Université de Nice Sophia-Antipolis

DEA SIC : Image - Vision

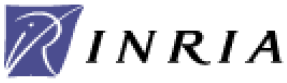
Rapport de stage

Utilisation de fonctionnelles de type rapport de coûts pour l'extraction de régions dans des images de télédétection

Lieu : Ariana, projet commun I3S/INRIA
Encadrants : Ian JERMYN et Josiane ZERUBIA

10 septembre 2004

Igor Rosenberg



Résumé

La segmentation d'image a subit de nombreuses avancées ces dernières décennies. On propose ici de s'intéresser à une nouvelle forme d'énergie, écrite comme un rapport de coût. L'intérêt principal en est de proposer un équilibre automatique entre deux énergies concurrentes. On exhibe ici deux manières de se servir de ce nouveau cadre, d'une part en minimisant globalement l'énergie sur des graphes, d'autre part en se basant sur le cadre variationnel. Cependant, le travail montre que la modélisation est grandement compliquée par le rapport des énergies, et que des interactions spécifiques sont à prendre en compte.

Mots-clés : rapport de coût, télédetection, segmentation, zones urbaines

Abstract

Image segmentation has gone through numerous steps forward these last decades. In this report, we propose to look into a new energy, written as a ratio cost. The main interest is to have an automatic equilibrium between two concurrent energies. We show here two different methods to use this new framework, one way being to globally minimize the energy on graphs, and the other way uses the variationnal methodology. But the work shows that the modelling is greatly complicated by the ratio cost, and that specific interactions should be taken into account.

Key-words : ratio cost, remote sensing, segmentation, urban areas

Table des matières

Chapitre 1

Introduction

1.1 Position du problème - but

Dans le cadre du stage de DEA, effectué dans l'équipe Ariana[1], on se propose d'extraire les regions correspondant à des zones urbaines dans une image satellitaire. On souhaite retourner des courbes fermées dont l'intérieur correspond à une partie de ville. Ce thème de recherche correspond à la demande d'analyse de zones urbaines, pour les besoins par exemple des télécommunications, ou la création de cartes d'occupation des sols, qui est à mettre en parallèle avec les modifications des villes (par exemple, Macapá, ville du Brésil, voit sa population augmenter de 10% par an).

La difficulté consiste à retrouver le contour exact d'une ville, ainsi qu'à savoir exclure des zones particulières à l'intérieur des villes (parcs, lacs...), dans un processus entièrement automatisé. De plus, la vérité de terrain n'est accessible qu'au travers d'un expert, ce qui complique le travail de vérification des résultats.

Le travail de stage repose sur deux méthodes développées au sein de l'équipe Ariana[1]. D'une part, les travaux de la thèse [20] d'Anne Lorette décrivent une méthode d'analyse statistique de la texture attribuant une valeur réelle à chaque pixel, valeur correspondant à un indice de confiance pour une zone urbaine. On utilise alors la méthode décrite par Jermyn et Ishikawa [14] (méthode de rapport de coût sur un bi-graphe), qui est capable d'extraire d'un graphe la région correspondant au minimum global d'une description à priori donnée sous forme d'énergie, pour définir où se trouvent exactement les frontières.

Le travail original réalisé lors du stage est d'une part de trouver des fonctions qui permettent de souligner les caractéristiques des villes sur des images satellitaires, afin que la région trouvée par la méthode du rapport de coût corresponde à une zone urbaine. Dans une deuxième partie, je me suis intéressé à explorer le comportement des rapports de coût dans le cadre des méthodes variationnelles. En effet, une telle formulation de l'énergie n'est mentionnée nulle part dans la littérature, bien qu'elle présente certains avantages. J'ai donc mis en parallèle les deux approches (graphe et variationnelle) afin de les comparer.

1.2 L'approche rapport d'énergies

Dans le cadre de l'extraction de contours, on s'intéresse à la **minimisation** d'une énergie $E = \frac{E_1}{E_2}$, rapport de deux fonctions d'énergie E_1 et E_2. Plus spécifiquement, on souhaite analyser la forme

$$E[R] = \frac{\int_R A(x)\mathrm{d}x}{\int_{\partial R} B(\gamma(s))\mathrm{d}s} \tag{1.1}$$

La fonction B est construite pour être toujours positive, et on souhaite que le dénominateur soit invariant par rapport à l'orientation de la courbe, tandis que A peut être négative, et doit changer de signe quand l'orientation change. Il existe des configurations où cette énergie est négative : si l'on trouve une région à énergie positive, il suffit d'inverser le sens de parcours pour que l'énergie devienne alors négative. Il

[1] INRIA Sophia-Antipolis `http://www-sop.inria.fr/ariana/`

est plus simple de voir le problème de minimisation sur contours orientés comme une maximisation de la valeur absolue, ce qui a pour effet de nullifier le rôle de l'orientation. On a ici l'inclusion de deux types de descripteurs de départ, en aboutissant à un équilibre entre la maximisation de la valeur absolue du numérateur et la minimisation du dénominateur (qui est toujours positif). On remarque que l'on n'a pas de paramètre qui relie les deux termes d'énergie. En effet l'équilibre entre les deux termes est faite par le quotient, et la minimisation de l'un et la maximisation de l'autre.

Pour donner un exemple, E_1 peut être choisie comme une fonction du gradient de la région, tandis que E_2 décrit le périmètre de la région, en choisissant $B = 1$. Il faut comparer cela à une écriture plus standard de la forme $E = E_1 + \alpha E_2$, où l'évaluation du paramètre prend un rôle prépondérant, car c'est lui qui indique quel est l'apport de chacune des énergies mises en jeu. Par exemple, lorque α a une grande valeur, la contribution de l'énergie 1 est minime, tandis que pour une valeur de α proche de zéro, la courbe subira plus l'influence de cette énergie.

En prenant des rapports d'énergies, on se retrouve avec une énergie qui peut être cjoisie comme invariante à la multiplication. Multiplier les données par une constante ne change rien, grâce au quotient d'intégrales, si les fonctions intégrées sont linéaires. Cette forme d'énergie est donc adaptée aux changements d'illuminations que l'on modélise généralement par la multiplication par un facteur. L'intérêt est aussi que l'on n'est pas biaisé vers des régions plus ou moins grandes : l'énergie du numérateur fait grandir la frontière, tandis que celle du dénominateur réduit la longueur de la frontière. L'équilibre se fait en fonction des données sous-jacentes. Par exemple, l'énergie d'une région donnée serait la même qu'une version réduite de moitié de la même région, en supposant que l'on ait dans les deux régions les mêmes propriétés.

L'avantage principal de cette formulation réside dans l'existence d'un algorithme pour trouver la solution globale au problème de minimisation, dont le cœur est décrit dans 3.1.3. Je vais introduire dans un premier temps une version d'un algorithme de minimisation de l'énergie sur des graphes. Je présenterai ensuite la description de la même problèmatique, dans le cadre des méthodes variationnelles.

1.2.1 Discrétisation sur un graphe

Le but est d'extraire d'une image une région qui minimise la forme d'énergie donnée en 1.1. Il existe plusieurs façons de résoudre le problème sur les images, qui est un domaine discret, alors que l'énergie est décrite de façon continue. L'utilisation de méthodes variationnelles habituelles, de type "level sets", est décrite dans la partie suivante. Pour la première partie, je me suis concentré sur l'utilisation d'une méthode qui permet de trouver le minimum global de la fonction d'énergie. En effet, en projettant l'image sur un graphe, on est alors capable de trouver le cycle qui minimise globalement l'énergie (1.1).

Les travaux de Jermyn et Ishikawa [14] se concentrent sur l'application d'un algorithme d'extraction du cycle de rapport minimal sur un graphe dont les arêtes possèdent deux poids (bi-graphe), lorsque les données initiales sont fournies par une image. On cherche le cycle C dont le poids $w(C) = \frac{\sum(\lambda)}{\sum(\tau)}$ est minimal, où λ et τ sont les poids du graphe.

L'algorithme présenté trouve le minimum **global** sur tous les cycles du graphe. On est donc assuré d'avoir la meilleure solution de la modélisation que l'on a proposée. Il faut mettre cette propriété en parallèle avec les techniques variationnelles habituelles, où l'on trouve généralement des minima locaux, car on utilise une descente de gradient, ce qui lie très fortement la réponse à l'initialisation. Pour la méthode utilisée ici sur les graphes, on n'a pas ce problème : il n'y a pas d'initialisation de la courbe, et l'on trouve toujours la région minimisant le critère.

Par contre, on doit trouver la fonction de projection des données, qui envoie une image sur un graphe. C'est là que la modélisation a lieu. Il faut proposer une fonction qui souligne les éléments importants de l'image, et qui atténue les bruits et les zones qui ne nous intéressent pas. En effet, l'algorithme sur les graphes fournit bien le minimum global, mais il ne présente aucune modélisation du phénomène auquel on s'intéresse, comme toute méthode. Il faut donc construire son entrée pour que le minimum corresponde à ce que l'on souhaite extraire. Un grand soin doit donc être apporté à l'écriture des fonctions A et B. On peut voir cette

projection comme le fait que l'algorithme est indépendant des données initiales, et donc applicable à toute sorte de données initiales. En effet, si l'on est capable de trouver une fonction qui projette les données sur un graphe, on peut extraire de ce graphe la région minimisant le rapport d'énergie. L'algorithme en lui-même ne se sert que du graphe généré. On peut donc réutiliser cette méthode sur des images médicales ou des images d'une séquence vidéo, sous réserve que l'on parvienne à projeter les données initiales sur un graphe. C'est d'ailleurs le travail de la première partie du stage : valider une réutilisation de la méthode d'extraction du cycle de rapport minimal sur des images satellites, dans le but d'extraire des régions correspondant aux zones urbaines.

1.2.2 Discrétisation variationnelle

La méthodologie variationnelle nous donne un cadre pour trouver des solutions partielles à des minimisations d'énergie sur des courbes. On commence par se donner une forme d'énergie, qui modélise le problème auquel on s'intéresse. La solution du problème est le minimum global de cette énergie. On construit alors une suite de courbes, reliées entre elles par une descente de gradient : chaque courbe est "meilleure" que la précédente parce qu'ayant une énergie plus faible. On tombe malheureusement dans un puits de potentiel, sans avoir de garantie sur la distance au minimum global de l'énergie.

Pour décrire l'évolution des courbes, on transforme la formulation d'énergie globale de la courbe en utilisant la méthode d'Euler-Lagrange. On obtient alors la dérivée temporelle du contour, que l'on décrit sous forme de force définie en chaque point du contour, et qui pousse cette courbe vers une énergie de valeur plus faible. La partie dérivation de l'énergie demande un tout petit peu d'attention, concernant le sens de certains vecteurs, ainsi qu'avec le signe des quantités manipulées. On a alors toutes les clés en main pour dérouler la machinerie variationnelle, qui est bien étudiée depuis les travaux fondateurs de [16].

Chapitre 2

Etat de l'art

Dans le travail d'extraction des zones urbaines d'une image, il y a deux parties bien distinctes. En premier lieu, il faut prétraiter les données image dont on dispose pour mettre en valeur les éléments sur lequels on souhaite s'attarder. Il faut trouver une méthode pour faire ressortir de l'image les pixels correspondant à de la ville, et diminuer l'importance de tous les autres. Bien évidement, cela repose sur un a priori des caractéristiques d'une ville. Ensuite, une fois qu'on a prétraité l'image, il faut extraire la région, en segmentant. On utilise les résultats de la description urbaines, et on essaie de séparer les zones réellement urbaines des autres zones. On a exploré dans ce stage deux méthodes qui découlent de la formulation sous forme d'énergie. Dans un premier temps, je me suis concentré sur la segmentation de graphes. Ensuite, j'ai utilisé le cadre variationnel avec la même énergie.

2.1 Description urbaine

Les principales méthodes d'extraction de zones urbaines sont basées sur l'analyse de texture. L'analyse de texture est très difficile car les textures réelles ne sont souvent pas uniformes, à cause des changements d'orientation et de taille. De plus, beacuoup d'algorithmes proposés sont d'une grande complexité algorithmique. On remarquera que l'on utilise souvent des champs de Markov, pour leur capacité à modeliser des interactions parfois lointaines.

On peut trouver des états de l'art dans [11], [3], [29]. [36] ont divisé l'analyse de texture en quatre catégories :
- statistique
- geometrique
- basé modèle
- analyse du signal.

Les méthodes statistiques analysent la distribution dans l'espace des niveaux de gris, en calculant des caractéristiques locales en chaque point de l'image, puis en calculant des statistiques de ces paramètres locaux. Des méthodes de premier, deuxième ordre, ainsi que d'ordre supérieur existent, et permettent de calculer moyenne, variance et ressemblance de structures. Les plus utilisées sont les paramètres de cooccurrence [12] et les différences de niveau de gris [39]. Des méthodes basées sur l'autocorrélation ont aussi été développées, mais pour des résultats décevants [5].

Les méthodes géométriques décomposent les textures en primitives, pour ensuite deviner des règles sur leur organisation spatiale. Ces primitives peuvent être extraites par détection de contours [35], ou par morphologie mathématique [32]. L'organisation des structures est alors apprise par des méthodes statistiques, mais aussi par diagrammes de Voronoï [35].

Les méthodes basées modèles font l'hypothèse de l'existence d'un modèle sous-jacent, régit par différents paramètres. On considère l'intensité des pixels comme fonction de deux éléments : un bruit additif aléatoire, et la structure de la surface de l'objet sur lequel repose la texture. On se réferrera à [3] pour de plus amples

renseignements. On trouve des modèles ou l'élément de base est la région, et d'autres où l'élément de base est le pixel ; [27] a proposé un modèle basé fractal.

Les méthodes basées sur l'analyse du signal analysent le contenu fréquenciel de l'image. Il existe des filtres de domaine spatial (mask de Laws [18]), mais les masques standards (Robert et Sobel) peuvent aussi être utilisés pour obtenir des informations de fréquence. On peut aussi extraire de l'information des moments [37]. L'utilisation de la transformée de Fourier par fenêtre permet de garder de l'information sur la position mais on peut aussi utiliser des transformée en ondelettes [22]. Lorsqu'on utilise des fonctions (somme de gaussiennes), on crée des bancs de filtres. Manjunath et Ma [23] ont amélioré la conception des filtres pour couvrir tout le spectre sans faire exploser la dimensionalité . On pourra se référer à [30] pour une plus longue description de ces méthodes d'analyse du signal.

Dans [26], il est proposé différents paramètres de texture pour l'analyse et l'identification de zones urbaines dans les images aériennes. Une étude complète sur une zone urbaine, ainsi qu'une étude intra-urbaine sont présentées.

La société SCOT[1], en accord avec Eurosat[2], propose dans sa gamme de logiciels des outils pour la délimitation d'agglomérations urbaines par le traitement automatique d'images satellitaires. Ils recherchent des logiciels et des algorithmes appropriés pour l'extraction de l'urbain, et proposent des méthodologies permettant de faire des tests sur différentes villes, avec différentes images.

Anne Lorette présente dans sa thèse [20] une méthode pour déterminer la position des villes sur des images satellites, en construisant des champs de Markov pour analyser la texture en chaque pixel, puis en utilisant un K-means flou pour classifier ses résultats. Sa méthode est presque entièrement automatisée, seule une étape de choix des classes réellement urbaine doit être réalisée par un opérateur.

2.2 Segmentation de graphe

Dans un article de 1993, Wu et Leahy [40] proposent une méthode pour segmenter un graphe en deux masses distinctes, en se basant sur la notion de "minimum cut", c'est-à-dire en choisissant la segmentation qui minimise le coût de séparation du graphe en deux agglomérats. Ce critère peut être utilisé pour obtenir de bons résultats de segmentation sur quelques images. Cependant, cette méthode a tendance à favoriser les petits ensembles de noeuds du graphe. Elle est donc biaisée vers les petites régions, ce qui n'est pas souhaitable.

Shi et Malik [34] ont poussé plus loin l'analyse en proposant de normaliser les régions par la mesure de la similarité entre régions, c'est à dire qu'il y a une certaine mesure de la taille de l'agrégat qui est prise en compte. On obtient ainsi une approche qui n'est plus biaisée vers les petites régions. Cependant, on ne peut pas inclure des information sur la frontière de la région considérée, car les régions sont vues comme des ensembles de points, et la spécificité des points sur le bord n'est pas exploitable en tant que telle. De plus, ils ne proposent qu'une solution approchée au problème qu'ils se posent. En effet, la résolution du "normalized cut" est un problème très dur (NP-complet).

Cox et al. [6] présentent une approche de rapport d'énergie, entre une aire généralisée et une longueur généralisée, et trouvent la courbe optimale en choisissant chaque point de l'image comme pivot, l'un après l'autre. Cependant, ils sont restreint aux aires généralisées, ie l'intégrale de fonctions positives sur la région. De plus, l'algorithme est très lourd à mettre en œuvre.

Utilisant une autre approche, Jermyn et Ishikawa [14], qui se basent sur les travaux de Lawler [17] et de Meggido [24], proposent de résoudre la minimisation d'un rapport d'énergie sur un graphe. Ils proposent une solution exacte, en **temps polynomial**, au problème qu'ils se sont posés. De plus, dans le même article, ils se servent d'une méthode donnée par Karp [15] pour appliquer une solution hautement parallélisable et

[1] Service et conception de systèmes en observation de la Terre : `http://www.scot-sa.com`
[2] `http://www.eurostat.org`

multirésolution au problème du rapport moyen minimum sur un cycle dans un graphe, appliqué sur des images. On est cependant limité dans les fonctions qui peuvent être utilisées pour décrire les régions. Par exemple, il n'est pas directement possible d'introduire de l'information du deuxième ordre. La détection de régions à trou, par exemple un tore, n'est pas possible, car dans leur formalisme une région est décrite par l'intérieur d'un cycle. De plus, la méthode n'est pas interactive, et ne permet pas à l'utilisateur de spécifier des points de recherche initiaux, du fait du minimum global. Cependant ces extensions peuvent être envisagées, en élagant le graphe des régions non désirées.

L'article de Wang et Siskind [38] présente une nouvelle fonction, le "ratio-cut", pour partitionner un graphe, ainsi qu'un algorithme (limité aux graphes planaires connexes) qui résoud le problème en temps polynomial, en se basant sur les travaux de Jermyn et Ishikawa [14] dans une de leurs étapes de réduction. De plus, ils étendent leur méthode afin de s'affranchir de bruit poivre-et-sel, et de bords flous, en construisant une heuristique. Ils accélèrent aussi la méthode en travaillant sur des sous-images qui sont ensuite fusionnées.

2.3 Segmentation par méthodes variationnelles

Une excellente introduction m'a été proposée dans le cours de DEA de M. Rachid Deriche[3].

Introduits par Kass, Witkin et Terzopoulos [16] en 1988 , les méthodes variationnelles sur les contours tentent de minimiser une énergie définie sur l'ensemble des contours d'une image. Comme trouver le minimum global est considéré comme un problème difficile (NP-complet), la méthode généralement employée est de prendre un minimum local obtenu par descente de gradient, en utilisant la méthode d'Euler-Lagrange. Typiquement, l'énergie s'écrit sous la forme

$$E(\gamma) = \alpha \int E_{int}(\gamma(s))\mathrm{d}s + \beta \int E_{image}(\gamma(s), I)\mathrm{d}s \qquad \text{où } \gamma : [0,1] \to C$$

L'énergie interne tend à lisser la courbe, tandis que l'énergie image rapproche la courbe des éléments importants de l'image.

Il existe de nombreuses extensions de l'idée de départ : "level-sets" [25], snakes [16], optimisations "Narrow band" et "extension velocities" [1].

Cependant, malgré le nombre élevé de publication, qui reflète l'essort prit par ces méthodes, peu de gens se sont intéressés à la modélisation de phénomènes incluant toutes la région sélectionnée. On pourra cependant citer les travaux de Stéphanie Jehan-Besson, Michel Barlaud et Gilles Aubert [13] qui proposent dans leur modèle Dream^{2}s de prendre comme fonctionnelle

$$J(\Omega_{in}, \Omega_{out}, \Gamma) = \iint_{\Omega_{out}} k^{(out)}(x, y, \Omega_{out})\mathrm{d}x\mathrm{d}y + \iint_{\Omega_{in}} k^{(in)}(x, y, \Omega_{in})\mathrm{d}x\mathrm{d}y + \int_{\Gamma} k^{(b)}(x, y)\mathrm{d}s \qquad (2.1)$$

En introduisant un paramètre τ qui gouverne les étapes de l'évolution, ils obtiennent une paramétrisation de l'évolution de l'énergie. La dérivation est alors possible, par rapport à τ, et donne une évolution fonction d'intégrales sur le contour et sur la région entière.

On pourra aussi citer les travaux de Marie Rochery [31]. Elle modélise son problème, l'extraction de résaux linéiques, par une énergie quadratique, qui est, en fait, une intégrale double d'une fonction bien choisie. Elle obtient alors dans son équation d'évolution des termes dépendant de la forme du contour. Ces travaux peuvent donc être mis en parallèle avec les idées présentées ici, du moins pour l'écriture des énergies.

[3] http://www-sop.inria.fr/robotvis/personnel/der/teaching.html

Chapitre 3

Eléments théoriques

Je présente dans cette partie les éléments théoriques nécessaires à la compréhension de mon travail de stage.

Dans la première partie, je me suis d'abord concentré sur une segmentation de graphes. Je présente brièvement donc la théorie des graphes standard, qui est enseignée généralement dans les premières années de faculté. J'introduis aussi l'utilisation de la méthode rapport de coût, qui est à la base de l'algorithme utilisé.

Ensuite, j'ai appliqué ces méthodes sur les travaux d'Anne Lorette, qui utilisent des champs de Markov. Je présente donc succintement sa méthode, et la théorie générale sur laquelle elle s'appuie.

Dans la dernière partie de mon stage, j'ai pu appliquer le principe du rapport de coût dans le cadre variationnel. Je décris donc, dans la troisième partie de ce chapitre, comment sont construites les méthodes variationnelles, de façon détaillée, afin de bien préciser ces techniques, qui ne sont enseignées que très tardivement dans un cursus universitaire.

3.1 Théorie des graphes

Deux excellentes références sur la théorie des graphes sont le livre de Gondran et Minoux [10], et celui de Ahuja et al. [2]. Après les définitions, je présenterai les algorithmes standards que j'utilise, puis je présenterais la méthode utilisée par Jermyn et Ishikawa dans [14], méthode que je reprends dans mon travail de stage.

3.1.1 Définitions

- Un graphe orienté $G = (S, A)$ est défini comme un ensemble S, dont les éléments sont appelés les sommets, et un ensemble A, ensemble ordonné de couples de sommets, appelés arêtes.

- Un chemin de longueur l est une suite de l arcs notée $C = \{a_1, \ldots, a_l\}$ avec $a_1 = (s_0, s_1), a_2 = (s_1, s_2), \ldots, a_l = (s_{l-1}, s_l)$. Un chemin peut être vu comme le parcours d'une suite de sommets en prenant les arcs du graphe.

- Un cycle est un chemin tel que $s_0 = s_l$.

- On peut construire une fonction poids $w : A \to E$ qui associe à chaque arête une valeur, par exemple un réel ou un couple d'entiers.

3.1.2 Algorithmes standards

Je présente ici quelques problèmes classiques en théorie des graphes, et des algorithmes qui les résolvent. Je les utilise lorsqu'il faut extraire un cycle de poids négatif d'un graphe. Je commence par l'algorithme

classique de Dijkstra, qui trouve la distance entre un sommet particulier nommé source, et tous les autres sommets. Ensuite, une version adaptée aux poids négatifs est présentée (correction d'étiquettes) et finalement je montrerai comment trouver un cycle négatif, s'il existe, dans un graphe de poids quelconques.

Plus court chemin
Soit G un graphe pondéré, $w : A \to \mathbb{R}^+$ une fonction de poids positive. Trouver le plus court chemin entre un sommet source s et tous les autres sommets.

```
DIJKSTRA()
 1  T := ∅;        T̄ := S;
 2  d(i) := ∞ pour chaque sommet i ∈ S;
 3  d(s) := 0;        pred(s) := ∅;
 4  tant que T̄ ≠ ∅ faire
 5            soit i ∈ T̄ tel que d(i) = min{d(j)|j ∈ T̄}
 6            T := T ∪ {i}
 7            T̄ := T̄ − {i}
 8            pour  toute arête (i, j) faire
 9                        si d(j) > d(i) + w_ij alors
10                           d(j) := d(i) + w_ij
11                           pred(j) := i;
```

Le principe est de garder deux ensembles de sommets, T et $\bar{T}$. Tous les sommets dans T ont déjà leur distance finale. On choisit alors dans l'ensemble $\bar{T}$ le sommet dont la distance est la plus petite. Ce nœud est à sa distance minimale : n'importe quel autre chemin sortant de l'ensemble T pour rejoindre ce sommet a forcément une longueur plus grande. On insère ce nœud dans l'ensemble T, on met à jour ses successeurs et l'on itère. Les distances sont bien calculées à partir du sommet source s, car il est le seul à n'être pas infini à l'initialisation, c'est avec celui-ci que les calculs de distance commencent. L'algorithme termine avec tous les sommets ayant un attribut d qui correspond à la distance de ce sommet à la source s.

L'algorithme de Dijkstra résoud le problème du *plus court chemin*, mais ne marche que pour des poids positifs car l'hypothèse qui constitue le socle de l'algorithme est que tous les nœuds de T ont déjà leur distance à la source finale. Or, si l'on a un arc de poids négatif, on risque de modifier la distance d'un des nœuds de T. Dans le cas de poids négatifs, on risque d'avoir un cycle de poids négatif dans le graphe. En le parcourant, on mettrait tout le temps à jour les nœuds dont la distance serait indéfiniment décroissante (vers $-\infty$). Il existe un algorithme pour le cas où les arêtes peuvent être négatives, mais où on garantit qu'il n'y a pas de cycles négatifs.

```
CORRECTION D'ETIQUETTES()
 1  d(s) := 0;        pred(s) := 0;
 2  d(i) := ∞ pour chaque sommet i ∈ S − {s};
 3  LIST := {s};
 4  tant que LIST ≠ ∅ faire
 5            enlever un sommet i ∈ LIST
 6            pour  toute arête (i, j) faire
 7                        si d(j) > d(i) + w_ij alors
 8                           d(j) := d(i) + w_ij
 9                           pred(j) := i;
10                           si j ∉ LIST alors
11                              ajouter noeud j à LIST
```

On choisit un sommet de la liste. On met à jour tous les successeurs de ce sommet. On met dans la liste les successeurs, car leurs fils ne sont peut-être plus à jour, il faudra donc les réactualiser.

Finalement, on a les outils pour trouver un cycle négatif :

Détection de cycles négatifs dans un graphe
On se donne G un graphe pondéré par $w : A \to \mathbb{R}$, une fonction de poids quelconque. Trouver un cycle négatif, s'il en existe un.

On a plusieurs façons de résoudre le problème, en se basant sur la mise à jour des distances des nœuds. Pour la simplicité de l'exposé, on supposera que le graphe est accessible à partir de la racine, ie pour tout sommet x, il existe un chemin de la source à x (dans le cas contraire, il faut chercher indépendamment dans chaque partie connexe).

- *Borne inférieure :* si le graphe ne contient pas de cycles négatifs, alors la distance de la source à tout nœud est supérieure à $|S| * Min$, où $Min = \min(\min\ w, 0)$ $(\star)$. On fait donc tourner un algorithme de correction d'étiquette, en choisissant arbitrairement le nœud source, et on attend un des évènements suivants :
  - la condition $(\star)$ est bafouée, un cycle négatif passe par le sommet de poids minimal.
  - l'algorithme termine, alors il n'y a pas de cycle négatif.

- *Recherche de cycle négatif :* On se contente de vérifier de temps en temps que le graphe des prédécesseurs $(pred(y) = x \Leftrightarrow d(y) = d(x) + w_{x,y})$ n'a pas de cycle, qui correspondrait alors à un cycle négatif dans le graphe original. Cette recherche peut être faite en "colorant" le graphe des prédecesseurs, la rencontre de la couleur déjà existante sur un nœud à colorier représentant un cycle.

3.1.3 Algorithme rapport de coût

Dans [14], Jermyn et Ishikawa appliquent à des images une méthode basée sur les travaux de Lawler [17] et de Meggido [24]. Ils y expliquent comment extraire le cycle de rapport minimal sur un bi-graphe $(G, A, S, (\lambda, \tau))$ dont les arêtes possèdent deux poids λ et τ. Le principe est de plonger la recherche de région dans un graphe. On se donne une image initiale. On construit alors une projection de l'image sur les bi-graphes, on résoud le problème du rapport minimal sur les cycles de ce graphe, puis on retourne dans l'espace des images, où le cycle apparaît alors comme la frontière d'une région. Le grand intérêt est que l'algorithme sur les graphes ne propose qu'un seul résultat, le minimum **global** sur tous les poids de cycles du graphe, le poids d'un cycle C étant défini par

$$w(C) = \frac{\sum_{a \in C} \lambda}{\sum_{a \in C} \tau} \tag{3.1}$$

L'algorithme est général, dans le sens qu'il n'a pas de paramétrisation liée aux données sur lequel il travaille. La contrepartie est que l'on doit trouver la fonction de projection des images, ie une méthode de construction d'un graphe (sommets et arêtes) qui reflète les données de l'image, et qui souligne les zones qui répondent au critère que l'on recherche. L'idée est que le cycle qui sera extrait sera composé d'arêtes dont la somme a une forte valeur. La modélisation a donc lieu lors de la construction de ce graphe, c'est là que la description de ce que l'on cherche est introduite.
On cherche à trouver la solution générale au problème suivant :

Cycle de poids minimal
Soit un bi-graphe $(G, S, A, (\lambda, \tau))$ vérifiant
- $\lambda : A \to \mathbb{Z}$,
- $\tau : A \to \mathbb{N}^\star$,
- Si $(u, (\lambda, \tau), v)$ est une arête de G, alors $(v, (-\lambda, \tau), u)$ en est une autre.
Trouver le cycle de poids $w = \frac{\sum \lambda}{\sum \tau}$ minimal.

On se restreint aux cycles qui ne font pas plusieurs boucles sur eux-mêmes, car ils ont le même rapport que le cycle qui ne fait qu'un seul tour. La solution ne peut pas être un cycle qui se croise, car sinon une des deux parties de cycles possède un rapport meilleur que la boucle. Cependant, on peut trouver des cas dégénérés où plusieurs cycles ont le même poids.

L'algorithme fut d'abord décrit par Lawler [17], puis généralisé par Meggido [24]. Il repose sur l'observation suivante : soit $w_t : A \to \mathbb{Q},\ w_t(a) = \lambda(a) - t\tau(a),\ t \in \mathbb{Q}$. Si t^* est le plus grand t tel l'équation

$w_{t^*}(C) < 0$ n'admet pas de solution sur l'espace des cycles de G, alors t^* est la valeur de t telle la solution au problème du *Cycle de poids minimal* est le cycle C qui vérifie $w(C) = t^*$.

On a donc réduit le problème à celui de trouver t^*, ie la valeur de t telle que le cycle de poids minimum, en considérant les poids w_t, est zero. Ceci revient alors à chercher la plus grande valeur (négative) de t telle que le graphe (G, w_t) n'a pas de cycle négatif. On remarquera bien que le graphe (G, w_t) a une fonction poids qui envoie les arêtes sur les rationels, et non plus sur un couple d'entiers.

```
RechercheCycleMin()
1  t = 0 // borne supérieure sur t
2  tant que (ilExisteUnCycleNegatif(G, t)) faire
3            t = rapport d'un cycle négatif deG
4       rendre trouverCycleDePoidsNul(G, t)
```

Concernant l'initialisation, on remarque que si les cycles ne sont pas tous de poids w nul, il existe toujours un cycle de poids négatif. En effet, si C est un cycle de poids $w(C) > 0$, alors $-C$, le même cycle, mais en prenant les arêtes à l'envers, est de poids négatif : $w(-C) < 0$. Il faut bien voir aussi que dans l'algorithme, le rapport des cycles n'est utilisé que pour mettre t à jour. Le poids d'un cycle, lors de la phase de recherche de cycles négatifs, est donné par $w_t(C) = \sum(\lambda(a) - t\tau(a))$

Il faut remarquer que, bien que le résultat cherché soit un cycle C, dont le poids est un rapport $w(c) = \frac{\sum\lambda}{\sum\tau}$, que l'on cherche à minimiser, l'algorithme travaille sur des poids $w_t(C) = \sum\lambda - t\tau$. Juste avant que l'algorithme ne rende son résultat, la valeur t est le plus petit rapport de poids des cycles de G. Il suffit alors d'extraire ce cycle du graphe. On peut le faire en considérant le graphe $(\tilde{G}, A, S, \tilde{w})$ où $\tilde{w}(u, v) = w(u, v) - d(v) + d(u)$. Dans ce graphe, les arêtes de poids non nul correspondent à des arêtes qui ne sont pas sur le chemin menant à un cycle. On peut donc les supprimer, et rechercher un cycle dans le graphe beaucoup moins dense (toutes les arêtes sont alors de poids nul) qui reste.

On remarquera que, de toutes manières, la valeur de t décroît jusqu'à atteindre t^*. On peut donc utiliser une dichotomie. On a une valeur d'où l'on peut commencer, car 0 est une borne supérieure. On peut donc commencer avec $t = -1$ et multiplier cette valeur tant qu'il y a des cycles négatifs dans le graphe. Lorsqu'on n'en trouve plus, t est plus grand que t^*. Par dichotomie, on se rapproche alors de t^* en comsidérant qu'un cycle négatif trouvé signifie que t^* est plus petit, et pas de cycle négatif signifie que t^* est plus grand. J'ai décidé d'arrêter la dichotomie lorsque l'intervalle de recherche est plus petit qu'une constante que l'on choisi (1 est une bonne valeur). On réutilise alors l'algorithme linéaire décrit ci-dessus pour trouver la valeur exacte de t^*, ce que ne peut pas faire la dichotomie (considérer le cas où $t^* = -\frac{5}{3}$, qui ne peut pas être atteint par une dichotomie, qui ne peut atteindre que des fractions de dénominateur en 2^k). L'avantage de la dichotomie est que, en un nombre raisonnable d'itérations, on se trouve proche de la valeur finale de t^*, ce qui n'est pas garanti par la version linéaire, qui pourrait essayer *tous* les cycles du graphe avant de trouver le bon. Cependant, la dichotomie teste souvent des valeurs plus petites que t^*. Il faut alors mener l'algorithme de correction d'étiquettes à son terme, ce qui est très couteux, car il faut trouver la distance de chaque nœud à la source. Cependant, l'expérience montre que la dichotomie est plus rapide.

3.1.4 Construction du graphe

On est capable, par l'algorithme décrit dans le paragraphe précédent, d'extraire le cycle de rapport minimal d'un bi-graphe. Cependant, la construction d'un tel graphe, et les raisons qui poussent certaines arêtes à appartenir au cycle élu n'ont pas été explicitées. Je montre dans cette partie comment placer les sommets du graphe, pour correspondre aux pixels, ainsi que comment calculer le poids de chaque arête.

Localisation des sommets

On souhaite obtenir une correspondance entre l'image initiale et un cycle dans le graphe. Pour la localisation spatiale des sommets, deux solutions se présentent (voir figure 3.1). Les deux versions sont implantées,

mais la version où les sommets du graphe sont les angles des pixels s'explique mieux intuitivement, et permet de mieux définir le poids des arêtes du graphe en fonction de l'image de départ.

Poids des arêtes

On va construire les arêtes telles que l'algorithme cherche la région R qui minimise l'énergie

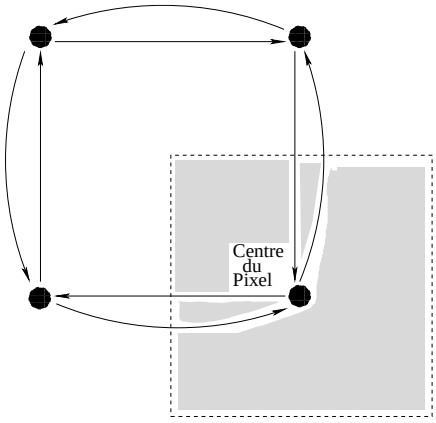
$$E[\partial R] = \frac{N[\partial R]}{D[\partial R]} = \frac{\int_S \gamma'(t)^\perp \cdot v(\gamma(t))\mathrm{d}t}{\int_S |\gamma'(t)| g(\gamma(t))\mathrm{d}t} \qquad (1) \tag{3.2}$$

avec
- ∂R la frontière de la region,
- $S = S^1$ un cercle,
- $\gamma : S \to \mathbb{R}^2$, qui représente la frontière de la région δR dans $\mathbb{R}^2$
- $v : \mathbb{R}^2 \to \mathbb{R}^2$ un champ de vecteurs,
- $g : \mathbb{R}^2 \to \mathbb{R}$.

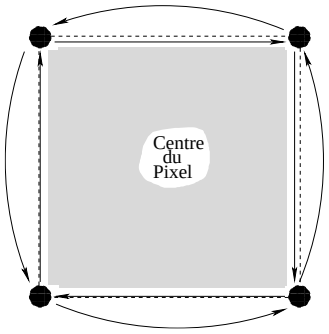
Cette énergie peut paraître étonnante au premier coup d'œil : mise sous cette forme, c'est l'expression d'une énergie d'une région dans le domaine continu. En fait, comme l'on souhaite rester le plus général possible, l'expression sur le plan est plus intéressante que celle sur un domaine discret. La forme finale de l'énergie (fonctions v et g) est à choisir en fonction d'à priori sur la région que l'on souhaite extraire, et doivent dépendre des données initiales. Ensuite, on discrétise cette énergie pour pouvoir la faire passer dans l'algorithme décrit plus haut. Cependant, je vais proposer une réécriture de l'énergie pour une manipulation plus aisée.

On conserve le dénominateur sous cette forme pour les informations sur la frontière. Donc pour calculer les valeurs de τ pour chaque arête, il faut intégrer la fonction g entre les deux sommets, sur une ligne droite. On calcule donc pour chaque arête $a : I \to \mathbb{R}^2$

$$\tau = \int_I g(a(s))\mathrm{d}s \tag{3.3}$$



On peut placer les sommets du graphe au centre des pixels. Les arêtes représentent alors les relations de voisinage entre pixels. Cheminer le long d'une arête revient à changer de pixel. Les régions obtenues seront alors composées des pixels sur le cycle, ainsi que des pixels à l'intérieur.



On peut placer les sommets du graphe dans les coins des pixels. Les arêtes correspondent alors au bord des régions, qui sont constituées des pixels à l'intérieur des arêtes du cycle.

FIG. 3.1 – Différents positionnements des sommets du graphe.

Par exemple, on peut prendre $g = 1$. On a alors $\tau \in \{1, \sqrt{2}\}$, si l'on est en 8-connexité. De plus, si l'on regarde alors la somme de ces valeurs sur les cycles, on trouve que l'on a une petite valeur pour des petites régions. En fait, pour $g = 1$, on mesure le périmètre. Cette énergie a tendance à créer des régions convexes et pousse la courbe à être moins tordue. En effet, beaucoup de circonvolutions feront augmenter la valeur du dénominateur, tandis que la ligne droite produira le dénominateur le plus petit.

Par contre, grâce au théorème de Green,

$$\int_S \gamma'^{\perp}(t) \cdot v(\gamma(t))dt = \pm \int_R \nabla \cdot v(x, y)dxdy$$

(le signe dépend de l'orientation de la région) on peut transformer une intégrale sur un contour en une intégrale sur une région. En prenant donc une fonction telle que

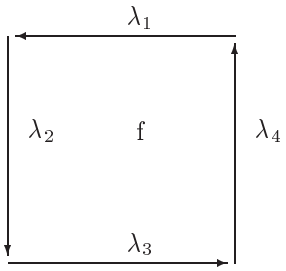
$$f = \nabla \cdot v, \tag{3.4}$$

on peut apporter des informations sur la région entière dans le numérateur, et on remarquera que le champ vectoriel $v : \mathbb{R}^2 \to \mathbb{R}^2$

$$v_f^x(x, y) = \int_0^x f(x', y)\mathrm{d}x'$$

$$v_f^y(x, y) = 0$$

permet de retrouver la relation 3.4. Cette forme est très élégante car ce champ de vecteurs n'a qu'une coordonnée significative, ce qui permet de mettre à zéro toutes les arêtes horizontales.



On construit alors les poids λ des arêtes par

$$\lambda_4 = -\lambda_2 + f \tag{3.5}$$

$$\lambda_3 = \lambda_1 = 0$$

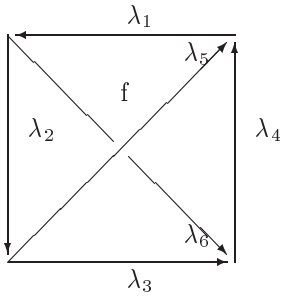
en n'oubliant pas de construire les symétriques :

$(u, (\lambda, \tau), v) \in G \Longrightarrow (v, (-\lambda, \tau), u) \in G$

On remarque que la somme des poids λ sur le cycle correspond à la somme des valeurs de f dans la région ,

$$\sum_{cycle} \lambda = \sum_{region} f \tag{3.6}$$

C'est l'analogue du théorème de Green sur les graphes. En effet, on a construit les valeurs λ des arêtes (en intégrant dans la relation (3.5)) pour que cette relation soit une conséquence de la construction. On peut aussi construire des diagonales (8-connexité), en ajoutant la valeur de la moitié du pixel traversé, ce qui conserve la propriété (3.6), si l'orientation est respectée.



On met come valeurs

$\lambda_5 = -\lambda_2 + \frac{f}{2}$

$\lambda_6 = \lambda_2 - \frac{f}{2}$

Pour résumer, on peut dire que l'on est parti d'un rapport d'intégrales autour d'une région dans le domaine continu, que l'on a discrétisé sur un graphe par le quotient de sommes sur des arêtes :

$$E[R] = \frac{\int_R A(x)\mathrm{d}x}{\int_{\partial R} B(\gamma(s))\mathrm{d}s} \approx \frac{\sum_{a \in cycle} \lambda(a)}{\sum_{a \in cycle} \tau(a)} \tag{3.7}$$

3.2 Champs de Markov

On se rappelle que le but final est d'extraire des zones urbaines d'images satellitaires. A l'aide des algorithmes présentés dans la partie ci-dessus, on sait trouver dans un graphe le cycle de rapport minimal, qui correspond à la région minimum global du critère utilisé. On a aussi dit que pour passer d'une image à un graphe, il fallait construire une fonction qui sache prendre en compte la modélisation des régions que l'on souhaite trouver.

On présente donc ici certains des travaux d'Anne Lorette sur les images de télédétection, qui décrivent une méthode utilisant des champs de Markov pour caractériser les pixels selon un indice de texture. Cet indice caractérise assez bien les zones urbaines, dans le sens qu'une valeur élevée de l'indice correspond très probablement à un pixel dans une ville, et un indice de valeur basse est synonyme de texture appartenant à autre chose qu'une zone urbaine. Cela va nous permettre d'avoir une première «impression» de la localisation des zones urbaines, qui seront alors segmentées par l'algorithme de rapport de coûts.

Dans la première partie, je décris brièvement les caractéristiques d'un champ de Markov, qui sont nécessaires pour présenter les travaux d'Anne Lorette centrés sur la détection de zones urbaines.

3.2.1 Définitions

On renvoie à l'ouvrage de Li [19] pour une présentation générale des champs de Markov et leur utilisation en vision par ordinateur, ainsi qu'à [28] pour le lien entre les champs de Markov et la théorie des graphes.

L'image est représentée comme un ensemble de sites S.

Définition :
V est un voisinage du site s si

1. $s \notin V(s)$
2. $s \in V(t) \Leftrightarrow t \in V(s)$

On considère le champ aléatoire $X = (X_1 \dots X_n)$, défini sur S, à image dans Ω.

Définition :
(X, Ω, P) est un champ de Markov par rapport au système de voisinage V s'il vérifie :

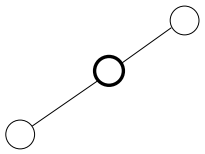


FIG. 3.2 – Voisinage pour le modèle uni-directionel

1. $\forall x \in \Omega, P(X = x) > 0$
2. $\forall s \in S,\ \forall x \in \Omega,\ P(X_s = x_s | \{X_r = x_r, r \in S - \{s\}\}) = P(X_s = x_s | \{X_r = x_r, r \in V(s)\})$

Cela signifie que la valeur d'un site du champ ne dépend que de la valeur de ses voisins. Cependant, on peut quand même introduire des proporiétés globales en imposant des contraintes locales.

Théorème d'Hammersley-Clifford :
Si (X, Ω, P) est un champ de Markov, alors sa distribution $P(X)$ est une distribution de Gibbs :

$$P(X) = \frac{e^{-U(X)}}{Z} \qquad \text{où} \quad Z = \sum_{X \in \Omega^s} e^{-U(X)} \text{ et } U(X) = \sum_{c \in C} V_c(\{X_s, s \in c\})$$
$$\text{et } C \text{ représente l'ensemble des cliques}$$

Pour être plus exhaustif, il faudrait présenter quelques modèles (Ising, Potts, ...), un exemple d'utilisation de champ de Markov (restauration), et les estimateurs habituels (MAP, MPM...). Ensuite la relaxation probabiliste avec Métropolis et l'echantilloneur de Gibbs, le principe du recuit simulé et quelques approximations deterministes de type ICM ou GNC. On pourra se référer au livre [4].

3.2.2 Indice de texture pour les zones urbaines

Dans sa thèse [20], et l'article [21], Anne Lorette, en se basant sur des travaux de Xavier Descombes [7], décrit une méthode permettant d'extraire les zones urbaines dans une image satellitaire, basée sur l'estimation d'un paramètre appelé température. Ensuite, elle segmente ce résultat, et présente aussi des résultats sur de la classification intra-urbaine. Cependant, je vais uniquement parler de la méthode pour décrire les régions urbaines, et je ne parlerai pas de la segmentation, car je ne m'en sers pas.

Anne Lorette, pour extraire son paramètre de texture, utilise un même modèle markovien en 1D, (qui prend en compte une direction spécifique) dans huit directions différentes, pour obtenir huit valeurs. Ensuite, elle combine ces valeurs pour ne donner qu'une seule valeur finale. Je vais donc décrire sa méthode en trois parties.

- Dans un premier temps, je décrirai le modèle unidirectionnel utilisé.
- Ensuite, je présenterai la méthode d'estimation des paramètres de ce modèle.
- Pour terminer, je montrerai comment Anne Lorette combine les informations données par huit versions différentes, une version par direction, pour extraire son paramètre final d'indice urbain.

Le modèle

Le paramètre de base que l'on souhaite extraire, la température, notée T, est fondé sur un modèle markovien. Le but est de construire des modèles tels que les zones urbaines, qui sont des zones à forte variance et peu corrélées, aient une température plus élevée que les zones de champs par exemple.

On définit le voisinage comme étant composé de deux voisins. Cela revient à choisir une direction privilégiée d ; le voisinage est alors constitué du pixel avant et du pixel arrière (voir Figure 3.2). On définit la

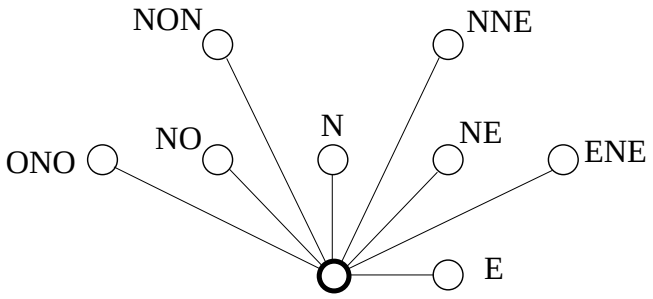


FIG. 3.3 – Les huit directions posibles

probabilité conditionnelle pour le modèle sur son voisinage par

$$\begin{aligned} P(X_s \mid \{X_r \in V^{(d)}(s)\}) &= P(X_s \mid m_s^{(d)}) \\ &= \frac{1}{Z_{V^{(d)}(s)}} \exp -\beta^{(d)} \left(\sum_{r \in V^{(d)}(s)} (X_s - X_r)^2 + \lambda^{(d)} (X_s - \mu)^2 \right) \\ &\equiv N\left(\frac{2m_s^{(d)} + \mu\lambda^{(d)}}{2 + \lambda^{(d)}}, \frac{1}{2\beta^{(d)}(2 + \lambda^{(d)})} \right) \end{aligned}$$

où $m_s^{(d)}$ désigne la moyenne des deux voisins du site s, et $\beta = \frac{1}{T}$, μ et λ sont les paramètres du modèle. La direction, qui détermine le voisinage, est notée (d). La probabilité conditionnelle suit une loi normale.

Prenons comme exemple le modèle qui prend en compte les voisins de droite et de gauche, c'est-à-dire le site à l'est et celui à l'ouest. La direction choisie est alors (E/O). La probabilité conditionnelle locale s'écrit :

$$P\left(X_s | X_r \in V_s^{(d=O/E)}\right) = P\left(X_s | \frac{1}{2}(X(O) + X(E))\right)$$

Le paramètre que l'on va extraire est la variance conditionnelle :

$$\sigma^2 = \frac{1}{2\beta(2 + \lambda)}$$

Estimation des paramètres

Il s'agit d'estimer la variance σ^2.

Anne Lorette estime les paramètres de texture, les variances conditionnelles locales, en utilisant la méthode des «queues de comètes». Cette méthode, fondée sur l'estimation des matrices des probabilités conditionnelles, a été proposée pour la première fois par Xavier Descombes dans [7], puis détaillée dans [8, 9]. La méthode consiste à estimer la matrice des probabilités conditionnelles dans une fenêtre glissante, centrée sur le pixel X_s. Il faut que la taille de la fenêtre soit assez grande pour avoir des résultats fiables, et assez petite pour éviter les mélanges de texture. Pour chaque position de la fenêtre (donc pour chaque pixel), on estime la variance conditionnelle locale à partir de la matrice des probabilités conditionnelles.

Anne Lorette dispose ainsi d'un modèle qu'elle sait estimer, sur un voisinage restreint à deux voisins.

Paramètre final

On sait maintenant, grâce au modèle à deux voisins présenté ci-dessus, construire une valeur pour une direction donnée. Anne Lorette propose alors de construire les valeurs pour un modèle appliqué dans chacune des huit directions présentées dans la rose des directions (voir Figure 3.3).

Le voisinage $V_s^{(d)}$ d'un pixel s reste composé de ses deux plus proches voisins dans la direction d considérée, un voisin devant, un voisin derrière. On peut alors se représenter le modèle global comme un voisinage non standard (ni 4, ni 8 connexe) autour de chaque site, ce qui est une originalité intéressante. On construit en fait huit modèles 1D pour chaque pixel.

On rappelle que le but final est de trouver un indice de texture répondant fortement aux zones urbaines. Pour cela, Anne Lorette a choisi de construire huit températures pour chaque pixel, dans huit directions différentes. Cependant, comme les voisinages ne sont pas isotropes, elle introduit aussi une étape de normalisation, en pondérant la valeur de chaque modèle mono-directionel, pour prendre en compte les distances aux voisins, qui est une des valeurs suivantes : 1, $\sqrt{2}$, ou $\sqrt{5}$. On peut alors comparer les huit valeurs entre elles.

L'hypothèse de base est que la zone urbaine répond fortement au modèle présenté ci-dessus dans toutes les directions. Cependant, prendre le max des valeurs, la moyenne, ou le min, ne donne pas de résultat satisfaisant. En fait, les huit valeurs extraites étant plus ou moins grandes, et sensibles à d'autres éléments que la ville, Anne Lorette a proposé de ne garder comme valeur finale que la moyenne des valeurs médianes, ie $\frac{1}{2}(p_4 + p_5)$, où les valeurs $p_i, i \in 1 \dots 8$ sont les paramètres extraits, triés par ordre croissant de valeur. Elle obtient ainsi une carte de valeurs, où il n'y a qu'un indice de texture urbaine pour chaque pixel.

Les images de valeurs d'Anne Lorette représentent une fonction qui, à chaque pixel, associe un réel positif. L'image de cette fonction est un candidat possible à l'entrée de l'algorithme de rapport de coût sur les graphes. En combinant les deux méthodes, on espère trouver les différentes régions de l'image originale, qui devraient correspondre à des villes. On tirera profit du fait que la réponse est construite pour que l'intensité la plus forte soit située sur les zones urbaines.

3.3 Méthodes variationelles

Je vais maintenant présenter les méthodes variationnelles. Dans une première partie, je m'attacherai à donner le cadre exact du problème à résoudre. Je donnerai ensuite deux méthodes employées habituellement pour faire varier les contours. Ensuite, je présenterai quelques méthodes utilisées pour accélérer les différents calculs. Pour finir, je montrerai quelle a été l'approche que j'ai employée pour calculer la force, dépendant d'une énergie sous forme de rapport de coûts.

3.3.1 Cadre

Afin de réaliser une segmentation d'image, on souhaite trouver une courbe Γ qui détermine une région Ω dans une image. Cette courbe doit répondre à des critères, qui sont les éléments donnés par la modélisation du phénomène. Pour la segmentation basée sur les contours, on considérera commme meilleure une courbe qui passe aux pixels dotés de forts gradients, tandis que pour la segmentation basée sur la texture, on pourra considérer un critère basé sur une région autour de chaque pixel.

Pour déterminer la courbe "meilleure" que les autres, on peut donner un critère de qualité à chaque courbe possible, et tenter de maximiser ce critère. La question qui se pose est alors : comment trouver cette courbe, plus satisfaisante qu'une autre ? On se propose de le faire en faisant évoluer au cours du temps une courbe initiale. On calcule la qualité de la courbe courante, puis on lui applique une transformation, en espérant que la courbe suivante est plus proche de l'objectif que l'on s'est fixé. Dans le cadre des méthodes variationnelles, cette évolution est déterminée en chaque point du contour par une force dépendant de la modélisation des phénomènes recherchés, et fonction de la localisation de la courbe sur l'image. Si l'on garde l'exemple de la segmentation basée sur le gradient, on fera bouger un point situé sur un faible gradient, tandis qu'un point sur un fort gradient subira une force beaucoup moins importante.

On se trouve avec plusieurs points difficiles, qui doivent être résolus :
- Expression du critère de qualité pour chaque courbe possible
- Discrétisation d'un schéma continu (les modèles sont habituellement continus, tandis que l'on se trouve en présence d'images discrètes)
- Expression de la force qui pousse chaque point de la courbe
- construction de l'évolution de la courbe, en fonction de la force
- Critère d'arrêt de l'évolution

L'expression du critère de qualité, et sa dérivation sous forme de force en chaque point, est dépendant du

problème que l'on souhaite résoudre. On présentera plutôt dans cette partie comment construire les courbes à partir d'une courbe initiale. Les deux méthodes principales seront présentées, nommément "**B**oundary **V**alue **P**roblem" et "**I**nitial **V**alue **P**roblem". La discrétisatisation sera alors évoquée, puis je terminerai ce chapitre par quelques accélérations possibles, au niveau algorithmique.

3.3.2 "Boundary Value Problem"

Si la force qui déplace la courbe est de signe constant, on peut définir le problème comme le temps d'arrivée en chaque point de l'onde de propagation du phénomème. On peut voir cela comme une onde sismique, qui donne un temps d'arrivée aux points du plan, à partir d'une courbe de départ initial, qui peut être vue comme la zone de départ du feu, selon une autre analogie. On modélise alors le problème par un ensemble $Init$ de points de départ, tels que $T(Init) = 0$, et une équation d'évolution

$$|\nabla T|\, F = 1 \tag{3.8}$$

Le temps d'arrivée au point p est $T(p)$. Si l'on s'intéresse à la courbe formée par l'onde au temps t, il faut isoler la courbe $\Gamma(t) = \{p \in \Omega \mid T(p) = t\}$.

Malgré la limitation du signe constant de la force F, cette approche est intéressante car elle admet un algorithme de résolution très rapide : "Fast Marching" (voir 3.3.6). On s'en sert pour calculer une fonction distance, lorsque $F = 1$, la fonction distance étant une fonction associant un réel à chaque point du plan, la distance de ce point au bord de la courbe initiale.

3.3.3 "Initial Value Problem"

On peut changer la perspective du problème en rajoutant une dimension. La courbe, que l'on fait évoluer, peut être vue comme le niveau 0 d'une surface de dimension supérieure. Cette surface évolue avec le temps, et la courbe évolue alors comme le niveau 0 de cette surface. Plus formellement, on se donne une surface $\Phi : \Omega \times \mathbb{R} \to \mathbb{R}$, et on définit à chaque instant t la courbe sur l'image comme étant l'ensemble des points $\Gamma(t) = \{p \in \Omega \mid \Phi(p, t) = 0\}$.

On souhaite alors obtenir une équation d'évolution, qui nous décrit précisément la surface Φ. En effet, on ne dispose, au départ, que de la courbe de niveau 0 initiale, $\{p \in \Omega \mid \Phi(p, 0) = 0\}$, ainsi que de la force qui pousse la courbe. On doit donc construire les valeurs de Φ pour les points qui ne font pas partie de la courbe, puis étendre la fonction dans le temps.

Lorsque l'on souhaite définir Φ sur tout l'ensemble Ω, on ne dispose que de la position du niveau zéro. On peut étendre les valeurs de Φ_0 à l'ensemble Ω tout entier en utilisant l'algorithme "Fast Marching" (voir 3.3.6), qui est parfaitement adapté à cette tâche, en résolvant $|\nabla\Phi| = 1$, ce qui correspond à construire la distance euclidienne.

Pour l'extension dans le temps, considérons la trajectoire $\xi(t)$ d'un point de la courbe. Par trajectoire d'un point, on entend la suite des positions successives au cours du temps de ce point, qui est toujours un point de la courbe du niveau 0 de Φ. Par construction,

$$\Phi\left(\xi(t), t\right) = 0$$

En dérivant cette expression, on obtient

$$\Phi_t + \nabla\Phi(\xi(t), t) \cdot \xi'(t) = 0$$

En posant

$$F = \xi'(t) \cdot n \quad \text{et} \quad n = \frac{\nabla\Phi}{|\nabla\Phi|}$$

on se retrouve avec la formulation de Osher et Sethian [25] :

$$\Phi_t + F|\nabla\Phi| = 0 \tag{3.9}$$

la fonction initiale $\Phi(p,0)$ étant donnée.

Il reste à expliquer comment résoudre ce problème dans le cas discret, et quelles sont les améliorations possibles.

3.3.4 Schéma de discrétisation

Dans la suite, la notation f_{ij}^n correspond à $f(i,j,n)$.

On s'est posé des problèmes à résoudre dans le domaine continu, en les écrivant comme des équations aux dérivées partielles. Cependant, on ne peut pas travailler en continu pour la résolution, par manque de moyens techniques, et aussi parce que la donnée initiale (une image), est déjà un objet discret. On s'intéresse donc à des méthodes d'analyse numérique, qui proposent des solutions approchées et discrètes au problème continu. Par exemple, on propose le calcul de la fonction Φ, qui dépend du temps, en discrétisant le temps (on calcule Φ^n, où $n \in \{t_1, t_2, \ldots\}$ décrit le temps), ainsi qu'en prenant une grille de points représentant l'image : au lieu de calculer $\Phi(a,n)$, pour $a \in \Omega$, on se contente de calculer $\Phi(i\Delta x, j\Delta x, n)$, où $(i,j) \in \mathbb{N}^2$.

Dans cette partie numérique, je ne présente que les équations qui permettent de résoudre les équations différentielles en 2 dimensions. Le lecteur se référera à [33] pour l'extension des calculs à d'autres espaces.

On a d'abord besoin de calculer plusieurs dérivées. Comme l'information autour du pixel qui nous intéresse n'est pas toujours disponible, il faut ruser pour trouver la dérivée qui donnera le moins d'erreur.

$$D^{+x}u \equiv \frac{u(x+\Delta x,y,t)-u(x,y,t)}{h} \tag{3.10}$$

$$D^{-x}u \equiv \frac{u(x,y,t)-u(x-\Delta x,y,t)}{h} \tag{3.11}$$

$$D^{+y}u \equiv \frac{u(x,y+\Delta y,t)-u(x,y,t)}{h} \tag{3.12}$$

$$D^{-y}u \equiv \frac{u(x,y,t)-u(x,y-\Delta y,t)}{h} \tag{3.13}$$

Ces définitions de dérivée permettent de calculer les évolutions des points, en ne se basant que sur de l'information dont on est sûrs. Par exemple, si le front "avance", on voudra utiliser une écriture "backward", (D^-), aussi appelé "upwind scheme".

3.3.5 "Extension Velocities"

Cette méthode est présentée dans le livre de Adalsteinsson et Sethian [1]. On souhaite étendre la force F de l'équation 3.9 à une force définie sur tout le domaine Ω. En effet, il est possible que F ne soit significative que sur le contour. Par exemple, si la force en chaque point du contour dépend d'une intégrale sur ce contour, comment doit-on définir la force ailleurs ? On a, cependant, besoin de la force sur tout le domaine pour calculer les variations de Φ en chaque point, pas uniquement sur la courbe de niveau 0. On doit donc calculer une extension F_{ext} définie sur l'ensemble Ω qui vérifie

$$\begin{cases} F_{ext} = F \text{ aux points vérifiant } \Phi = 0 \\ \forall a \in \Omega, \Phi(a) = 0 \Rightarrow \lim_{x \to a} F_{ext}(x) = F(a) \end{cases} \tag{3.14}$$

On se propose de construire la fonction F_{ext} telle que si la fonction Φ vérifie $|\nabla \Phi| = 1$, alors calculer la fonction Φ à l'instant d'après ne perturbe pas ce gradient unitaire. Il faut résoudre

$$\nabla F_{ext} \cdot \nabla \Phi = 0 \tag{3.15}$$

La stratégie est alors la suivante :

- On se donne une fonction $\Phi(\cdot,t)$, t un instant donné, dont on n'utilise qu'une petite partie : la position du niveau 0.

– Construire une distance $\Phi^{temp}(\cdot, t)$ autour du niveau zéro donné précédemment. On souhaite résoudre

$$\begin{cases} \forall a \in \Omega, \Phi(a,t) = 0 \Rightarrow \Phi^{temp}(a,t) = 0 \\ |\nabla \Phi^{temp}| = 1 \end{cases} \tag{3.16}$$

La méthode "Fast Marching" (voir 3.3.6) est bien appropriée.
– Construire F_{ext} grâce à l'équation (3.15).
– On peut alors construire la fonction $\Phi(\cdot, t + \Delta t)$ par l'équation aux dérivée partielles (3.9).

On a ainsi une méthode pour construire la version discrétisée de la fonction Φ sur tout son domaine. Ce calcul a été rendu possible par l'extension d'une force donnée uniquement sur le contour, force que l'on a alors étendu à tout le domaine Ω.

3.3.6 "Fast Marching"

Cette méthode algorithmique est aussi décrite dans le livre de Sethian [33]. On souhaite résoudre l'équation Eikonale (3.8). On peut utiliser l'approximation numérique suivante :

$$\sqrt{max(D_{ij}^{-x}T, -D_{ij}^{+x}T, 0)^2 + max(D_{ij}^{-y}T, -D_{ij}^{+y}T, 0)^2} = \frac{1}{F_{ij}} \tag{3.17}$$

La clé de l'algorithme de "Fast Marching" réside dans le tri des valeurs à mettre à jour. En effet, comme on utilise une méthode "upwind", il suffit de propager l'information des petites valeurs aux grandes. Cela nous permettra d'être certains de ne pas avoir besoin de revenir sur des valeurs déjà calculées. On peut classer les valeurs des pixels, selon que les valeurs sont acceptées, à l'essai, ou inconnues. Plus précisement, on applique l'algorithme suivant :

```
FAST MARCHING()
 1  ACCEPTE = courbe initiale
 2  ESSAI = voisins(ACCEPTE)
 3  INCONNU = Ω − ESSAI − ACCEPTE
 4  tant que ESSAI ≠ ∅ faire
 5        Choisir a ∈ ESSAI le point de valeur T minimum
 6        ACCEPTE = ACCEPTE ∪ a
 7        ESSAI = ESSAI − a
 8        pour tous les voisins v de a faire
 9              si v ∉ ACCEPTE alors
10                 resoudre (3.8) pour v
11                 si v ∉ ESSAI alors
12                    ESSAI = ESSAI ∪ v
13                    INCONNU = INCONNU − v
```

On voit que la clé de l'efficacité de cet algorithme réside dans la recherche du pixel avec T minimal. Cela peut se faire en utilisant un tas. Le tas considéré est un arbre binaire complet, où le père est toujours de valeur T inférieure à ses fils. On insere et enlève des éléments à cette structure en $O(M)$, où M est le nombre d'éléments dans le tas. On voit donc que visiter chaque pixel une fois suffit pour calculer son temps d'arrivée.

On peut mettre cet algorithme en parallèle avec l'algorithme de Dijkstra. Ils sont basés sur la même idée, qui consiste à fixer la plus petite valeur, puis mettre à jour ses voisins. Cependant, "Fast Marching" garde le principe de l'équation aux dérivées partielles sous-jacente, ce que ne permet pas l'algorithme de Dijkstra.

3.3.7 Discrétisation du "Initial Value Formulation"

Sethian propose dans son livre [33] d'utiliser le schéma suivant :

$$\Phi_{ij}^{n+1} = \Phi_{ij}^{n} - \Delta t[max(F_{ij}, 0)\nabla^{+} + min(F_{ij}, 0)\nabla^{-}] \tag{3.18}$$

où on a pris

$$\nabla^{+} = \sqrt{max(D_{ij}^{-x}, 0)^2 + min(D_{ij}^{+x}, 0)^2 + max(D_{ij}^{-y}, 0)^2 + min(D_{ij}^{+y}, 0)^2} \tag{3.19}$$

$$\nabla^- = \sqrt{max(D_{ij}^{+x},0)^2 + min(D_{ij}^{-x},0)^2 + max(D_{ij}^{+y},0)^2 + min(D_{ij}^{-y},0)^2} \tag{3.20}$$

Pour simplifier, j'ai laissé les opérateurs D sans les appliquer à la fonction. Cette écriture est basée sur le raccourci $D_{ij}^{\bullet} \equiv D^{\bullet}\Phi_{ij}^{n}$.

Comme on ne veut pas que le front de propagation traverse plus de deux cases à la fois, on impose que

$$\max_{\Omega} F\Delta t \leq \Delta x \tag{3.21}$$

Ceci doit être vérifié en tous les points que l'on met à jour, pas uniquement sur le niveau 0. On pourra, dans la mise en œuvre, vérifier rapidement en parcourant les pixels que cette condition est bien vérifiée. Sinon, on choisira un échantillonage du temps plus petit.

3.3.8 Bande étroite

Cette idée pour obtenir une vitesse de calcul plus grande est appelée "Narrow band" en anglais. Le but est d'accélérer le calcul itératif de la fonction Φ. On constate que, pour chaque temps t, on met à jour chaque point, y compris ceux situés très loin de la courbe de niveau 0. L'idée consiste donc à construire une bande autour de la courbe de niveau 0, et d'ignorer les points qui n'en font pas partie, en considérant qu'ils sont trop éloignés.

Plus précisement, on gèle les valeurs au bord de la bande, en indiquant qu'il faut arrêter les itérations lorsqu'un de ces pixels est atteint. Cela se produit lorsque la courbe de niveau 0 passe par un de ces points. Il faut alors reconstruire la bande (réinitialiser). On recalcule les valeurs de F pour la nouvelle bande, bande qui a été reconstruite autour de la dernière position connue du niveau 0, lorsqu'un pixel gelé a été atteint.

On doit choisir la largeur de la bande. Une largeur infinie correspond à prendre toute l'image, on met à jour tous les pixels, le processus n'est donc pas accéléré. A l'opposé, une largeur d'un pixel oblige à réinitialiser à chaque itération. Un bon compromis, dicté par l'expérience, consiste à prendre une largeur de 6 pixels autour de la courbe de niveau zéro.

Une variante à cette méthode consiste à choisir la largeur de la bande en fonction de la valeur de la force F. Ainsi, si la force locale est faible, la courbe ne bougera pas vite, on peut donc utiliser une largeur étroite, tandis que lorsque la force est localement importante, la courbe bougera plus vite, il faut inclure plus de points, donc faire un bande plus large autour de ce point-là.

Une autre variante propose d'ajouter des points à la bande, lorsque les points sur le bord passent sous un certain seuil ($|\Phi| < \theta_1$). Symétriquement, on enlève les points qui ont une valeur trop grande ($|\Phi| > \theta_2$).

3.3.9 Energie comme rapport d'énergie

On se rappelle que l'on souhaite trouver la force gouvernant l'évolution de la courbe, ceci à partir de la seule donnée de l'énergie. L'énergie décrit la qualité de la courbe considérée. A énergie faible (voir négative), "bonne" courbe.

Pour notre travail, nous introduirons l'énergie à minimiser sous la forme d'un rapport d'énergie. Cette approche est novatrice, car habituellement, plusieurs énergies sont combinées linéairement, avec l'introduction d'un paramètre. En utilisant le rapport, on peut mettre en concurrence deux énergies sans devoir attribuer de coefficient de pondération.

Pour retrouver l'évolution à partir de l'énergie, la méthode habituelle consiste à dériver l'énergie, pour obtenir les équations d'évolution :

$$E(\gamma) = \int F[\gamma, I]$$

$$\frac{\delta E}{\delta \gamma(t)} = G[\gamma(t), \gamma'(t), \gamma''(t), I]$$

Cette dérivée nous indique dans quel sens la modification est la plus grande. En effet, dans la direction du gradient, la variation d'énergie est la plus importante. On a donc les éléments nécessaires pour faire une descente de gradient.

La variation d'énergie ne dépend habituellement que d'éléments locaux comme la valeur des quelques pixels sous-jacents, de la tangente à la courbe, ou du rayon de courbure. Par contre, dans le cadre de rapports d'énergies

$$E(\gamma) = \frac{E_1(\gamma)}{E_2(\gamma)} ,$$

on se retrouve dans l'équation d'évolution avec des termes dépendant de tous les points de la région, ainsi que d'autres dépendant du contour. Cela va présenter une difficulté supplémentaire, et surtout un temps de calcul accru. Avant de montrer comment on dérive l'énergie, pour obtenir l'évolution du contour, je présente deux références concernant des énergies qui ont la même difficulté de nécessité d'un calcul global pour les déplacements locaux.

Dans le cadre des intégrales sur toute la région, Jehan-Besson et al. ont proposé [13] des énergies tenant compte de l'intérieur de la région (moyenne). Ce genre d'information pourrait être inclu dans le numérateur d'une énergie.

On pourra aussi se référer aux travaux de Marie Rochery [31]. Elle propose de regarder des énergies quadratiques, c'est-à-dire s'écrivant sous forme d'intégrale double sur le contour. Elle obtient alors une force qui s'écrit comme un intégrale sur le contour, et qui dépend de l'interaction entre le point courant et tous les autres points du contour.

Dérivation de l'énergie

On pourra trouver des justifications dans l'article [13].
Notations :
- N est le vecteur orthogonal au contour Γ, dirigé vers l'intérieur de la courbe,
- v est la vitesse de la courbe au point considéré.
- κ est le rayon de courbure au point de la courbe,

$$\kappa = \frac{x'y'' - y'x''}{(x'^2 + y'^2)^{3/2}}$$

On part de

$$E = \frac{E_1}{E_2} \tag{3.22}$$

La dérivée de cette fraction est (on note $\frac{\partial E}{\partial \tau} = E'$)

$$E' = \frac{E_1' \cdot E_2 - E_1 \cdot E_2'}{(E_2)^2} \tag{3.23}$$

Si les énergies s'écrivent sous forme d'intégrales sur la région et sur le contour

$$E_1 = \iint_\Omega f(x,y)\mathrm{d}x\mathrm{d}y \tag{3.24}$$

$$E_2 = \int_\Gamma g(s)\mathrm{d}s \tag{3.25}$$

leurs dérivées s'écrivent

$$E_1' = \iint_\Omega f' - \int_\Gamma f(v \cdot N) \tag{3.26}$$

$$E_2' = \int_\Gamma (-g\kappa + \nabla g \cdot N)(v \cdot N) \tag{3.27}$$

(le premier terme de E_1'est nul, car la fonction f ne dépend pas du temps τ). On peut alors écrire l'équation d'évolution sous la forme

$$E' = \frac{-\int_\Gamma g \int_\Gamma f(v \cdot N) - \iint_\Omega f \int_\Gamma (-g\kappa + \nabla g \cdot N)(v \cdot N)}{(\int_\Gamma g)^2} \tag{3.28}$$

On le réécrit en factorisant le terme $(v \cdot N)$ et l'intégrale sur le contour :

$$E' = -\frac{\int_\Gamma \left((\int_\Gamma g) f + (\iint_\Omega f)(-g\kappa + \nabla g \cdot N) \right) (v \cdot N)}{(\int_\Gamma g)^2} \tag{3.29}$$

Puis on se rappelle que le but est de trouver v comme étant le vecteur qui maximise Γ'. La partie d'évolution du contour est donc le terme de l'intégrale, moins le $(v \cdot N)$:

$$\boxed{\Gamma' = \frac{(\int_\Gamma g) f + (\iint_\Omega f)(-g\kappa + \nabla g \cdot N)}{(\int_\Gamma g)^2}} \tag{3.30}$$

On remarque que comme annoncé précédement, on garde des intégrales dans la force d'évolution du contour. On pourra cependant se contenter de calculer ces quantité qu'une seule fois par contour, et éviter de les recalculer lors du calcul de la force en chaque point.

Chapitre 4

Application et comparaison

Les deux techniques, à savoir la méthode employant les graphes, et celle utilisant les méthodes variationnelles, ont été implantées lors du stage. Je présenterai dans cette partie quelques détails techniques concernant l'implantation proprement dite, avant de passer à la comparaison des méthodes. J'ai employé les mêmes images pour pouvoir avoir une base solide de comparaison.

On rappelle que l'on utilise l'algorithme de rapport de coûts pour extraire des régions d'une image, la recherche étant dans le choix de fonctions pour décrire les comportements que l'on souhaite voir ressortir de l'image. Ceci est valable dans les deux cas de figure, que ce soit pour le variationnel ou le graphe.

4.1 Mise en œuvre des graphes

On part d'un problème sur une image, à savoir en extraire une région satisfaisant le minimum d'une énergie donnée sous forme de rapport de coût. On doit alors plonger le problème sur un bi-graphe représentant l'image originale, résoudre le problème et calquer le résultat sur l'image.

Il est facile de projeter des images directement sur une grille : on construit un graphe dont les sommets sont les coins des pixels, et les arêtes les bords des pixels. Les deux fonctions f et g sont ici utilisées pour construire les poids τ, et λ respectivement, poids attribués aux arêtes du bi-graphe. On se référera à la partie 3.1.4 ainsi qu'aux équations 3.3 et 3.5 pour la méthode employée.

On intègre la valeur du niveau de gris des pixels pour construire la fonction λ, tandis que la fonction τ est la distance entre deux noeuds (en 8-connexité, $\tau \in \{1, \sqrt{2}\}$). Lorsque que l'arête se trouve sur le contour d'une région, on souhaite qu'elle appartienne au cycle qui sera finalement extrait. Il faut donc construire la fonction g telle qu'elle ait des petites valeurs sur un contour, tandis que la fonction f aura des grandes valeurs dans la région, de telle sorte que la somme de la fonction v sur le cycle prenne une grande valeur.

J'ai écrit intégralement le code réalisant ces opérations en JAVA. Le choix de ce langage reflète une envie de m'en servir pour développer entièrement une application. On pourrait aussi citer les arguments habituels qui sont la portabilité et les librairies standard. Cependant, j'ai été très déçu par la vitesse d'exécution, qui met parfois plus d'une demi-heure[1], sur des images de dimensions réduites (100x100).

4.2 Mise en œuvre de l'approche variationnelle

Pour cette approche, je bénéficiais de tout un cadre théorique solidement bâti et bien connu, la seule chose à modifier étant l'expression de l'énergie, d'après l'équation 3.30. J'ai donc utilisé un code sur les "level sets" déjà écrit, en n'ayant que l'expression de la force à changer. Je me suis servi du code C++ de Marie Rochery. Une toute petite étape de nettoyage a été nécessaire (son code, utilisé pour des énergies quadratiques, contient des calculs d'intégrales sur le contour entier pour chaque point du contour - cf [31] pour plus de détails).

[1] Les temps sont mesurés sur un processeur à 2GHz

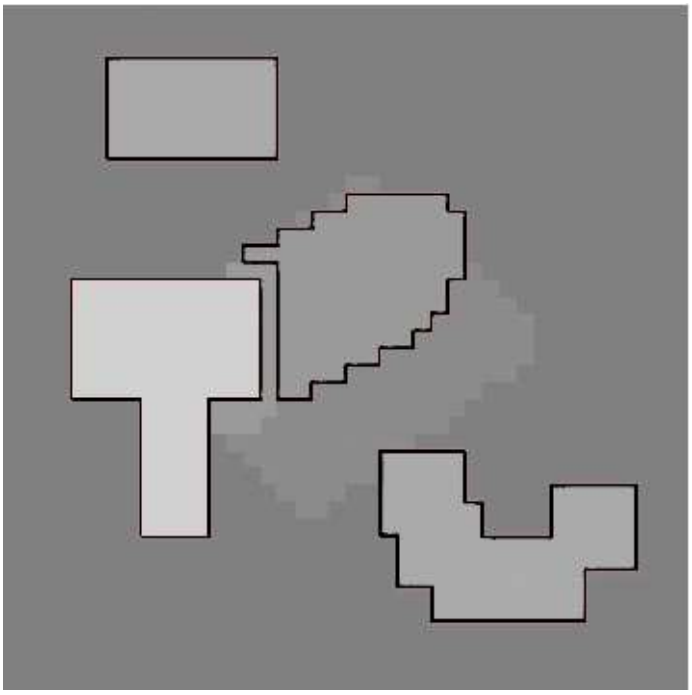


FIG. 4.1 – Extraction de régions

En partant d'une fonction Φ, son code permet de
- retrouver les points du contour, et la région associée
- calculer les quantités nécessaires au calcul de la force (courbure, vecteur normal, ...)
- $\star$ calculer la norme de la force à appliquer, pour chaque point du contour
- étendre cette force définie sur le contour à tous les points du plan
- faire évoluer la fonction Φ selon la force étendue

En initialisant la fonction Φ, par exemple par une ellipse, on obtient le comportement d'un contour actif habituel. La partie qui demandait une réelle modification a été ($\star$). Comme je n'ai eu finalement que très peu de modifications à faire, j'ai pu très vite commencer à étudier le comportement de la force que l'on propose.

4.3 Comparaison sur des images synthétiques

4.3.1 Méthode sur les graphes

On obtient de bons résultats (voir figure (4.1) en prenant

$$f = |\nabla I| \qquad g = 1$$

où $|\nabla I|$ est le gradient de l'intensité de l'image synthétique I de départ. En effet, cela correspond à attribuer une forte valeur aux contours. La région à l'intérieur des contours est donc choisie car inclure les bords augmente la valeur absolue de l'énergie, tandis qu'aller au-delà augmente la longueur sans ajouter au numérateur, ce qui a pour résultat de faire diminuer la valeur absolue. Prendre le gradient pour f correspond à avoir pris comme fonction initiale le Laplacien, ΔI. En effet, cela vient de l'équation 3.4 qui lie la fonction f au champ de vecteur v.

On constate que la méthode marche bien pour extraire les contours, ce qui est le résultat attendu (voir figure (4.1)). Dans cet exemple, l'algorithme a été executé plusieurs fois, afin d'extraire plusieurs objets de l'image.

De plus, on voit que le modèle est bien adapté aux régions en longueur. Par exemple, les queues ne sont

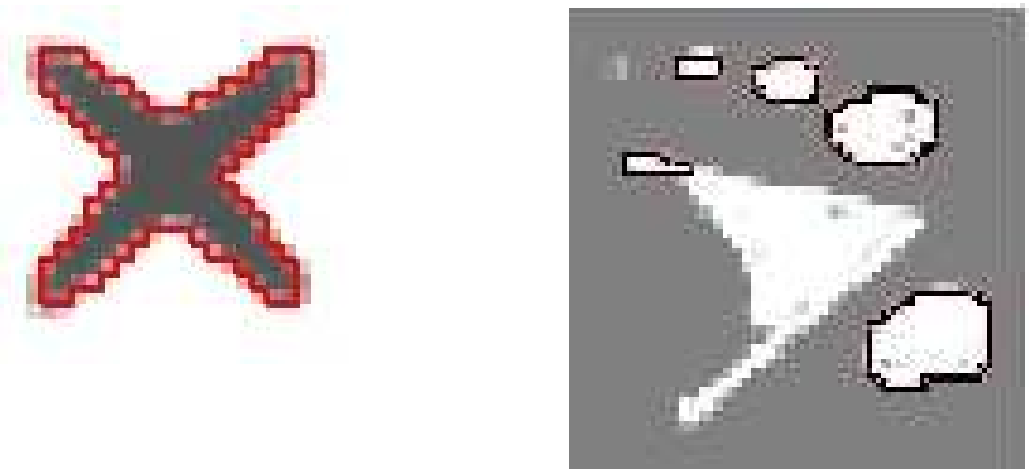


FIG. 4.2 – Extraction des régions, bords flous

FIG. 4.3 – Original et remplissage du chat

pas coupées, et font partie de la région extraite de l'image 4.4. Même lorsque les contours ne sont pas très clairs, l'algorithme prend quand même la partie importante de la région : le résultat visuel est satisfaisant (figure 4.2).

4.3.2 Méthode variationnelle

On a d'abord pris les mêmes fonctions :

$$f = -|\Delta I| \qquad g = 1$$

Ici encore, on favorise les régions où le gradient est important, en ajoutant un terme de longueur pour avoir des régions au périmètre raisonnable. Cependant, on a ici un effet étonnant : toutes les zones concaves d'une image sont gommées. Ceci s'explique par le fait que les zones sans gradient, ie les zones plus lisses, ne sont pas pénalisées, et contribuent de 0 à la double intégrale sur la région. Donc sur cette intégrale, rajouter ou non des pixels blancs ne change strictement rien. Cepenadnt, il reste le terme du dénominateur. Celui-ci tend vers les régions à court périmètre. Or les concavités rajoutent de la longueur, tandis que le ligne droite qui traverse la zone concave est bien plus courte. C'est l'effet qu'on peut observer sur l'exemple du chat (voir Figure 4.3) : les zones au-dessus de la queue, et autour des moustaches, sont comblées. J'insiste sur le fait que ce résultat vient de la méthode, qui ne trouve qu'un minimum local. Dans ce cas-ci, l'énergie doit d'abord augmenter, en atteignant les bords de la concavité, avant de décroître, et de dépasser la valeur précédente, lorsque tout le contour est bien extrait.

La construction brutale de la méthode variationnelle n'est donc pas satisfaisante. On essaie donc de dévaloriser (avec un paramètre à déterminer) les zones extérieures, en affectant un poids positif aux pixels de dehors (Figure 4.4).

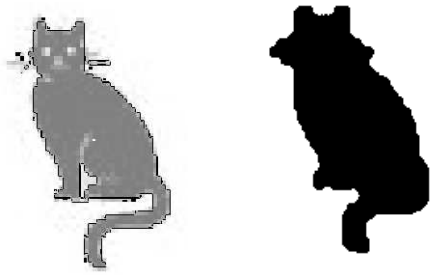


FIG. 4.4 – Méthode du graphe et variationnelle

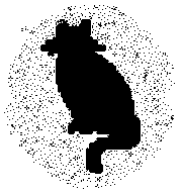


FIG. 4.5 – Artefact pour α trop important

$$f = \begin{cases} \alpha & \text{si } \nabla I = 0 \\ -|\Delta I| & \text{sinon} \end{cases} \qquad g = 1$$

Le paramètre α vient ici pondérer la contribution de rejet des pixels que l'on peut considérer comme mauvais. Il décrit à quel point un pixel de l'extérieur est à exclure. Si ce paramètre est trop fort (par exemple 100 fois le gradient maximal), on obtient des artefacts (Figure 4.5). S'il n'est pas assez important, on corrige, mais pas assez, l'effet de remplissage des concavités. On voit bien cet effet sur la Figure 4.4 où la queue n'est pas bien segmentée. Il faut donc trouver la juste mesure. Une étude du comportement de l'algorithme serait bienvenue, et permettrait sûrement de déterminer automatiquement ce paramètre, en fonction de l'image d'entrée.

4.4 Application à des images de télédétection

4.4.1 Données brutes

Si l'on entre directement dans l'algorithme les valeurs des variances conditionnelles données par la méthode d'Anne Lorette (que l'on notera Φ), la région qui est invariablement trouvée est l'image entière. En effet, les valeurs des pixels sont trop grandes par rapport à la longueur d'une arête du graphe. Comme le modèle initial, qui ne comporte pas de paramètre, ne permet pas de trouver de région dans les données fournies, on est amenés à modifier la fonction de projection sur le graphe. On introduit des modèles plus compliqués, par exemple incluant le gradient, pour espérer extraire les zones urbaines. Les premiers essais ont porté sur la soustraction d'une constante, ce qui revient à recentrer les données, et sur une modification du dénominateur, en donnant à chaque arête une longueur dépendant du gradient sous-jacent.

4.4.2 Ajout d'une constante

On se rappele que le modèle est invariant à la multiplication par un scalaire, mais pas à l'addition : les résultats obtenus avec la fonction f ne sont pas forcément les mêmes que ceux obtenus avec $f + \alpha$, pour $\alpha \in R$. La première idée pour obtenir des resultats sur la sortie d'Anne Lorette est donc de rajouter une constante. On obtient effectivement des résultats différents, avec le meilleur à l'oeil nu étant $f = \Phi - 450$. Mais cette valeur dépend de l'image, et une méthode pour déterminer ce paramètre devrait être envisagée pour utiliser cette méthode. Cette fonction trop simpliste a donc été rejettée, car bien qu'elle présente une

avancée par rapport aux données brutes, elle donnait des résultats encore trop loin de ce qu'on était en droit d'espérer, avec en plus l'ajout d'un paramètre.

Je présente quelques résultats (voir Figure 4.6) pour l'ajout d'une constante, car ils sont intéressants du point de vue du comportement de l'algorithme. Lorsqu'on ajoute 0 à la fonction, on se retrouve avec toute la région. On essaie donc avec d'autres valeurs, plus petites. On remarque que pour $\alpha < -400$ on commence à trouver une région. Il semble y avoir une région minimum vers -450, puis la région recommence à grandir, et finit par englober à nouveau toute l'image. En fait, comme l'ajout d'une constante recentre les données, on voit qu'il y a une "meilleure" valeur qui correspond à recentrer les valeurs, de telle manière que la contribution des points «ville» soit réellement plus importante que celle des points extra-urbains. En effet, comme la valeur des points extra-urbain n'est pas forcément nulle, ils contribuent quand même aux régions, et les prendre en compte n'est pas mauvais au regard de l'algorithme, sauf si justement on recentre les données. On remarque qu'ensuite enlever une trop grande valeur produit l'effet mirroir. Les valeurs que l'on ne souhaite pas voir sont à nouveau sollicitées, car en changeant l'orientation de la région, la valeur est grande à nouveau pour des pixels extra-urbains.

L'aspect très carré du résultat vient de l'utilisation de la 4-connexité. Cela permet d'avoir un contour qui respecte la structure des pixels sous-jacents de l'image, mais on perd certaines améliorations qui pourraient être obtenues en arrondissant les coins.

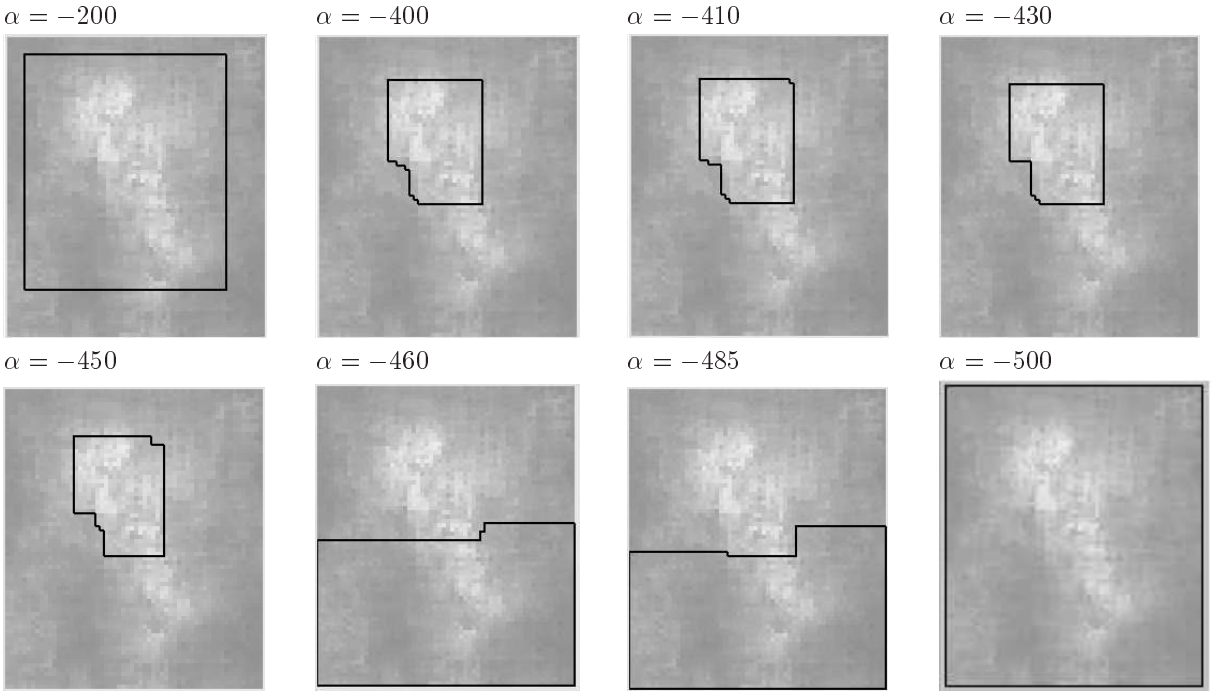


FIG. 4.6 – Différents valeurs de α, où $f = \Phi + \alpha$

4.4.3 Gradient de l'image

En utilisant non pas la valeur de Φ brute, mais le gradient de l'image ($f = |\nabla\Phi|$), on se retrouve avec des petites zones, car certaines régions (généralement au centre des villes) répondent très fortement. On n'obtient pas de plus grandes régions car la méthode n'est pas biaisée vers les régions de grande taille. On se retrouve plutôt avec les pics de la fonction Φ, qui sont entourés de forts gradients dans toutes les directions. Cette méthode n'est donc pas satisfaisante, car on ne recherche pas les centres des zones urbaines, mais les zones entières, en soulignant la frontière avec le non urbain (voir figure 4.7).

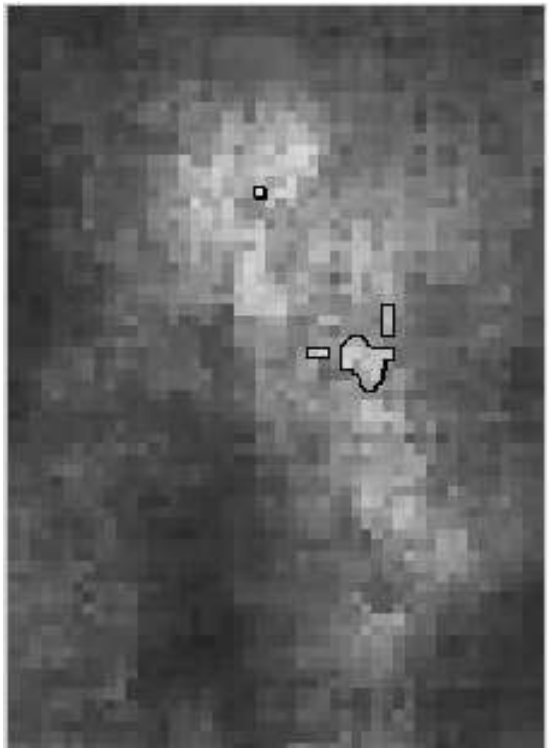


FIG. 4.7 – Une partie d'une image satellite, où on utilise uniquement le gradient de la fonction Φ pour extraire les zones urbaines

4.4.4 Longueur d'une arête en fonction du gradient

On souhaite avoir un poids d'arête important sur le contour. On essaie d'inclure cette contrainte dans le dénominateur. La fonction $g = 1$ agit comme un terme d'élasticité, en ressérant la courbe, mais ne tient pas compte du tout de l'image sous-jacente. Comme on cherche les bords de régions, et que sur le contour d'une région, le gradient a une valeur importante, on a choisi pour la fonction du dénominateur

$$g = \frac{1}{\beta + |\nabla I|} \qquad \begin{array}{l} \text{si } |\nabla I| \ll 1 \text{ alors } g \sim 1 \text{ et } \frac{\lambda}{\tau} \sim \lambda \\ \text{si } |\nabla I| \gg 1 \text{ alors } g \ll 1 \text{ et } \frac{\lambda}{\tau} \gg 1 \end{array}$$

La valeur de β change en fait la contribution relative d'un grand gradient par rapport à un petit gradient. Pour β petit (proche de 1 par exemple), il y aura une grande variation dans les valeurs de τ, tandis que pour β égal au gradient moyen, on n'aura qu'une variation du simple au triple. Cependant, on se retrouve avec à nouveau des paramètres à regler. Je ne présente pas de résultats pour cette méthode, car je n'ai pas encore eu le temps de bien regarder quels étaient les réglages qui convenaient le mieux.

Chapitre 5

Conclusion et perspectives

Nous avons proposé un survol de l'étude d'une énergie sous forme de rapport de coût dans ce stage de DEA. Premièrement, un algorithme de minimisation sur les graphes a été implémenté, pour être ensuite combiné à une analyse de texture pour obtenir une extraction de zones urbaines à partir d'images de télédétection. Ensuite, une approche variationnelle a été proposée, pour obtenir une comparaison des résultats. Cependant, les deux méthodes se heurtent à trop de simplicité dans la modélisation. Comme on ne propose qu'un cadre de développement d'énergie, sans s'attarder sur les énergies à employer, on constate que le problème a simplement été déplacé. L'approche permet d'envisager de nouvelles manières de combiner les informations, mais ne donne pas de solution miracle au problème très général de la segmentation.

Je vais donc présenter ici les différents points difficiles qui ont étés soulevés, en essayant de donner des pistes pour leur résolution.

5.1 Immobilité de la courbe variationnelle

Supposons qu'à un moment donné de l'évolution de la courbe, la région séléctionnée ait une intégrale qui vaille zéro : $\iint_R f = 0$. L'énergie est alors de zéro. Si un petit déplacement ne change rien de cette intégrale double, qui vaut toujours zéro, la courbe va alors cesser d'évoluer. On a un plat dans l'énergie, dont on n'arrive pas à sortir. On peut alors proposer de supprimer de tels cas de figure, où un plat empêche la courbe d'évoluer, en rajoutant un terme constant à chaque pixel de l'image. La fonction f contient alors un terme d'aire, et sa minimisation revient à choisir une region de petite aire (pour une maximisation de l'énergie, on ajouterait un terme négatif). Ce terme permet d'assurer la poursuite de l'évolution, mais introduis une contrainte sur la courbe qui n'est pas forcément nécessaire.

Ce problème n'apparaît pas avec la méthode sur les graphe car la solution trouvée est le minimum global, et on ne s'arrête donc jamais dans des puits, ou sur des plateaux de l'énergie.

5.2 Choix des fonctions f et g

Toute la partie modélisation du problème est reporté sur le choix de f et de g. Ces fonctions sont directement reliées au résultat final, et leur construction nécessite de bien poser le modèle que l'on souhaite implanter.

J'ai d'abord utilisé des fonctions très simples ($f = |\nabla I|$ et $g = 1$), gouvernant uniquement en fonction du gradient de l'image sous-jacente. Malgré des résultats encourageants sur des images de synthèse très simples, on se heurte très vite au problème de la sensibilité au bruit. En effet, un bruit impulsionnel créé une région minuscule à énergie très négative. On a alors plusieurs solutions qui viennent à l'esprit. On peut soit lisser l'image originale, ce qui aura pour effet d'étaler ces signaux impulsionnels. On peut aussi choisir d'exclure ces petits artefacts de la recherche de régions en configurant les fonction f et g pour n'accepter que des régions d'une certaine taille. Par exemple, rajouter une constante à la fonction f permet de rajouter un terme d'aire, ce qui a pour effet de privilégier les grandes régions. En contrebalançant cet effet par une fonction g tendant

vers des petites régions, on obtient un rapport que l'on peut biaiser plus ou moins en fonction de la taille espérée.

L'introduction de plus d'éléments dans les fonctions f et g permettrait sûrement de se rapprocher d'une solution à un problème spécifique, mais on risque alors de se limiter à une certiane classe de courbe, en perdant de vue certaines autres solutions moins évidentes, mais aussi correctes.

On a aussi vu que un des intérêts de la méthode rapport de coût, comparé à des méthodes plus standards, est la suppression d'un paramètre. Cependant, lors de notre étude, nous nous sommes rendus compte que cela créait des effets indésirables sur les solutions trouvées. On a alors été obligés de rajouter un paramètre pour supprimer la tendance à lisser les zones concaves.

5.3 Critère de qualité

Comme indiqué plus haut, on ne proposait pas une nouvelle solution à un problème donné. On proposait plutôt un nouveau cadre de développement de nouvelles énergies qui elles résoudrait des problème bien spécifiques. Or rien n'a été proposé dans ce cadre là, ce qui fait que choisir une énergie plus qu'une autre s'est fait sans réelle distance.

On se retrouve donc sans échelle pour mesurer la qualité de cette nouvelle approche. Il faudrait se pencher sur la recherche d'un problème qui pourrait avoir une meilleure solution en utilisant ce cadre, où la formulation rapport de coût prendrait tout son sens. On aurait ainsi la certitude d'une avancée, en proposant cette approche.

5.4 Dernier mot

Pour clore ce rapport, je dirai que l'approche rapport de coût présente des propriétés qui pourraient se révéler interessantes, mais que l'analyse s'en trouve compliquée. Une étude plus approfondie serait requise pour extraire précisement les caractéristique de cette approche.

Bibliographie

[1] Adalsteinsson, D., and Sethian, J. A. The fast construction of extension velocities in level set methods. *Journal of Computational Physics 148* (1999), 2–22.

[2] Ahuja, R., Magnanti, T., and Orlin, J. *Network flows.* Prentice Hall, 1993.

[3] Chellapa, R., and Jain, A., Eds. *Markov Random Fields - Theory And Application.* Academic Press, 1993.

[4] Cocquerez, J., and Phillip, S. *Analyse d'images : filtrage et segmentation.* Masson, 1995.

[5] Conners, R. W., Trivedi, M. M., and Harlow, C. A. Segmentation of high-resolution urban scene using texture operators. *CVGIP 25* (1984), 273–310.

[6] Cox, I. J., Rao, S. B., and Zhong, Y. Ratio regions : A technique for image segmentation. *Proceedings 13th International Conference on Pattern Recognition* (1996), 557–564.

[7] Descombes, X. *Champs markoviens an analyse d'image.* Thèse de doctorat, ENST, December 1993.

[8] Descombes, X., and Prêteux, F. Topology and parameter estimation in markov random field modeling. *SPIE, Neural and Stochastic Methods in Image and Signal Processing II* (1993), 156–166.

[9] Descombes, X., Sigelle, M., and Prêteux, F. Estimating Gaussian Markov random field parameters in a non-stationary framework : Application to remote sensing imaging. *IEEE Trans. on Image Processing 8*, 4 (1999), 490–503.

[10] Gondran, M., and Minoux, M. *Graphs and Algorithms.* Wiley interscience, 1988.

[11] Haralick, R., and Shapiro, L. *Computer and Robot Vision, volume II*, vol. 2. Addison-Wesley, 1992.

[12] Haralick, R. M., Shanmugam, K., and Dinstein, I. Textural features for image classification. *IEEE Trans. on Systems, Man, and Cybernetics 3*, 6 (1973), 610–621.

[13] Jehan-Besson, S., Barlaud, M., and Aubert, G. Dream2s : Deformable regions driven by an eulerian accurate minimization method for image and video segmentation. *International Journal of Computer Vision 53* (2003), 40–70.

[14] Jermyn, I., and Ishikawa, H. Globally optimal regions and boudaries as minimum ratio weight cycles. *IEEE Trans Pattern Analysis and Machine Intelligence 23*, 10 (2001), 1075–1088.

[15] Karp, R. A characterization of the minimum cycle mean in a digraph. *Discrete Math. 23* (1978), 93–112.

[16] Kass, M., Witkin, A., and Terzopoulos, D. Snakes : Active contour models. *International Journal of Computer Vision* (1988), 321–331.

[17] Lawler, E. L. Optimal cycles in doubly weighted linear graphs. *Proceedings of International Sympsosium on Theory of Graphs* (1966), 209–213.

[18] Laws, K. *Texture Image Segmentation.* PhD thesis, University of Southern California, Jan. 1980.

[19] Li, S. *Markov Random Field modeling in computer vision.* Springer-Verlag, 1995.

[20] Lorette, A. *Analyse de texture par méthodes markoviennes et par morphologie mathématique : application à l'analyse des zones urbaines sur des images satellitales.* Thèse de doctorat, Université de Nice - Sophia Antipolis, 1999.

[21] Lorette, A., Descombes, X., and Zerubia, J. Texture analysis through a markovian modelling and fuzzy classification : Application to urban area extraction from satellite images. *International Journal of Computer Vision 3*, 36 (2000), 221–236.

[22] Mallat, S. G. A theory of multiresolution signal decomposition : the wavelet representation. *IEEE trans. on pattern analysis and machine intelligence 11*, 7 (1989), 674–693.

[23] Manjunath, B. S., and Ma, W. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 18*, 8 (Aug 1996), 837–42.

[24] Meggido, N. Combinatorial optimization with rational objective function. *Math. Operations Research 4* (1979), 414–424.

[25] Osher, S., and Sethian, J. A. Fronts propagating with curvature dependent speed : Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics 79* (1988), 12–49.

[26] Paquerault, S. Traitement d'images aériennes : Reconaissance des zones urbaines sur des critères de texture. *rapport de DEA, ENST Paris* (1994).

[27] Pentland, A. P. Fractal-based description of natural scenes. *IEEE Trans. on pattern analysis and machine intelligence 6*, 6 (Nov. 1984), 661–672.

[28] Pérez, P. *Modèles et algorithmes pour l'analyse probabiliste des images.* Habilitation à diriger les recherches, Université de Rennes 1, December 2003.

[29] Reed, T., and du Buf, J. A Review of Recent Texture Segmentation and Feature Extraction Techniques. *CVGIP : Image Processing 57* (1993), 359–372.

[30] Reed, T., and Wechsler, H. Segmentation of textured images and gestalt organization using spatial/frequency representation. *IPAMI* (1990), 1–12.

[31] Rochery, M., Jermyn, I. H., and Zerubia, J. Étude d'une nouvelle classe de contours actifs pour la détection de routes dans des images de télédétection. In *GRETSI* (Paris, France, sep 2003).

[32] Serra, J. *Image Analysis and Mathematical Morphology*, vol. 1. Academic Press, 1982.

[33] Sethian, J. A. *Level Set Methods and Fast Marching Methods : Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision and Materials Science.* Cambridge University Press, 1999.

[34] Shi, J., and Malik, J. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (2000).

[35] Tuceryan, M., and Jain, A. K. Texture segmentation using voronoi polygons. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-12*, 2 (1990), 211–216.

[36] Tuceryan, M., and Jain, A. K. *Handbook of Pattern Recognition and Computer Vision.* Chen, Pau et Wang, Editeurs; World Scientific Company, 1993.

[37] Tuceryan, M., Jain, A. K., and Ahuja, N. Supervised classification of early perceptual structure in dot patterns. In *11th IAPR International Conference on Pattern Recognition* (The Hague, Sept. 1992), vol. II, IEEE Computer Society Press, Los Alamitos, CA,, pp. 88–91.

[38] Wang, S., and Siskind, J. M. Image segmentation with ratio cut - extended version. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (December 2002).

[39] Weszka, J., Dyer, C., and Rosenfeld, A. A Comparative Study of Texture measures for Terrain Classification. *TransSMC 6* (1976), 269–285.

[40] Wu, Z., and Leahy, R. An optimal graph theoretic approach to data clustering : Theory and its application to image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell. 15*, 11 (1993), 1101–1113.