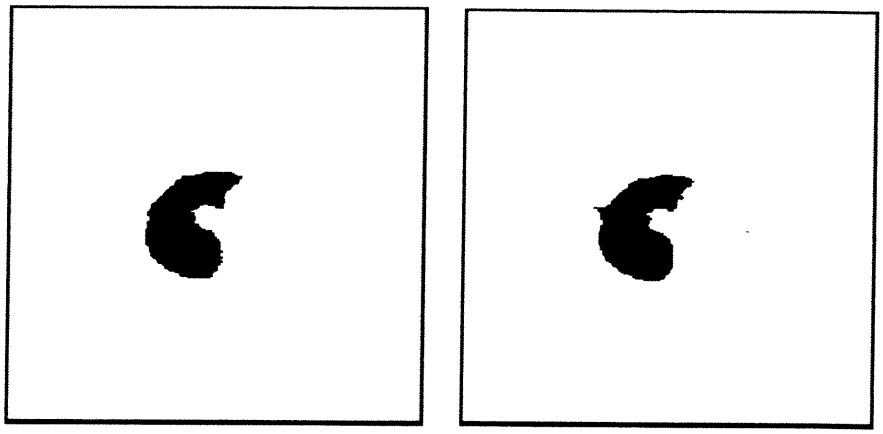


(c) *Contour retrouvé après avoir utilisé la méthode à progression rapide conjointement avec l'analyse multirésolution.*

(d) *Zoom du contour retrouvé après avoir utilisé la méthode à progression rapide conjointement avec l'analyse multirésolution.*

FIG. 4.10. *Récupération du contour à l'aide de différentes méthodes.*

et par la méthode combinée. La Figure 4.12(a) montre l'image pour laquelle nous voulons retrouver un contour. Les Figures 4.12(b-c) sont les résultats obtenus. On remarque immédiatement que le contour obtenu par les méthodes à progression rapide seulement semble plus "bruité" que celui retrouvé lorsque nous utilisons également l'analyse multirésolution. Un résultat semblable à celui de la Figure



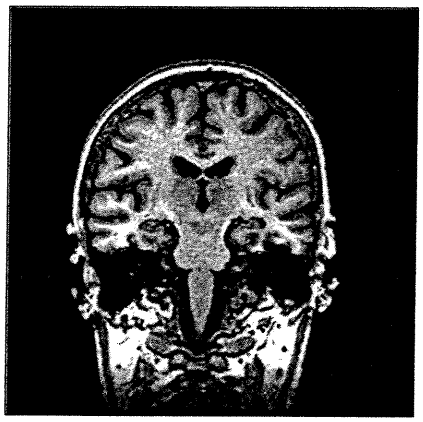
(a) *Contour retrouvé après avoir utilisé la méthode à progression rapide seule.*

(b) *Contour retrouvé après avoir utilisé la méthode à progression rapide avec l'algorithme multirésolution.*

FIG. 4.11. *Récupération du contour à l'aide de différentes méthodes. Au lieu d'utiliser un algorithme de contour, nous ne colorions que les pixels ayant une valeur plus petite qu'un certain seuil.*

4.12(c) pourrait être obtenu en convoluant l'image originale avec une gaussienne pour diminuer le bruit, lisser certains détails et ensuite n'utiliser que les méthodes à progression rapide.

Puisque la différence entre les deux images n'est pas très élevée, nous allons de nouveau colorier les pixels plutôt que d'utiliser des algorithmes de contour, comme montré à la Figure 4.13. De cette façon, nous pouvons mieux observer le résultat de l'algorithme utilisé. De plus, il nous est possible de cette manière de calculer rapidement l'aire de la surface englobée par le contour, ce qui peut être intéressant dans une grande quantité d'applications. Lorsque nous colorions les pixels, nous pouvons donc les compter, comme expliqué juste avant. Nous sommes donc en mesure de dire que si nous n'utilisons que la méthode à progression rapide, nous trouvons 30299 pixels, de plus il semble y avoir beaucoup de bruit dans l'image. Cette méthode retrouve beaucoup de points à l'intérieur que nous aimerions voir disparaître. Tandis que si nous utilisons la méthode à progression rapide avec



(a) *Image dont on veut retrouver le contour.*

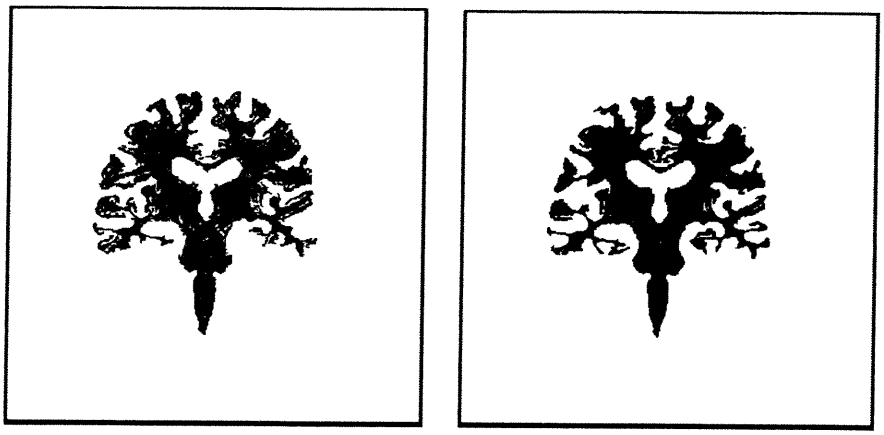
(b) *Contour retrouvé par la méthode à progression rapide seule.*

(c) *Contour retrouvé par la méthode à progression rapide et l'analyse multirésolution.*

FIG. 4.12. *Nous voulons récupérer le contour de cette image d'une coupe du cerveau.*

l'analyse multirésolution, nous découvrons 31002 pixels, tout en ayant une image beaucoup moins bruitée.

Étudions ce qui arrive si nous bruitons l'image traitée. La Figure 4.14(a) représente l'image bruitée, de la tranche IRM précédente, avec un bruit gaussien d'écart-type de 10. Nous voulons retrouver le contour dans cette image. Nous remarquons, qu'en présence de bruit, l'algorithme élaboré dans ce mémoire est



(a) *Contour retrouvé après avoir utilisé la méthode à progression rapide seule.*

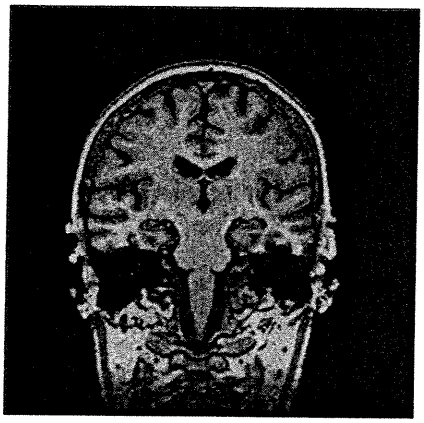
(b) *Contour retrouvé après avoir utilisé la méthode à progression rapide avec l'algorithme multirésolution.*

FIG. 4.13. *Récupération du contour à l'aide de différentes méthodes. Au lieu d'utiliser un algorithme de contour, nous colorions les pixels ayant une valeur plus petite qu'un certain seuil.*

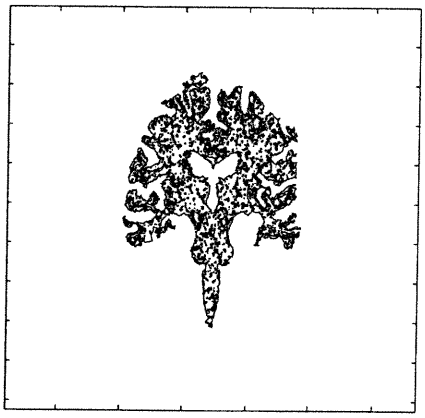
beaucoup plus performant que si nous n'utilisons que les méthodes à progression rapide.

La Figure 4.14(b) nous donne un contour contenant une grande quantité d'artefacts provenant uniquement du bruit. Nous ne voulons pas retrouver ces artefacts, ceux-ci n'étant que le résultat du bruit introduit dans les données. On remarque par contre, que la Figure 4.14(c) quant à elle, nous donne un contour se rapprochant beaucoup de celui trouvé à la Figure 4.12(c). L'algorithme conjoint parvient ainsi à "débruiter" le contour voulu tout en gardant l'image telle qu'elle est, c'est-à-dire sans la convoluer pour lisser ce bruit. La méthode introduite dans ce mémoire semble donc beaucoup plus robuste par rapport au bruit que la méthode à progression rapide.

Contrairement à l'exemple précédent, nous n'afficherons pas les images obtenues par coloriage des pixels, la Figure 4.14 nous permettant de bien distinguer le contour bruité de celui débruité.



(a) *Image bruitée avec bruit gaussien d'écart-type 10, dont on veut retrouver le contour.*



(b) *Contour retrouvé par la méthode à progression rapide seule.*

(c) *Contour retrouvé par la méthode à progression rapide et l'analyse multirésolution.*

FIG. 4.14. *Nous voulons récupérer le contour de cette image bruitée d'une coupe du cerveau.*

Pour terminer, nous rappelons qu'il nous est impossible de trouver un critère *quantitatif* permettant de mesurer l'efficacité de notre méthode sur des images réelles. Par contre, il est possible d'étudier le rapport entre les points retrouvés grâce à la méthode à progression rapide seule et les points retrouvés si nous utilisons la multirésolution. Lorsque nous faisons cette étude, nous nous retrouvons

face à un problème. Nous étions intéressés à calculer le rapport des points retrouvés en fonction de l'écart-type du bruit. Nous voulions, en fait, voir la différente segmentation de l'image que les méthodes parvenaient à faire. Nous devions, pour cela, retrouver "à peu près" toujours le même contour. Mais, lorsque l'écart-type du bruit augmente, la méthode à progression rapide seule à tendance à dépasser le contour voulu. Il devient ici presqu'impossible de trouver la valeur pour laquelle la méthode offre le meilleur contour. Il est donc difficile (et même sans intérêt car nous ne comparons plus les mêmes choses), de construire un graphe de cette situation.

4.4. Conclusion

Ce chapitre nous a permis d'expliquer en plus grand détail l'algorithme utilisé. Nous avons également validé notre méthode sur des images artificielles, dans le but de quantifier son efficacité, et nous l'avons fait sur des images réelles. Ces images ont permis de juger de l'efficacité de la méthode en cas réel et de voir les divers avantages que cette méthode peut avoir sur les méthodes traditionnellles, dans certains cas précis. Par exemple, cet algorithme nous permet de traverser certains obstacles, selon que leur taille et leur intensité se trouve dans une certaine plage de valeurs. Il permet aussi de "débruiter" les contours retrouvés sans à avoir à faire de traitement préalable de l'image.

Chapitre 5

CONCLUSION

En guise de conclusion, nous reviendrons rapidement sur les résultats obtenus dans ce mémoire. Nous discuterons également des voies futures qui pourraient être explorées.

5.1. Résumé

L'approche utilisée dans ce mémoire est basée sur deux techniques bien établies. La première technique est la méthode à progression rapide introduite par Sethian dans [1]. Cette technique est utilisée de plus en plus fréquemment pour résoudre des problèmes très divers. Les applications comprennent ainsi la détection de contours, le suivi d'onde sismique, le parcours de labyrinthes, etc. La seconde technique est l'analyse multirésolution, qui a été introduite par Mallat et qui est discutée en détail dans [10]. Cette technique est aussi très populaire présentement et ses utilisations ne se comptent plus.

La nouveauté de ce mémoire est d'utiliser ces deux méthodes en même temps pour obtenir des contours qui approximent mieux la forme étudiée et aussi qui permettent d'obtenir des résultats moins bruités. En effet, nous remarquons que l'utilisation de l'analyse multirésolution permet de "lisser" certains détails, de telle sorte que le contour peut se propager plus facilement et rapidement sans rencontrer des obstacles, comme du bruit, qui ne font pas partie du contour voulu recherché. De plus, ces contours, bien que moins bruités, n'entraînent pas de perte d'information dans l'image, ce qui serait le cas par exemple lorsque nous convoluons une image avec un filtre gaussien.

Ces techniques permettent donc de retrouver des contours moins bruités, bien que l'image étudiée le soit toujours. De plus, elles permettent de "dépasser" certains obstacles jugés inintéressants, et ne faisant pas partie du contour voulu.

5.2. Futur

Le premier prolongement intéressant de la méthode développée dans ce mémoire serait l'application de celle-ci en trois dimensions. Il serait utile de construire un algorithme en trois dimensions et de l'appliquer à différents volumes dans l'espace.

Une autre voie intéressante nous est montrée à la Figure 5.1. Lorsque nous

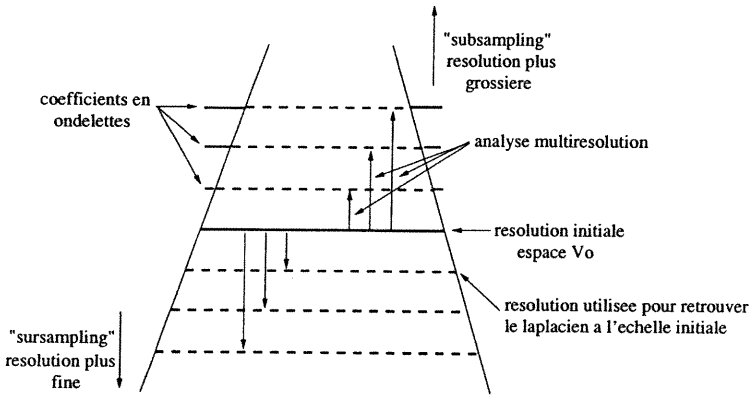


FIG. 5.1. *Schéma d'une voie future à explorer.*

utilisons la multirésolution, nous remontons dans le schéma, ce qui revient à lisser notre image et à enlever du bruit. Par contre, comme nous le savons, il est possible de "descendre" dans le schéma en ajoutant des pixels, et de la résolution, à notre image. Dans ce cas, nous ne donnons pas d'information supplémentaire à notre image, mais interpolons celle-ci. Nous descendons déjà d'une résolution dans notre schéma pour retrouver le Laplacien à la résolution de l'image de départ. Rien ne nous empêche donc de continuer encore vers des résolutions supérieures. En faisant cela, nous agrandissons les structures se trouvant à l'intérieur de l'image. De cette manière, il est possible que la détection de structures fines se fasse plus facilement. En effet, il est très difficile pour une méthode comme la nôtre, de découvrir des structures fines. Par contre, en utilisant l'algorithme multirésolution "à l'envers", nous pourrions peut-être obtenir de meilleurs résultats.

BIBLIOGRAPHIE

[1] Sethian, J.A., *A Fast Marching Level Set Method for Monotonically Advancing Fronts*, Proc. Nat. Acad. Sci., 93, 4, pp.1591-1595, 1996.

[2] Osher S. and Sethian, J. A., *Fronts propagating with curvature dependent speed : Algorithms based on Hamilton-Jacobi formulation*, J.Comput.Phys., vol. 79, pp. 12-49, 1988.

[3] Sethian, J. A., *A review of recent numerical algorithms for hypersurfaces moving with curvature dependent speed*, J.Differential Geom., vol. 31, pp. 131-161, 1989.

[4] Sethian, J.A., *Level set methods.* Cambridge,1996.

[5] Malladi, R., Sethian, J. A. and Vemuri, B.C., *Shape modeling with front propagation : a level set approach*, IEEE Trans. Pattern Anal. Machine. Intel., vol. 17, pp. 158-175, 1995.

[6] Rouy, E. et Tourin, A., *A Viscosity Solution Approach to Shape-From-Shading*, SIAM J. Num. Anal., 29, 3, pp. 867-884, 1992.

[7] Sedgewick, R., *Algorithms*, Addison-Wesley, Reading, MA, 1988.

[8] Danielsson P., *Euclidian distance mapping*, Computer Graphics and Image Processing, 14 :227-248, 1980.

[9] Mallat S., *A theory for multiresolution signal decomposition : The wavelet representation*, IEEE Trans. Pattern Anal. Machine. Intel., vol. 11, July 89.

[10] Mallat S., *A Wavelet Tour of Signal Processing*, Academic Press, 1998.

[11] Rosenfeld A., *Multiresolution Image Processing and Analysis*, Springer-Verlag, New-York, 1982.

[12] Lina J-M., *Image Processing with Complex Daubechies Wavelet*, CRM Report 2335, 1996.

[13] Lina J-M., *Complex Daubechies Wavelets : filters design and applications*, CRM Report 2449, 1996.

[14] Burt P.J. and Adelson E.H., *The Laplacian pyramid as a compact image code*, IEEE Trans. Commun., 31(4) :532-540, April 1983.

[15] Daubechies I., *Ten lectures on Wavelets*, Philadelphia, PA : SIAM, 1992.